Méthode de conception de produit intégrant ses services en phase conceptuelle appliquée aux projets de construction

Cyril Mauger

2014-ENAM-0046

École doctorale n° 432 : Sciences des Métiers de l'Ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité "Conception"

présentée et soutenue publiquement par

Cyril MAUGER

le 19 décembre 2014

Framework for Integration of Services in Product Requirements Definition Applied to Public Buildings

Directeur de thèse : **Jean-Yves DANTAN**
Co-encadrement de la thèse : **Éric DUBOIS, Ali SIADAT**

Jury

Mme Lucienne BLESSING, Professeur, RUES, University of Luxembourg (LU)	Présidente
M. Benachir MEDJDOUB, Professeur, CVTR Lab, Nottingham Trent University (GB)	Rapporteur
M. Jean-François BOUJUT, Professeur, G-SCOP, Grenoble INP (FR)	Rapporteur
M. Roel WIERINGA, Professeur, CTIT, University of Twente (NL)	Rapporteur
M. François VERNADAT, HDR, Cours des Comptes Européennes (LU)	Examinateur
M. Jean-Yves DANTAN, Professeur, LCFC, Arts et Métiers ParisTech (FR)	Examinateur
M. Eric DUBOIS, Professeur, Université de Namur (BE)	Examinateur
M. Ali SIADAT, Maitre de Conférences, LCFC, Arts et Métiers ParisTech (FR)	Examinateur
M. Sylvain KUBICKI, Docteur, SSI, CRP Henri Tudor (LU)	Invité
M. Christophe GOBIN, Docteur, Vinci Construction France (FR)	Invité

T H È S E

Arts et Métiers ParisTech - Centre de Metz
Laboratoire Conception Fabrication Commande

‘‘Le plus grand obstacle à la vie est l’attente, qui espère demain et néglige aujourd’hui.’’

– Sénèque*, De la brièveté de la vie*

DEDICATION

To all my teachers, who fed me with their knowledge for so many years without any proper feedback or gratefulness. Their altruism is exemplar and too often neglected by their students.

To my parents, who did a great job.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank the Fond National de la Recherche (FNR) Luxembourg for funding this research project through the Aide à la Formation Recherche (AFR) grant scheme dedicated to PhD research project. This thesis would have been impossible without their support. I am also grateful to the Public Research Centre Henri Tudor for hosting me during these nearly five years and for providing me with appropriate equipment and resources. Together with the Laboratoire Conception Fabrication Commande of Arts et Métiers ParisTech, they properly completed the FNR support.

I would like to thank the members of jury of my thesis for their time, effort, and consideration. I warmly thank Professor Lucienne BLESSING, not only for "*accepting*" to be the President of my jury but also for her wise advices and feedbacks on various occasions along these four years. It really meant something to me to have you at my Viva, as a closure. I would like to thank Professor Jean-François BOUJUT, Professor Benachir MEDJDOUB, and Professor Roel WIERINGA for accepting to be the reviewers of my dissertation. Their interesting and detailed reviews provided me great confidence to prepare my defence. I am grateful to Dr François VERNADAT and Dr Christophe GOBIN who kindly accepted to be part of that great jury in charge of judging the results of my research. It was an honour to be assessed by all of you regarding your respective domain of expertise.

I am heartedly thankful to my 4 supervisors for their patience, understanding and support. I cannot say how long it took us to be able to understand each other (or rather to understand me). This was quite a challenge but we managed it together. I am indebted to my main supervisor, Professor Jean-Yves DANTAN, who provided me with guidance when I was lost, support when I was alone, and advices when I was confused. I am thankful to Dr Ali SIADAT, second supervisor from Arts et Métiers ParisTech, for all the professional and personal discussions, for his availability and encouragements. Being my Master Thesis supervisors, I spent more than five years with the two of them. They unconditionally supported my way of thinking, my ideas, and me. There would not have been any thesis without you, the first ones who really believed in me.

My sincerest gratitude also goes to my two co-supervisors from Henri Tudor who joined the team for the PhD: Professor Eric DUBOIS for his high interest and trust about the topic, for the insights from Software Engineering, and for having naturally shared his scientific network; and Dr Sylvain KUBICKI for his commitment on this new topic not only during but also after the PhD, for his expertise on AEC issues on which I still have a lot to learn from him, and for his more than welcome critical feedbacks. Your additional support and expertise were essential to the success of this project.

I warmly thank Professor Daniel BERRY for his commitment and interest, for the precious case study and advices that helped me go through the PhD, and for his cheerful support. We met at the right time, when things got more difficult for me; you pushed me further and were an unexpected source of motivation for me. I am indebted to my friend Dr Ahmed Jawad QURESHI for sharing his knowledge and network with me, for all the discussions we had, and most importantly for his friendship. You did a lot to help me behind the scene, and I am utterly grateful to you for that. I also thank Dr Alain ETIENNE for his time and expertise on the matters of database and metamodeling. I am grateful to Mr André RIFAUT for all the discussions, references and advices regarding GORE.

This research would not have been a success without interesting case studies. I express my gratitude to Messrs Thomas SCHWARTZ and Lahcène HARBOUCHE for accepting to take me as a Master Thesis trainee to work on the Luxembourgish secondary school project which became later my first case study. The initiation of this PhD project would not have been possible without their support. I address my special thanks to my uncle, Mr René MAUGER, for his support by introducing me to the city council of Betton which led me to my second case study: the multimedia library in France. I would like to thank the persons from the city council of Betton: its

mayor, Mr Michel GAUTIER, for accepting to let me use the information about the multimedia library; Messrs Michael PIOLAIN and Patrice VALLEE as well as the head of the multimedia library, Mrs Muriel PIFFETEAU, for their time and all the documents on the case study. I also thank Professor Daniel BERRY and the rabbi from the Canadian synagogue kitchen case study, third case study of this thesis. I would like to thank the Fonds Belval for accepting to let me use the University Library of Luxembourg as a fourth case study, as well as Mrs Marie-Pierre PAUSCH-ANTOINE and Mrs Julie WILLEMS from the University Library for their time, inputs and feedbacks.

I would like to thank all the AEC professionals for their time, interest about, and feedbacks on my research: Mr Matthieu BOURDON; Messrs Serge VANNEYRE, Alain GRANDJEAN, and all the other trainers from the GEPA PAMO training; as well as Vinci Construction France, Polaris Architect and Schroeder & Associés for their support on the topic before the FNR. I also thank Dr Sepideh GHANAVATI, Mrs Margot HARTMAN, Mrs Anja DUMJAHN and the other secretaries from Tudor for the proofreading of this thesis.

I express my gratitude to all my colleagues and friends who provided me with technical, logistical, and moral support along these 4 to 5 years spent between Metz and Luxembourg: Jean-Baptiste, Jinna and Renaud during my stays in Metz; Kiki & Clémence and Bubu & Marianne (+ Eva) during my stays in Paris; Hervé & Olivier for the essential daily coffee breaks; Alek, Alex(s), Amandine, Amir, Annie, Antoine(s), Bruno, Céline, David, Dong, Edwin, Georgios, Hella, Muriel, Pierre, Simon(s), Younes, Yves, Zak… and all the others I forgot to mention in here for your participation in making this journey less difficult (almost pleasant) to complete.

Finally, I am grateful to my parents, Odile & Patrick, my brothers, Xavier & Bastien, and all my (growing) family for their constant support and encouragement despite the distance that never stopped to increase each year since I left Brittany. The distance might continue to increase in the next years but I will never stop to love you for what you have done with me. Last but not least, I would like to thank my cat Grumpfeth for its constant effort to distract me from finishing this damn PhD (it did a great job to promote procrastination with a few other folks). Without it, I might have finished earlier but without comfort. Adopting it was the best thing I did to slow down a little bit, just enough to start to really enjoy the journey.

PREFACE

This PhD research project takes its roots in the master thesis performed by the applicant on "*adapting conceptual design methods to construction projects*" in 2010. The first objective was to analyse and compare Value Management, the APTE® method, Quality Function Deployment (QFD), and Requirements Engineering regarding the architectural programming of Architecture-Engineering-Construction projects. This project was developed side-by-side with a real architectural programming case study: the refurbishment and extension of a secondary school in Luxembourg, the Lycée Technique de Bonnevoie (LTB). Based on the first encouraging results, a PhD grant application was successfully accepted to deepen the research and it gave birth to the following research topic.

The National Research Fund of Luxembourg financed the present project under the AFR grant scheme (Aides à la Formation-Recherche). The application ID is 1000660 associated to the AFR-PhD 2010-2 call for proposals. The original grant covered three years of research from February 2011 to January 2014 with an agreed extension for a fourth year from February to October 2014.

The PhD research project broadens the original scope by aiming to identify trails for the improvement of architectural programming practices through knowledge transfer from engineering domains such as Mechanical Engineering, Software Engineering and Industrial Engineering to Architecture-Engineering-Construction domain. These improvements mainly concern functional requirements definition of public construction projects. Proof of concepts on either post ex facto or current case studies are used to validate those trails. The development of a demonstrator is out of this research scope but part of the research perspectives.

Identified trails constitute a research roadmap to deepen with more qualified experts, professionals, and partners from the required domains on a higher scale. This research project is the first initiative leaded by both the Service Science and Innovation (SSI) department of Public Research Centre Henri Tudor and the Laboratoire Conception Fabrication Commande (LCFC) of Arts et Métiers ParisTech on the architectural programming topic. The major challenge of this project is to develop a multi-disciplinary approach which takes into account the services provided by the building during its conceptual design. This objective will be achieved by the adaptation and the development of a framework based on the interoperability of multi-disciplinary approaches.

The SSI department hosts the following domains of expertise: Software Engineering (e.g. Goal-oriented Requirements Engineering) and Architecture-Engineering-Construction (e.g. ICT, CAD, and BIM). The LCFC hosts the following domains of expertise: Mechanical Engineering (e.g. conceptual design) and Industrial Engineering (e.g. Enterprise Architecture, information systems). Each domain of expertise is represented by one person in the supervision team:

- Pr. Jean-Yves DANTAN, lead supervisor: Mechanical Engineering
- Pr. Eric DUBOIS, research advisor: Software Engineering
- Dr. Ali SIADAT, research advisor: Industrial Engineering
- Dr. Sylvain KUBICKI, research advisor: Architecture-Engineering-Construction

The applicant has a general engineering background with major knowledge in Mechanical Engineering and Industrial Engineering. A professional training was taken in architectural programming at the very beginning of the research project to fill the gap of knowledge on the topic of interest. These are all the ingredients gathered in this research project to provide an original and innovative research and practical contribution.

To comply with the requirements of the European Doctorate Label, the following document was written in English, and the defence was performed in English with a Jury composed of a European committee. A substantial abstract in French completes this document in agreement with the Toubon Law (*Loi n°94-665 du 4 août 1994 relative à l'emploi de la langue française*).

PUBLICATIONS ARISING FROM THE RESEARCH

Referred Conference Papers

1. Mauger, C., Schwartz, T., Dantan, J.-Y., and Harbouche, L., 2010. Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: The elicitation process, in 2010 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), IEEE, pp. 310–314.
2. Mauger, C., and Kubicki, S., 2013. A Conceptual Model for Building Requirements Processing, in Proceedings of the 11th International Post-Graduate Research Conference (IPGRC), University of Salford, UK, pp. 318–330.
3. Mauger, C., 2013. A Design Artifact for Functional Assessment of Construction Projects, in Hadju, M. and Skibniewski, M. eds., Proceedings of the Creative Construction Conference 2013, 6-9 July 2013, Budapest, Hungary, pp. 505–515.
4. Mauger, C., and Dantan, J.-Y., 2013. Buildings Definition as Product-Service Systems, in Proceedings of the Congrès Français de Mécanique 2013, August 26-30th 2013, Bordeaux, France, Bordeaux, France, pp. 1–6.
5. Mauger, C., Dantan, J.-Y., and Dubois, E., 2014. GFBS : A PSS Design Model Applied to the Briefing Process of Construction Projects, in Marjanović, D., Storga, M., Pavkovic, N., and Bojcetic, N. eds., Proceedings of the International Design Conference - Design 2014, Dubrovnik, Croatia, May 19-22 2014, Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, The Design Society, Glasgow, pp. 905–914.
6. Mauger, C., and Berry, D.M., 2014. Lessons Learned from and for Requirements Engineering and Building Construction : A Case Study of Requirements Engineering for a Synagogue Kitchen with Use Cases and Scenarios, in Proceedings of the Software Summit SwSTE 2014 The IEEE Computer Society International Software Conference, 11-12 June 2014, Tel-Aviv, Israel, IEEE Computer Society CPS, pp. 67–76.
7. Mauger, C., and Kubicki, S., 2014. Intégration de la notion de Service dans un processus de modélisation adapté à la programmation architecturale, in Kubicki, S., Halin, G., and Bignon, J.-C. eds., Proceedings of the Séminaire de Conception Architecturale Numérique (SCAN'14), 18-20 Juin 2014, Luxembourg, PUN - Editions Universitaire de Lorraine, pp. 125–135.

Poster Session

1. Mauger, C., and Berry, D.M., 2013. Requirements Engineering and Building Construction : Requirements Engineering for a Synagogue Kitchen with Use Cases and Scenarios, in Poster session REFSQ2013, pp. 15.

Doctoral Symposium

1. Mauger, C., 2012. Method for the conceptual phase of an Integrated Product and Service Design applied to Construction Project, in Dörr, J., Fricker, S., and Paech, B. eds., Proceedings Requirements Engineering: Foundation for Software Quality - 18th International Working Conference (REFSQ 2012), Volume 2 - Workshops, Doctoral Symposium, Empirical Track, Essen, Germany, pp. 365–372.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ACRONYMS & ABBREVIATIONS

AEC	Architecture-Engineering-Construction
AFR	Aide à la Formation Recherche
AIA	American Institute of Architects
APICS	American Production and Inventory Control Society
AQC	Agence Qualité Construction
BIM	Building Information Modelling
BPMN	Business Process Modelling Notation
CAD	Computer-Aided Design
CERTU	Centre d'Etudes sur les Réseaux, les Transports, l'Urbanisme, et les constructions publiques
CRP	Centre de Recherche Public
CRS	Clients Requirements Specification
DGUHC	Direction Générale de l'Urbanisme, de l'Habitat et de la Construction
DRM	Design Research Methodology
DRS	Design Requirements Specification
DS	Descriptive Study
FAST	Function Analysis System Technique
FBD	Function Block Diagram
FBS	Function Behaviour Structure
FFB	Fédération Française du Bâtiment
FRS	Functional Requirements Specification
GEPA	Groupe pour l'Education Permanente des Architectes
GFBS	Goal Function Behaviour Structure
GORE	Goal-Oriented Requirements Engineering
ICT	Information and Communication Technology
IDEF0	Integration DEFinition
IE	Industrial Engineering
IEEE	Institute of Electrical and Electronics Engineers
IFC	Industry Foundation Classes
ISO	International Standard Organisation
ITIL	Information Technology Infrastructure Library
JORF	Journal Officiel de la République Française
KAOS	Knowledge Acquisition in automated specification or Keep All Objectives Satisfied
LCFC	Laboratoire de Conception, Fabrication et Commande
LTB	Lycée Technique de Bonnevoie
ME	Mechanical Engineering
MIQCP	Mission Interministérielle pour la Qualité des Constructions Publiques
MOA	Maitrise d'Ouvrage (i.e. paying client)
MOE	Maitrise d'Œuvre (i.e. design team)
NASA	National Aeronautics and Space Administration
PAMO	Programmation Architecturale Assistance à Maîtrise d'Ouvrage (professional training)
PBS	Product Breakdown Structure
PRS	Production Requirements Specification
PS	Prescriptive Study
PSS	Product-Service System
QFD	Quality Function Deployment
RC	Research Clarification
RIBA	Royal Institute of British Architects
SADT	Structured Analysis and Design Technique
SE	Software Engineering
SoPSS	Service-oriented Product-Service System
SPOS	Service de Psychologie et d'Orientation Scolaire
SSI	Service Science & Innovation
SYPAA	Syndicat des Programmistes en Architecture et en Aménagement
UK	United Kingdom
UML	Unified Modelling Language
USA	United States of America

Part A – Résumé Etendu en Français

Chapitre I – CONTEXTE

Ce premier chapitre introduit le contexte de quatre années de recherche sur le sujet de la programmation architecturale et de la définition des besoins en ingénierie. La Section 1 définit, positionne et analyse la programmation architecturale en tant que sujet de recherche. La Section 2 introduit les objectifs industriels et scientifiques poursuivis tout au long de ces travaux. La Section 3 présente la méthodologie mise en place pour atteindre ces objectifs. Enfin, la Section 5 décrit l'organisation du résumé étendu.

1. Introduction

Le cycle de vie d'un bâtiment est composé d'activités complexes allant de l'idéation à la démolition. Chaque activité implique de nombreux acteurs hétérogènes sur une période de temps relativement courte. L'organisation de ces activités peut être multiple en fonction de différents critères (p. ex. type de bâtiment, cadre contractuel, secteur public ou privé, temps et budgets alloués, etc.). Dans ces travaux de recherche, une organisation séquentielle de ces activités est considérée afin de limiter la complexité du sujet, même si cela ne reflète pas les pratiques actuelles.

Cette section propose une vue globale de la phase conceptuelle des projets de construction appelée programmation architecturale. Cette section couvre la définition de la programmation architecturale, quelques spécificités liées au secteur de la construction reflétant sa complexité, ainsi qu'un état de l'art décrivant le contexte de cette recherche.

1.1. La Programmation Architecturale

La programmation architecturale consiste à définir le cadre et les attentes d'un projet de construction sur base des exigences formulées par la Maitrise d'Ouvrage (MOA) (c.-à-d. le client) (Abdul-Kadir & Price 1995). Cette tâche incombe généralement à la MOA qui peut se faire accompagner d'un professionnel de l'Assistance à MOA (AMOA) appelé programmiste (Certu 2010). La programmation architecturale est un processus riche en informations à générer, à manipuler et à gérer (Voordt & Wegen 2005). Son but est d'informer la MOA lors de différentes phases décisionnelles du projet de construction (Kelly et al. 2002).

La phase de programmation architecturale, comme toute phase conceptuelle, est basée sur une grande quantité d'informations vagues et incomplètes qu'il est nécessaire d'analyser et de clarifier (Wang et al. 2002). Macmillan propose une structuration de cette phase en deux étapes (Macmillan et al. 2001) : la première consiste à traduire les besoins métiers de haut niveau en une stratégie de conception tandis que la seconde concerne le développement de cette stratégie de conception en propositions de concepts architecturaux (Figure 1). La programmation architecturale se concentre sur la première étape de cette représentation de la phase conceptuelle.

La programmation architecturale est reconnue comme étant l'une des phases les plus importantes dans le déroulement des projets de construction (Shen et al. 2004; Yu et al. 2005; Tzortzopoulos et al. 2006). Les études préliminaires, la programmation architecturale et les études de conception représentent près de 10% du coût total d'un projet de construction, le reste étant dédié à la construction du bâtiment (Certu 2010). Au cours de ces phases, près de 75% des coûts globaux du projet sont engagés (Faatz 2009; Certu 2010). Paulson, et plus tard MacLeamy, ont déjà démontrés que ces phases amont ont la plus grande influence sur le projet de construction tout en ayant les plus faibles coûts en termes de modifications (Paulson 1976). Il s'agit là de la meilleure opportunité d'amélioration pour un projet de construction (Figure 2).

1.2. Spécificités des Projets de Construction

La programmation architecturale se distingue de la phase conceptuelle des autres domaines d'ingénierie de par la nature même de son objet d'étude. Un bâtiment est défini par de nombreuses exigences et contraintes à satisfaire à un niveau social, urbain, fonctionnel, technique autant que économique, d'insertion dans le paysage et de protection de l'environnement, relatives

à sa réalisation ainsi qu'à son utilisation (DGUHC & Certu 1999; JORF 2007). Tous ces éléments constituent un ensemble considérable d'informations hétérogènes, imprécises et incertaines, issues de sources multiples qu'il est nécessaire de rassembler et traiter tout au long du processus de programmation architectural (Shen et al. 2004).

La programmation architecturale se base sur une définition incertaine des besoins métiers de la MOA. Ces besoins métiers dépendent fortement du contexte local du projet de construction. La stratégie opérationnelle de la MOA qui en découle est étroitement liée aux exigences bâtiments qui sont à définir (Tzortzopoulos et al. 2006). La compréhension de cette stratégie est rarement complète, même pour la MOA qui éprouve des difficultés pour la décrire (Cooper & Press 1995; Barrett & Stanley 1999). Tout au long du projet, cette stratégie opérationnelle est également soumise à de nombreux changements et évolutions.

La programmation architecturale se confronte à de nombreux acteurs de nature complexe et manquant de coordination (Newcombe 2003; Tzortzopoulos et al. 2006) :

- La **MOA** et ses parties prenantes représentent trois types de **clients** : le ''client payeur'' (dit MOA), le ''client utilisateur'' (dit utilisateur), et le ''client usager'' (dit usager). Dans le cas des bâtiments publics, ces trois entités sont représentées par des personnes bien distinctes ayant un rapport différent avec le bâtiment. *Dans l'exemple d'un lycée, le MOA est le Ministère de l'Education Nationale qui finance le projet. L'utilisateur est représenté par le personnel de l'établissement (p. ex. les enseignants, le directeur, le concierge…). Ce personnel est nécessaire à la transmission de connaissance aux élèves, usagers de l'établissement.*
- La **Maitrise d'œuvre** (MOE) (c.-à-d. les **concepteurs**) est généralement représentée par une équipe de conception regroupant un ou plusieurs cabinets d'architectes, divers bureaux d'études et autres consultants allant de l'économiste de la construction aux experts légaux. Cette équipe de conception n'est souvent pas encore définie au moment de la programmation architecturale (p. ex. dans le cas des concours d'architecture).
- Les **entrepreneurs** sont aussi représentés par une équipe de construction temporaire qui évolue tout au long du projet de construction. Chaque membre de cette équipe se voit assigner un certain nombre de taches interdépendantes au niveau du temps et du séquencement ce qui amène à des temps de construction longs et irréguliers (Kubicki et al. 2006).

Enfin, les bâtiments se distinguent également des autres types de produits (logiciel ou manufacturier) de par sa durée de vie relativement longue comparée à son usage intensif ainsi qu'aux coûts résultants de son cycle de vie (de sa conception à sa démolition). En général, un bâtiment a plusieurs secondes vies à travers la rénovation et la réhabilitation. De telles caractéristiques sont à prendre en compte au plus tôt dans le processus de développement de projet, ce qui impacte la phase de programmation architecturale.

1.3. Etat de l'Art

D'après la littérature, les projets de construction souffrent de nombreux problèmes en termes de coût, de qualité et de délai impactant directement la satisfaction de la MOA. Il s'avère qu'une part importante de ces problèmes est originaire de la phase de programmation architecturale (Allégret & Guilleux 2000; DGUHC 2000; Love & Li 2000; Yu et al. 2008; Feng & Tommelein 2008; Certu 2010; AQC 2013). Un panorama historique (Bousbaci 2004), pratique (Duerk 1993; Cherry 1999; Hershberger 1999; Barrett & Stanley 1999; Allégret et al. 2005a; Certu 2010) et littéraire (Kamara et al. 1999; Green & Simister 1999; Kamara et al. 2002; Shen et al. 2004; Yu et al. 2005; Kiviniemi 2005; Yahya et al. 2007; Bogers et al. 2008; Chung et al. 2009; Huovila & Hyvarinen 2012; Koutamanis 2013; Hassanain & Juaim 2013) de la programmation architectural met en évidence de nombreuses difficultés relatives à cette phase des projets de construction.

Les pratiques de programmation architecturale sont actuellement fortement dépendantes de l'expérience des architectes et des programmistes. Les bonnes pratiques, méthodes et approches

de programmation architecturale sont encore peu communiquées notamment car elles représentent un avantage concurrentiel aux professionnels offrant cette mission d'AMOA.

La MOA se repose entièrement sur la mémoire et les compétences du programmiste. Le processus de programmation architecturale est encore principalement réalisé de manière manuelle et sous format papier. Par conséquent, la cohérence globale et la complétude du programme restent souvent difficiles à assurer et manquent de clarté, sans oublier le temps perdu dans l'exercice. Quelques solutions sont apportées au niveau de la recherche sur les technologies de l'information et de la communication (p. ex. programmation architecturale assistée par ordinateur) avec plus ou moins de succès. Cependant, même si ces initiatives sont trop souvent focalisées sur le bâtiment plutôt que sur les besoins métiers et opérationnels de la MOA, elles nécessitent plus d'attention (Koutamanis 2013).

1.4. Synthèse

La programmation architecturale, au même titre que la phase conceptuelle en ingénierie, est une étape essentielle dans un projet de construction. En effet, des décisions critiques doivent être prises par le MOA sur base d'une quantité importante d'informations vagues, instables et incomplète. Au travers de cet état de l'art, nous avons montré qu'il y a encore des améliorations à apporter en termes de pratiques, recherches et connaissances. Malgré une grande quantité de travaux sur le sujet, quelques pistes sont encore à explorer, notamment en ce qui concerne la relation entre les besoins stratégiques de la MOA (besoins métiers, opérationnels) et les exigences détaillées des clients utilisateurs et usagers (exigences bâtiment) (Kiviniemi 2005). Par conséquent, ces travaux de recherche se limitent à la formulation des exigences bâtiments sur base des besoins métiers de la MOA (Figure 3). Basée sur ces différents éléments, la Section 2 présente les objectifs industriels et scientifiques qui ont guidés ces travaux.

2. Objectifs

Cette section est divisée en deux parties : la première partie présente les objectifs industriels de ces travaux, tandis que la seconde en précise la problématique scientifique et les questions de recherche traitées dans ces travaux.

2.1. Objectifs Industriels

Les objectifs industriels présentent le cadre de ces quatre années de recherche en se basant sur des problèmes pratiques. Ces objectifs reflètent notre compréhension de la programmation architecturale sur base d'un état de l'art du domaine et d'observations faites sur le terrain (c.-à-d. entretiens, pratiques et formation professionnelle).

Le cadre de ces travaux se limite aux **bâtiments publics multi-utilisateurs** dans les limites de la Loi sur la MOA Publique (Loi MOP) française (JORF 2010). Dans ce contexte, la programmation architecturale se doit d'être réalisée par la MOA (assistée d'un professionnel de la programmation : le programmiste) avant consultation de la MOE qui ne peut y être impliquée. Le programme est le seul document transmis à la MOE lui permettant de comprendre et de répondre aux besoins de la MOA. Au-delà d'un certain coût, un concours d'architecture est obligatoirement mis en place. Dans ce contexte, la MOA se retrouve devant plusieurs difficultés : définir un programme seul (ou presque), évaluer des propositions architecturales et les comparer sur base de ce programme. De plus, ses connaissances sur le projet sont souvent limitées à une compréhension partielle de ses propres besoins métiers et une appréciation encore plus restreinte des besoins et activités des utilisateurs et usagers. Ces entités sont généralement inconnues ou indisponibles au moment de la programmation architecturale.

Cette recherche se focalise sur sept problèmes pratiques relatifs au programme et aux méthodologies de programmation architecturale (Figure 4) :

a. les programmes ‘‘**copier-coller**’’ où un même programme est utilisé pour deux projets d'un même type sans réelle prise en compte du contexte local (SC-Construct Project),

b. les **conflits** entre exigences dans le programme (Barrett & Stanley 1999; Tzortzopoulos et al. 2006),

c. le **manque de précision** dans la définition de ces exigences (Barrett & Stanley 1999; Tzortzopoulos et al. 2006),

d. les exigences **implicites** (Barrett & Stanley 1999),

e. le **manque de complétude** des programmes (Barrett & Stanley 1999; Kamara et al. 2002; Shen et al. 2004; Yu et al. 2007),

f. **l'absence de formalisme** et de **normes** en programmation architecturale (formation PAMO du GEPA), ainsi que

g. le **manque de réelle compréhension commune** des besoins métiers de la MOA, des utilisateurs et des usagers (Kelly et al. 2002; Kamara et al. 2002; Yu et al. 2006; Kalay 2001).

Les problèmes pratiques relatifs aux comportements humains des parties prenantes tels que leur degré d'implication (Egan 1998), la communication entre parties prenantes (Yahya et al. 2007; Tzortzopoulos et al. 2006), ou les problématiques de collaboration (Kalay 2001) ne font pas partie du cadre de ces travaux.

La résolution de chacun de ces problèmes pratiques n'est pas réalisable en un seul projet. Ce projet de recherche se limite donc à la définition d'un **cadre** et d'un **outillage méthodologique** pour la **partie fonctionnelle** du **programme** dans le cas des **constructions publiques**. La partie fonctionnelle du programme se concentre sur la description des activités à réaliser dans le futur bâtiment (Gobin 2003) menant à la définition des espaces nécessaires et de leurs relations en termes de dépendance.

Ce but se décompose en plusieurs objectifs. Ces objectifs sont à considérer comme des principes à prendre en compte dans la définition du cadre et des outils méthodologiques. Ils représentent des indicateurs de performances de la solution proposée. L'objectif de ces travaux n'est pas de les satisfaire complètement mais plutôt d'analyser la façon dont chacun d'entre eux est abordé dans la proposition finale :

Objectif 1 – Aider la MOA, les utilisateurs et les usagers à **exprimer** leurs exigences en prenant en compte les aspects de **complétude**, de **pertinence** et de **fiabilité** de ces exigences (Barrett & Stanley 1999; Kelly et al. 2002; Kamara et al. 2002; Shen et al. 2004; Tzortzopoulos et al. 2006; Yahya et al. 2007; Yu et al. 2007; Feng et al. 2009).

Objectif 2 – Supporter l'**analyse** et le **traitement** de ces exigences en termes de **consistance**, **maturité**, **traçabilité** et **dépendances** entre elles afin de permettre une **aide à la décision** tout au long du processus de programmation (Barrett & Stanley 1999; Kamara et al. 2002; Gobin 2003; Tzortzopoulos et al. 2006; Yahya et al. 2007; Yu et al. 2007; Feng et al. 2009; Certu 2010).

Objectif 3 – **Spécifier** les exigences de manière **claire** et **utile** afin de permettre leur **validation** par la MOA, leur **utilisation** par la MOE. Cette spécification doit refléter l'état des pratiques de la MOE par rapport à ses outils actuels (p. ex. outils et langages informatiques d'aide à la conception).

Objectif 4 – **Vérifier** que les exigences demandées par la MOA sont bien implémentées par la MOE (adéquation programme-projet de manière objective).

Chacun de ces objectifs concerne une étape différente du processus de programmation architecturale. Les contributions issues de ces travaux de recherche constituent une première étape dans la poursuite de cette stratégie.

2.2. Objectif Scientifique

A partir des éléments présentés dans les sections et sous-sections précédentes, il nous semble que la programmation architecturale fait référence à un processus de traduction des besoins

métiers de la MOA en exigences bâtiment. Ceci implique qu'il existe un lien entre les besoins stratégiques de haut niveau (c.-à-d. orientés métiers) et les exigences techniques de plus bas niveau (c.-à-d. orienté bâtiment). Dans ce mémoire, les ''*besoins*'' sont considérés comme une expression brut de ce que la MOA veut, souhaite, ou désire, tandis que les ''*exigences*'' sont considérées comme une définition structurée et formalisée de ces besoins à destination de la MOE.

Dans le domaine de l'Architecture-Ingénierie-Construction, ce processus de traduction des besoins en exigences est fortement lié à l'expérience du programmiste. Au regard de l'état de l'art sur la programmation architecturale, les approches actuelles se caractérisent par une formalisation plutôt tardive des besoins en exigences (Figure 5). Les besoins métiers sont généralement traduits en besoins bâtiment (c.-à-d. en termes d'espaces) avant d'être formalisés et structurés sous forme d'exigences bâtiment. Le processus de programmation a en effet tendance à être rapidement orienté vers la génération de principes de solutions architecturales.

Dans les autres domaines d'ingénierie, cette transition entre les besoins de haut niveau et les exigences de bas niveau est bien mieux tracée et formalisée. En Génie Mécanique, les besoins sont formalisés par des **fonctions** avant d'être déclinées en exigences produit. En Génie Logiciel, les besoins sont formalisés sous forme d'**objectifs** avant d'être opérationnalisés par des exigences sur les logiciels. En Génie Industriel, les besoins sont formalisés par des **processus** et des **activités** avant d'être instanciés par des exigences sur les ressources. Ces fonctions, objectifs, processus et activités sont des exigences de haut niveau, c'est-à-dire des formalisations de haut niveau des exigences sur leurs systèmes respectifs. Ces exigences de haut niveau sont de nature abstraite ce qui retarde le raisonnement sur les principes de solutions à mettre en place. La définition de ces exigences est pleinement supportée par un ensemble d'approches, méthodes, outils et techniques propres à chaque domaine.

L'objectif scientifique de ces travaux de recherche est de développer une méthode de programmation architecturale basée sur les connaissances issues de ces domaines d'ingénierie (c.-à-d. les Génies Mécanique, Logiciel et Industriel). Cette méthode devra expliquer la ou les **relations** entre les **besoins** métiers de la MOA et les **exigences** bâtiment, ainsi que la **transition** entre les **besoins de haut niveau** et les **exigences de bas niveau**. La méthode proposée vise à une formalisation amont (c.-à-d. tôt dans le processus) des besoins en exigences (Figure 6).

Pour atteindre un tel objectif, trois questions de recherche structurent la démarche :

QR1 : Quels sont les concepts nécessaires pour décrire un bâtiment et son domaine de définition ?

QR2 : Comment modéliser un bâtiment et son domaine de définition tout en intégrant les différents points de vue des parties prenantes impliquées ?

QR3 : Comment dériver les exigences bâtiment à partir des besoins métiers ?

La première question de recherche fait référence aux informations nécessaires pour décrire l'objet d'étude. La deuxième question de recherche porte sur le périmètre de l'objet d'étude et la cohérence des différents points de vue sur l'objet d'étude pouvant être pris. La dernière question de recherche demande plus de détails sur le développement des exigences bâtiments en partant des besoins métiers de la MOA.

2.3. Synthèse

Dans cette section, nous avons présenté le cadre dans lequel cette recherche s'inscrit. Les motivations, intérêts et aspects prospectifs de cette recherche ont été établis au travers de divers constats au niveau des pratiques et des connaissances scientifiques sur la définition des besoins en Ingénierie. La volonté de conserver un aspect applicatif a pu être exprimée dans la description des objectifs industriels. Dans ce contexte, la problématique de recherche abordée dans ces travaux se focalise sur le transfert de connaissances sur la définition des besoins issues de divers domaines d'ingénierie vers le domaine de l'Architecture-Ingénierie-Construction.

3. Méthodologie

L'approche mise en place pour atteindre les objectifs présentés dans la Section 2 a été guidée par notre compréhension et perception de la programmation architecturale. Celle-ci est influencée par un certain nombre d'observations sur les similarités entre le domaine de l'Architecture-Ingénierie-Construction et les autres domaines d'ingénierie. Il en résulte l'identification de possible transfert de connaissances issues des Génies Mécanique, Industriel et Logiciel vers l'Architecture-Ingénierie-Construction afin de faire progresser la programmation architecturale. Cette approche s'appuie également sur une méthodologie de recherche issue de la *Design Science* et de la *Design Research*, ainsi qu'un certain nombre de cas d'études.

3.1. Sciences de la Conception et Recherche en Conception

La méthodologie suivie au cours de cette recherche est un mélange de deux approches : l'approche de la *Design Science* proposée par (March & Smith 1995) et l'approche de *Design Research* proposée par (Blessing & Chakrabarti 2009). L'approche de March & Smith est structurée selon deux axes (Table 3) : le premier axe présente les activités de recherche : construire, évaluer, découvrir, et justifier, tandis que le second présente les résultats de recherche : concept, modèle, méthode et instanciation.

Nos travaux de recherche se limitent à l'activité de construction d'artefacts pour répondre aux objectifs industriels et scientifiques. Il s'agit là de poser les premiers éléments pour une recherche sur le long terme. Les autres activités de recherche sont réservées pour les perspectives. Cette activité de construction d'artefact est étendu sur l'ensemble des résultats de recherche définis par March & Smith afin d'assurer une continuité entre ces artefacts.

L'approche DRM (*Design Research Methodology*) introduite par (Blessing & Chakrabarti 2009) est utilisée comme élément structurant de la méthodologie de recherche appliquée dans ces travaux. Cette approche est composée de quatre étapes : la clarification de la recherche (RC), une première étude descriptive (DS-I), l'étude prescriptive (PS) et une seconde étude descriptive (DS-II). L'approche DRM est présentée comme un processus itératif avec diverses combinaisons possibles (sept types de recherche identifiés). La combinaison suivie dans cette thèse correspond au type 3 incluant : une RC basée sur une revue de la littérature, une DS-I également basée sur une revue de la littérature, une PS exhaustive et une initialisation de DS II (Figure 7).

3.2. Cas d'Etudes

Ces travaux de recherche se sont appuyés sur trois cas d'études ex post facto et un cas d'étude en cours : un lycée luxembourgeois, une médiathèque française, une synagogue canadienne, et une bibliothèque universitaire luxembourgeoise. En Figure 8 sont représentés les documents utilisés dans chaque cas d'étude en tant que source d'information.

Le premier cas d'étude est celui qui a motivé ces travaux de recherche. Il s'agit de la rénovation et l'extension d'un lycée au Luxembourg (projet d'environ 10 M€). L'objectif de ce cas d'étude était d'évaluer l'impact d'une nouvelle organisation pédagogique à mettre en place dans le lycée sur les exigences bâtiment. Ce cas d'étude a permis d'établir un premier lien entre les objectifs de la MOA, les processus internes de l'établissement et les exigences spatiales.

Le deuxième cas d'étude est une médiathèque en France (projet d'environ 2 M€). L'objectif de ce cas d'étude était de retracer un certain nombre d'exigences fonctionnelles à partir du programme dans les synthèses réalisées par le programmiste (dans le diagramme fonctionnel) et par les différents cabinets d'architecture ayant répondus au concours (dans leurs projets architecturaux). Au travers de la modélisation du cycle du livre, ce cas d'étude a permis de développer une des contributions principales de ces travaux de recherche permettant de modéliser des liens informationnels entre le programme, le diagramme fonctionnel et les projets architecturaux notamment en s'appuyant sur les notions d'espaces et de relations fonctionnels.

Le troisième cas d'étude est une cuisine de synagogue au Canada (projet d'environ 50 k€). L'objectif de ce cas d'étude était d'évaluer deux propositions d'agencement de cuisine sur base d'un certain nombre de cas d'utilisation d'une cuisine et au regard des exigences spécifiques issue

de la religion juive. Ce cas d'étude a permis de voir comment les besoins d'un cuisinier pouvaient être modélisés via différents modèles d'ingénierie puis être retracés dans les différentes propositions d'agencement pour en faire une comparaison.

Le dernier cas d'étude est une bibliothèque universitaire au Luxembourg (projet d'environ 60 M€). L'objectif de ce cas d'étude était d'aider les bibliothécaires à préparer l'allocation spatiale des collections (environ 142 000 ouvrages physiques) dans les zones prévues à cet effet (neuf zones réparties sur quatre étages) selon un certain nombre de critères (p. ex. suivi de la classification Dewey au travers des étages). Ce cas d'étude a permis de voir comment la phase d'adéquation programme-projet pouvait être abordée en s'appuyant sur une modélisation de certaines exigences du programme d'une part et de caractéristiques du bâtiment d'autre part.

Chacun de ces cas d'étude nous apporte différents types d'informations allant d'un niveau relativement global jusqu'à un niveau très détaillé (Figure 9). De plus, chaque cas d'étude couvre seulement une partie du processus de programmation architecturale ce qui fait que la somme des quatre permet d'en couvrir la totalité.

3.3. Transdisciplinarité

Chaque domaine d'ingénierie a ses propres spécificités. De ces spécificités ont été développées des pratiques et des théories propres à chaque domaine. Plutôt que de se focaliser sur ces spécificités, ces travaux de recherche se concentrent sur l'identification de points communs entre ces domaines. Ces similarités, accompagnées d'un certain nombre d'approximations, rendent possible l'utilisation de modèles, outils ou techniques en dehors de leur domaine de prédilection. Dans le cadre de ces travaux, quatre domaines d'ingénierie sont considérés : le Génie Mécanique, le Génie Industriel, le Génie Logiciel et l'Architecture-Ingénierie-Construction.

Les premiers points communs entre ces quatre domaines ont été identifiés en amont de ces travaux. Ils concernent notamment les problèmes rencontrés durant la phase conceptuelle ainsi que les solutions (p. ex. concepts, modèles, théories, méthodes, outils ou techniques) apportées par chaque domaine pour améliorer et outiller la définition des besoins sur leur propre système. Quel que soit le domaine, la phase conceptuelle y est considérée comme étant la phase la plus importante dans le processus de développement d'un système. Elle propose le levier le plus intéressant pour l'amélioration d'un projet en termes de coûts/impact (Boehm & Papaccio 1988; Abdul-Kadir & Price 1995; Hsu 2000; Wang et al. 2002). Cependant, il s'agit également de la phase la moins bien comprise, plus particulièrement dans le domaine de l'Architecture-Ingénierie-Construction (Macmillan et al. 2001).

Dans ces travaux, les solutions existantes dans les autres domaines d'ingénierie sont réutilisées et adaptées pour améliorer le processus de programmation architecturale avant de commencer à en inventer de nouvelles. Ainsi chaque chapitre de ce mémoire fait référence à des éléments en dehors du domaine d'application de ces travaux. Ces éléments viennent supporter les contributions proposées en termes de faisabilité ou de valeur ajoutée.

4. Plan du Résumé Etendu

Ce résumé étendu suit la même organisation que la version complète en anglais en cinq chapitres (Figure 10 – Table 4). Le Chapitre I introduit le contexte de ces travaux de recherche et pose quelques bases sur la programmation architecturale via son état de l'art. Le cadre, les lignes directrices et la méthodologie suivie, ainsi que les objectifs industriels et scientifiques y sont décrits. Les chapitres suivant sont organisés autour des trois questions de recherches traitées dans ces travaux.

Le Chapitre II se concentre sur le domaine de définition de l'objet bâtiment et de la programmation architecturale. Il introduit l'objet d'étude au travers d'une révision des concepts requis pour le décrire. Les concepts proposés sont issus d'une revue multidisciplinaire des approches de développement de systèmes.

Le Chapitre III traite de la modélisation de l'objet d'étude et de son processus de définition en deux parties. La première partie propose un ensemble de modèles conceptuels permettant de définir le périmètre de l'objet d'étude. La seconde partie se concentre sur un ensemble de modèles opérationnels permettant de représenter de manière pratique les exigences sur l'objet d'étude selon plusieurs points de vue.

Le Chapitre IV illustre les contributions des Chapitres II et III en s'appuyant sur les cas d'études. Il est structuré en trois parties. La première partie présente une méthode de programmation architecturale permettant de dériver les exigences bâtiment à partir des besoins métiers de la MOA. La deuxième partie présente des instanciations possibles de cette méthode sous forme de modèles de processus selon plusieurs points de vue : séquences d'activités, flux d'information et collaboratif.

Le Chapitre V termine ce résumé étendu par deux conclusions : une première orientée recherche et une seconde orientée industrielle. Ces conclusions reviennent sur les objectifs définis dans le Chapitre I et proposent un ensemble de perspectives pour la suite des travaux.

Chapitre II – DECRIRE UN BATIMENT

Le Chapitre I nous a permis d'identifier un manque à un niveau de connaissances scientifiques (théorique). En effet, la programmation architecturale s'apprend principalement via l'expérience et s'améliore avec la pratique et le temps. Le Chapitre II propose un ensemble de concepts nécessaires à la structuration des informations permettant de définir un bâtiment dans le cadre de ces travaux de recherche (c.-à-d. programme fonctionnel).

La Section 1 introduit la structure du Chapitre II par une clarification de la première question de recherche et une courte synthèse de l'existant. La Section 2 propose une analyse transdisciplinaire de processus de développement de système afin d'identifier les concepts principaux utilisés pour décrire un objet d'étude dans leur domaine de référence. A partir d'une sélection de concepts structurés autour de questions simples, la Section 3 présente la synthèse d'une analyse multidisciplinaire de ces concepts et de leurs définitions ayant menée à la formulation de définitions cohérentes propres à notre objet d'étude. Ces concepts et leurs relations en forment le domaine de définition sur lequel la Section 4 conclue ce chapitre.

1. Introduction

Quels sont les concepts nécessaires pour décrire un bâtiment et son domaine de définition ?

Selon (March & Smith 1995), les concepts (*constructs*) forment le vocabulaire du domaine utilisé pour décrire les problèmes et pour spécifier leurs solutions dans ce domaine. Le domaine de référence de ces travaux est la programmation architecturale, tandis que l'objet d'étude est un bâtiment. L'objectif de ce chapitre est donc d'identifier le vocabulaire nécessaire pour décrire un bâtiment sous forme d'exigences à partir des besoins de la MOA.

Selon les guides et bonnes pratiques du domaine, il est conseillé d'utiliser le même vocabulaire que celui de la MOA pour la phase de programmation architecturale (DGUHC 2000). Cependant, il est aussi nécessaire d'adopter un premier vocabulaire pour permettre aux programmistes de comprendre ce qui est à définir pendant la phase de programmation quel que soit le type de projet ou la MOA du projet. Ce vocabulaire ‘‘métier’’ serait ainsi propre aux programmistes, sa traduction dans le langage de la MOA se faisant à travers le programme.

En France, la Loi MOP définit un cadre légal et certaines consignes à destination de la MOA à propos du programme :

> *‘‘Le maître de l'ouvrage définit dans le programme les objectifs de l'opération et les besoins qu'elle doit satisfaire ainsi que les contraintes et exigences de qualité sociale, urbanistique, architecturale, fonctionnelle, technique et économique, d'insertion dans le paysage et de protection de l'environnement, relatives à la réalisation et à l'utilisation de l'ouvrage.’’*
> (JORF 2007)

Pour définir ces éléments, la MOA peut se faire accompagner d'une AMOA, c'est-à-dire un programmiste. Sur base de cette consigne et au regard de la littérature sur la programmation architecturale, une quantité importante de concepts est identifiée : But, Objectif, Fonction, Activité, Espace, Coût/Economie, Temps, Valeur, Fait, Besoin, Idée, Problème, Forme, Concept, Performance, etc. Dans ces travaux de recherche, nous nous limitons aux concepts relatifs à la qualité fonctionnelle d'un bâtiment. Nous définissons cette qualité fonctionnelle par la capacité du bâtiment à satisfaire les besoins métiers et opérationnels de la MOA.

Quelques chercheurs en Architecture-Ingénierie-Construction proposent des ontologies mais ces dernières sont généralement focalisées sur les aspects gestion de projet et processus de construction (El-Diraby et al. 2005; El-Gohary & El-Diraby 2011) ou encore la conception du bâtiment (Peachavanish et al. 2006) plutôt que sur le bâtiment en lui-même et ce à quoi il sert.

Du côté de la Conception Assistée par Ordinateur, les *Industry Foundation Classes* (IFC) sont proposées comme modèles de données destinés à décrire un bâtiment et ses données de construction sous forme d'objets (buildingSMART 2013). Ces objets sont structurés en six

concepts fondamentaux répondant à six questions simples : Acteurs (Qui), Contrôles (Pourquoi), Groupes (Quoi), Produits (Où), Processus (Quand) et Ressources (Comment). Ces concepts ne couvrent pas la description des usages d'un bâtiment mais seulement les exigences bâtiment (c.-à-d. les informations de sortie de la programmation architecturale). La relation liant les besoins de la MOA (c.-à-d. les besoins de haut niveau) et les exigences bâtiment (c.-à-d. les exigences de bas niveau) est absente de ce modèle de données (Kiviniemi 2005). Même si une partie des concepts correspondants couvrent la problématique traitée dans ce chapitre, le point de vue sur l'objet d'étude est différent. Les IFC sont définies avec un point de vue conception-construction du bâtiment alors que dans ces travaux, il s'agit d'un point de vue programmation architecturale. Cette différence de points de vue est développée dans les sections suivantes.

2. Développement de Systèmes : Vue d'Ensemble Multidisciplinaire

Dans cette section, nous proposons une analyse transdisciplinaire de processus de développement de système afin d'identifier les concepts principaux utilisés pour décrire un objet d'étude (ou système) dans leur domaine de référence. Les approches analysées sont présentées dans les Tableaux 5 et 6.

Les Figures 12 et 13, reprises sur un A3 en Annexe A, représentent les résultats des deux premières analyses qui ont consistées à positionner les concepts manipulés dans chaque approche sur un axe quasi-temporel partant de concepts abstraits vis-à-vis de la description de l'objet d'étude (p. ex. but du système) vers des concepts plus concrets (p. ex. dimensionnement des composants).

Sur base de ces résultats, une troisième analyse en trois étapes a été menée, d'un point de vue global puis local. La première étape se concentre sur un point de vue processus résultant à l'identification de trois étapes dans le processus de développement de système (Figure 14) : l'analyse externe, l'analyse interne, et l'analyse industrielle.

La deuxième étape se concentre sur les livrables produits au cours du processus de développement. Il en ressort la génération de cinq livrables (Figure 15) : le cahier des charges général (CRS), le cahier des charges fonctionnel (FRS), le cahier des charges concepteur (DRS I), le dossier de conception (DRS II), et le dossier produit (PRS).

La troisième étape se concentre sur les informations définies en amont de ces livrables. Cette dernière étape nous donne une certaine structuration des concepts utilisés pour décrire un objet d'étude quel que soit le domaine d'ingénierie selon les mêmes six questions fondamentales utilisées pour structurer les IFC (Figure 16) : Pourquoi, Quoi, Comment, Quand, Qui et Où. Ces questions sont reformulées afin de correspondre au contexte de ces travaux :

- Pourquoi avons-nous besoin de ce système ?
- Que doit faire le système ?
- Comment doit-il le faire ?
- Quand doit-il le faire ?
- Qui ou quelle partie de ce système doit le faire ?
- Où est-ce que cela doit-il être fait ?

Pour répondre à ces questions, un premier ensemble de concepts est identifié sue base des analyses précédentes. Ces concepts sont choisis par rapport à leur sens et leur importance dans leur domaine respectif, chaque concept étant associé à une question : But-Pourquoi, Fonction-Quoi, Activité-Comment et Quand, Ressource-Qui et Espace-Où (Figure 17). Il est à noter que la question du Qui associé au concept de Ressource couvre autant les entités humaines que matérielles.

Ces concepts ne portent pas la même définition selon leur domaine d'utilisation. En se basant sur un état de l'art de leurs différentes définitions, la section suivante propose une nouvelle

définition adaptée au contexte de recherche de ces travaux, c.-à-d. à la programmation architecturale.

3. Clarification des Concepts

Dans la section précédente, cinq concepts permettant de répondre à six questions ont été identifiées. Dans cette section, ces cinq concepts sont définis à partir d'une revue de leur définition et de leur sens à travers les différents domaines d'ingénierie où ils sont communément utilisés. Dans cette version résumée, seule la synthèse des analyses menant à la définition de chaque concept pour la programmation architecturale est présentée. Pour faciliter la compréhension des définitions, des exemples basés sur le cas d'étude du Lycée Technique du Luxembourg sont proposés.

3.1. But – Pourquoi

Le concept de *But* se retrouve, en termes de définition et d'usage, dans quatre domaines : la Programmation Architecturale en Architecture-Ingénierie-Construction, la Conception en Génie Mécanique, le Management Engineering en Génie Industriel et les approches d'ingénierie des exigences orientée objectif (GORE) en Génie Logiciel.

L'analyse transdisciplinaire du concept de *But* (ou *Objectif*) nous montre une distinction claire en termes de sens avec le concept de *Fonction* qui y est souvent assimilé. Le concept de *But* se réfère à la justification de la nécessité du système tout en esquissant la situation désirée (TO-BE). La confusion qui existe entre le concept de *But* et celui de *Fonction* semble spécifique aux domaines manipulant des systèmes essentiellement physiques (Rosenman & Gero 1998). Dans ces domaines, la *Fonction* est vue comme la justification de la nécessité du système. Dans le cadre de ces travaux de recherche, le système bâtiment n'est pas seulement associé à ces domaines mais aussi aux domaines des services et du business management (Mauger et al. 2013). Les domaines des services ajoutent un niveau d'abstraction supplémentaire (c.-à-d. de plus haut niveau) focalisé sur la question du *Pourquoi* à travers le concept de *But* et en structurant cette réponse sous forme d'une arborescence d'objectifs menant à la définition des exigences de bas niveau de l'objet d'étude. Ce niveau supplémentaire d'information représente un prérequis et une aide à la décision dans le processus de définition des exigences bâtiment.

Définition 1 : Un **But** est un état ou une condition définie par le ou les clients (p. ex. la MOA) que le système doit satisfaire au travers d'une coopération de ses composants et exprimant la raison d'être du système.

En s'appuyant sur la littérature, un **But** peut se raffiner en un ou plusieurs objectifs. Cinq types d'objectifs sont repris de la littérature : les objectifs de type *accomplissement*, les objectifs de type *interruption*, les objectifs de type *maintien*, les objectifs de type *évitement* et les objectifs de type *optimisation* (Figure 19). Cette typologie d'objectifs, illustrée dans le Tableau 7, indique une considération plus globale des justificatifs et des limites cachés derrière la nécessité du système.

3.2. Fonction – Quoi

Il existe un nombre très important de définition pour le concept de *Fonction*. En effet, ce concept est très largement discuté au niveau de la recherche, notamment en Conception. Dans ses travaux de thèse, Eisenbart rapporte un très bon état de l'art multidisciplinaire de ces définitions (Eisenbart 2014) Son corpus couvre un large panel de domaine, incluant notamment : le Génie Mécanique, l'Electronique, l'Ingénierie Système et le Génie Logiciel. A partir de cette analyse, il définit le concept de *Fonction* via la notion de fonctionnalité telle que définie par Warell :

"the combination of all its effects, actions, functions, and properties and their behaviour, that contribute to making the system useful for an intended purpose." (Warell 1999)

Dans le cadre de ces travaux de recherche, cette définition générique est considérée comme trop large. Le but ici est différent des travaux d'Eisenbart. Vis-à-vis des six questions de base, le concept de *Fonction* se doit d'être plus spécifique pour répondre à la question de ce que doit faire le système.

En s'appuyant en partie sur l'état de l'art d'Eisenbart, cette section résume l'analyse du concept de *Fonction* au travers de cinq domaines : la Programmation Architecturale, l'Architecture et la Construction pour l'Architecture-Ingénierie-Construction, ainsi que les Génies Logiciel et Mécanique de manière générale.

L'état de l'art proposé met en évidence un nombre important de définitions pour le concept de *Fonction*. Il est d'ailleurs reconnu que plusieurs définitions coexistent en Ingénierie (Vermaas 2011). Vermaas résume ces différentes interprétations en trois catégories associées aux notions de capacité, de comportement et de finalité (Vermaas 2013). Plutôt que de prendre une définition générique couvrant ces trois aspects, ces travaux nécessitent une définition plus précise. La notion correspondant à la nature de ces travaux est celle de capacité telle que définie par Vermaas. La notion de *finalité* est déjà couverte par le concept de **But** (Section 3.1) tandis que la notion de *comportement* fait plutôt référence à la question suivante du *Comment* adressée par le concept d'*Activité* (Section 3.3).

Une fonction est donc considérée comme une capacité objective et indépendante du contexte. Elle est utilisée pour expliciter, structurer mais également abstraire ce que le système doit être capable de faire ou permettre. Ceci fait référence au concept de *Fonction Externe* en Analyse de la Valeur. Nous la considérons comme un prérequis à la définition d'un bâtiment. Par contre, la réponse au *Comment* le système va le faire est subjective et dépendante du contexte et des moyens à disposition. Celle-ci fait référence au concept de *Fonction Interne* en Analyse de la Valeur. La définition suivante, illustrée par la Figure 21, est donc proposée pour le concept de *Fonction* :

Définition 2 : Une **Fonction** est une capacité d'action offerte par le système à ses clients, leur permettant de changer leur environnement d'un état A à un état B, de conserver ou d'éviter un état C, ou encore d'optimiser un état D.

L'état désiré correspond au concept de **But** tel que défini dans la section précédente. La relation entre un **But** et une **Fonction** est illustrée Figure 22. Elle se traduit de cette façon : ''*un **But** peut être atteint via une ou plusieurs **Fonctions**, tandis qu'une **Fonction** peut permettre d'atteindre un ou plusieurs **But***''. Les cinq types d'objectifs définis dans la section précédente se reflètent dans les différentes possibilités qu'offre une fonction dans sa définition (c.-à-d. changer, cesser, conserver, éviter, ou optimiser).

3.3. Activité – Comment et Quand

Dans le domaine de l'Architecture-Ingénierie-Construction, les concepts d'*Activité* et d'*Espace* sont étroitement liés. La Programmation Architecturale est principalement concentrée sur le concept d'*Activité* tandis que l'Architecture se focalise plus sur le concept d'*Espace*. L'*Activité* est la donnée d'entrée du processus de programmation architecturale, l'*Espace* sa donnée de sortie. Un espace ne peut se comprendre que par les activités qui s'y déroulent. Cependant, la relation entre ces deux concepts implique un certain nombre de confusions vis-à-vis de leur usage et formulation. Cette section a pour objectif de préciser le concept d'*Activité* dans le cadre de ces travaux de recherche à partir d'une revue de ses définitions dans les domaines de la Programmation Architecturale, de l'Ingénierie Système et du Génie Logiciel, ainsi qu'en Architecture d'Entreprise.

Comme précisé ci-dessus, le concept d'*Activité* est primordial lors de la programmation architecturale. Il n'est cependant pas suffisant de savoir ce que font localement les gens dans un bâtiment pour pouvoir le définir. En programmation architecturale, une activité est généralement présentée et décrite seule, sans réelle formalisation de ses dépendances avec d'autres activités (p.

ex. au sein d'un processus). La description précoce des espaces se fait généralement au détriment de celle des activités. Ces deux concepts sont bien différents, le concept d'*Activité* répondant aux questions du *Comment* et *Quand* est réalisé une fonction alors que l'*Espace* se réfère à *Où* cette activité à lieu. Les exigences spatiales caractérisant un espace sont spécifiées sur base de la description des activités qui y auront lieu. D'après l'état de l'art, la description d'une activité nécessite cependant de nombreux autres concepts pour être complète. L'objectif dans ces travaux n'est pas de rendre plus complexe la description d'une activité mais plutôt de donner suffisamment d'information complémentaire pour en permettre sa bonne compréhension. Il en résulte que seul trois concepts sont conservés ici vis-à-vis des domaines analysés : l'*Activité*, le *Processus* et l'*Opération*.

Le concept de *Processus* apporte la dimension temporelle et séquentielle manquant actuellement en programmation architecturale vis-à-vis de la description d'une activité. D'après la littérature, un processus est déclenché par un évènement. Ce concept d'*Evènement* n'est pas intégré dans le cadre de ces travaux de recherche en tant que concept à part entière mais plutôt en tant que caractéristique du concept de *Processus*. De cette façon, nous limitons la complexité du domaine de définition en réduisant le nombre de concepts représentés à un niveau conceptuel. Le concept de *Processus* est donc utilisé pour répondre à la question du *Quand*.

A partir de l'ensemble des définitions analysées à un niveau transdisciplinaire, les définitions suivantes sont proposées pour les concepts d'*Activité*, de *Processus* et d'*Opération*. Il est à noter que le concept d'*Activité* est considéré comme encapsulant vis-à-vis des concepts de *Processus* et d'*Opération*, chacun étant un type d'activité (Figure 25).

Définition 3 : Une **Activité** est un ensemble de tache requise pour réaliser une fonction.

Définition 3.1 : Une **Opération** est une tache élémentaire.

Définition 3.2 : Un **Processus** est un ensemble d'activités ordonnées.

Ainsi, si l'on reprend les concepts de **Fonction** et **But** pour positionner ces derniers concepts, la question répondue par le concept d'**Activité** peut se reformuler ainsi : comment le système va-t-il assurer les fonctions requises pour atteindre le but (ou les objectifs) défini ? Les fonctions peuvent être réalisées par une opération unique ou un ensemble d'activité (c.-à-d. un processus). Cependant, une activité ne permet pas nécessairement de réaliser une fonction. Dans le cas d'un processus, les activités qui le composent ne sont pas suffisantes pour réaliser individuellement la fonction. En termes de bonnes pratiques, la relation entre les fonctions et les activités devrait suivre l'axiome d'indépendance de l'approche Axiomatique Design (Suh 1990) (c.-à-d. ''*Maintenir l'indépendance des exigences fonctionnelles*''). Un modèle conceptuel reprenant ces remarques sur les liens entre les concepts de **But**, **Fonction**, **Activité**, **Processus** et **Opération** est proposé en Figure 26.

3.4. Ressource – Qui/Avec Quoi

Qui est la cinquième question de base utilisée pour structurer la description de l'objet d'étude. En partant des définitions précédentes, cette question se reformule de la façon suivante :

''Qui doit assurer les activités permettant de réaliser les fonctions requises pour atteindre le but (ou les objectifs) défini(s) ?''

Le terme *Qui* désigne en général une personne humaine. Dans le cadre de ces travaux, la réponse à cette question est plus large. Elle inclue toute entité pouvant ou permettant d'assurer une activité. Cette entité peut être une personne ou un artefact (p. ex. un automate) qui peut être actif (c.-à-d. réalise l'activité) ou passif (c.-à-d. en support à l'activité). Pour permettre de couvrir toutes ces notions, le concept de *Ressource* a été désigné pour répondre à la question du *Qui*.

Le concept de *Ressource* est défini dans cette section au travers d'une analyse de ses définitions dans les domaines suivants : l'Architecture-Ingénierie-Construction, l'Ingénierie Système et le Génie Logiciel, ainsi que l'Architecture d'Entreprise et la Gestion des Opérations en Génie Industriel.

Cette analyse transdisciplinaire met en évidence que le concept de *Ressource* n'est pas seulement utilisé pour désigner l'acteur d'une activité (c.-à-d. actif), mais aussi les artefacts nécessaires à la bonne réalisation d'une activité (c.-à-d. passif). Pour retranscrire ces deux aspects, une définition générique du concept de *Ressource* est proposée pour ces travaux de recherche :

Définition 4 : Une **Ressource** est une entité (interne ou externe au système) requise pour réaliser une opération.

Ainsi, le concept de **Ressource** est rattaché au concept d'**Activité** via le concept d'**Opération** (Figure 28). Une opération requière (au moins) une ou plusieurs ressources pour être réalisée tandis qu'une ressource peut être requise par plusieurs opérations. Une ressource n'est pas nécessairement un artefact unique, elle peut être composée d'autres ressources. Deux types de ressources sont considérés pour ces travaux de recherche : les ressources humaines et les ressources matérielles.

Afin de bien distinguer les choses et ne pas mélanger ou démultiplier le nombre de types de ressources, la notion de ressource spatiale n'est pas considérée dans ces travaux. Le concept d'Espace y est préféré en réponse à la question de base *Où* traitée dans la section suivante.

3.5. Espace – Où

Le concept d'*Espace* est propre au domaine de l'Architecture-Ingénierie-Construction. Les autres domaines de l'ingénierie analysés pour les concepts précédents n'en proposent pas de définitions utiles ou pertinentes pour ces travaux. Le focus de cette section est donc sur l'analyse des définitions du concept d'*Espace* au niveau de la Programmation Architecturale, de l'Architecture, et de la Construction.

Les Architectes donnent plus de sens au concept d'*Espace* que les Ingénieurs. D'après Lefebvre, un espace est un ''*produit social*'' (Lefebvre 1974). Il met en place un contexte qui structure la vie, les activités et les relations de ses utilisateurs et usagers (Lawson 2001). Cette dimension sociale représente la logique derrière l'espace au travers de la relation homme-espace (Dursun 2012b). L'**Activité** est peut-être le concept central de la programmation architecturale, mais l'*Espace* est clairement le concept clé de la conception architecturale (Dursun 2009).

Dans le cadre de ces travaux de recherche, la dimension sociale de l'espace est mise de côté pour un autre projet de recherche qui se concentrerait sur la phase de conception. Ainsi, seules les propriétés physiques de l'espace sont incluses dans sa description tandis que sa logique (fonctionnelle) est reportée dans le concept d'**Activité**. En effet, les activités contiennent l'essentiel des informations nécessaire à la description des espaces.

Définition 5 : Une **Espace** est le lieu où les activités se déroulent.

En programmation architecturale, trois types d'espaces peuvent se distinguer et sont repris dans ces travaux : l'*Espace Fonctionnel* issue de la synthèse des besoins de la MOA, l'*Espace Demandé* représentant une quantification du nombre d'espaces fonctionnels requis, et l'*Espace Proposé* par l'architecte en réponse au programme (Figure 30).

3.6. Domaine de Définition

Dans cette section, nous avons analysé les définitions de cinq concepts choisis pour décrire notre objet d'étude. Cette analyse transdisciplinaire nous a permis de définir ces concepts selon notre objet d'étude mais aussi le contexte de ces travaux de recherche : la programmation architecturale. La taxonomie résultante forme le domaine de définition requis pour arriver à définir et à décrire des exigences bâtiment selon différents points de vue sur l'objet d'étude :

Le couple **Pourquoi-But** établie la raison d'être (c.-à-d. les justifications et limites) de l'objet d'étude, la situation désirée par la MOA.

Le couple **Quoi-Fonction** abstrait les capacités attendues de l'objet d'étude permettant d'atteindre, d'interrompre, de maintenir, d'éviter ou d'optimiser les buts (objectifs) définis, indépendamment des solutions techniques pouvant y répondre.

Le couple **Comment/Quand-Activité** décrit la manière (c.-à-d. la solution) dont l'objet d'étude assure ces fonctions.

Le couple **Qui-Ressource** décrit les moyens nécessaires pour réaliser les activités et précisent comment la solution est implémentée.

Enfin, le couple **Où-Espace** décrit le bâtiment en lui-même, le lieu où sont réalisées les activités.

4. Synthèse

Dans ce chapitre, nous proposons un ensemble de concepts pour définir et décrire le domaine de définition des exigences bâtiment : le **But**, la **Fonction**, l'**Activité**, la **Ressource**, et l'**Espace**. Chaque concept répond à une des six questions de bases structurant ce domaine de définition : *Pourquoi*, *Quoi*, *Comment*, *Quand*, *Qui* et *Où*. Chaque couple question-concept représente un point de vue différent sur le même objet d'étude. Ces points de vue sont organisés par niveaux d'abstraction vis-à-vis de l'objet d'étude, du plus haut niveau orienté métier (*Pourquoi-But*) au plus bas niveau orienté bâtiment (*Où-Espace*) (Figure 32).

Les concepts proposés et définis sont structurés au sein d'une taxonomie. Leurs relations sont présentées dans un modèle conceptuel pour faciliter l'identification de leurs dépendances entre concepts et l'informatisation de la taxonomie. De cette façon, les processus de décision et la gestion des changements d'information pourraient être facilités. Cette structure de concepts représente le domaine de définition des exigences bâtiment, premier résultat issu de ces travaux (Tableau 10).

Chapitre III – MODELISER UN BATIMENT

A partir du domaine de définition, le Chapitre III propose un ensemble de langages de modélisation pour formaliser la transition des besoins vers les exigences selon chacun des points de vue définis dans le chapitre précédent. Outre la Section 1 qui introduit ce chapitre, le Chapitre III est organisé en deux parties (Figure 33). La première partie présente les modèles conceptuels permettant de définir le périmètre de l'objet d'étude (Section 2) et de structurer le processus de définition des besoins (Section 3). La seconde partie présente une série de modèles opérationnels pour outiller la formulation des besoins en exigences via les Sections 4 à 6.

1. Introduction

Comment modéliser un bâtiment et son domaine de définition tout en intégrant les différents points de vue des parties prenantes impliquées ?

Dans le Chapitre II, un ensemble de concepts est proposé pour décrire le domaine de définition associé à la programmation architecturale et à l'objet bâtiment mais sans en définir le périmètre. Dans ce troisième chapitre, un ensemble de modèles est proposé pour établir les limites (internes et externes) de l'objet d'étude (Section 2), pour en articuler sa description (de l'extérieur vers l'intérieur) (Section 3 et 5), et pour fournir des points de vue (semi-)formels sur son domaine de définition (Section 4 et 6).

Dans la Section 2, le périmètre et les limites de l'objet d'étude sont modélisés au travers de la systémique et du paradigme de Système Produit-Service. Ce paradigme permet de comprendre la nature bicéphale des bâtiments (c.-à-d. différencier les parties statique et dynamique des bâtiments).

La Section 3 propose une modélisation des transitions entre les concepts définis dans le Chapitre II en s'appuyant sur le paradigme FBS de Gero pour distinguer la réalisation d'une fonction par la partie Produit, la partie Service, ou une combinaison des deux. Ce paradigme apporte de nouveaux concepts permettant de compléter le domaine de définition proposé précédemment.

La Section 4 présente une analyse de l'état de l'art des langages de modélisation dans les différents domaines d'ingénierie déjà explorés dans le Chapitre II. Ces langages de modélisation peuvent être utilisés pour représenter le domaine de définition selon des points de vue complémentaires. Les représentations graphiques issues de ces langages de modélisation sont une façon de faciliter la compréhension du domaine de définition par les diverses parties prenantes de la programmation architecturale.

S'appuyant sur des observations et la conclusion de la Section 4, la Section 5 introduit un nouveau concept pour permettre de structurer la transition entre les concepts d'Activité et d'Espace : le Méta-Espace. Ce Méta-Espace permet la formalisation et la structuration de la transition de l'Activité vers l'Espace mais aussi de l'Espace vers l'Activité, ceci assurant une certaine traçabilité des décisions prises pendant la programmation architecturale.

La Section 6 présente le langage de modélisation associé à ce nouveau concept : le Diagramme de Méta-Espace. Ce diagramme assure la transition entre les livrables principaux de la programmation architecturale manipulant les concepts d'Activité et d'Espace (p. ex. le programme et les plans d'architecte). Un de ses usages permet notamment de vérifier la cohérence des informations contenues dans les différents documents selon plusieurs points de vue.

2. Modéliser l'Objet d'Etude : Le Bâtiment en tant que Système

L'objet bâtiment est déjà considéré comme un système complexe multicouches (Bertelsen 2003). Son premier niveau de complexité s'observe sur les étapes de conception, industrialisation et construction notamment à cause des services techniques (p. ex. ventilation, air conditionné, chauffage) qui le composent (Kubicki et al. 2006). Ces travaux de recherche se focalisent sur un second niveau de complexité concernant la programmation architecturale. Cette complexité est

liée à la nature bicéphale de l'objet bâtiment. En effet, un bâtiment n'est pas un objet autonome mais un système complexe séparable en deux parties : une partie statique (décrite par les concepts d'Espace et de Ressource) et une partie dynamique (décrite par le concept d'Activité).

Dans cette section, cette complexité est abordée de manière progressive via la systémique (Section 2.1), permettant de définir le périmètre de l'objet bâtiment, le paradigme Système Produit-Service (Section 2.2), permettant de décomposer l'objet bâtiment en deux composantes principales, et le principe d'alignement (Section 2.3), permettant d'assurer la cohérence entre les composantes de l'objet bâtiment.

2.1. Point de Vue Systémique

Le concept de *Système* est utilisé pour modéliser des objets complexes de façon abstraite afin de les rendre plus faciles à aborder. Ce concept apporte un point de départ neutre en termes de solution pour la définition et la résolution d'un problème de conception. Pour définir un système, trois concepts sont à prendre en compte : les besoins-exigences, les frontières et l'architecture (Faisandier 2011).

Les besoins-exigences font références à l'explicitation de ce qui est attendu en termes de services et de contraintes. Il s'agit du principal résultat de la définition des besoins. En programmation architecturale, ce résultat est compilé dans le programme, livrable sur lequel se base les architectes pour concevoir la partie bâtiment de ce système complexe. Cependant, ce bâtiment n'est pas suffisant pour adresser les besoins réels de la MOA. Pour comprendre les besoins réels de la MOA, le programmiste à besoin de comprendre le contexte et comment la MOA veut atteindre son but (et ses objectifs). Les besoins-exigences sont donc l'ensemble des informations concernant le domaine de définition des exigences bâtiment tel que défini dans le Chapitre II.

Les frontières explicitent les limites de l'objet étudié ainsi que ses interfaces avec le monde extérieur. Ces frontières sont explicitées dans ces travaux en s'appuyant sur trois concepts supplémentaires venant compléter le domaine de définition proposé : l'*Environnement*, les *Eléments Externes*, et le *Système*. Ces concepts sont définis dans ces travaux en s'appuyant sur une analyse de quelques références issues de l'état de l'art (Jackson & Zave 1993; AFNOR 1996; INCOSE 2010; IEEE 2010).

Définition 6 : L'**Environnement** d'un système est le contexte ''AS-IS'' dans lequel le Système s'inscrira et sur lequel le Système aura un impact ou sera impacté de manière directe ou indirecte de par son existence. Ce contexte inclue tout élément externe ou facteur environnemental (incluant les circonstances, conditions et connaissances) qui existe indépendamment de l'existence du système.

Définition 7 : Un **Elément Externe** est un artefact (physique ou non) dont l'existence n'est pas subordonnée à celle du système mais dont le comportement ou la condition impactera ou sera impacté par la présence du système.

Définition 8 : Un **Système** est l'objet d'étude vu tout d'abord comme une boîte noir représentant une collection de composants (c.-à-d. sous-systèmes) structurée et organisée. Ces composants tangibles ou intangibles contribuent à la réalisation des fonctions du systèmes par rapport à son environnement. Pour être considéré comme dépendant, chacun de ses composants doit être conçu, embauché, acheté, ou créé dans le cadre du processus de développement du système.

L'Architecture explicite le fonctionnement interne de l'objet d'étude et son organisation structurelle en termes de composants et de relations entre ses composants. Les composants de l'objet d'étude (c.-à-d. le système) sont appelés *Eléments Internes* en opposition au concept d'*Elément Externe* ou de *Resource* tel que défini dans la taxonomie du Management de la Valeur (AFNOR 1996). Le concept d'*Elément Interne* se distingue de celui de *Ressource* en ce qu'une ressource peut être interne ou externe au système. Ces concepts font référence à deux points de vue différents mais complémentaires. La Figure 36 illustre cette différence sur l'exemple du Lycée Technique au Luxembourg.

Définition 9 : Un **Elément Interne** est un artefact (physique ou non) dont l'existence est entièrement subordonnée à celle du système, et contribuant à son bon fonctionnement.

Dans cette section, le modèle systémique introduit de nouveaux concepts permettant de définir le périmètre de l'objet d'étude en distinguant les éléments appartenant au système étudié (**Eléments Internes**) et les éléments appartenant à son environnement (**Eléments Externes**). Les activités de programmation architecturale et de conception diffèrent selon le type d'élément. Les **Eléments Externes** sont décrits à partir de l'existant (informations factuelles) tandis que les **Eléments Internes** sont définis par le programmiste puis conçus par les concepteurs, achetés par la MOA, engagé par les utilisateurs, ou construits par les entrepreneurs. Leur matérialisation n'est effective qu'au moment où le système prend forme. Les **Eléments Internes** sont sujets à la volonté de la MOA tandis que les **Eléments Externes** en sont indépendants mais évoluent avec le temps. Les relations entre ces nouveaux concepts sont modélisées Figure 37.

Dans le cadre de ces travaux de recherche, la nature des éléments internes du système n'est pas limitée au simple objet bâtiment mais intègre également des entités organisationnelles (c.-à-d. les services proposées par son intermédiaire). Deux parties sont distinguées dans le système : la partie physique (c.-à-d. le bâtiment) et la partie organisationnelle (c.-à-d. les services proposés). L'objet d'étude est considéré comme un ''*système opérationnel*'' dont la programmation architecturale vise à définir la partie physique. La dépendance entre ces deux parties et leurs interactions sont développées dans la section suivante.

2.2. Paradigme Système Produit-Service

Dans la littérature, le Système Produit-Service est principalement associée à une notion de modèle économique (modèle d'affaire). Au lieu de vendre un produit aux consommateurs, un usage ou une fonctionnalité lui est proposé (Meier et al. 2010). L'idée est donc de développer des offres mélangeant des produits et des services pour (conjointement) satisfaire les besoins du client (Goedkoop et al. 1999). Le Tableau 11 reprend les principales différences entre un produit et un service d'après la littérature (Bachmann 1998; Moritz 2005). Les Systèmes Produit-Service sont classés en trois catégories : orienté produit, orienté usage, et orienté résultat (Manzini & Vezzoli 2003; Yang et al. 2009; Tukker & Tischner 2006). D'autres classifications plus détaillées sont présentées dans la littérature (Meijkamp 1998; Brezet et al. 2001; Zaring et al. 2001; Mont 2002; Behrendt et al. 2003) mais ne reflètent pas notre vision du système bâtiment.

Dans le cadre de ces travaux, le paradigme Système Produit-Service est utilisé comme une façon de percevoir notre objet d'étude, c.-à-d. le système bâtiment, de façon plus large qu'un simple objet physique limité au bâtiment. Ainsi, les entités physiques telles que le bâtiment, le mobilier, les équipements et les personnes (c.-à-d. les ressources qui composent ce système bâtiment) en représentent la partie *Produit* (Tableau 12).

Définition 10 : Un **Produit** est une ressource (tangible ou intangible) ou un espace nécessaire pour réaliser une activité contribuant à la délivrance d'un service.

Le concept de *Service* a deux sens dans le domaine de l'Architecture-Ingénierie-Construction qui diffèrent de celui du Système Produit-Service. Le premier concerne les services techniques du bâtiment (p. ex. ventilation, chauffage) qui donnent vie au bâtiment. Le second fait référence aux services (publics) fournit par les utilisateurs du bâtiment à ses usagers (p. ex. l'éducation fournie par les enseignants aux élèves). Ces deux sens se retrouvent dans une typologie associée au concept de *Service* dans le cas des Systèmes Produit-Service orientés résultat (Yang et al. 2010). Cette typologie distingue les *Services Principaux* des *Services Supports* (Figure 40).

Définition 11 : Un **Service** est une activité (humaine ou automatisée) réalisée par et utilisant un ensemble de ressources (tangibles ou intangibles) à destination d'autres humains ou systèmes et leur apportant une certaine valeur ajoutée.

Définition 11.1 : Un **Service Principal** est un service à destination des usagers dont ils tirent une valeur ajoutée.

Définition 11.2 : Un **Service Support** est un service à destination du produit ou un service secondaire nécessaire à la bonne réalisation d'un service principal. Son existence est soumise à celle d'un service principal ou du produit.

Dans le cas des constructions publiques, un bâtiment n'est qu'un des moyens nécessaires à la fourniture d'un service à la communauté et non pas une fin en soi. Ce service représente le point de départ menant à la définition de ce bâtiment. Au vu des classifications de Systèmes Produit-Service existantes (dont le point de départ est le produit), un nouveau type est proposé correspondant au cas des bâtiments publics afin de refléter notre perception du système bâtiment : le Système Produit-Service orienté Service (Mauger et al. 2013).

A partir de la définition du service à fournir, les autres composants du système sont définis. La partie **Service** est décrite au travers du concept d'**Activité** tandis que la partie **Produit** est décrite en utilisant les concepts de **Ressource** et d'**Espace** (Figure 41). Le concept d'**Espace** fait référence au bâtiment et est le dernier concept à traiter dans la définition du Système Produit-Service orienté Service. Le raisonnement proposé implique donc une plus grande réflexion par la MOA sur les services qu'il désire fournir et sur son système bâtiment dans son intégralité et non plus uniquement sur l'objet bâtiment.

2.3. Principe d'Alignement

Le principe d'alignement tel qu'appliqué dans ces travaux de recherche est tiré de l'Architecture d'Entreprise (Vernadat 1996b; Op't Land et al. 2009; Greefhorst & Proper 2011) où la notion d'alignement Business-IT (Zachman 1987) est reprise ici pour la définition du système bâtiment. Ce principe est transposé à l'alignement entre la partie **Produit** et la partie **Service** du système bâtiment et propagée à l'ensemble de ses composants (**Eléments Internes**).

2.4. En Résumé

Dans cette section, l'objet d'étude est modélisé en tant que système complexe afin de pouvoir appréhender sa nature bicéphale. Ce système complexe appelé Système Produit-Service est décomposé en deux parties, une partie **Service** (dynamique) et une partie **Produit** (statique). Le concept d'**Activité** est rattaché à la partie **Service** tandis que les concepts de **Ressource** et d'**Espace** sont associés à la partie **Produit** (Figure 42). La définition de la partie bâtiment est subordonnée à celle de la partie **Service**. Cette subordination est modélisée via le Système Produit-Service orienté Service. La gestion des interactions entre la partie **Produit** et la partie **Service** est quant à elle associée au principe d'alignement.

Le paradigme de Système Produit-Service met en évidence une nouvelle problématique dans le processus de définition du système bâtiment : l'allocation des fonctions (Muller et al. 2007) à la partie **Produit**, à la partie **Service** ou à une combinaison des deux. La section suivante propose de

s'appuyer sur un autre paradigme, celui du FBS, pour modéliser cette allocation des fonctions et structurer la transition entre le concept de **But** et celui d'**Espace**.

3. Un Modèle pour la Programmation Architecturale

L'approche FBS développé par Gero modélise le processus de conception à partir de relations entre ces trois concepts : *Fonction*, *Comportement* et *Structure* (Gero 1990). Cette approche est utilisée dans ces travaux pour modéliser et structurer la transition entre les besoins métiers (c.-à-d. le but et les objectifs) et les exigences bâtiment (c.-à-d. les espaces) ainsi que l'allocation des fonctions aux différentes parties du système bâtiment en tant que Système Produit-Service orienté Service.

L'approche FBS de Gero a été choisie sur base d'une analyse critique de notre domaine de définition ainsi qu'une revue (non-exhaustive) de la littérature sur les méthodes et modèles de conception et les Systèmes Produit-Service. L'approche de Gero introduit un concept-clé permettant d'adresser la notion d'allocation des fonctions et de définition des exigences bâtiments : le concept de *Comportement*. Ce concept inclue la notion de *propriété* d'un produit (Weber 2005; Eder 2008) (c.-à-d. comportement du produit), notion absente du domaine de définition tel que défini dans les sections précédentes (Figure 42). Une *propriété* de produit est considérée comme un comportement passif pouvant réaliser une fonction. C'est une alternative au concept d'**Activité** qui peut être considéré comme un comportement actif du système bâtiment. Ainsi, nous proposons de prendre le concept de *Comportement* pour modéliser et structurer l'allocation des fonctions au **Produit** via les *propriétés* du produit ou au **Service** via les activités (Figure 45).

Définition 12 : Un **Comportement** est une activité (action ou réaction) ou une propriété d'un système (ou sous-système) sous certaines conditions ou suite à un évènement déclencheur, associée ou dérivée d'un de ses composants expliquant comment et quand le système assure une fonction donnée.

De plus, le concept de *Structure* apporte un niveau de conceptualisation supplémentaire à notre domaine de définition en couvrant les concepts de **Ressource** et d'**Espace** (Figure 46). Cette *Structure* est décrite à partir des attributs caractéristiques des composants, c.-à-d. les attributs pouvant être influencés ou déterminés par le concepteur (p. ex. matériaux, forme, dimension, etc.) (Weber 2005). Ils correspondent aux ''*propriétés internes*'' (Eder 2008) et aux ''*paramètres de conception*'' (Suh 1998) dans d'autres théories de conception.

Définition 13 : Le concept de **Structure** se réfère aux composants (physiques et virtuels) du système, leur description (statique) (c.-à-d. leurs caractéristiques), et leurs relations décrivant sa composition et son organisation interne. Il répond de manière plus globale à la question du qui (c.-à-d. quelle ressource) doit faire quelque chose dans le système pour assurer une fonction quand elle est nécessaire, et du lieu où cette action doit être réalisée.

Les concepts proposés par Gero répondent également à une partie des six questions de base définies dans le Chapitre II. Le concept de **Fonction** adresse le **Quoi**, le concept de **Comportement** couvre le **Comment** et le **Quand**, et le concept de **Structure** répond aux questions du **Qui** et du **Où**. Seule la question du **Pourquoi**, que nous avons associée au concept de **But**, n'est pas adressée par le paradigme FBS. Afin de couvrir l'ensemble du domaine de définition défini, l'approche FBS de Gero est complétée par le concept de **But** (G pour *Goal*) pour obtenir le modèle.

Ainsi, le modèle conceptuel du domaine de définition est mis à jour en intégrant les concepts de **Comportement** et de **Structure**. Un But (ou Objectif) est toujours atteint via des fonctions. Les fonctions sont désormais réalisées au travers de comportements soient du service via les activités, soient du produit via les propriétés. Ces comportements sont ensuite associés à des éléments de structure pouvant être des ressources matérielles ou humaines, ou des espaces. Ces éléments de structure sont décrits au travers de leurs caractéristiques (Figure 47).

A partir de ce nouveau domaine de définition, un modèle pour le processus de programmation architecturale est proposé en croisant les concepts GFBS et PSS. Les concepts GFBS sont utilisés pour structurer le processus tandis que les concepts du PSS sont utilisés pour structurer l'objet d'étude. Les concepts de But et de Fonction sont associés au système global (c.-à-d. au système Produit-Service dans son intégralité). Le concept de Comportement est d'abord associé au système global avant d'être distribué entre la partie Produit et la partie Service du système bâtiment. Afin de limiter la complexité du modèle, le concept de Structure est maintenu à un niveau macro (c.-à-d. sans détail). Ce modèle GFBS est présenté sous forme d'une théorie structurée en six étapes (Figure 48). Cette théorie sera ensuite appliquée sur le processus de programmation architecturale en y intégrant le principe de ''*zigzag*'' (c.-à-d. aller-retour) issu de la théorie Axiomatic Design (Suh 1998) (cf. Chapitre IV).

Les concepts issus du GFBS et du PSS apportent un second niveau de formalisation conceptuelle sur le processus de programmation architecturale (Figure 50). Ce niveau structure et formalise la transition entre les besoins métiers et les exigences bâtiments à un niveau conceptuel. La section suivante introduit un ensemble de modèles opérationnels pour outiller cette transition.

4. Modèles Opérationnels pour une Formalisation Pratique Multi-Vue

Lors de la programmation architecturale, une quantité énorme d'informations est générée, des besoins métiers aux exigences bâtiment (Shen et al. 2004). Actuellement, ces informations sont transmises aux architectes et équipes de conception au travers du programme, un document textuel complété par quelques diagrammes dont la représentation graphique varie d'un programmiste à l'autre. La compréhension et la gestion de cette énorme quantité d'informations reste un challenge en termes de consistance, complétude et redondance.

Dans la première partie de ce Chapitre III, nous avons proposé un ensemble de modèles conceptuels pour structurer le domaine de définition au travers des paradigmes FBS et PSS. Cette seconde partie se concentre sur des modèles opérationnels (semi-formels) pour outiller de façon pratique le processus de programmation architecturale. Plutôt que de créer un nouveau modèle intégrant tous les concepts du domaine de définition, nous proposons d'utiliser les modèles existants dans les différents domaines d'ingénierie pour fournir des vues locales sur le même objet d'étude. Chaque modèle existant couvre un ensemble de concepts différents et propose un point de vue différent sur l'objet d'étude.

Cette section n'a pas pour vocation d'établir un état de l'art multidisciplinaire des modèles de représentation graphique ou de faire une analyse critique des langages de modélisation mais plutôt de démontrer l'intérêt de ces modèles et langages vis-à-vis de la représentation graphique des informations sur le système bâtiment. N'importe quel modèle de chaque domaine d'ingénierie pourrait être appliqué lors de la programmation architecturale pour autant qu'il intègre une partie des informations de son domaine de définition. D'ailleurs, il n'est pas strictement nécessaire que toutes les informations représentées par ces modèles soient présentes dans le domaine de définition. Une application partielle des formalismes de chaque modèle peut suffire à aider le programmiste ou la MOA.

Le concept de But (Objectif) est principalement modélisé dans les approches GORE (*Goal-oriented Requirements Engineering*) en Génie Logiciel (Giunchiglia et al. 2001; Kavakli & Loucopoulos 2003; Anwer & Ikram 2006; Lapouchnian 2005) (p. ex. Tropos, i* ou encore KAOS). Le degré de couverture du domaine de définition du bâtiment et de la programmation architecturale de ces approches est illustré sur la Figure 53.

Le concept de Fonction est modélisé via un certain nombre de modèles dit fonctionnels. Chaque approche de conception systémique a sa propre définition du concept de Fonction mais aussi ses propres langages de modélisation plus ou moins opérationnels (Ross & Schoman 1977; Gero 1990; Pahl & Beitz 1995; Umeda & Tomiyama 1997; Suh 1998). Diverses recherches proposent déjà un état de l'art de ces modèles fonctionnels (Erden et al. 2008; Eisenbart et al. 2012; Eisenbart et al. 2013; Eisenbart 2014). Nous nous limitons ici à en introduire quelques-uns ayant un intérêt particulier pour ces travaux et de les positionner sur notre domaine de définition (Figure 58) : le diagramme d'interaction (de la Bretesche 2000; Société APTE 2007), l'arborescence Fonction-Moyens (Tjalve 1978; Buur 1990; Bracewell & Sharpe 1996), le diagramme FAST (AFNOR 2000), ou encore le Bloc Diagramme Fonctionnel (Maussang 2008).

Le concept de Comportement possède également un grand nombre de langages de modélisation. Dans ces travaux, le focus est mis sur trois langages principaux : l'IDEF (National Institute of Standards and Technology 1993; Mayer, Crump, et al. 1995), le BPMN (White 2004; OMG 2011a) et l'UML (Booch et al. 1998; OMG 2011b). Chacun de ces langages intègre différents modèles plus locaux pour permettre de couvrir la modélisation de leur système respectif dans son ensemble. Le cas du BPMN est quelque peu particulier puisqu'il est déjà adopté dans le domaine de l'Architecture-Ingénierie-Construction pour représenter les processus métiers et les échanges d'informations au cours des projets de construction via l'IDM pour les usagers du BIM (Wix 2007; Weise et al. 2009; Berard & Karlshøj 2011). Les modèles tirés de ces langages de modélisation sont positionnées sur le domaine de définition en Figure 63.

Le concept de Structure est analysé en deux fois afin de distinguer les langages de modélisation permettant de représenter le concept de Ressource et celui d'Espace. Nous nous limitons à trois types de langages de modélisation pour le concept de Ressource : les diagrammes structurels UML, plus particulièrement le Diagramme de Classes (Figure 64) et le Diagramme d'Objets (Figure 65), le Diagramme Entité-Relation (Chen 1976) (Figure 66) et l'Organigramme Technique de Produit (Figure 67). Ces langages peuvent être utilisés pour représenter tout type d'objet, physique aussi bien que virtuel. Leur positionnement sur le domaine de définition est représenté au travers de la Figure 68.

Le concept d'Espace est modélisé de différentes façons en fonction du processus de développement du projet de construction. Lors de la programmation architecturale, le plus commun est le Diagramme Fonctionnel (Figure 69) (Certu 2010; Mauger & Kubicki 2013). Pour le processus de conception, on y retrouve un plus large panel : la Matrice de Proximités (Figure 71) (Akinc 2005), le Graphe (Figure 72) (Euler 1736) ou Diagramme Réseau, le Diagramme Bulle (Figure 72) (Lawson 2005), le Diagramme de Venn (Figure 72) (Chilakamarri et al. 1996), le Plan Schématique (Figure 73) (Ruch 1978), ou encore le Diagramme de Zone ou d'Affinité (Figure 74) (White 1986; Alread & Leslie 2006). Le degré de couverture de ces langages de modélisation vis-à-vis du domaine de définition est présenté Figure 76.

Sur base d'une analyse multidisciplinaire de ces langages de modélisation, nous avons vu que le domaine de définition pouvait être représenté graphiquement selon différents points de vue, chaque point de vue couvrant un ensemble différent de concepts (Figure 77). Pour permettre d'utiliser n'importe quel langage de modélisation dans le cadre de la programmation architecturale, nous proposons de lier ces différents modèles par l'intermédiaire des concepts qui composent le domaine de définition. Ainsi, ce n'est plus le sens originel des concepts vis-à-vis de leur domaine d'application courant qui est considéré mais la définition proposée dans le cadre de ces travaux. Plutôt que d'essayer de créer un nouveau langage de modélisation intégrant tous ces points de vue, nous avons privilégié le détournement des langages existants par l'intermédiaire d'une redéfinition des concepts modélisés. En guise de synthèse, la couverture en termes de concepts de chaque type de modèle (classé par rapport à leur concept principal) est illustrée sur le modèle conceptuel du domaine de définition en Figure 78.

Cependant, il s'avère que l'ensemble des langages de modélisation présentés, même s'il n'est pas exhaustif, ne couvre pas une transition essentielle que nous cherchons à modéliser/formaliser dans ces travaux : la transition entre le concept Comportements et le concept de Structure, plus

particulièrement la transition amenant au concept d'Espace. Cette transition est assurée par les programmiste notamment sur base de leur expérience et n'est actuellement pas formalisée (Figure 79). Dans les sections suivantes, nous proposons donc un nouveau langage de modélisation permettant de formaliser cette transition et de couvrir ce manque dans la continuité de modélisation du domaine de définition défini pour la programmation architecturale (Figure 80).

5. Du Programme au Projet : Le Concept de Méta-Espace

Dans la section précédente, différents langages de modélisation des exigences bâtiment sont analysés. Chaque langage apporte des modèles donnant un point de vue limité à un ensemble d'information sur le domaine de définition du système bâtiment. Dans les pratiques, ces informations sont retranscrites dans le programme sous forme de textes descriptifs mais aussi sous forme graphique notamment via un diagramme fonctionnel. Durant la phase de conception, les architectes manipulent ces mêmes informations au travers de divers modèles avant de produire leurs propres plans. Chaque fois que l'on passe d'un modèle à un autre, une partie de l'information d'origine est perdue (Figure 81). Ceci est dû à la quantité énorme d'informations à manipuler associée aux limites des hommes ainsi qu'au manque d'interopérabilité entre les différents langages de modélisation. Dans cette section, nous proposons un nouvel artefact, le méta-espace, afin de faciliter la manipulation de cette quantité d'information au travers de ces différents modèles (traçabilité, consistance, objectivité, complétude, intégrité des informations).

La fonction principale d'un bâtiment est de ''*protéger les personnes en train de faire certaines choses d'une certaine façon*'' (Koutamanis 2013). Le concept d'activité est un moyen abstrait approprié pour expliquer à un architecte ce qu'il est attendu de son projet architectural en termes de résultats. Chaque activité nécessite un espace pour être réalisée. Afin de pouvoir évaluer le nombre d'espaces nécessaires et suffisants pour la réalisation de toutes les activités, un artefact intermédiaire est introduit : le Méta-Espace. Le Méta-Espace représente une abstraction commune aux concepts d'Activité et d'Espace facilitant l'analyse et la manipulation des informations liées à ces concepts.

Définition 14 : Un **Méta-Espace** est un espace virtuel idéal caractérisé par toutes les exigences nécessaires à la bonne réalisation d'activités élémentaires (c.-à-d. opérations) ainsi que les buts et fonctions associées.

Ainsi, un méta-espace est associé à chaque opération. Les méta-espaces sont ensuite regroupés sur base des exigences les caractérisant afin d'obtenir un nombre réduit d'espaces. Le regroupement des méta-espaces se base sur un ensemble de règles logiques et métiers mettant en évidence les exigences compatibles. La Figure 82 représente de manière simplifiée ce processus de traitement des exigences faisant le lien entre des activités et un espace physique.

Le Méta-Espace est un artefact intermédiaire. Pour clarifier ses différents usages tout au long du processus de programmation architecturale, nous proposons d'en définir une typologie. Trois types de méta-espaces sont considérés. Chaque type fait référence à un livrable en relation avec la programmation architecturale : le Méta-Espace Opérationnel (lié au programme) (Figure 83), le Méta-Espace Fonctionnel (lié au diagramme fonctionnel) (Figure 83) et le Méta-Espace de Conception (ou Physique) (lié aux plans de l'architecte) (Figure 84).

Définition 14.1 : Un **Méta-Espace Opérationnel** est un espace virtuel idéal associé à une opération (c.-à-d. tache élémentaire) permettant de représenter la transition entre les concepts d'Activité (via l'Opération) et d'Espace.

Définition 14.2 : Un **Méta-Espace Fonctionnel** est un espace virtuel idéal associé à un espace fonctionnel et composé de méta-espace opérationnels élémentaires. Il permet de représenter la transition entre la partie textuelle du programme et le diagramme fonctionnel.

Définition 14.3 : Un **Méta-Espace de Conception** est une abstraction d'un espace conçu (dessiné) par l'architecte et représenté sur un plan ou schéma 2D.

Chacun de ces méta-espaces modélise un ensemble d'informations différent à différents moments du processus de programmation architecturale. Cette distinction a pour objectif d'aider à la définition des espaces fonctionnels, à l'évaluation des propositions d'architectes et à faciliter la comparaison des différents livrables liés à la programmation architecturale. Chaque type de méta-espace peut permettre de représenter le même artefact mais avec une quantité d'information différente. Cet artefact est, par exemple, un bureau où un certain nombre d'activités sont réalisées par des personnes avec des ressources particulières. Les points de vue conception et programmation sur ce bureau ne sont pas les mêmes et diffèrent notamment par la qualité des informations utiles à chaque phase. Or, au final, il s'agit du même artefact devant satisfaire la MOA.

La correspondance entre ces différents points de vue est actuellement assurée par le programmiste. Dans la Section 6, nous proposons de nous appuyer sur ces méta-espaces (Figure 85 et 86) pour outiller le programmiste dans sa démarche.

6. Modèle Transitoire : Le Diagramme de Méta-Espace

Cette section est organisée autour des trois types de méta-espaces définis dans la Section 5. Chaque type de méta-espaces représente un ensemble d'informations différent à différents moment du processus de programmation. Il en ressort un langage de modélisation a visée pratique appelé Diagramme de Méta-Espace. En s'appuyant sur l'abstraction apportée par le Méta-Espace, ce langage de modélisation permet d'outiller la transition d'un livrable à un autre. Ceci nous permet de formaliser ces transitions mais également d'assurer une certaine traçabilité des informations et de leurs dépendances d'un livrable à l'autre (continuité de l'information).

Un diagramme de Méta-Espace intègre les propriétés du graphe et du diagramme bulle mais utilisé dans un contexte différent et pour des objectifs différents (ceux de la programmation architecturale). L'agencement des méta-espaces composant le diagramme n'a pas d'importance. Le diagramme de méta-espace se contente de représenter de manière graphique les dépendances entre méta-espace, sachant que les méta-espaces contiennent les informations relatives aux activités qui y sont associées (Figure 87). Trois types de diagramme de méta-espaces sont proposées en adéquation avec les types de méta-espaces définis dans la Section 5 : le Diagramme de Méta-Espace Opérationnel, le Diagramme de Méta-Espace Fonctionnel et le Diagramme de Méta-Espace de Conception (ou Physique).

Le Diagramme de Méta-Espace Opérationnel est un outil d'abstraction des modèles d'ingénierie permettant de représenter les informations du programme (p. ex. IDEF0, Diagramme d'Activité, ou Diagramme d'Etat-Transition). Le diagramme de méta-espace opérationnel se débarrasse des formalismes propres à chaque modèle fonctionnel pour les résumer à un simple graphe (Figure 90). Les informations contenues dans ces modèles nécessites d'être conservées en arrière-plan pour une utilisation ultérieure (p. ex. lors du traitement des exigences permettant de réduire le nombre de méta-espaces avant obtention du diagramme fonctionnel). Une solution informatisée (p. ex. base de données) est à privilégier pour la conservation de ces informations. La restitution de ces informations peut se faire via différentes façons graphiques comme, par exemple, l'utilisation de couleurs ou de types de traits pour identifier des flux d'objets entre méta-espaces (Figure 91 et 92).

Le Diagramme de Méta-Espace Fonctionnel permet de faire le lien entre le diagramme de méta-espace opérationnel et le diagramme fonctionnel (Figure 94). L'obtention de ce diagramme se fait par le traitement des exigences contenues dans les méta-espaces opérationnels. Les méta-espaces opérationnels sont regroupés en méta-espaces fonctionnels sur base des règles logiques et métiers de regroupement des exigences compatibles. Le diagramme de méta-espace fonctionnel peut également se déduire du diagramme fonctionnel. Chaque espace fonctionnel s'abstrait en un

méta-espace fonctionnel et leurs relations sont ensuite retranscrites dans le diagramme de méta-espace fonctionnel (Figure 95).

Le Diagramme de Méta-Espace de Conception (ou Physique) est utilisé pour modéliser la proposition (2D) de l'architecte. Chaque espace physique proposé par l'architecte est modélisé par un méta-espace physique et les accès entre espaces physiques sont retranscrits en lien entre les méta-espaces physiques. Le but est d'obtenir un formalisme identique entre le programme, le diagramme fonctionnel et les plans de l'architecte pour en faciliter la comparaison (Figure 96). La modélisation de trois propositions d'architectes sous ce formalisme facilite leur comparaison et l'identification des différences en termes d'agencement (Figure 97).

L'ensemble de ces trois diagrammes peut servir à outiller le processus de programmation architectural en traçant les informations issues des besoins de la MOA (c.-à-d. le programme) et retranscrites dans les projets architecturaux. Les méta-espaces sont utilisés pour modéliser des activités et des espaces tandis que les diagrammes de méta-espaces en modélisent les relations (p. ex. processus ou accessibilité entre espaces). L'ensemble de ces deux artefacts permet de formaliser la transition entre le concept d'Activité et celui d'Espace (Figure 98).

7. Synthèse

Dans ce chapitre, deux ensembles de modèles ont été introduits (Table 14). Les modèles conceptuels ont apporté un niveau de structuration supplémentaire au domaine de définition. Le paradigme Système Produit-Service a permis de modéliser l'objet d'étude et son périmètre. Le paradigme FBS a permis de modéliser, structurer, et compléter le domaine de définition ainsi que les transitions entre concepts au travers du modèle GFBS.

Les modèles opérationnels apportent un niveau de formalisation (graphique) des concepts du domaine de définition à partir de langages de modélisation existants. Ces langages de modélisation ne couvrant pas la transition entre les concepts d'Activité et d'Espace, un nouveau langage de modélisation a été proposé : le diagramme de méta-espace. A partir de l'ensemble de ces langages de modélisations, l'intégralité des concepts du domaine de définition peut être modélisée via plusieurs points de vue. Chaque modèle couvre son propre ensemble d'information avec un objectif différent. La combinaison de ces modèles apporte une vue plus complète sur le même objet d'étude mais de manière statique. Dans le chapitre suivant, ces différents modèles sont orchestrés/articulés au sein de méthodes appliquées sur deux étapes de la phase de programmation architecturale : la définition des exigences et l'évaluation de propositions architecturales.

Chapitre IV – DÉFINIR UN BÂTIMENT

Les chapitres précédents ont introduit un ensemble de concepts et de modèles pour structure et décrire le domaine de définition d'un bâtiment. Ce chapitre propose deux méthodes basées sur ces artefacts (Figure 99). Elles représentent des instanciations possibles de la théorie GFBS sur deux étapes de la programmation architecturale : la définition des besoins et l'évaluation de propositions architecturales. Ces deux méthodes sont illustrées à partir des données issues des différents cas d'études introduits dans le Chapitre I Section 4.4.

Dans la Section 1, un référentiel de processus de programmation architecturale est proposé sous forme de deux vues : une vue processus et une vue information, afin de positionner nos contributions. La Section 2 présente un exemple illustré de méthode de programmation architecturale s'appuyant sur les artefacts des Chapitres 2 et 3. La Section 3 modélise cette méthode de programmation architecturale selon différents points de vue complémentaires : une vue *processus*, une vue *flux d'information* et une vue *collaboration*. La Section 4 présente un exemple de méthode pour évaluer les propositions architecturales au regard des informations du programme en s'appuyant sur les même éléments que la Section 2. La Section 5 conclue ce chapitre.

1. Introduction

Le but de la programmation architecturale est d'assurer la meilleure qualité possible du futur bâtiment (Certu 2010). Il n'existe actuellement aucun processus normalisé ou méthode formelle pour encadrer ou outiller cette phase. En effet, la programmation architecturale s'apprend, se pratique et se développe principalement via l'expérience des architectes et programmistes. Elle dépend également fortement du type de projet de construction et de ses parties prenantes. Pour permettre de se positionner, nous proposons un processus de programmation générique (Figure 100) issue d'une interprétation des pratiques françaises telles qu'elles sont présentées dans le cadre de formations professionnelles (GEPA, PAMO).

Sur ce modèle sont représentés les échanges d'informations entre les principaux acteurs de la programmation architecturale. La plupart de ces échanges se font sous forme d'entretiens ou de groupes de travail (Heintz & Overgaard 2007) ou de documents plus ou moins contractuels (DGUHC 2000). Ce modèle simpliste ne représente pas les bouclages et autres itérations qui arrivent au cours du processus de programmation architecturale. Ces échanges d'informations font l'objet d'une standardisation en cours menée par BuildingSMART (Figure 101) (Jerving 2011).

Dans ces deux cas, les besoins et exigences se basent sur les activités à réaliser dans le futur bâtiment. La valeur ajoutée de ces travaux de recherche se situe sur :

- L'intégration de la notion de Service au processus de programmation architecturale permettant une meilleure compréhension et documentation des usages et exigences bâtiment, ainsi que
- La mise en relation et en adéquation entre les exigences de haut niveau (c.-à-d. Buts et Fonctions) et les exigences de plus bas niveau (c.-à-d. Comportements et Structure) pour permettre une aide à la décision tout au long du processus de programmation.

Les informations relatives à la partie bâtiment (c.-à-d. produit) et à la partie service sont distinguées et mise en relation au travers des différentes contributions faites dans ces travaux.

Ces travaux de recherche forment une base théorique et scientifique riche permettant la structuration et l'analyse des informations sur le système bâtiment à partir de plusieurs distinctions : produit-service, exigences-solutions, ou encore la dépendance ou non avec le contexte du projet de construction.

2. GFBS appliqué à la Définition des Besoins

Dans cette section, la théorie GFBS proposée dans le Chapitre III Section 3.4 est appliquée sur la phase de définition des besoins. Il en résulte une méthode de définition des besoins en huit étapes structurée selon quatre niveaux de perception du système bâtiment (c.-à-d. points de vue) (Figure 102). Le premier niveau fait référence à l'environnement tel qu'*observé* avant création du système bâtiment (c.-à-d. la situation AS-IS). Le deuxième niveau représente le résultat *attendu* en termes de système bâtiment et de son environnement (c.-à-d. la situation TO-BE). Le troisième niveau précise ce qui est *demandé* par la MOA au sujet du système bâtiment. Le dernier niveau représente ce qui est *proposé* par l'architecte ou l'équipe de conception.

Cette méthode est illustrée dans la version anglaise sur base des différents cas d'étude présentés dans le Chapitre 1 Section 3.2 (Figure 103). Bien que la présentation dans cette section soit séquentielle, il est à considérer que l'application de cette méthode se fait de manière récursive et itérative telle que dans l'approche Axiomatic Design (Suh 1998) (Figure 104). Cette récursivité et les différentes itérations sont représentées dans la Section 3 via les flux d'information modélisé en IDEF0.

2.1. Identification de l'Environnement (étape 0)

La première étape de la programmation architecturale est de comprendre le contexte du projet de construction. Dans ces travaux, la compréhension de ce contexte est limitée à l'environnement du système bâtiment. L'objectif de cette étape est de définir et de clarifier la situation actuelle menant à la nécessité de création du système bâtiment. Les informations sur ce contexte peuvent être structurées selon les concepts G_{Env}, F_{Env}, B_{Env} et S_{Env} (Figure 105). Ces informations concernent uniquement des éléments déjà existants et sont généralement récoltées au moment des études préliminaires. Les techniques d'élicitation des exigences issues du Génie Logiciel (Table 15) (Coulin 2007) peuvent être utilisées pour outiller cette récolte d'information.

2.2. Définition du But (étape 1)

La définition d'un but (G_{Sys}) pour un système bâtiment n'est pas chose aisée. En général, une multitude d'objectifs (p. ex. politiques, sociaux, culturels, urbains, architecturaux) sont définis pour un même projet de construction. En termes de bonnes pratiques, nous suggérons de définir un but principal et un ensemble limité d'objectifs secondaires. L'objectif de cette étape est donc de clarifier quel est le but commun des parties prenantes du projet vis-à-vis du système bâtiment (c.-à-d. la situation TO-BE). Ce but doit en définir la nécessité d'existence ainsi que son espérance de vie. L'atteinte de ce but n'étant pas évidente en soi, ce but peut se voir décomposer en un ensemble d'objectifs plus précis via notamment les méthodes issues du GORE ou encore de la *Theory Of Constraints* (TOC) (p. ex. arbre des réalités futures ou arbre des prérequis).

2.3. Génération de Fonctions (étape 2)

Un objectif se formule comme un état ou une condition à satisfaire. Lorsque celui-ci est suffisamment précis et ne nécessite pas plus de décomposition, cet état ou condition est à reformuler en tant que fonction du système bâtiment (F_{Sys}), c.-à-d. en tant que capacité fournie par le système bâtiment au MOA, aux utilisateurs ou usagers. L'objectif de cette étape est de conceptualiser la transition entre la situation AS-IS (G_{Env}, F_{Env}, B_{Env}, S_{Env}) et la situation TO-BE (G_{Sys}) sous une forme neutre en termes de solutions. Ceci peut se faire en utilisant le diagramme pieuvre du Génie Mécanique (Figure 112 et 113). Chaque fonction (F_{Sys}) est ensuite caractérisée par un ensemble de critères, niveau et flexibilité reflétant la bonne atteinte des buts et objectifs de manière plus concrète.

2.4. Allocation des Fonctions (étape 3 et 4)

Cette étape consiste à définir des concepts programmatiques (B_{Sys}) (c.-à-d. des principes de solutions) permettant la réalisation des différentes fonctions générées. L'objectif est de définir comment chaque fonction (F_{Sys}) peut être implémentée concrètement par les principaux éléments du système bâtiment. Le niveau de détail de ces principes de solutions (B_{Sys}) est peu important. Il

s'agit là d'esquisser, de donner l'intuition des solutions possibles et de choisir les plus pertinentes en accord avec les objectifs définis. L'Arbre des Voies Technologiques, l'Arborescence Fonction-Moyen ou encore les premiers niveaux du SADT/IDEF0 peuvent faciliter la représentation des différentes possibilités.

2.5. Décomposition des Comportements (étape 5)

Les concepts programmatiques (B_{Sys}) sont des principes de solutions de haut niveau. L'étape de Décomposition des Comportements consiste à détailler ces principes de solutions en termes de solutions techniques. Ces solutions techniques sont réalisées à partir de comportements de la partie produit (B_{Pro}), de la partie service (B_{Ser}) ou d'une combinaison des deux. A partir de la description de ces solutions techniques sont déduites, entre autres, les exigences bâtiment. On en déduit également les exigences sur tous les éléments internes du système bâtiment.

2.6. Traitement des Exigences (étape 6)

La décomposition des Comportements (B_{Pro} et B_{Ser}) génère une quantité immense d'informations sur le système bâtiment, ses composants et leurs comportements. Les exigences bâtiment font référence aux espaces nécessaires à la réalisation des différentes activités et à leurs caractéristiques. L'objectif de cette étape est de réduire la quantité d'exigences bâtiment (c.-à-d. d'espaces) en utilisant le Méta-Espace comme moyen d'abstraction et de transition entre le concept d'Activité et celui d'Espace. Ainsi, chaque opération à réaliser est modélisée sous forme d'un méta-espace opérationnel (B_{Pro}). Les dépendances entre les opérations sont retracées dans les diagrammes de méta-espaces opérationnels (p. ex. flux d'information, processus, flux de ressource). Chaque méta-espace opérationnel (B_{Pro}) regroupe les exigences relatives à la bonne réalisation de l'activité. Les méta-espaces opérationnels (B_{Pro}) sont ensuite regroupés en méta-espaces fonctionnels (B_{Pro}) en fonction de règles de regroupement et de compatibilité entre exigences (Figure 121 à 124).

2.7. Synthèse Fonctionnelle (étape 7)

La synthèse fonctionnelle consiste à définir la typologie et les attributs des espaces fonctionnels (F_{Pro}) ainsi que leurs relations à partir des diagrammes de méta-espaces fonctionnels obtenus à lors de l'étape précédente. L'objectif est de résumer les informations fonctionnelles du programme en une liste d'espaces et d'exigences spatiales. Le résultat de cette synthèse se retrouve sous la forme d'un diagramme fonctionnel (Figure 127).

2.8. Synthèse Physique (étape 8)

La synthèse physique consiste à formaliser la quantité d'espaces (S_{Pro}) nécessaire selon les exigences formulées par le client. Sur base des exigences fonctionnelles, le programmiste quantifie les espaces physiques (S_{Pro}) afin de permettre la réalisation de l'ensemble des activités (B_{Ser}) prévues en accord avec les données de l'environnement (p. ex. le nombre d'élèves prévus dans l'école). Le résultat de cette synthèse se retrouve dans le tableau des surfaces (Table 17).

2.9. En Résumé

Dans cette section, nous avons présenté un exemple illustré de méthode de programmation architecturale basée sur la théorie GFBS (Table 18). Cette méthode est illustrée à partir de données issues de quatre cas d'étude complémentaires et s'appuie sur l'ensemble des concepts, techniques et modèles présentés dans les chapitres précédents. La section suivante propose trois points de vue différents sur cette méthode de programmation afin de représenter plus en détail le séquencement des étapes au niveau opérationnel, les flux d'informations entre les étapes et les interactions entre les parties prenantes de la programmation architecturale.

3. Instanciation sur des Vues Processus

Dans la Section 2, nous avons présenté un exemple de méthode de programmation architecturale basé sur la théorie GFBS. Dans cette section, nous proposons de représenter cette méthode selon trois vues détaillant l'instanciation de cette méthode. La première vue présentée est

une vue processus représentant les liens dynamiques entre les étapes de la méthode. Cette vue est modélisée en BPMN. La deuxième vue représente les flux d'informations entre les étapes de la méthode. Cette vue est modélisée en IDEF0 ce qui nous permet de représenter également les boucles de contrôle et les itérations possibles. La troisième vue est une réactualisation du processus IDM présenté en Section 1 représentant les aspects collaboratifs.

3.1. Vue Processus

La méthode de programmation architecturale GFBS fait intervenir différentes parties prenantes. Dans cette section, nous allons représenter ses principales étapes vis-à-vis des interactions entre deux de ses acteurs principaux : les clients (MOA, utilisateurs et usagers) et le programmiste. Ces étapes et interactions sont modélisées dans le langage BPMN. La méthode proposée est structurée en quatre parties : l'Analyse Externe, l'Analyse Interne, le Traitement des Exigences et la Synthèse des Exigences (Figure 130 et Annexe H).

L'Analyse Externe fait référence à l'Analyse Fonctionnelle. Le principe est de définir les fonctions de l'objet d'étude (considéré comme une boite noire) à partir de l'analyse de son environnement. Cette première partie regroupe les étapes 0 à 2 de la méthode de programmation GFBS, c.-à-d. d'Identification de l'Environnement, de Définition du But et de Génération des Fonctions. Les informations fournies par les clients (p. ex. études préliminaires, informations sur le contexte et le projet, stratégie de la MOA) sont récoltées et traitées par le programmiste qui les traduit en Buts (G_{Sys}) et en Fonctions (F_{Sys}) avant de les restituer aux clients pour validation (Figure 131).

L'Analyse Interne a pour objectif de décrire comment le système bâtiment va fournir les fonctions (F_{Sys}) issues de l'Analyse Externe. Cette deuxième partie regroupe les étapes 3 à 5 de la méthode de programmation GFBS, à savoir l'Allocation des Fonctions à la partie Produit ou Service du système bâtiment, la Définition du Service et la Décomposition des Comportements. La description du fonctionnement interne du système bâtiment se fait en étroite collaboration avec les clients. A l'issue de l'étape de définition des Services, le programmiste édite le préprogramme qui synthétise l'éventail des principes de solutions possibles (Figure 132). Ce n'est que lors de la Décomposition des Comportements que les principes de solutions les plus pertinents seront développés en termes d'exigences bâtiment.

Le Traitement des Exigences a pour objectif de formaliser la transition entre les Comportements Produit (B_{Pro}) et Service (B_{Ser}) du système bâtiment et les exigences bâtiment et de rationaliser les exigences bâtiment via les Méta-Espaces (étape 6). Cette troisième partie se décompose en trois étapes : la génération de méta-espaces opérationnels, la création de diagrammes de méta-espaces opérationnels et la création de diagrammes de méta-espaces fonctionnels via le regroupement des méta-espaces opérationnels en méta-espaces fonctionnels (Figure 133).

La Synthèse des Exigences regroupe les dernières étapes de la méthode de programmation (étapes 7 et 8) de Synthèse Fonctionnelle, Synthèse Physique et de Rédaction du Programme. La Synthèse Fonctionnelle vise à la création du diagramme fonctionnel à partir du diagramme de méta-espace fonctionnel. Il s'agit de la synthèse graphique des exigences fonctionnelles sur le système bâtiment. La Synthèse Physique consiste à quantifier le nombre d'espaces nécessaires pour le système bâtiment. Le programme est ensuite rédigé (synthèse textuelle) et soumis à la MOA pour validation avant transmission aux architectes (Figure 134).

Cette vue processus indique les points de synchronisation et les interactions entre le programmiste et les clients tout au long de la méthode de programmation GFBS. Des échanges réguliers entre ces deux entités sont essentiels pour le bon déroulement de la méthode proposée.

3.2. Vue Information

Dans cette section, le langage IDEF0 est utilisé pour modéliser les flux d'informations entre les activités reprises du BPMN avec un focus sur les activités du programmiste. Ses interactions avec les clients sont représentées par un contrôle réalisé par la MOA sur les activités concernées.

Sur ce modèle, les outils et techniques issus des divers domaines d'ingénierie (Chapitre III) complètent la description des activités au travers des mécanismes.

La Figure 135 modélise la méthode de programmation architecturale à un niveau macro en reprenant les quatre parties définies dans la section précédente. Chaque partie est ensuite détaillée vis-à-vis des étapes qui la composent et des flux d'informations internes incluant les bouclages entre activités : Analyse Externe (Figure 136), Analyse Interne (Figure 137), Traitement des Exigences (Figure 138) et Synthèse des Exigences (Figure 139).

Cette représentation IDEF0 de l'exemple d'application de la théorie GFBS sur le processus de programmation architecturale nous permet de montrer plus concrètement à quel moment les outils et techniques de l'ingénierie peuvent être utilisés. Le principe d'aller-retour issu de l'approche *Axiomatic Design* y est représenté par l'intermédiaire des boucles d'informations entre les activités, ceci pour indiquer que le processus de programmation n'est pas linéaire.

3.3. Vue Collaboration

Pour compléter les deux vues précédentes, cette section propose de positionner la valeur ajoutée de l'approche GFBS au sein de la programmation architecturale en complétant le processus IDM proposé par BuildingSMART (Jerving 2011). Dans la proposition de BuildingSMART (Figure 101), le concept d'*Activité* est primordial dans la génération d'informations telles que les *Fonctions*, les *Exigences Spatiales*, Les *Exigences Systèmes* ou encore les *Exigences d'Equipements*. Le modèle actuel reflète l'état des réflexions et des solutions existantes autour de la programmation architecturale de par l'absence de modélisation des activités des diverses parties prenantes. Seules les informations résultantes sont modélisées (p. ex. er_exchange_requirements) et transmises aux concepteurs afin qu'ils puissent modéliser les espaces et leurs contraintes.

L'approche GFBS pour la programmation architecturale introduit dans ce modèle la notion de **Service** via la modélisation des activités issues des données fournies par les diverses parties prenantes sous forme de processus (Figure 140). Il en résulte un *modèle de Service* qui est transmis aux concepteurs pour leur permettre de tester leur proposition architecturale tout au long de sa conception. Ce *modèle de Service* a également pour objectif d'améliorer la compréhension des activités des parties prenantes (c.-à-d. MOA, utilisateurs et usagers) via leur formalisation. Une illustration du processus d'enrichissement du modèle numérique de bâtiment est représentée Figure 141 au travers d'un diagramme de séquence.

3.4. En Résumé

La méthode de programmation architecturale basée sur la théorie GFBS proposée en Section 2 a été modélisée selon trois vues : une vue processus (BPMN), une vue information (IDEF0) et une vue collaboration (processus IDM). Chaque vue est une instanciation qui nous apporte un complément d'information sur l'utilisation concrète de cette méthode. Ces instanciations constituent des éléments de vérification de la méthode proposée. Pour en obtenir une validation, une application concrète sur un cas d'étude sera nécessaire.

La section suivante propose une autre façon d'appréhender la théorie GFBS en l'appliquant à l'évaluation fonctionnelle de proposition d'architectes dans la cadre notamment de concours d'architecture. Il s'agit là d'une autre forme de vérification de la méthode.

4. GFBS appliqué à l'Evaluation de Projets

Lors d'un concours d'architecture, une partie de l'évaluation des propositions architecturales porte sur les usages. Cette évaluation fonctionnelle et objective est réalisée par le Comité Technique à partir des informations contenues dans le programme. Dans cette section, nous nous proposons d'appliquer l'approche GFBS pour structurer cette évaluation. Pour faire le lien avec la programmation architecturale GFBS, un niveau intermédiaire est ajouté entre ce qui est *proposé* par l'architecte et ce qui a été *demandé* par le programmiste et la MOA. Il s'agit des informations *analysées* par le Comité Technique lors de l'évaluation des propositions (Figure 142).

L'évaluation GFBS est structurée en trois phases : l'Analyse Fonctionnelle, l'Evaluation Fonctionnelle et l'Evaluation Quantitative (Figure 143). Deux variantes sont proposées pour les phases d'Analyse Fonctionnelle et d'Evaluation Fonctionnelle afin de mettre en valeur l'intérêt du concept de Méta-Espace. Ces deux variantes reflètent également deux cas de figure dans lesquels peuvent se retrouver le Comité Technique en fonction des informations complémentaires données par les concepteurs sur leur proposition respective. Les cas d'études présentés dans le Chapitre I Section 3.2 sont utilisés pour illustrer ces variantes (Figure 144).

4.1. Analyse Fonctionnelle

L'Analyse Fonctionnelle consiste à identifier les espaces fonctionnels des propositions architecturales. Ces espaces fonctionnels et leurs relations sont représentés dans le programme via le diagramme fonctionnel. L'objectif pour le Comité Technique est de vérifier comment chacune des propositions répond à cet ensemble d'exigences fonctionnelles.

Dans la première variante (Figure 146), les architectes n'ont pas indiqué dans leur proposition quelle est la fonction de chacun des espaces composant la proposition. C'est le cas de la bibliothèque universitaire. Pour faire l'analyse fonctionnelle de cette proposition, le Comité Technique va devoir identifier les espaces fonctionnels par lui-même (Figure 147 à 149) et modéliser la proposition architecturale sous forme de diagramme de méta-espace (Figure 150).

Dans la seconde variante (Figure 152), les architectes ont identifié eux-mêmes les espaces fonctionnels dans leur proposition. C'est le cas de la médiathèque. Pour réaliser l'Analyse Fonctionnelle, le Comité Technique peut directement créer le modèle des propositions sous forme de diagrammes de méta-espace (Figure 153).

4.2. Evaluation Fonctionnelle

L'Evaluation Fonctionnelle consiste à évaluer de manière objective les propositions architecturales. Cette évaluation peut être réalisée à deux niveaux : global ou local (Figure 154).

L'évaluation globale consiste à comparer les propositions architecturales au diagramme fonctionnel (c.-à-d. espaces fonctionnels et leurs relations). Le diagramme de méta-espace est ici utilisé pour abstraire les deux livrables dans un langage commun pour faciliter leur comparaison (Figure 156). Le diagramme de méta-espace permet également de faciliter la lecture fonctionnelle des différentes propositions (Figure 157).

L'Evaluation Locale consiste à appliquer les processus définis dans le programme (p. ex. le cycle du livre) sur les propositions architecturales. Chaque opération qui compose un processus peut être rattachée à un espace fonctionnel. Suivre un processus revient donc à regarder les espaces traversés tout au long des activités qui le composent. Le suivi des processus est facilité par l'utilisation des diagrammes de méta-espace physique et fonctionnel qui, pour le premier, modélise les propositions architecturales et, pour le second, modélise les différents usages du bâtiment (Figure 159).

4.3. Evaluation Quantitative

L'Evaluation Quantitative consiste à vérifier les quantités et les caractéristiques des composants du système bâtiment entre les propositions architecturales (S_{Pro} *proposé*) et le programme (S_{Pro} *demandé*) (Figure 160). Une telle évaluation a été réalisée dans le cas de la bibliothèque universitaire afin de vérifier comment les collections de documents de la bibliothèque pouvaient être distribuées dans le bâtiment proposé par l'architecte (Figure 161 et Annexe J).

4.4. En Résumé

Sur base des conclusions des évaluations fonctionnelles et quantitatives, le Comité Technique peut éditer un document recensant les différences entre le programme et les propositions architecturale de manière plus précise (via les cas d'utilisation formalisés) et objective. Ces résultats peuvent ensuite être présentés à la MOA et au jury du concours pour discussions et sélection de la meilleure proposition architecturale.

5. Synthèse

Dans ce chapitre, l'ensemble des artefacts de conception (concepts, modèles et méthode) créés dans les chapitres précédents a été déployé et mis en relation au sein d'un exemple de méthode de programmation architecturale couvrant la définition des exigences et l'évaluation des propositions architecturales. Celle-ci est illustrée à partir des informations issues des cas d'étude. Après une présentation théorique, cette méthode de programmation architecturale a été modélisée selon trois points de vue (processus, information, collaboration) afin de montrer comment elle pourrait s'appliquer concrètement. La valeur ajoutée des différents artefacts de conception est mise en valeur via les cas d'étude. La prochaine étape consisterait à appliquer cette méthode de programmation architecturale sur un cas d'étude en situation réelle (type Living Lab). Les différents points de vue présentés indiquent un certain intérêt à informatiser la méthode pour bénéficier pleinement des modèles introduits dans le Chapitre III. La quantité d'informations générées à chaque étape et l'interopérabilité nécessaire pour articuler les outils et les techniques proposés en font un point essentiel en termes de perspectives.

Chapitre V – CONCLUSION

Ce chapitre conclut ce résumé étendu en deux sections. Chaque section résume comment chaque contribution issue de ces travaux de recherche aborde les questions et observations faites dans le chapitre d'introduction (Chapitre I) ainsi que quelques perspectives. La Section 1 se concentre sur les aspects scientifiques tandis que la Section 2 aborde les aspects industriels.

1. Synthèse Scientifique

Le problème de recherche abordé dans ces travaux se concentre sur le transfert de connaissances en définition des besoins issues de divers domaines d'ingénierie vers le domaine de l'Architecture-Ingénierie-Construction. Le Chapitre I nous a dressé un panorama de l'état de l'art coté Architecture-Ingénierie-Construction ayant mené à la conclusion que les pratiques de programmations architecturales sont fortement liés à l'expérience des programmistes, souvent orientées solutions et sans réel bénéfice des outils informatiques disponibles actuellement. Le Chapitre III nous a confirmé cette tendance via un état de l'art des modèles pratiques utilisés dans le domaine. Enfin le Chapitre II nous a souligné un certain manque de clarté dans la définition des concepts permettant de décrire l'objet bâtiment et son processus de définition. D'un autre côté, la richesse des connaissances des domaines d'ingénierie tels que le Génie Mécanique, Industriel et Logiciel a pu être constaté au travers des Chapitre II et III. Les concepts et modèles existants permettent de structurer, tracer et formaliser la définition des besoins de leur propre objet d'étude tout au long du processus de développement.

Le but de ces travaux de recherche était donc de développer un raisonnement pour la programmation architecturale à partir des connaissances issues de ces domaines d'ingénierie. Pour atteindre ce but, nous avons répondu à trois questions de recherche portant sur (1) les concepts nécessaires pour décrire notre objet d'étude, (2) les modèles permettant de le représenter ainsi que son domaine de définition selon des points de vue multiples et (3) une méthode de programmation architecturale s'appuyant sur ces concepts et modèles. Les concepts nécessaires (1) ont été identifiés et définis au sein d'un domaine de définition dans le Chapitre II. Un ensemble de modèles (2) a été proposé dans le Chapitre III afin de préciser le périmètre de l'objet d'étude ainsi que divers moyens de représenter son domaine de définition dans son intégralité et selon plusieurs points de vue. Enfin un exemple illustré de la méthode (3) appliquant la théorie GFBS et s'appuyant sur les concepts et modèles des Chapitre II et III a été proposé dans le Chapitre IV.

Le caractère prospectif de ces travaux de recherche nous a permis d'identifier plusieurs pistes de recherche à approfondir pour la suite, principalement pour le domaine de l'Architecture-Ingénierie-Construction. Les pistes évoquées concernent : (1) l'analyse sémantique basée sur les ontologies pour enrichir ou valider le domaine de définition proposé, (2) l'étude de la relation entre les diagrammes de méta-espace et les modèles de graphes utilisés dans le cadre de la génération et de l'optimisation d'agencements spatiaux en conception architecturale, (3) l'analyse de l'intérêt du concept de méta-espace en conception paramétrique afin de mettre en relation les besoins de la MOA et des paramètres de conceptions, (4) la mesure de l'influence de l'utilisation de modèles d'ingénierie sur le processus de conception architecturale, (5) l'analyse de la mise en place d'une gestion des connaissances en programmation architecturale en termes d'usages et de données, ou encore (6) l'étude du lien entre le domaine de définition proposé et les modèles et logiciels IFC et BIM existants.

2. Synthèse Industrielle

Sur le plan pratique, le but de ces travaux de recherche était de structurer et d'outiller la programmation architecturale fonctionnelle de bâtiments de type publics. Le cadre est apporté par le domaine de définition (Chapitre II) ainsi que la méthode GBFS appliquée à la programmation architecturale (Chapitre IV). L'outillage est quant à lui proposé au travers de la multitude de langages de modélisation issue de domaines de l'ingénierie (Chapitre III) qui ont été positionnés tout au long de la proposition de méthodes de programmation architecturale (Chapitre IV).

Les différentes contributions ont essayé d'aborder les quatre objectifs industriels définis dans le Chapitre I. L'objectif 1 portait sur l'expression des exigences, le partage et la compréhension commune de ces exigences ainsi que leur complétude. Les aspects de fiabilité et de pertinence n'ont pas pu être traités dans le cadre de ces travaux. L'objectif 2 se concentrait sur l'outillage de l'analyse et du traitement des exigences en termes de consistances, de maturité, de traçabilité et de dépendance entre exigences. L'objectif 3 se focalisait sur les aspects de formalisation des exigences bâtiment pour en permettre l'usage et la vérification par les concepteurs avec des critères de clarté et d'utilité. Enfin, l'objectif 4 avait pour vocation de faciliter la vérification par la MOA ou le programmiste de la bonne implémentation des exigences bâtiment dans les propositions architecturales et ce de manière objective. L'idée derrière ces objectifs industriels n'était pas de les satisfaire mais plutôt d'orienter les propositions faites dans ces travaux afin d'en assurer une certaine applicabilité.

Les perspectives industrielles de ces travaux s'orientent vers un élargissement du cadre d'application à tout type de bâtiment ainsi que vers une industrialisation des contributions proposées. Tout d'abord, nous pouvons citer (1) l'extension du domaine de définition proposé aux informations ''non-fonctionnelles'' pour obtenir une description complète du système bâtiment, (2) une application concrète et complète de la méthode et des modèles sur un cas d'étude en vue d'une validation et d'une formalisation plus avancée, (3) la formalisation des règles de regroupement pour les méta-espaces sur base des attributs et propriétés qui décrivent concrètement le domaine de définition, (4) le développement d'une solution informatique pour permettre de vraiment bénéficier de l'intérêt des langages de modélisation proposés, (5) l'intégration et les interactions avec les pratiques et solutions BIM et enfin (6) la validation de l'intérêt de telles contributions par les professionnels ainsi que l'évaluation du degré d'acceptation de telles solutions par cette communauté.

Part B – English Full Version

Chapter I – BACKGROUND

The following chapter introduces the background of four years of research work on the topic of architectural programming and requirements definition in engineering. Section 1 defines, positions and reviews architectural programming as a topic of research. Section 2 introduces the scope and practical issues that led to the definition of industrial goal and objectives used as practical guidelines during the research. Based on these elements, Section 3 presents the research objective and questions tackled in this dissertation. Section 4 describes the methodology followed to address these research objective and questions. Finally, Section 5 presents the layout of the dissertation.

1. Introduction

A construction project life-cycle is composed of complex activities, from inception to demolition, involving numerous and heterogeneous actors during relatively short periods. The organisation of these activities can be multiple depending on several criteria (e.g. the building type, the contractual framework, the public or private stakeholders, the budget and time allocated, etc.). In this research, the focus is on a specific early phase of this life-cycle: the conceptual phase. As a result, to limit the complexity, a consecutive organization of the construction project life-cycle is considered even if it is not in practice.

This section provides an overview of the conceptual phase of construction projects, aka architectural programming (USA) or briefing process (UK). Both terms are used in the dissertation with the same meaning. In the first part, architectural programming is defined and positioned, and its importance is emphasized. A few specificities of architectural programming are then introduced to apprehend its complexity. The section concludes with a state-of-the-art on architectural programming that details the context of this research.

1.1. Definition

The briefing process consists of defining the framework and expectation of a construction project through a statement of clients' requirements (Abdul-Kadir & Price 1995). This task is generally performed by the clients or by a professional, the programmer, under the clients' responsibility. It is an extensive information process (Voordt & Wegen 2005) which purpose is to inform decision-making and (critical) decision implementation about the project development (Kelly et al. 2002).

"Architectural programming is the research and decision-making process that defines the problem to be solved by design." (Cherry 1999)

This phase, as any conceptual phase, is based on very fuzzy and incomplete information to gather, analyse and clarify (Wang et al. 2002). It constitutes the core of clients' satisfaction about the future project. Macmillan proposes a framework that divides this phase in two stages (Macmillan et al. 2001); the first one consists of developing business needs into a design strategy whereas the second develops the design strategy into a concept proposal (Figure 1). Architectural programming is more focused on the first stages of the conceptual phase.

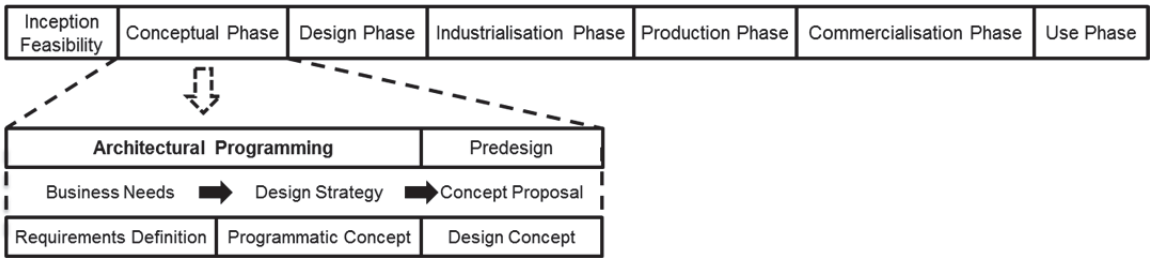


Figure 1 – Conceptual phase of construction projects: architectural programming (adapted from (Macmillan et al. 2001)*)*

The briefing process is recognised as one of the most important phase of construction projects (Shen et al. 2004; Yu et al. 2005; Tzortzopoulos et al. 2006). The preliminary (i.e. inception and feasibility), architectural programming and design studies represent around 10% of the total operation cost of a construction project. The rest of it is dedicated to the construction of the building (Certu 2010). During the architectural programming and design phase, around 75% of the global cost of the construction project is engaged (Faatz 2009; Certu 2010). Paulson, and later MacLeamy, already demonstrate that this earlier phase constitutes the highest influence on construction project while expenditures are at the lowest (Paulson 1976). This phase presents the greatest opportunity for improvement (Figure 2).

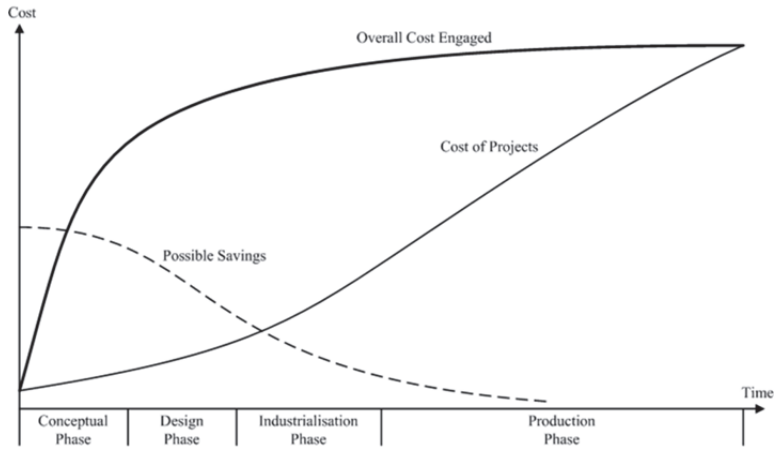


Figure 2 – Evolution of costs and savings along construction projects development based on (Certu 2010; Faatz 2009)

1.2. Construction Project Specificities

Architectural programming differs from the conceptual phase of other engineering domains due to the very nature of its object of interest. A building faces many constraints and requirements to satisfy in terms of social, urban, functional, technical and economical quality, as well as landscape insertion and environmental impact regarding both its realization and usage (DGUHC & Certu 1999; JORF 2007). All of these elements constitute a huge amount of heterogeneous, fuzzy and unstable, multi-format and multi-source information to gather and process during the briefing process (Shen et al. 2004).

Architectural programming is not based on "*solid*" market studies as for manufacturing or software products. It is based on clients' (unstable) business needs that are highly context specific. Indeed, the definition of building requirements is tightly linked to the clients' business operation and strategy (Tzortzopoulos et al. 2006). The understanding of this business strategy and operation is rarely thorough even from the clients' side who have difficulties to describe them (Cooper & Press 1995; Barrett & Stanley 1999). It is generally subjected to many changes and evolution during the project development.

Architectural programming has to deal with complex and uncoordinated (multi-headed) actors and multiple stakeholders (Newcombe 2003; Tzortzopoulos et al. 2006) to gather this information:

- The **client** is rarely unique. A distinction is generally made between the **paying clients**, the **user clients**, and the **customer clients**. In most engineering projects, at least two of them are the same person but in the case of public construction, the three of them are totally different entities. Each one of them has a distinct relationship to the building that required to be handled in a specific way depending on the context. *In a school building example, the paying client is the Ministry of National Education who pays for a new school. The user client is the personnel (e.g. teachers, director, janitor…) required to provide (more or less directly) education to the pupils, customer clients of the school.*
- The **designer** is generally represented by a temporary design team which is composed of one or several architecture firms, various expert consulting engineer offices (e.g. structural engineers, thermal engineers…), and other independent consultants from construction economist to legal experts. This design team is often not defined at the programming stage, e.g. in the case of an architectural design competition.
- The **building contractor** is also represented by a temporary and evolving construction team whose assigned tasks are highly interrelated and interdependent in time and sequence leading to long and irregular production period (Kubicki et al. 2006).

Buildings also distinguish themselves regarding their long and specific life span compared to their rather intensive usage as well as regarding the overall resulting costs (e.g. from inception to dismantling). It, generally, leads more systematically to several second lives through refurbishment and rehabilitation, or call for specific dismantling procedures. Such characteristics have to be integrated early in the project development, which impact the briefing process.

1.3. State of Art and Issues

In this section, a literature review on architectural programming is presented. First, its rather recent origins are presented. Then, to further contextualize the development of architectural programming, the practices in four countries of interest are introduced before presenting the various (global) existing approaches. This is followed by a synthesis of limitations observed by researchers regarding books and guidelines that formalize these approaches for clients and practitioners. Finally, the most recent ICT solutions developed to improve architectural programming are presented.

1.3.1. A little bit of history

Historically, the briefing process as a formalization of the clients' demand always existed (Kumlin 1995; Cherry 1999; Berrada 2006). However, till the mid-19th century (Viollet-le-Duc 1863), the focus in architecture was on the project with no deep considerations about the brief (Bousbaci 2004). The rise of architectural programming as a topic of investigation in research and practice seems partly tied to the "*brief/project*" model and its influence (supremacy) in architecture, and the rise of the design methods movement in the mid-20th century.

The "*brief/project*" model and its composing terms are strongly "*rooted*" in architectural culture, language and history (Hall 1987). The "*brief/project*" model embodies the best the *Académie des Beaux-Arts* traditional pedagogical process which essentially prepared architectural students for competitions (Pérouse de Montclos 1984). The studies were punctuated by a series of competitions (Conan 1997) based on a brief provided by the teacher which students had to answer through a sketch and a project developing the sketch (Bousbaci 2004).

Even if its influence on current architectural design competition is not clearly established in literature, the link between them cannot be totally avoided considering its supremacy in architectural education and its continuous impact on architectural practices. Architectural design competition provides an interesting variety of projects for the same unique brief (RIBA 2013a). Even though, it exists since a very long time (Jong & Mattie 1995), their regulation remains recent and took more and more importance in later years to promote fairness (AIA 1988; ISO 1994). Such formalization of the architectural design competition could only foster the development of research and practice in architectural programming. This is particularly the case in France (see Section 1.3.2).

The design methods movement designates a post-war school of thought interested in the emerging systems sciences to rethink the design process and activity (Upitis 2008). Three generations of design methods are considered (Rittel 1973; Cross 1981). The first generation (1960s) is about the generalization of US spatial and military approach (NASA) to other fields of (civil) application. The second generation (1970s) moved from the definition of optimal solution toward satisfactory or appropriate ones with the concept of "*wicked problems*" (Simon 1969) or "*ill-structured problems*" (Simon 1973) more relevant to architecture and planning (Rittel & Webber 1973) than to engineering and industrial design. This second generation marks the divergence between architecture and engineering (1980s) regarding design methodologies (Cross 1993). The third generation (architecture, 1980s) integrates the subjectivity of the designers ignored by previous generations (Broadbent 1984) and considers a rather trial and error approach inspired from Popper's theory (Popper 1936). Results or consequences of the design methods movement are introduced in Section 1.3.2 regarding international practices and in Section 1.3.3 regarding architectural programming approaches.

1.3.2. Review of International Practices

Architectural programming is not performed the same way depending on the country. Each one of them has developed a set of regulations, guidelines, or standards of their own. These practices are related to: the architects' education and their culture, (public) contracting/tender offers, public owner and regulators "impulse" and association of owners, and Facility Management policies. This section presents a review of four countries to position this research: the United States of America and the United Kingdom (as most of the literature comes from there), Luxembourg (as part of the research context), and France (regarding its influence on the Luxembourgish practices).

USA

The development of architectural programming in United States is grounded in post-World War II issues regarding urban planning qualified as insensitive (i.e. disconnected from social and human concerns) (Gans 1962) exemplified by the famous Pruitt-Igoe project failure (Duerk 1993), and the impact of the baby boom on U.S. school facilities (Cherry 1999) that initiated the user participation in facility planning and later on their involvement in the design process through the development of design methods. The term architectural programming is born from the term architectural analysis firstly introduced by William Peña and William Caudill in *Architectural analysis – Prelude to Good Design* (Pena & Caudill 1959) then further developed by others (Table 1). The result of these developments is a list of architectural programming approaches classified in four categories (Hershberger 1999): Design-based architectural programming, Knowledge-based architectural programming, Agreement-based architectural programming, and Value-based architectural programming (the different approaches are further developed in Section 1.3.3). The various existing positions and approaches regarding architectural programming in the United States supports the needs of continued development of this specialty (Cherry 1999).

Table 1 – Main U.S. pioneers of architectural programming, taken from (Cherry 1999)

U.S. pioneers	Main References
William Peña William Caudill Steven Parshall	(Pena 1959; Pena 1969) (Pena & Caudill 1959) (Pena & Parshall 2001)
Gerald Davis	(Davis 1972; Davis 1978; Davis 1982; Davis & Szigoti 1985)
Herbert McLaughlin	(McLaughlin 1976; McLaughlin 1979)
Edward T. White	(White 1972; White 1991)
Wolfgang F.E. Preiser	(Preiser 1972b; Preiser 1972a; Preiser 1977; Preiser 1985)

UK

Briefing practices are investigated since the 1960s by both academic researchers and practitioners. However, during 30 years, little improvement or changes have been made on the way the briefing process is performed (Barrett et al. 1999). In their book *Better Construction Briefing*, Peter Barrett and Catherine Stanley expose the various issues related to the briefing process and the limits of current best practices (Barrett & Stanley 1999). The need of improvements on the briefing process is a recurring one as stated in UK reports on the Construction Industry (Latham 1994; Egan 1998; Kelly et al. 2002; Wolstenholme 2009). A general conclusion of these research and reports is the need to involve, commit, and empower the building clients during and through the briefing process. The Royal Institute of British Architects (RIBA) emphasizes the importance of the briefing process by integrating it as a specific part of its RIBA "*Plan of Work*" under the label "*Preparation*". The RIBA "*Plan of Work*" is the definitive UK model for the building design and construction process developed since 1963 (RIBA 2013b).

France

The situation in France is quite different from USA and UK. Architectural programming was formally introduced in 1973 through a public decree (JORF 1973) relative to engineering tasks (a.k.a. the Engineering Reform of 1973) and formalised in 1985 through the Public Contract Law (JORF 1985; JORF 2010). This law defines the relationships between the contracting owner (i.e.

paying client) and the general contractor (i.e. architect, designer). It imposes a strict separation between the paying client, the architect, and the contractors in order to (1) implicate the paying client in the briefing process (i.e. responsibility), (2) foster fully developed proposal by the architect, and (3) reduce excessive profits by contractors (based on Cabanieu 2011, GEPA, PAMO). Architectural programming and design are therefore dissociated. The paying client becomes responsible for the architectural programming and the architects are pushed to provide fully developed projects before construction.

Letters of Nobility of Architectural Programming (France):

"The contracting owner defines in the brief the objectives of the construction project and the needs it has to satisfy as well as the constraints and requirements of social, urban, architectural, functional, technical, and economic quality, of landscape integration and environmental conservation, regarding the realization and usage of the work." (translated from (JORF 1985)*)*

In addition, architectural competitions become a legal prerequisite for public construction projects exceeding a certain budget (Cabanieu 1994). Its purpose is to foster the architectural quality of design proposals by putting in competition several design teams (MIQCP 2012). Such formalization of architectural programming practices led to the rise of a new autonomous profession: programmer, syndicated in 1994 (SYPAA 2005). A programmer helps the contracting owner with architectural programming. He has an architecture, social science, economy, or engineering background. Despite the legal push, education and trainings on architectural programming still lack (Allégret et al. 2005a) and tend toward urban planning or clients' awareness of its importance (SYPAA 2011). Architectural programming in France seems to be still in an awareness campaign and struggles to open up (MIQCP 2008; Certu 2010; DGUHC 2000). In 2005, a French survey has shown that around 70% of the surveyed professional programmers claimed to be autodidacts (Allégret et al. 2005b).

Luxembourg

Architectural programming is a very recent topic of research in Luxembourg (Harbouche et al. 2008). A first study on architectural programming in Luxembourg school buildings was initiated in 2008 by the Public Research Centre Henri Tudor (through the SC-Construct project, which gathered 25 actors of the public and private sectors in Luxembourg). This study reveals that most of the briefs were copy-paste of the previous construction projects based on a standard document (Luxembourg Building Ministry 2006). The result is a global overcapacity review of the current school buildings with occupation rate from 65% in the briefing phase to 50% in the building usage. This is mainly explained by two aspects: the need to have dedicated spaces for certain activities as well as a maximum flexibility regarding the other spaces of the building, and the lack of alignment between a desired operating that led the brief and the resulting operating in the end building. This situation, acceptable a few years ago, becomes more difficult to assume nowadays.

Based on these first results, a survey on architectural programming in Luxembourg was initiated in 2013 in the same spirit as the RAMAU project in France (Allégret et al. 2005a) to understand how it is perceived and performed. The briefing mission seems to be largely handled by architects with a tendency toward specialisation on this mission. Indeed, architectural programming is still considered by many as a competitive advantage by architects. The importance of architectural programming is recognized as a key element to ensure quality and satisfaction of clients' needs. Contracting owners as well as prime contractors are numerous to acknowledge that architectural programming will become an activity single-handedly ensured by more specialised and autonomous profiles (Guerriero & Fry 2014). The described future of architectural programming in Luxembourg corresponds to the current situation in France. The quintessentially French architectural programming practices are used as a basis to develop the Luxembourgish ones especially through professional trainings (CRP Henri Tudor 2012).

Conclusion

Despite its recognized importance, there is still a lack of interest in architectural programming by the Architecture-Engineering-Construction community. Whether pushed by practitioners,

researchers, governments, or regulation, architectural programming still needs to be developed. It is not a well-established "*discipline*". First of all, it is rather absent of architecture education considering the various (strewn) efforts put in it since the 1980s. Architectural programming is still mainly learned on the (first) job and experience-based. Secondly, originally integrated into the architects' or designers' mission as an additional service, it becomes more and more obvious that architectural programming requires to be handled by a dedicated and trained specialist (team). Finally, the lack of "*neutral*" or core foundations through theoretical knowledge and community does not help its diffusion. These are main difficulties architectural programming has to face before trying to convince the clients of its usefulness. However, it seems that the clients represent the best leverage to further develop and establish the discipline.

1.3.3. Architectural Programming Approaches

The first architectural programming "*approach*" was the traditional one performed by clients: the **client-based architectural programming**. It resulted in a short list of spaces, requirements, relationships, completed with light explanations about the various purposes of the project. The intuition of the architects was enough to understand what was needed. In the mid-20th century, the complexity of buildings made it less effective and inadequate (Hershberger 1999). Since then, architects started to propose architectural programming as an additional service with their own approaches to achieve more reliable and valid programs. In his book, *Architectural Programming and Predesign Management*, (Hershberger 1999) presents four kinds of architectural programming approaches:

Design-based Architectural Programming: most of the programmatic efforts are deployed during the design process, only a minimum set of information is gathered upstream. This can be defined as a trial and error "*co-design*" process where the client refines his requirements while the architect makes various design schemes. It was the most used approach back in the days with a great success in terms of clients' satisfaction. A major drawback of this approach is the time and efforts spent by the architect to design the "*perfect*" building.

Knowledge-based Architectural Programming: this kind of approach takes benefit from research methods, tools, and techniques from social and behavioural science to define the brief. It aims to develop knowledge about the users through statistical analysis of a large amount of data gathered on them. This knowledge concerns the environmental needs of each user client groups. However, such focus on the user clients may tend to hinder the definition of architectural requirements (e.g. site, climate, economics, time, and technology). The planning, data collection and analysis also tend to be time consuming and costly for typical building projects (e.g. offices, houses, warehouse).

Agreement-based Architectural Programming: the programmer performs the architectural programming with a representative group of users (i.e. committee) on a limited set of programmatic information (i.e. the ones required by the designer) till a jointly accepted set of design requirements are defined. This is a focused approach. The "*problem seeking*" approach developed by the CRS of Houston is the most notable example. The limited set of information is represented by a programming matrix with five information areas: Goals, Facts, Concepts, Needs, and Problem statement, to decline regarding four considerations: Function, Form, Economy, and Time, each one of them being refined by three keywords. It represents a framework of 36 cards to inform. An information index is provided with more accurate keywords dedicated to each matrix card to seek out appropriate information (Pena & Parshall 2001). The agreement concerns the definition of "*brown sheets*" (i.e. activity sheets) regarding 24 recurring programmatic concepts that will lead the architect in his design process.

Value-based Architectural Programming: this approach is presented as a mix of the best aspects of the three above approaches (Hershberger 1999). Values and goals led the programmer through the briefing process. Values are key elements that should be addressed by the architect (i.e. main design issues). Hershberger identifies eight value areas (i.e. human, environmental, cultural, technological, temporal, economic, aesthetic, and safety) but recognizes that only a unique set of value is generally applicable to each project, giving more latitude than Pena's

problem seeking. Their prioritization depends on the client. Based on the most important values, a programming matrix is also used to decline them regarding four information areas quite similar to Pena's: Goals, Facts, Needs, and Ideas.

Other architectural programming approaches are identified in our literature review:

Issue-based Architectural Programming: Duerk develops her own approach that fills the gap between behavioural science and practice based on Pena's approach, quite alike Hershberger, but focused on the concept of design issues (Kumlin 1995). Her programming matrix is organized around five information categories: Facts, Values, Goals, Performance requirements, and Concepts, declined into an unlimited number of design issues (Duerk 1993). 15 principles complete her approach. Other attempts have been made using programming matrices with different viewpoints and main concerns (Palmer 1981) or simple checklists based on recurrent information (White 1986; Kumlin 1995).

User-based Architectural Programming: in France, other kinds of approaches are considered. Most of them are more related to social and human sciences making them quite similar to the knowledge-based architectural programming. In this case, user-based approaches consist of involving the user and customer clients as well as any concerned stakeholder (e.g. citizens or neighbourhood) in the briefing process at different degree. The intensity degree of the involvement can be evaluated on a nine rating scale from "*authoritarian regime*" to "*direct democracy*" (Dimeglio et al. 2005): 1-Information, 2-Communication, 3-Consultation, 4-Concertation, 5-Participation, 6-Co-decision, 7-Co-production, 8-Co-management, and 9-Self-management (based on Zetlaoui-Léger's professional training support, GEPA, PAMO, 2011). Examples of such approaches are the Generative Architectural Programming (Conan 1988; Conan 1989), the Generative and Participative Architectural Programming (Dimeglio 2001) and the Participative and Concerted Architectural Programming (Zetlaoui-Léger 2002; Dimeglio et al. 2005).

Other kind of architectural programming approaches can be found but with less details in the literature such as **Evidence-based Architectural Programming** (a.k.a. Progressive Programming) (Lang & Moleski 2010; The Practitioner's Resource 2014), or **Environmental Programming** (Farbstein 1976). The latest ones are related to behavioural and social sciences (Moleski & Goodrich 1972) but seems to differ from Knowledge-based Architectural Programming. The terms are only briefly evocated in the literature. Due to the lack of more accurate information about them, these approaches are not further presented nor considered in this research work.

Most of these approaches are formalized in books and guides intended for clients and practitioners. Section 1.3.4 presents a synthesis of the latest researchers' viewpoints on these books and guides in the literature.

1.3.4. Books & Guides

As expected through the various existing approaches, there is a plethora of guides on the briefing process (Gobin 2003; Shen et al. 2004; Yu et al. 2008; Certu 2010). The problem is that very few of them give enough details to be practically applied. They are considered too general and implicit (Duerk 1993; Barrett & Stanley 1999; Kamara et al. 2002; Yu et al. 2005), and unclear and inefficient (Hunter & Kelly 2003; Bogers et al. 2008; Hassanain & Juaim 2013). Their frameworks lack comprehensiveness (Barrett et al. 1999; Kamara et al. 2001; Yu et al. 2005; Shen & Chung 2006) whereas briefing practices are considered inadequate and highly limited by researchers (Green & Simister 1999; Kamara et al. 2002; Shen et al. 2004; Yahya et al. 2007; Yu et al. 2008). Most of the existing approaches have a solution-oriented thinking process (Kamara et al. 2002; Yahya et al. 2007) focused on the building. As a result, the defined requirements look more like solution-rationales instead of pure requirements guided by the client's business needs (Kamara et al. 1999).

Nevertheless, researchers made some proposals based on these observations using state-of-the-art tools and techniques more specifically by integrating computers and automation.

1.3.5. Computer-Aided Architectural Programming

Existing approaches formalized in books, guides, or standards are mostly "*paper-based*". Information gathered through interviews, group work, or any other technique is processed manually. It is then communicated to the clients (for validation) and the architects (for design) in a paper-based form, far from state-of-the-art practices in AEC. Considering the already established benefit of computer integration in AEC (Howard et al. 1989; Björk 1992b; Miyatake & Kangari 1993; Evbuomwan & Anumba 1996; Koutamanis 2013) , various ICT solutions for architectural programming have already been developed (Table 2) (Chung et al. 2009; Huovila & Hyvarinen 2012).

Table 2 – List of ICT solutions supporting architectural programming

ICT solutions	**References**
SEED-Pro	(Akin et al. 1994; Akin et al. 1995; Flemming & Snyder 1997)
CRPM	(Kamara et al. 1999; Kamara & Anumba 2001; Kamara et al. 2002)
Trelligence Affinity	(Ciscon et al. 2002; Ciscon et al. 2004)
PREMISS	(Kiviniemi et al. 2004; Kiviniemi 2005)
HospiTool	(Nykänen et al. 2008)
EcoProp	(VTT 2008)
Needscape	(VTT 2012)
ICT for Architectural Briefing	(Chung et al. 2009; Koutamanis 2013)
dRofus	http://www.drofus.no/en/product.html (dRofus 2013)
PTS	http://pts.lj.se/web/In_English.aspx

The computed information is based on current architectural programming practices and deliverables. It results in room data sheets, equipment lists, functional and technical programs centralized in a database or library of components. Depending on the software, a requirement checking module is generally included with, from time to time, an integrated conceptual design tool (giving basic shape and layout to rooms and surfaces). This information is mainly used by the architects or designers to facilitate the design, to list the required resources, or to support the requirements checking of proposals. However, the evolution of Building Information Modelling usages tends toward a (clients-oriented) structuration of the information to support facility management, decision making along the development process and information traceability.

What is missing in these ICT solutions is the trace of the programmer's own processing. Their input corresponds to the programmer's output. The input of the programmer (of the architectural programming) is a lot of information not directly related to the building (e.g. clients' purpose and objectives, business processes and activities…). Based on this kind of information, the programmer helps the clients to define building requirements (i.e. output of architectural programming). In fact, most of these ICT solutions are design-oriented rather than really focused on architectural programming.

1.3.6. In Summary

According to the literature, construction projects suffer from several quality, cost, and delay problems that directly impact on clients' satisfaction. An important part of them are related to difficulties that occurred during the briefing process (Allégret & Guilleux 2000; DGUHC 2000; Love & Li 2000; Yu et al. 2008; Feng & Tommelein 2008; Certu 2010; AQC 2013). In this first section, we captured the various difficulties related to architectural programming through its history and review.

Architectural programming is currently a highly experience-based activity full of tacit knowledge acquired through time. Best practices, approaches and methods are still poorly communicated for various raisons, including the fact that they represent a competitive advantage to professionals offering architectural programming services.

The contracting owner fully relies on the programmers' competencies and memory. Despite the massive amount of information to identify and communicate, the whole process is still mostly performed manually and in paper form by programmers. As a result, the overall coherence and

comprehensiveness often remains obscure and difficult, not to mention the time wasted in the process. A few research proposals explored computer-aided architectural programming with more or less success but these initiatives often focus on the building rather than the clients' business needs and operations. Nevertheless, ICT solutions require more attention (Koutamanis 2013).

1.4. Synthesis

Architectural programming, as any conceptual phase in engineering, is an essential step of construction project development. Based on a huge amount of fuzzy, evolving, and incomplete information, critical decisions have to be taken by a "*multi-headed*" client. Through its state-of-the-art, we show that there is still some progress to be made in terms of practice, research, and knowledge. Despite the huge variety of work on the topic, a few things remain unexplored, especially the relationship between high-level strategic paying clients' needs (i.e. business needs and operations) and detailed user and customer clients' requirements (i.e. building requirements) (Kiviniemi 2005). The focus of this research is, therefore, limited to the development of building requirements based on the paying clients' business needs (Figure 3). Based on these statements, Section 2 presents the industrial goals and objectives that guided the performed research.

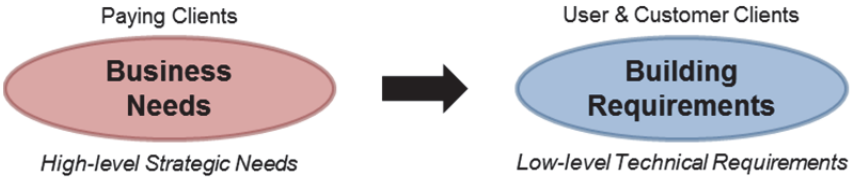


Figure 3 – From Paying Clients' Business Needs to User and Customer Clients' Building Requirements

2. Industrial Goal & Objectives

This section aims to define the "*tendering specification*" of this research. It presents the guidelines followed during four years to contribute to architectural programming in a useful and practical manner based on a set of practical issues. These industrial goals and objectives represent our understanding of architectural programming based on a literature review and on the field observation (interviews, practices, and professional training).

2.1. Scope

In this work, the industrial scope is limited to multi-users **public construction projects** in the frame of the French public contract law (JORF 2010). Construction projects such as dwelling houses or council estates are not considered in the scope. Specific difficulties arise from the French public contract law frame. The briefing process has to be **performed by the paying client** (or under his responsibility). The **design team cannot be involved** in it. An architectural design competition is generally organized to select the design team (mandatory above certain costs).

The **brief** is the **only document given** to the design teams to understand the clients' needs and make their proposal. This brief is a **"living" document** that evolves along the briefing process as well as during the design and construction phases. Despite perpetual changes, the brief has to ensure the **permanent consistency** between the paying clients' demand and the designers' proposal.

The competition rules ensure equity and anonymity of the participants during the competition by **limiting the interactions with the clients**. Therefore, the paying client is in a **difficult situation** in which he has to define a brief, evaluate architectural proposals, and then compare them without the specific background, knowledge, or competencies to do it at first and generally only **once in his lifetime**.

His **main knowledge** is often **limited** to a **partial understanding** of his own business processes and operations and a limited appreciation of the user and customer clients' activities and needs. Moreover, these user and customer **clients** are **not necessarily known or available** at

the time of the briefing process which makes the definition of their requirements more challenging.

2.2. Practical Issues

We focused on seven practical issues to ground this research (Figure 4):

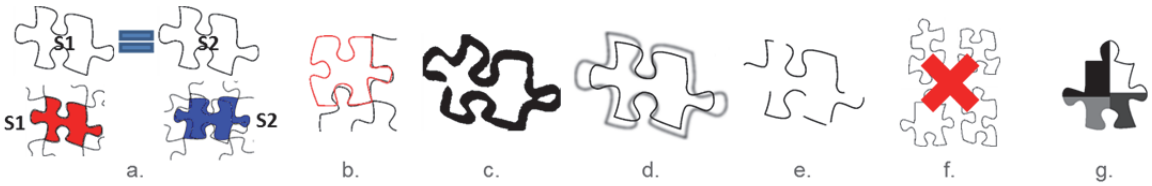


Figure 4 – Conceptualization of the seven focused issues of this research

a. A lot of briefs, especially for small public construction projects, are just scaled to the context **copy-paste** of previous existing solutions or standard briefs, based on the type of building (Figure 4.a). Unfortunately, the contextualization is limited to an adaptation of quantities (e.g. school building for x students) without real understanding of the local "*qualities*" which lead to unsatisfying solutions (based on empirical observations from the SC-Construct project).

b. **Conflicts** between requirements or even stakeholders' interests are identified late and hence are difficult to solve (Barrett & Stanley 1999; Tzortzopoulos et al. 2006) (Figure 4.b).

c. There is a **lack of accuracy** in requirements (Barrett & Stanley 1999; Tzortzopoulos et al. 2006) (Figure 4.c) making their validation or verification by the paying clients quite challenging.

d. There are a lot of **implicit requirements** (Barrett & Stanley 1999) mainly due to the lack of requirements documentation. The briefing process performed by the programmer remains quite obscure in the transition between the clients' business (purpose and operations) and the building requirements (i.e. his value added) (Figure 4.d).

e. Briefs **lack comprehensiveness** (Barrett & Stanley 1999; Kamara et al. 2002; Shen et al. 2004; Yu et al. 2007) mostly due to the time allocated for the briefing process but also due to its paper-based and synthesis nature (Figure 4.e). The brief only carries essential information to the designers.

f. Architectural programming practices suffer from an **absence of formalism** and **standardization** regarding their methods, tools, and techniques. Each project brings its own methodological approach. There is a serious lack of systemic and systematic practices compared to the other engineering domains. Both issues contribute to the absence of architectural programming as a discipline (Figure 4.f).

g. There is **lack of real and shared understanding of the clients' business needs**, objectives and operations (Kelly et al. 2002; Kamara et al. 2002; Yu et al. 2006; Kalay 2001) between the clients, the architects, and the contractors (Figure 4.g).

This research focuses on these seven issues because they are related to the content of the brief (i.e. information contained in the deliverable). This content is considered more related to the architectural programming practice and approach (i.e. methodology) rather than the human behaviour of the stakeholders. Human-related issues such as commitment of and to people (Egan 1998), communication between parties (Yahya et al. 2007; Tzortzopoulos et al. 2006), or collaboration (Kalay 2001), are not aimed to be solved or dealt with in here despite their recognized importance. Based on these seven practical issues, we propose to frame the research with a main practical goal and a set of industrial objectives.

2.3. Main Goal

All issues cannot be solved through one research project. Therefore, the main practical goal of this PhD research is scoped to **framing** and **supporting** the **functional part of public construction briefing process**. In the functional part, the building is just considered as a support to perform activities (Gobin 2003) (i.e. clients' business operations). We consider it as a **minimum to ensure** for public buildings. Moreover, the first thing design teams aim at is to produce a fully functional building that satisfy clients' expectations (Clift 1996; Oliveira et al. 2008). In this research, the "*non-functional*" parts refer to qualities of the building such as subjective information (e.g. aesthetic, feelings or impression, architectural image), political orientations (e.g. environmental impact, energy consumption, social considerations)... Project-related information (e.g. time, cost, management) is also not considered in the scope.

"Probably the most fundamental purpose of an architectural brief is to describe what takes place in the building to be designed (i.e. the activities of the users)." (Koutamanis 2013)

2.4. Industrial Objectives

This main goal refines into a set of ambitious practical objectives partially supported by the literature. These objectives have to be considered as guidelines or governing principles of this PhD research. Each one of them is a potential indicator of success or failure of the proposed contribution. Satisfying all of them at 100% is not the purpose but trying to address them as much as possible is.

Objective 1 – Help the paying, user, and customer clients to **express** their requirements (Tzortzopoulos et al. 2006; Yahya et al. 2007) in a shared local as well as global understanding: this expression should tackle **comprehensiveness** (Kelly et al. 2002; Kamara et al. 2002; Shen et al. 2004; Tzortzopoulos et al. 2006; Yu et al. 2007; Yahya et al. 2007), **relevancy**, and **reliability** (Kamara et al. 2002; Tzortzopoulos et al. 2006; Yu et al. 2007; Yahya et al. 2007; Feng et al. 2009) issues of requirements (Barrett & Stanley 1999).

Objective 2 – Support **analysis** and **processing** of the requirements: the **consistency** (Barrett & Stanley 1999; Tzortzopoulos et al. 2006) and **maturity** (Barrett & Stanley 1999; Kamara et al. 2002; Yu et al. 2007; Yahya et al. 2007; Tzortzopoulos et al. 2006; Feng et al. 2009) of requirements should be addressed. The **traceability** (Kamara et al. 2002; Gobin 2003; Certu 2010) and **dependencies** (Gobin 2003; Tzortzopoulos et al. 2006; Yahya et al. 2007) between them are required to support decision making along the briefing process.

Objective 3 – **Specify** the requirements so they can be **validated** by the clients and used/checked by the architects/designers by ensuring their **clarity** and **utility**. This specification should be supported by design state-of-the-art practices (i.e. Information and Communication Technologies ICT, Computer-Aided Design CAD, and Building Information Modelling BIM).

Objective 4 – **Verify** requirements implementation by the architects/designers in the design proposals. **Compliance** between the brief and the design proposals should be supported in an objective way. **Non-compliances** should be highlighted to support discussions between the paying, user, and customer clients, and architects/designers.

The stated objectives concern different steps of the briefing process. The idea is to make a contribution that could be valuable throughout the process so that effort at the beginning bear fruit later.

2.5. Synthesis

In this section, through the scope, practical issues, industrial goal and objectives, we aim to clarify the motivation, interest and prospective aspects of this applied research project. It lays the foundation for more research projects on architectural programming. The real fulfilment of each single objective would take a full PhD research project and additional resources. Integrating such industrial objectives provide a clearer and strategic long term view on the research to be

performed on this very specific topic. It also brings an insight into how and what can be transferred from science to the market, grounding the research in the real world.

3. Research Objective and Questions

Based on Sections 1 and 2, architectural programming refers to the process of translating information about clients' business needs into building requirements. This assumes that there is a link between high level strategic needs and low level technical requirements. In this dissertation, "*needs"* are considered as raw statements (usually in natural language) of what the clients want, desire or wish whereas "*requirements"* are considered as structured and formalized definitions of these wants, desires, and wishes. *"High-level strategic"* refers to information about the paying clients' business (i.e. domain information, business-oriented) whereas *"low-level technical"* refers to building information (i.e. building-oriented). In this section, the research objectives and the various questions it raises are introduced based on the statements and conclusions from Section 1 and Section 2.

3.1. Research Problem Statement

In Architecture-Engineering-Construction, the transition from business needs to building requirements is supported by the programmer's experience. Regarding the state-of-the-art, current approaches can be characterized by arrow (1) in Figure 5. Business needs are translated into space needs (i.e. low-level needs) then formalised into building requirements. These current approaches are rather solution-oriented with little and late formalisation of the transition from high-level needs to low-level requirements.

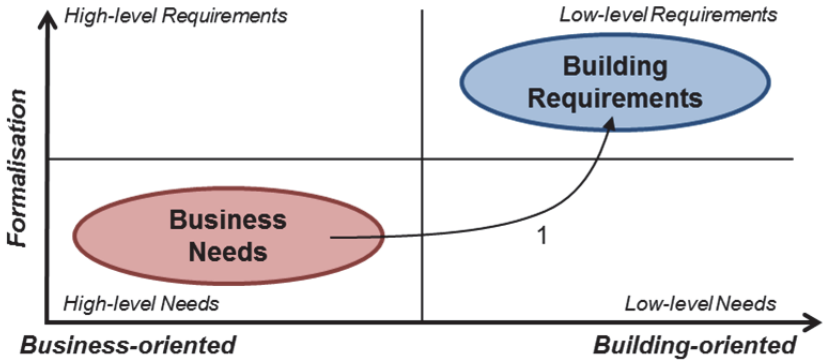


Figure 5 – Representation of the Transition from Business Needs to Building Requirements regarding the Current Situation

On the other hand, the transition between high level strategic needs and low level technical requirements is better traced and formalized in other engineering domains. In Mechanical Engineering, high level needs are formalized through **functions** detailed by product requirements. In Software Engineering, needs are formalized through **goals** operationalized by software requirements. In Industrial Engineering, needs are formalized through **processes** and **activities** then translated into resource requirements. Functions, goals, processes, and activities are high-level requirements, i.e. a high-level formalisation of their respective system requirements. They are rather abstract which delay the reasoning about solutions.

Their requirements definition process is fully supported by a set of approaches, methods, tools, and techniques. Automation is already set from clients' needs to process variables, passing through requirements and design parameters. Engineers' education has also been changed to integrate such contributions. In architects' education, for various reasons, the situation did not change so much since the 80's despite several attempts to integrate or develop architectural programming.

The starting point of architectural programming, as well as its object of interest, is different from other engineering domains. The stakeholders and their backgrounds are also totally different.

As a result, their scientific approaches, methods, tools, and techniques cannot be directly or strictly applied without certain adjustments and precautions. Our assumption is that architectural programming can still benefit from this knowledge but a specific reasoning has to be developed.

3.2. Research Objective & Questions

The research objective of this project is to develop an architectural programming framework based on existing knowledge from engineering disciplines. This framework should explain the **relation** between clients' business **needs** and building **requirements**, and the **formal transitions** from **high-level needs** to **low-level requirements**. The proposed framework aims towards an **early formalisation** of the needs into requirements (arrow (2) in Figure 6).

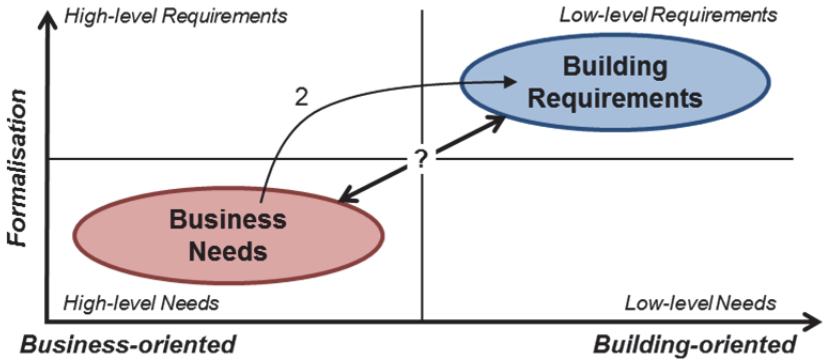


Figure 6 – Illustration of the Research Objective

To achieve such a research objective, three research questions structure the project:

RQ1: Which constructs are required to describe the building and its definition domain (i.e. high-level and low-level needs and requirements)?

RQ2: How to model the building and its definition domain according to the multiple (views) stakeholders involved?

RQ3: How to derive building requirements from business needs?

The first question refers to the information required to describe the object of study. The second question arises from the perimeter of the object of study and the consistency of the various viewpoints that can be taken on it. The last question asks for details about the development of building requirements starting from clients' business needs.

3.3. Synthesis

The research problem tackled in this PhD work focuses on the transfer of knowledge from multiple engineering domains to AEC domain regarding the requirements definition of systems. There is a cultural gap between these domains, to the detriment of the paying, user, and customer clients. Architectural programming could learn from the requirements definition of various engineering domains to reduce this gap for the benefit of the clients. The will is somehow to bring architectural programming and requirements engineering closer to produce synergies and knowledge transfer, mainly from engineering domains towards architecture.

4. Methodology

This research is based on a specific approach and perception of architectural programming. It is grounded on an engineering background and observations of similarities across Architecture-Engineering-Construction (AEC) and other engineering disciplines. It resulted in the identification of potential transfers of knowledge from Mechanical, Industrial, and Software Engineering to AEC in order to support architectural programming. This mind-set leads the research outcomes. This approach is supported by a methodology based on Design Research and Design Science.

Research methodology is about how the research is performed, why it is performed that way, and what is involved in it. The answers to these questions aim to make people understand how the research is designed. This section presents answers about these questions leading to this research methodology by first introducing the data collection (Section 4.1), then presenting the followed design science framework (Section 4.2) and the design research methodology used (Section 4.3). This is followed by a presentation of the four case studies used to develop and illustrate this research (Section 4.4). The section concludes with the transdisciplinary viewpoint taken during this research (Section 4.5).

4.1. Data Collection

When reading about methodology, two main types can be found: quantitative and qualitative research. This mainly refers to the collected data and their analysis. Quantitative research deals with numbers and statistics whereas qualitative research deals with descriptions, interpretations, and observations. Both research methodologies have their own advantages and drawbacks. As a result, it is quite often that both of them are used in a research project to balance each other.

In this project, drawbacks of quantitative research were judged too risky. The return on (time) investment was estimated definitely too low. This is mainly explained by the context: lack of experience in architectural programming per se by the host institutions, the supervisors, and the applicant, as well as a poor maturity level of Luxembourg regarding architectural programming. This assumption was confirmed by a quantitative survey performed in parallel by another team at CRP Henri Tudor (Guerriero & Fry 2014). It took more than one year from the survey definition to a partial result publication with mitigated conclusions. This survey is still interesting but requires deepest investigations taking the form of interviews (i.e. qualitative research). As a consequence, only qualitative research was performed. It implies limits but it is sufficient for the first exploratory research on the topic aiming toward practical solutions. Future research includes quantitative surveys based on the first results presented in this dissertation.

On the other hand, when looking at research methods, the range of alternatives is wider: experiments, surveys, field studies, action research, aggregation research, etc. Research methods are concerned with the way data are collected and research is performed. Several research methods were used to gather data at different points in time with different purposes. Indeed, this research is based on the results from surveys on architectural programming already performed around the world (Allégret et al. 2005a; Yu 2007; Juaim & Hassanain 2011), field studies in Luxembourg performed in the frame of the SC-Construct project at CRP Henri Tudor from 2008 to nowadays (Harbouche et al. 2008; Mauger et al. 2010), aggregation research, brief (i.e. architectural programming deliverable) content analysis, and professional trainings on architectural programming in France and Luxembourg.

4.2. A Design Science Framework

This research is performed following a design science framework. Among those available, March and Smith's framework is adopted (March & Smith 1995). Their framework is structured around two axes with a distinction between design science and natural science (Table 3). Natural science is about how things are whereas design is about how things ought to be, with devising artefacts to attain goals (Simon 1996). Their first axis concerns the research activities: build and evaluate on the design science side, and theorize (or discover) and justify on the natural science side:

> *"Build refers to the construction of a design artefact and aims at demonstrating that such an artefact can be constructed. Evaluate refers to the development of criteria and the assessment of the artefact's performance against those criteria. […] Discover […] refers to the construction of theories that explain how and why something happens. […] Justify refers to theory proving."*
> (March & Smith 1995)

Their second axis concerns the research outputs: construct, model, method, and instantiation. Constructs or concepts define the vocabulary (i.e. language) of the studied domain for both problem description and solution specification. A model states relationships among constructs. It

is used to represent problems as well as solution. A method is based on constructs and models in the studied domain. It is composed of successive steps used to perform tasks leading to solve a problem. An instantiation is the concrete creation of an artefact in its "*physical*" environment. It makes constructs, models, and methods operational.

Table 3 – Design Science Framework (March & Smith 1995)

		Research Activities			
		Build	**Evaluate**	**Theorize**	**Justify**
Research Outputs	**Constructs**				
	Model				
	Method				
	Instantiation				

Regarding the industrial objectives (Section 2) and research problem statement (Section 3), this research addresses the creation of new artefacts (design science) for the AEC domain to achieve the main goal of partially supporting the briefing process. Moreover, this PhD research addresses the identification of research trails regarding these statements and objectives. As a result, March and Smith framework was limited in this research to the build activity regarding the four kinds of design artefacts: constructs, models, methods, and instantiation.

As this PhD topic is highly forward looking, a breadth-first search is chosen to ensure the continuity and valorisation of the research contributions. It is mainly due to the research context and its novelty in both hosting institutions. A depth-first search would have been riskier regarding its "*durability*". Based on these first results, other research projects can be developed to deepen each identified trails in a coordinated effort.

4.3. A Design Research Methodology

Design Research Methodology (DRM) is introduced by Blessing and Chakrabarti as a help to make design research more effective and efficient (Blessing & Chakrabarti 2009). Four stages compose the methodology: Research Clarification (RC), Descriptive Study I (DS-I), Prescriptive Study (PS), and Descriptive Study II (DS-II). DRM is presented as an iterative process with many possible variations.

To sum up DRM, the Research Clarification stage aims to clarify the knowledge on the domain of study and the initial situation in order to formulate a realistic and worthwhile research goal. The Descriptive Study I stage aims to describe the existing situation (AS-IS) whereas Prescriptive Study aims to describe a desired situation (TO-BE) and propose to improve the AS-IS situation. The Descriptive Study II investigates the impact of and ability to apply results of the Prescriptive Study. Seven types of research are identified within DRM, covering various combinations of two to four of the stages with iterations in some cases (Blessing & Chakrabarti 2009). The methodology is completed with various diagrams, models and sub-steps for guidance but they are not presented in this research.

This research project can be considered as a Type 3 including a review-based Research Clarification, a review-based Descriptive Study I, a comprehensive (i.e. literature review and support development) Prescriptive Study, and an initial Descriptive Study II (Figure 7).

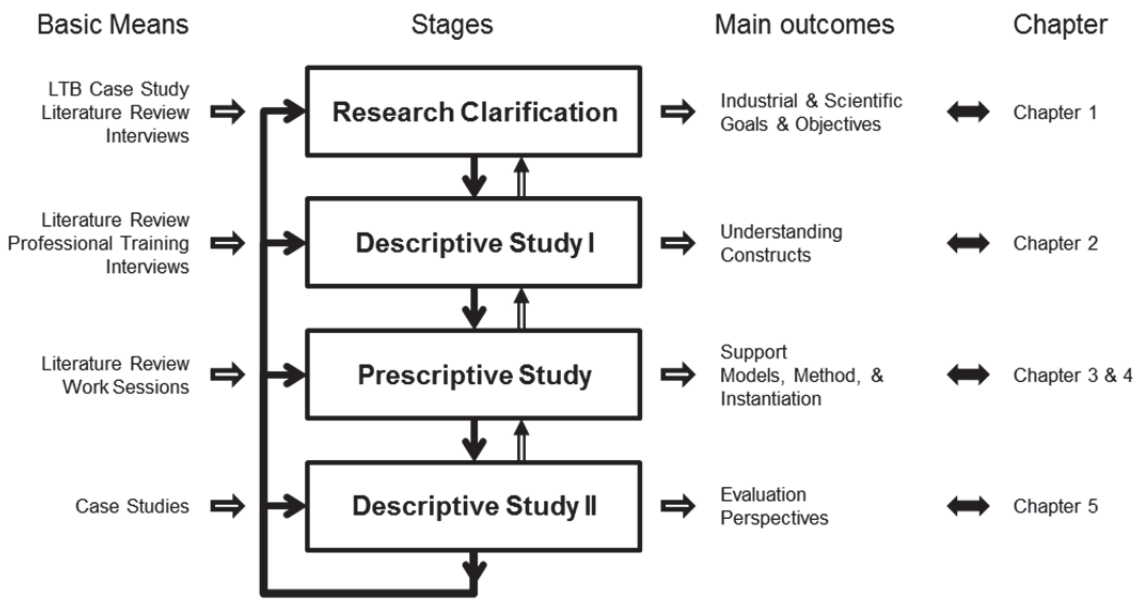


Figure 7 – DRM framework adapted from (Blessing & Chakrabarti 2009)

Research Clarification: Data collection was based on the support of a real architectural programming case study during the Master thesis, upstream of the PhD. Observation and participation in the process (i.e. field study) made me highlight issues and difficulties in gathering the requirements needed to define the future building and organization. This stage was enriched through a literature review and a set of interviews with professionals. The output of this stage can be read in Sections 2 and 3 of Chapter I.

Descriptive Study-I: Data collection was performed through an extensive literature review on architectural programming (more specifically on guides and practices), through attending to a professional training on architectural programming, and further interviews with professionals. The output of this stage is summed up in Section 1 of Chapter I as well as in the Chapter II (i.e. constructs).

Prescriptive Study: Data collection was performed through a wider literature review on conceptual design in Mechanical, Industrial, and Software Engineering, completed with work sessions with experts from each discipline. Chapters II, III, and IV presents the results of this Prescriptive Study taking the shape of design artefacts (i.e. models, methods, and instantiation).

Descriptive Study-II was limited to an overview of fragmented practical applications of the design artefacts. The practical application was based on information available from the three case studies. It used existing software tools to sketch instantiations of the design artefacts. This study is highly limited but its output provides sufficient elements to define research perspectives. Those perspectives are gathered in Chapter V.

4.4. Case Studies

As the availability and duration of public construction projects (from 3 to… many years depending on its complexity) are not compatible with the time frame and duration of a PhD thesis (3 to 4 years), this research project was based on three ex post facto case studies and an extra on going one: a Luxembourgish secondary school, a French multimedia library, a Canadian synagogue kitchen, and a Luxembourgish university library. Figure 8 represents the available deliverables used as source of information for each case study.

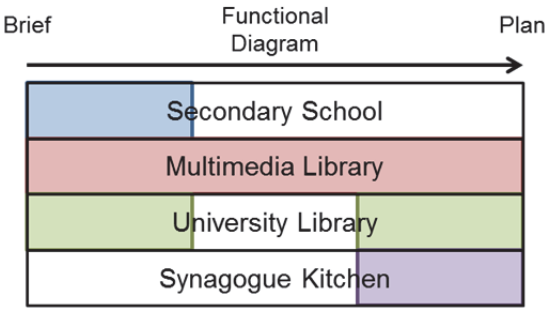


Figure 8 – Sources of Information for Each Case Study

4.4.1. Secondary School

The very first case study that motivated this research is the refurbishment and extension of a secondary school in Luxembourg (# 10,000,000€ project). This school is particularly interesting because of the major role of the psychological services and educational guidance (SPOS) and the irrelevance of the current building according to the school objectives and pupils profiles. This project is concerned by a poor occupancy rate, redundancy, and lack of adaptability and flexibility of the current spaces. Requirements are restrained to the secondary school business process and objectives in order to evaluate the impact of new pedagogic methods on building requirements.

One of the main issues in this case study is to manage the huge amount of unstructured, fuzzy, and ephemeral information. The output of this case study is the first framework that structured information regarding three main concepts: processes and activities, services delivered by the building, and spaces and their characteristics. This structure allows linking pedagogical concepts to requirements on the building. However, the proposed structure is only a draft reworked in the frame of this PhD research.

The main transcription of the case study in this research focuses on a particular group work session. This session is about the safety of pupils inside the school regarding issues with bullying and "*lawless areas*" in the previous school. The working group is composed of 5 teachers, the director and his superintendent. Two colleagues deal with the group session. I am involved in the session as an observer and technical support.

4.4.2. Multimedia Library

The second case study is a multimedia library built in 2008 in France (# 2,000,000€ project). The briefing process was performed in 2003, the building was built in 2008, and it has been used since 2011. Information on the case study comes mainly from the brief, interviews with the multimedia library director, and architectural design competition documents provided by the city hall, paying client of the project. The main issue developed through this case study focuses on the functional diagram, the modelling of the brief content, and the comparison of the design proposals.

This case study is used to develop the contribution on two phases of architectural programming: the brief synthesis into a functional diagram (from text to abstract graphical model), and the architectural design competition (from abstract graphical model to explicit representation). It provides information on quite a high level of the construction project, mainly the relationship between functional spaces regarding the viewpoint of cultural materials and their life cycle in the multimedia library.

4.4.3. Synagogue Kitchen

The third case study is the kitchen of a Synagogue in Canada (# 50,000€ project). This case study provides more detailed information about a single functional space. The focus is on understanding activities performed by the Rabbi inside this space and their impact on functional requirements definition regarding the layout, equipment and furniture. The main issue addressed

in this case study is the lack of requirements engineering performed before the design of a kitchen (Mauger & Berry 2013). An analysis of the plan (i.e. 2D drawing) proposed by the architect reveals a set of practical issues underlining the absence of proper requirements definition. The aim is to show how Requirements Engineering techniques such as use cases and scenarios could support the requirements definition of a kitchen (Mauger & Berry 2014). Details on the case study are provided in Appendix J.

4.4.4. University Library

The fourth case study is a university library which building is under construction (# 60,000,000€ project). This library is part of a more complex project relative to the whole university. The project aims to transform a brownfield into a new dynamic urban area. This is an on-going project started in the early 2000s'. The library is under construction and should be opened in 2017.

The issue dealt with in the frame of this research concerns the brief-project alignment. It focuses on the distribution of book collections inside the designed building. The design, the position of furniture, and the storage capacity constrain the distribution. The requirements to satisfy include information about: the classification system (i.e. Dewey), the relationships between the classifications, the documentary policy, and the quantity of documents to store. The library represents a collection of around 142.000 physical documents (directly accessible, i.e. without considering the storehouse) in 2013 that will grow to a critical mass of around 160.000 physical documents in 2018. The aim is to organize the collection in the building based on the briefing information and regarding the design constraints. Details are provided in Appendix J.

4.4.5. Synthesis

Each case study provides different kinds of information through different scales in terms of construction project as well as in terms of briefing process and information. This allows me to show that the contributions cover a large part of the briefing process, from high-level (i.e. global, strategic) needs to low-level (i.e. local, operational) requirements. However, regarding the amount of information that defines a building and the limited resources on this research project, only a fraction of each case study was used to illustrate and validate the feasibility of the contributions.

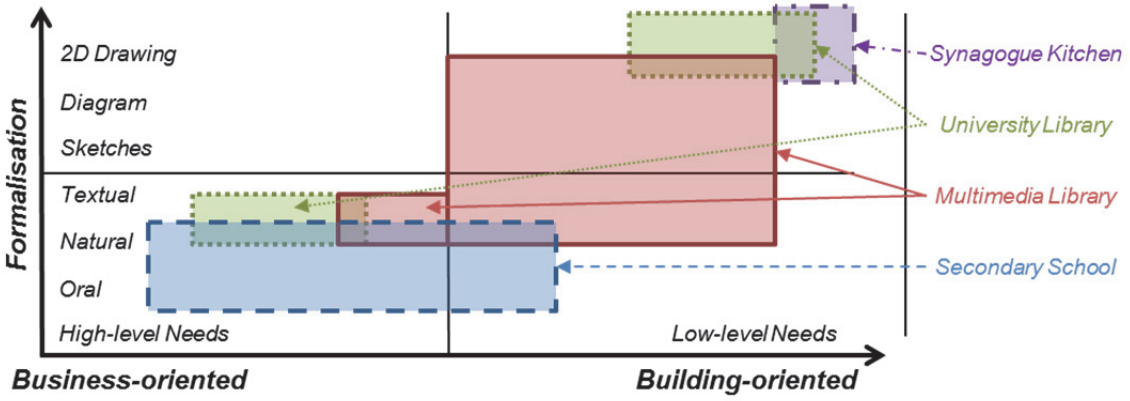


Figure 9 – Information Coverage Synthesis of Case Studies

Figure 9 illustrates the positioning of information used in case studies depending on their source: oral (i.e. collected during workshop), natural (i.e. structured by interviews), textual (i.e. extracted from the brief), sketches (i.e. raw graphical synthesis), diagram (i.e. structured in graphical synthesis), and drawings (i.e. extracted from the architects 2D drawings). A higher level validation is thus required in the perspective.

4.5. Transdisciplinary Research

Construction is already considered as a complex system (Bertelsen 2003) especially in its construction phase (Kubicki et al. 2006). The Built Environment is already accepted as a multidisciplinary field (Kalay 2001). For a while, it has been influenced by Management developing a strong Management-led crossdisciplinarity and now it calls for interdisciplinarity (Chynoweth 2006). Urban studies have already been subjected to transdisciplinarity on research experiences in France and Canada (Ramadier 2004).

The research in this project goes further in this direction by proposing transdisciplinary research (i.e. through the following disciplines: Mechanical Engineering, Industrial Engineering, Software Engineering, and Architecture-Engineering-Construction) about another phase of the Built Environment: architectural programming (i.e. requirements definition part of buildings' conceptual design phase).

> *"Transdisciplinary research is research that includes' cooperation within the scientific community and a debate between research and the society at large. Transdisciplinary research therefore transgresses boundaries between scientific disciplines and between science and other societal fields and includes deliberation about facts, practices and values."*
> (Hadorn et al. 2008)

Each engineering domain has its own specificities. They guide and constrain the conceptual design phase in terms of practices as well as theories. They dictate how to proceed regarding the object of study/design and often make people enclose themselves inside a box. This box represents the acceptable limits of their associated domain in terms of thinking. These boxes are more or less filled with ideas, theories and results of their own. Instead of focusing on their specificities, this research directly aims at their similarities. These similarities, escorted by a set of approximations, make boundary transgressions possible. In this research, only four engineering domains are considered: mechanical engineering, industrial engineering, software engineering, and architecture-engineering-construction. Addition of further scientific disciplines is considered as a perspective.

This research identifies similarities about the system development process in each domain, issues regarding the conceptual design phase, and solutions developed to support/improve the requirements definition activity (i.e. concepts, models, theories, methods, tools, and techniques). Indeed, the conceptual phase of engineering projects is recognized as the most important phase of the system development process and life-cycle. It provides the biggest leverage for project improvement in terms of costs and ability to influence the project over the time (Abdul-Kadir & Price 1995; Boehm & Papaccio 1988; Hsu 2000; Wang et al. 2002). However, it seems to be the least understood, especially in Architecture-Engineering-Construction (AEC) (Macmillan et al. 2001).

As an example, architectural programming presents interesting analogies with the domain of software development and its associated discipline called Software Engineering (IEEE Computer Society 2005). The main focus of this discipline, which emerged in 1968, is about the design, development and maintenance of software. During the years, it has been identified that one of the main complexities of software development is to design software meeting customers' needs. Studies such as the Standish Group (reviewed in (Anton 2003)) have shown that about 40% to 60% of software project failures were due to a bad understanding of stakeholders' expectations and of the properties of the software's environment (e.g. business and legal). This led in the mid 80's to the emergence of the Requirements Engineering (RE) discipline (Theyer & Dorfman 1990) considered as a very distinct sub-field from Software Engineering focusing on the problem domain and its associated requirements rather than on the software solution. Today, Requirements Engineering has developed techniques, models, methods and tools for supporting the elicitation of requirements, their formalisation and organisation as well as their management. Of a particular interest are frameworks associated with Goal Oriented Requirements Engineering (GORE) (Lamsweerde 2009) which highlight the systemic view needed when approaching the elicitation of requirements.

We believe that architectural programming is following, or has to follow, the same kind of path. As solutions exist in other disciplines, we think that architectural programming can learn from them to speed up the process and to avoid reinventing the wheel. As a result, in each chapter of this dissertation, some references to other disciplines than Architecture-Engineering-Construction (AEC) are used to support contributions in terms of feasibility or value added. These other disciplines are limited to Mechanical Engineering, Industrial Engineering, and Software Engineering. In particular, Human and Social Sciences are not considered in the scope of this research but could be considered in the perspectives. Following the same spirit of forward looking research, proper scientific validation of the contributions is kept for the research perspectives.

5. Layout of the dissertation

The thesis dissertation is organized in five distinct chapters (Figure 10 – Table 4). Chapter I introduces the background of the research performed during these four years on architectural programming and more generally on system requirements definition. It provides the scope, guidelines, mind-set, and methodology followed to reach the defined research as well as industrial objectives. The three research questions resulting from the state-of-the-art on architectural programming structure the rest of this dissertation.

Chapter II focuses on the first research question about the definition domain of architectural programming. It introduces the object of study by revising the concepts (i.e. constructs) required to define and describe it. The proposed constructs are defined and interrelated based on a multidisciplinary overview of conceptual design methods.

Chapter III answers the second research question about the modelling of building requirements in two main parts. The first part focuses on conceptual models to define the perimeter of the object of study. The second part develops operational models to practically support the conceptual models and to represent the building requirements according to the multiple viewpoints on the object of study.

Chapter IV is based on and illustrates contributions from Chapters II and III using the case studies. It is divided in three main parts that address the third research question. The first part presents the framework (i.e. method) proposed to derive building requirements from business needs. This framework is based on the conceptual models and supported by the operational models. The second part introduces instantiations of the framework by proposing a set of model views of its theoretical application on a briefing process using case studies data. The third part adapts the proposed framework from the briefing process to the assessment of proposals.

Chapter V concludes this thesis dissertation with two viewpoints on the contributions. The first viewpoint presents the research conclusions and perspectives regarding the research objective and questions. The second viewpoint presents the industrial conclusions and perspectives according to the industrial goal and objectives.

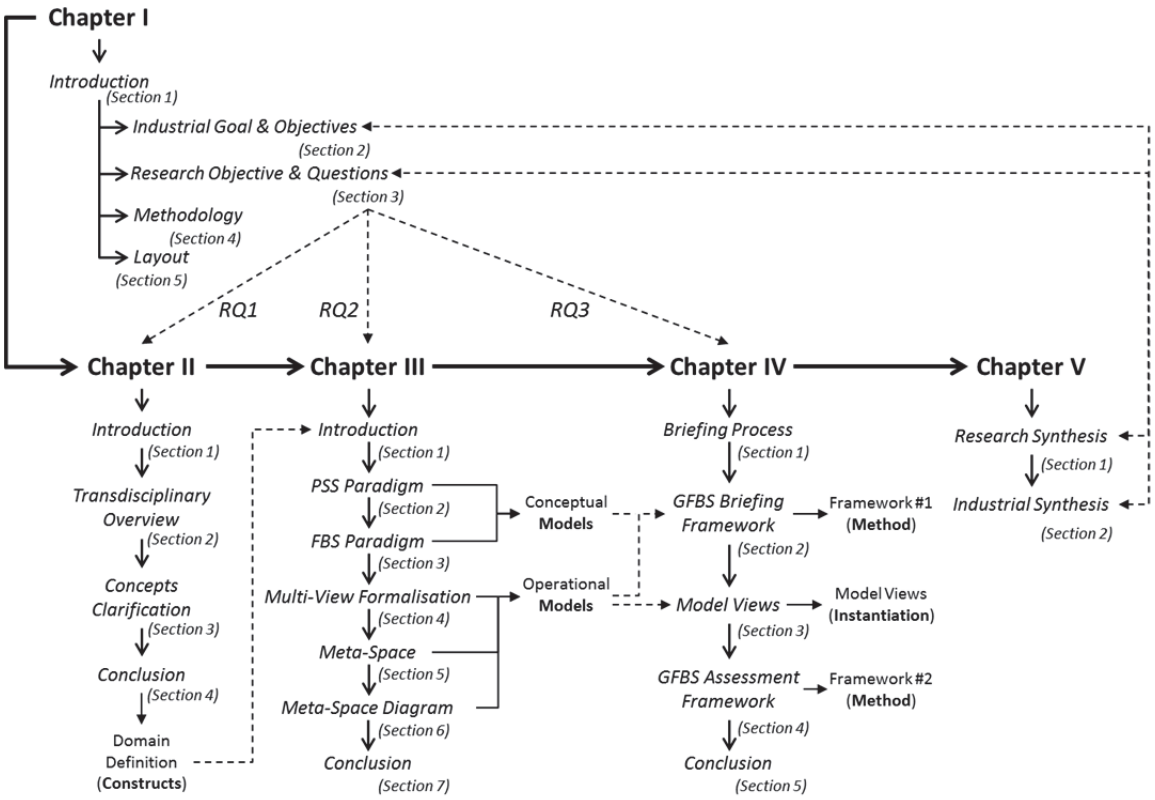


Figure 10 – Dissertation Layout & Relationships between Sections

Table 4 – Dissertation Layout & Design Artefacts Synthesis

Research Outputs	Research Activities: Build
Constructs *Chapter II*	*Definition Domain*
Models *Chapter III*	*Conceptual Model* / *Operational Model*
Method *Chapter IV*	*Frameworks*
Instantiation *Chapter IV*	*Process Model Views*

Chapter II – DESCRIBE A BUILDING

As stated in the Chapter I, architectural programming lacks scientific theoretical foundations. It is mainly an experience-based activity that improves through practice and time. Chapter II aims to define a set of constructs (i.e. concepts) to structure the information required to describe a building in the scope of this research work (i.e. functional brief).

This chapter proposes a set of constructs and their relationships in order to define and describe the building requirements and its domain. It should be noted that the degree of details of the constructs is limited to a conceptual level (i.e. abstract classes). The aim of this research is to structure this abstract level. Information such as the kinds of architectural spaces (e.g. room, stairs, floor) are not part of this prospective research but should be addressed in the perspectives. Section 1 introduces the structure of Chapter II through a clarification of the first research question and a short literature review of existing solutions. Section 2 proposes a transdisciplinary overview on system development processes to identify the main concepts used to describe an object of study in its early phases. Based on a selection of concepts structured by primitive interrogatives, Section 3 presents a multidisciplinary analysis of the used concepts and their definitions across engineering before proposing a set of consistent definitions adapted to our object of study. The constructs and their relationship are used to form the definition domain. Section 4 concludes this chapter on the definition domain resulting from this taxonomy.

In this chapter, a set of conceptual models are used to graphically represent the definitions and taxonomy of each concept. To help the reader, the graphical convention used for the conceptual models is presented in Figure 11.

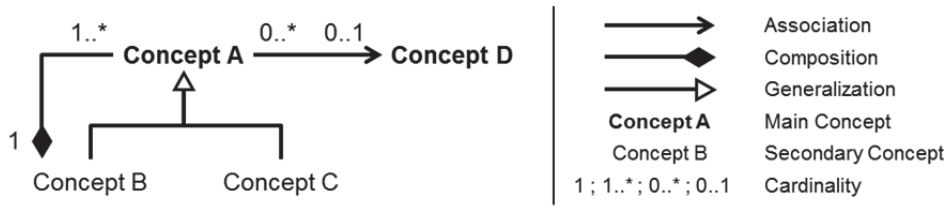


Figure 11 – Graphical Convention used in Chapter II

Figure 11 should be read this way: "*The main concept A is a generalization of concept B and concept C. Respectively, secondary concepts B and C are specializations of the main concept A. Concept B is composed of (at least) one to several concepts A. Therefore, concept B can be composed of concepts B and/or C. The main concept A is optionally linked to maximum one main concept D. The main concept D is optionally linked to several main concepts A*".

1. Introduction

This first section introduces a few things about this chapter. The research question tackled in this chapter is: "what constructs are required to define and describe the building requirements and its domain?" To answer this question, we first define the terms construct, the domain of interest, and the object of study in the frame of this research. Then, we propose a short literature review of artefact describing buildings and their limitation regarding this research. Finally, we revisit the methodological elements governing this chapter based on these clarifications.

1.1. Clarification of RQ1

*What **constructs** are required to define and describe the building requirements and its domain?*

Constructs correspond to the first created design artefact following the design science framework defined by March and Smith. It represents the main contribution of this chapter. This output will be used as a basis to build the next chapters and design artefacts (i.e. models, method, and instantiations).

"Constructs or concepts form the vocabulary of a domain […] used to describe problems within the domain and to specify their solutions." (March & Smith 1995)

Regarding the research question, the domain of interest in this chapter is the briefing process, and the object of study is a building. Therefore, the aim is to identify the vocabulary used, or rather required, to define first buildings and their related requirements, then to further describe the tackled problems and finally to specify their solution.

The first construct used in the briefing process or more generally during the requirements definition is the term *requirement* followed by the term *specification*.

A *requirement* is defined as a condition or capability that must be met or possessed by a system, component, product, service, or result to satisfy the clients' needs (IEEE 2010).

A *specification* is "*a document that fully describes a design element or its interfaces in terms of requirements (functional, performance, constraints, and design characteristics) and the qualification conditions and procedures for each requirement*" (IEEE 2005; IEEE 2010).

In this research, *Requirement* is a general concept that includes all the information about the object of study during the definition of the clients' expectation. *Specification* goes further and focuses on the object of study during its design. *Requirements* are client-oriented whereas *specifications* are designer-oriented. Therefore, all the constructs defined in this chapter that describe the object of the study are considered as *requirements*. The *specification* is not included in this chapter.

1.2. Literature Review

According to guides and best practices, the vocabulary used by programmers during the briefing process should suit with the stakeholders' understanding (DGUHC 2000). However, an initial vocabulary is required for the programmer to understand what should be defined during the briefing process. This vocabulary can be used by the programmer and later translated in the stakeholders' language through the brief.

This brief should define: objectives of the operation, needs to satisfy, constraints and requirements in terms of social, urban, functional, technical and economical, landscape integration and environmental protection qualities regarding the construction and usage of the building (JORF 2007). According to this statement and regarding the literature on architectural programming, a consequent number of concepts appear: Goal, Function, Activity, Space, Cost/Economy, Time, Values (human, environmental, cultural, technological, temporal, economic, aesthetic, and safety), Fact, Needs, Ideas, Problem Statement, Form, Concepts, Performance Requirements, Issues, etc. This research focuses on and is limited to concepts related to the functional quality of a building. The functional quality in this research refers to the ability of the building to satisfy clients' business operations and processes.

Research into Architecture-Engineering-Construction proposed ontologies (El-Diraby et al. 2005; El-Gohary & El-Diraby 2011) but they mainly focus on project information and the construction process of a building rather than the building itself. Other initiatives focus on the building (Peachavanish et al. 2006) but they are limited to its design, i.e. ignoring the usage of the building, an important information in architectural programming.

In Computer-Aided Design (CAD), there are the Industry Foundation Classes (IFC) (buildingSMART 2013). It is a data model that intends to describe building and construction industry data. The data covers the object definition (i.e. tangible object occurrences and types), the relationships among objects, and their properties. The objects are divided into six fundamental concepts answering six basic interrogatives: actors (Who), controls (Why), groups (What), products (Where), processes (When), and resources (How). These concepts are used to describe the building as an object (i.e. product) and the construction process (i.e. project and facility management). Information concerning the briefing process is limited to building requirements (i.e. the output of the briefing process). It does not include the usage and the relation between the paying clients' needs (i.e. high-level strategic needs) and the client users' and customers' requirements (i.e. low level building requirements) (Kiviniemi 2005). Even though some of these concepts already address part of the problems defined in this chapter, the view point taken to describe the building is slightly different. Current IFC is defined with a design point of view on the building whereas what we are looking for is a briefing point of view. The differences will be provided through Sections 2 and 3 of this chapter (i.e. meaning of the interrogatives and relative concepts).

1.3. Methodological Conclusion

In order to identify the main concepts required to describe a building, a problem domain analysis of conceptual design methods across engineering disciplines is proposed in Section 2. This problem domain analysis takes the form of a transdisciplinary overview. Based on a corpus of conceptual design methods from various engineering domains, a set of concepts was identified and positioned regarding a system development process. An extensive literature review on architectural programming for AEC and conceptual design methods in other engineering disciplines was used to constitute the corpus.

2. Multidisciplinary Overview of System Development Processes

Architectural programming corresponds to the requirements definition of conceptual design in other engineering domains. This section presents an overview of upstream concepts used by conceptual design methods in various engineering disciplines. In Chapter I, we saw that there is a lack of structure in the transition from business needs to space needs (mainly experience-based) in architectural programming. The conceptual phase of other engineering disciplines was therefore explored to identify what describes or characterizes their own object of study. The object of study is named "*system*" in this section.

The first analysis performed is a multidisciplinary analysis. It focuses on a limited set of conceptual design methods in Mechanical, Industrial, and Software Engineering. Most of them focus on requirements definition or processing. They can be considered as seminal in their respective domain. Each conceptual design method is first analysed to identify the concepts used to describe or characterize their respective system. Table 5 gathers the conceptual design methods that were analysed.

Table 5 – Corpus of Conceptual Design Methods Analysed

Main Discipline	Conceptual Design Method	References
Mechanical Engineering	*Problem Solving Technical System*	(Pahl & Beitz 1995)
	Quality Function Deployment	(Akao 1993)
	House of Quality	(Prasad 1998)
	Axiomatic Design	(Suh 2001)
	LCPN Approach	(Aoussat & Le Coq 1998)
	Value Analysis	(Miles 1972)
Software Engineering	*V-Model*	(IABG 1997)
	Requirements Engineering	(Glinz 2011)
	GORE	(Lamsweerde 2001)
Industrial Engineering	*CIMOSA*	(Vernadat 1996b)
	System Engineering	(Faisandier 2011)

The set of concepts from the various conceptual design methods are then positioned on a generic system development process using the description of the methods and, when possible, the proposed development process attached to the methods (Figure 12 and Appendix A for higher scale). The alignment of concepts is also based on the meaning of the concepts in the context of each method. The detail analysis of each conceptual design method is not part of this dissertation. Only the result of the analyses is presented. Similar comparison analyses are performed regarding various design processes (Blessing 1996; Scaravetti 2004; Tomiyama et al. 2009; Gericke et al. 2013), functional modelling approaches (Eisenbart et al. 2013), or business process modelling techniques (Aguilar-Savén 2004) with slightly different corpuses.

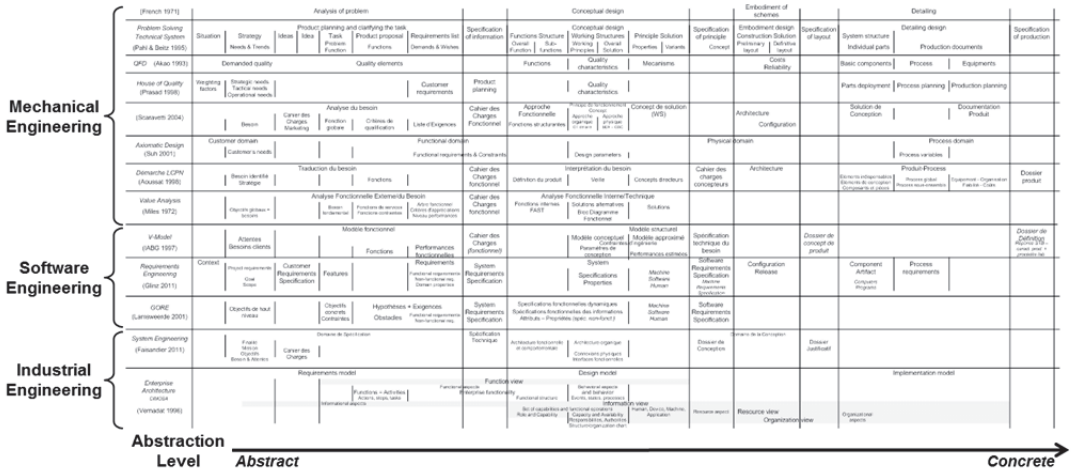


Figure 12 – Positioning of the Concepts used by Conceptual Design Methods to Describe a System (Appendix A)

Based on the first analysis and positioning of mechanical, industrial, and software engineering concepts, a set of architectural programming approaches were analysed in the same way. The set of AEC approaches (Table 6) represent various aspects of architectural programming state-the-of-art from practical approaches recognised (Pena & Parshall 2001) or still applied (Grandjean 1980) by programmers, a few approaches from research (Kamara et al. 2002; Gobin 2003; Yu 2007), two French guides (MIQCP 2001; Certu 2010), and an international standard (ISO 1994). The list is far from exhaustive but provides sufficient concepts to position.

Table 6 – Corpus of Analysed and Positioned AEC Architectural Programming Methods

Main Discipline	Conceptual Design Method	References
Architecture Engineering Construction	*Functional Analysis*	(Gobin 2003)
	Value Management	(Yu 2007)
	Problem Seeking	(Pena & Parshall 2001)
	Construction process in France	(MIQCP 2001; Certu 2010)
	Client Requirements Processing Model	(Kamara et al. 2002)
	Functional Analysis and Future Life	(Grandjean 1980)
	ISO-9699	(ISO 1994)

Figure 13 represents the result of the second analysis of the concepts used by various conceptual design methods across engineering. The concepts are positioned on an axis that represents their respective system development process from an abstract level (regarding the system to develop) to a concrete level (detailed description of the system). In the early stages, the concepts are rather abstract, fuzzy, and difficult to fully grasp. In the later stages, the concepts are rather concrete, material, and easy to apprehend.

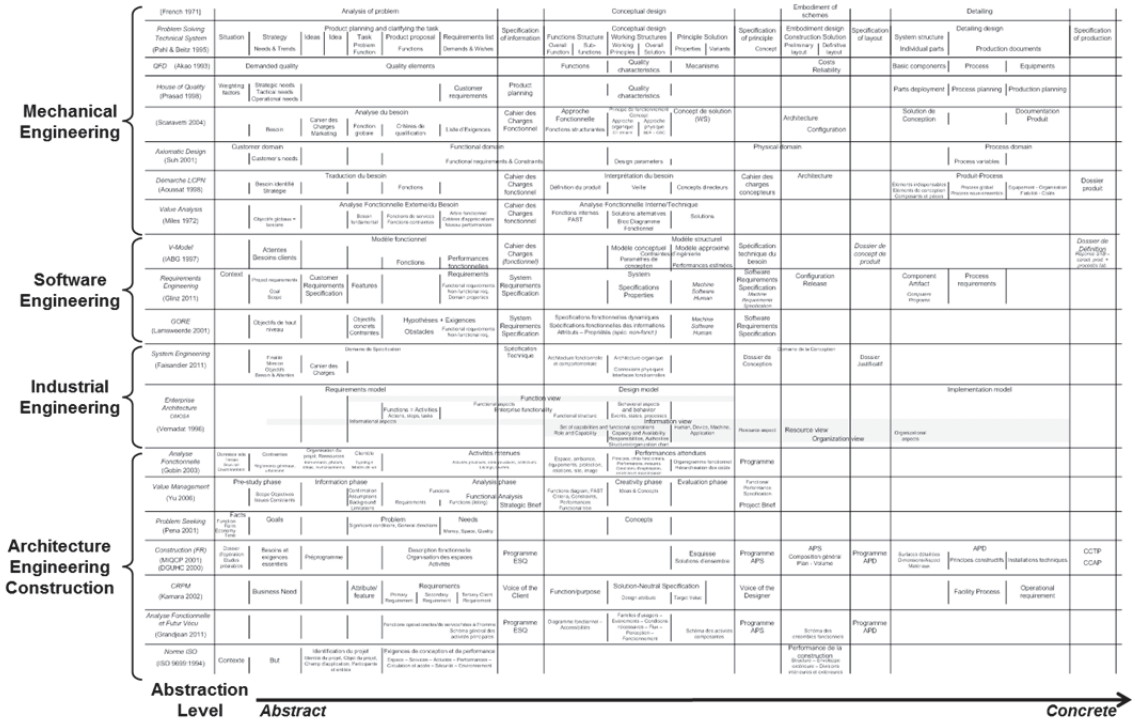


Figure 13 – Concepts used in Various Conceptual Design Methods across Engineering

Even if the corpus of conceptual design methods is not exhaustive and clearly led by our Mechanical Engineering background, it provides a first idea about the concepts required to describe a system and the nuances made across engineering domains.

Based on this figure, a third analysis is performed in three steps, from global to local. The first step focuses on the process. The second step focuses on the deliverables, and the third step focuses on their upstream content. This upstream content provides a certain structuration of the concepts used to describe an object of study across engineering domains.

First, three phases are identified across conceptual design methods: external analysis, internal analysis, and production analysis (Figure 14). External analysis focuses on the context and perimeter of the system to define. Internal analysis focuses on the detail description of the inside of the system. Its architecture, components, and their behaviours are developed to address elements from the external analysis. The last phase concerns the analysis of the components production and the system assembly or construction. Their identification allows the first verification and adjustment of the concepts alignment across conceptual design methods.

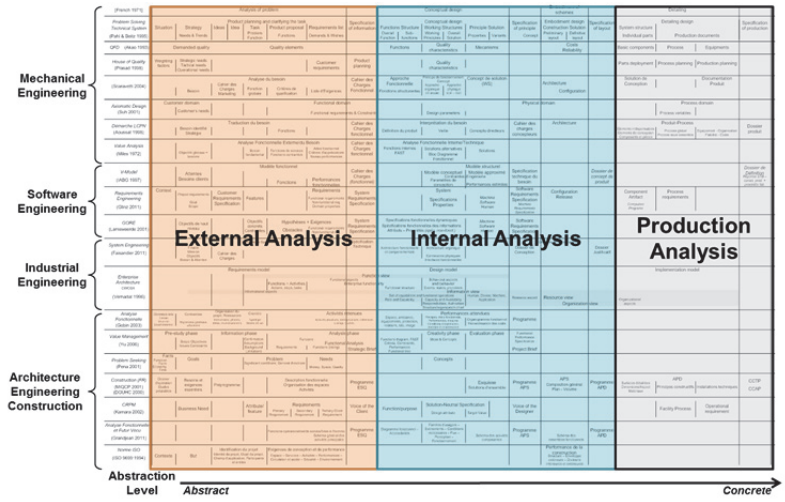


Figure 14 – Three Analysis Phases of Conceptual Design Methods

In a second step, it appears that almost all conceptual design methods were punctuated by a similar set of deliverables. The content and objectives of each deliverable across engineering discipline were noticeably alike. Figure 15 presents the positioning of these deliverables along the system development process. This second element allows further adjustments of the concepts alignment. The Clients Requirements Specification (CRS) states the initiation of the system development in terms of feasibility and opportunity. The Functional Requirements Specification (FRS) represents the synthesis of the external analysis. The second Design Requirements Specification (DRS II) synthesizes the internal analysis of the system with the higher level of details (embodiment design) compared to the DRS I (predesign). The Production Requirements Specification (PRS) gathers information about the production analysis of the system that develops the links between the system (including its components) and its production process.

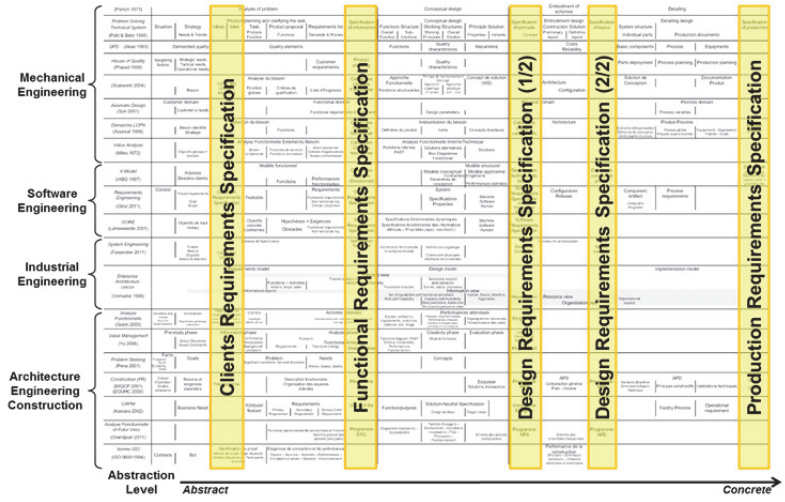


Figure 15 – Deliverables that Punctuate the Conceptual Design Methods

Six main topics of interest regarding the system to define can be distinguished from the analysis of the various deliverables based on their upstream concepts. These topics correspond to the six primitive interrogatives (Figure 16): why, what, how, when, who, and where. These six interrogatives are widely used to develop requirements in conceptual design (Kossiakoff et al. 2011) as well as to structure domain description approaches (Zachman 1987; Breaux et al. 2008). This last result from the analysis suggests that across conceptual design methods a system can be characterized by answering these six questions. The boundaries of each question match with the position of the key previously identified deliverables.

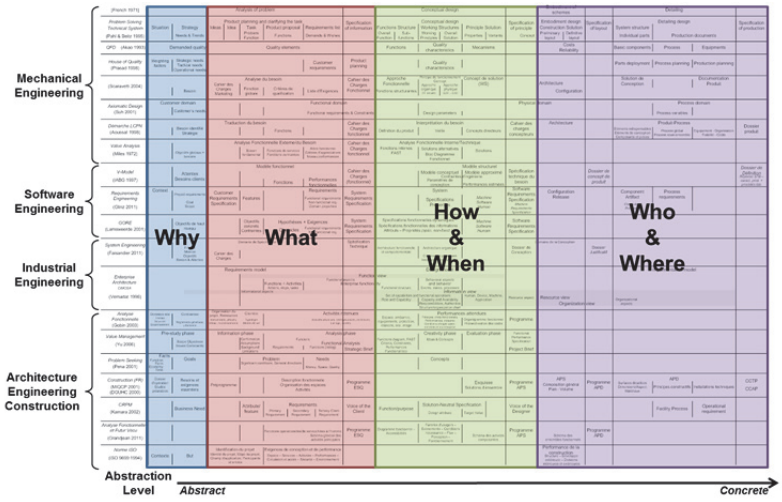


Figure 16 – Positioning of the Six Interrogatives from Conceptual Design

The six interrogative questions are proposed as a basis to characterize a system across engineering based on their analysis. Their order is taken from their position on the multidisciplinary overview of system development processes (Figure 16). It has to be noticed that the "*Where*" interrogative is specific to AEC and is therefore separately treated from the "Who" interrogative. In Section 1.2, the same interrogatives were used in the IFC to structure the description of the building and construction data. In order to describe the briefing data in the context of this research, the six interrogatives are formulated in a different way than for IFC:

- Why do we need the system?
- What should the system do?
- How would it do it?
- When would it do it?
- Who (which part of it or component) would do it?
- Where would it be done?

To address these questions, a first set of five main concepts is identified based on the multidisciplinary overview. The concept selection is based on their meanings, occurrence, and importance across engineering conceptual design methods. The concepts are: Goal, Function, Activity, Resource, and Space (Figure 17).

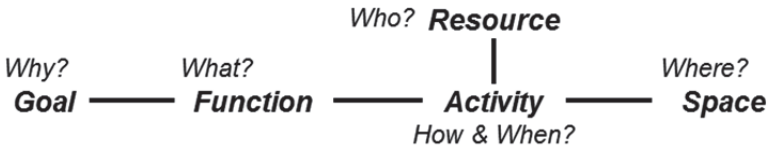


Figure 17 – Main Concepts from the Multidisciplinary Overview

Goal refers to the "*Why*" question. *Function* refers to the "*What*" question. *Activity* refers to both the "*How*" and "*When*" questions. *Resource* refers to the "*Who*" question that includes both human and material elements. *Space* refers to the "*Where*" question of particular interest regarding buildings. Despite their synchronous use across engineering, the definitions of the concepts differ from one engineering discipline to another. In the next section, a definition is provided to each concept in the frame of this research. The proposed definition is based on a literature review of their existing meaning across engineering.

3. Concepts Clarification

In Section 2, we introduced five main concepts to answer the six interrogatives. In this section, each concept is defined based on a review of its definitions across engineering disciplines starting with AEC to underline issues with current definitions. The aim is to introduce issues, implicit meanings and relative concepts across disciplines before providing definitions adapted to our understanding of Architectural Programming. To facilitate the understanding of the concepts and their definition, an illustrative example is provided after each definition based on the Luxembourgish secondary school case study (details of cases studies can be found in Chapter I Section 4.4).

3.1. Goal – Why

This section aims to define the concept of *Goal* as an answer to the "*Why*" interrogative of conceptual design. The concept of *Goal* resurfaces from four disciplines in terms of the use and definition: Architectural Programming in AEC, Design in Mechanical Engineering, Engineering Management in Industrial Engineering, and Goal-Oriented Requirements Engineering (GORE) in Software Engineering. Through a literature review, a definition of the concept of *Goal* is proposed that suits with our meaning of the "*Why*" interrogative (i.e. Why do we need the system?).

3.1.1. Architectural Programming: Goal & Objective

In a brief, the first part concerns the objectives and goals of the construction project. Peña defines *Goal* as "*the end toward which effort is directed. It suggests something attained only by prolonged effort*" (Pena & Parshall 2001). According to him, several synonyms can be used in architectural programming such as objective, mission, aim, purpose, reason, philosophy, aspiration, or policy. They are used to generate statements about what the clients want to achieve and why. An objective is defined as "*a more detailed delineation of a particular goal. It implies something tangible and immediately attainable*" (Pena & Parshall 2001). *Goals* represent a long term general ends whereas objectives represent short terms specific and somehow quantified ends (Cherry 1999).

One of the main problems with *goals* and *objectives* in practice is identified by Koutamanis (Koutamanis 2013). Due to their abstract and broad formulation, goals and objectives often remain self-evident to any building or trivial regarding the building type. The goal of a school is to provide education to the local children. The link between the goals and objectives, and the requirements on the building is not clearly developed. Goals almost stand for themselves without any consideration in the requirements. Their achievement is totally implicit through the requirements satisfaction.

3.1.2. Mechanical Engineering – Design: Purpose vs. Goal

The concept of *Goal* is not widely used in Mechanical Engineering. The first key concept is usually the concept of *Need* or *Function*. Instead of using the *Goal* concept, a few approaches refer to the concept of *Purpose*, often mixed with the concept of *Function* or integrated in its definition.

In the FBS approach, *Purpose* is defined as an intentional concept (Dorst & Vermaas 2005) expressing the need of a customer about what he wants to achieve with the system. It is first introduced and distinguished from the concept of *Function* as the answer to the question "*Why the system does what it does or what it is for*" (Rosenman & Gero 1998). Both concepts answer different questions and the confusion generally comes from the physical disciplines. However, the concept of *purpose* was replaced by the concept of *Requirement* as an input of the design process (Gero & Kannengiesser 2004), proving uncertainty about the clear distinction between the concept of *Purpose* and *Function* (Vermaas & Dorst 2007). The *Purpose* concept is clearly defined as a state of affairs users of the artefact seek and is supposed to be achieved by performing it (Vermaas & Dorst 2007).

3.1.3. Industrial Engineering – Engineering Management

In management philosophy, the concept of *Goal* is widely used and considered as a key driver (Goldratt & Cox 1984). An "official" definition in the field of Theory of Constraint (ToC) echoes with the concept of *Purpose*:

- Goal – "*the purpose for which the system was created as determined by the owner of the system*" (Cox III et al. 2012).

From time to time, a *Goal* is described as a continuing activity which it is not. Its statement should be specified as a condition to reflect outcomes, e.g. increasing profitability, now and in the future, cost-effective improvement of the overall health of the community (Dettmer 2007). Both definitions of the *goal* are gathered in business management under the definition of *business goal*:

- Business Goal – "*a state or condition the business must satisfy to reach its vision*" (IIBA 2009).

3.1.4. Software Engineering – GORE

In the more global context of Systems and Software Engineering, the concept of *Goal* is just defined as "*an intended outcome*" (ISO/IEC 2001). In Requirements Engineering, a *Goal* is defined as "*a state of affairs that the stakeholder wants to achieve*" (Glinz 2011). Of a particular interest are frameworks associated with Goal-Oriented Requirements Engineering (GORE) (Lamsweerde 2009) which focus on understanding why a system is needed instead of what a system should do or how it should do it (Lapouchnian 2005). The general concept of *Goal* was then introduced as a driver for software requirements acquisition.

- Goal – "*A goal is a prescriptive statement of intent the system should satisfy through cooperation of its agents*". Goals are non-operational objectives. "*Non-operational means that the objective is not formulated in terms of objects and actions available to some agent in the system; in other words, a goal as it is formulated cannot be established through appropriate state transitions under control of one of the agents*." (Dardenne et al. 1993)

Because of the complexity of targeted software systems, most of the GORE techniques (Lamsweerde 2009; Yu et al. 2010) organise *goals* into hierarchical trees or directed acyclic graph models (Figure 18) showing the how higher-level goals are satisfied by the lower-level ones (goal refinement) and, conversely, how the lower-level goals contribute to the satisfaction of the higher-level ones (goal abstraction) (Lamsweerde 2001).

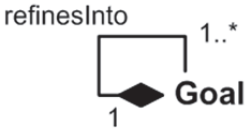


Figure 18 – Conceptual Model of the Goals Refinement-Abstraction

Moreover, five patterns of *Goals* are defined in the literature to support guidance during their elicitation, reuse, and checking: *achieve*, *cease*, *maintain*, *avoid*, and *optimize* (Dardenne et al. 1993). These patterns impact the possible behaviours of the system and detail the state transition desired by the stakeholders:

*"*Achieve *and* Cease *goals generate behaviors,* Maintain *and* Avoid *goals restrict behaviors, and* Optimize *goals compare behaviors"* (Dardenne et al. 1993)

Through the integration of explicit goal representations in these goal-based reasoning frameworks, a criterion for requirements completeness is introduced: "*the requirements are complete if they are sufficient to establish the goal they are refining*" (Yue 1987; Lamsweerde 2000). *Goals* provide other benefits to requirements definition (Lamsweerde 2001; Lapouchnian 2005): rationale for requirements that operationalize them and support for early requirements

analysis, precise criterion for requirements pertinence (Yue 1987), traceability links from high-level strategic objectives to low-level technical requirements, natural mechanism for structuring complex requirements documents (Lamsweerde 2001), basis for the detection and management of conflicts among requirements (Robinson 1989; Lamsweerde 1996), excellent way to communicate requirements to customers through its refinement process, and means to separate stable from volatile information: goals are much more stable than requirements or operations (Antón et al. 1994; Lamsweerde 2001).

3.1.5. Proposed Definition

Following this literature review, the *Goal* concept is clearly distinct from the concept of **Function**. The concept of *Goal* refers to the rationales behind the system's existence and sketches the desired TO-BE situation. As highlighted by (Rosenman & Gero 1998), the confusion seems to exist only in physical disciplines. In these disciplines, **Function** is perceived as the rationale. In this research, the building system is no longer attached only to physical disciplines but gets closer to business, management, service-related disciplines (Mauger et al. 2013). These disciplines add a higher level of information focusing on the "*Why*" interrogative through the concept of *Goal* and developing the relationships with lower level requirements. This new layer of information represents prerequisites and rationales to guide the requirements definition process and support the decision making process.

Definition 1: A **Goal** is a state of affair or condition defined by the clients that the system (i.e. planned facility) must satisfy through cooperation of its components (i.e. sub-systems) and establishes the raison d'être of the system (i.e. answer the "Why do we need it?" question).

Based on the literature review, a **goal** can be refined into one or several other **goals**. Five kinds of goals are considered in this research: *achieve goals*, *cease goals*, *maintain goals*, *avoid goals*, and *optimize goals* (Figure 19). This taxonomy of **Goals**, illustrated in Table 7, indicates a broader consideration of the rationales and limits behind the order for a (new) facility.

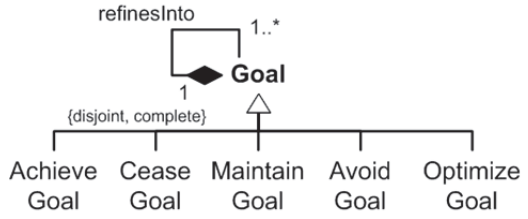


Figure 19 – Conceptual Model of the Goal Concept in GORE

Table 7 – Examples of goals from the LTB Case Study

Initial State A	**Goal Pattern**	**Goals** *(Desired State B)*
There are too many pupils for the current building.	*Achieve*	The new building is sized for the amount of pupils.
Pupils bully each other.	*Cease*	Pupils no longer bully each other.
Preparatory classes can have breaks when pupils are too agitated, without disturbing other classes.	*Maintain*	Preparatory classes can have breaks when pupils are too agitated, without disturbing other classes.
Unauthorized people should not enter the school.	*Avoid*	Unauthorized people should not enter the school.
Preparatory pupils walk too much (between their classrooms and the cafeteria) during school which tires them in class afterwards.	*Optimize*	Preparatory pupils should walk as less as possible during school to spare them unnecessary efforts that tire them in class afterwards.

3.2. Function – What

In research, the concept of *Function* is widely discussed and plenty of definitions can be found. In his PhD dissertation, Eisenbart proposes a rich multidisciplinary overview of its definition in literature (Eisenbart 2014). His corpus covers inter-alia: Mechanical Engineering, Electrical Engineering, and System and Software Engineering. Based on his overview, he adopted Warell's generic definition of *functionality* to define the *function* concept in his thesis:

> *"the combination of all its effects, actions, functions, and properties and their behaviour, that contribute to making the system useful for an intended purpose."* (Warell 1999)

In the frame of my PhD, this generic definition is considered too broad. The aim of this research is different. Regarding the six interrogatives defined in Section 2, *Function* as the answer to the "*what should the system do*?" requires a more specific definition.

Partly based on Eisenbart's overview, this section analyses the concept of *Function* with viewpoints from five disciplines to provide a definition compliant to our meaning of the "*What*" interrogative: Architectural Programming, Architecture, and Construction for AEC, Software Engineering in general, and Design Theories in Mechanical Engineering.

3.2.1. Architectural Programming

In architectural programming, Function refers to "*how the design product will work to do the job it is supposed to do"*, its performance (Pena & Parshall 2001). It is a "*how to*" question to answer the goals and objectives defined in the brief. Some approaches stay fuzzy about the concept of function, using it as a secondary concept, a range of other concepts such as *values* (Hershberger 1999) or *goals* (Cherry 1999). Some eventually evocate the *functional quality* or *essential functions* instead of a proper *Function* concept (Kumlin 1995). *Essential functions* of a building relate to the satisfaction of different domain needs: biological needs, psychic needs, and social needs; otherwise this is not architecture but only construction (de Beauvais 1995). These domains needs are further developed in the Maslow's hierarchy of needs (Maslow 1943) and human and social sciences. These needs are not in the scope of this research. However, they are still considered as essential but complementary.

3.2.2. Architecture

In architecture, it is not that easy to find an established definition of what is *Function*. Even in the debates around the "*form follows function*", the concept of *Function* is not properly but implicitly defined. According to Sullivan, a *function* is something that does not change (Sullivan 1896). When digging a little bit in history (Michl 1997), the concept of *Function* seems to have been introduced by Lodoli in architecture around 1750 with the principle of "*working part of the structure*" (Rykwert 1976) as a critic to the overuse of ornament and decoration (Michl 1997). For Lodoli, *function* is a synonym for *truth* (Neveu 2005) which somewhat confirms the implicit or philosophical meaning of the concept. However, Rykwert gives an interesting description of *function* through its etymology: "*From the Latin* fungor *(I perform) it had been used in a number of European languages to mean activity or performance in general, or the specific activity of certain things or persons, particularly the carrying out of any ritual or ceremonial action*" (Rykwert 1976). This description resonates with the "*how to*" of architectural programming as well as with some of the dictionary definitions. Other references associate the concept of *Function* to "*what people do*" (Hillier et al. 1976), i.e. activity or performance in general. This definition highlights a relation between two main concepts: *Function* and *Activity*, but it still has a lack of clarity in its meaning and definition. This ambiguous relation is clarified by Wurzer through the introduction of two notions: *capability* and *action* (Wurzer 2010). *Capability* refers to an ability to perform something whereas *action* refers to its implementation. Both senses are considered by architects through the concept of *Function*.

Aside from these conceptual discussions, the concept of *Function* is further developed by authors such as Neufert and Alexander. Since 1936, Neufert publishes a handbook that gathers sketches and drawings of abstract solutions to functional design issues (Neufert 1936; Neufert

1996; Neufert & Neufert 2012). Through its various editions, this handbook provides a state-of-the-art, the essence of techniques, projects, and science of buildings. Alexander proposes a set of *patterns* to address the same functional design issues (Alexander et al. 1977). Problem is that they never define what a *function* is but provide abstract solutions to realize it during design.

3.2.3. Construction

In France, a user-focused referential was initiated in the 1980s based on the concept of *Usage Function* (Gobin 2001). This concept was proposed by Value Analysis specialists (APTE consulting office) in collaboration with professionals from the Fédération Française du Bâtiment (FFB) and an industrial, GTM construction. The idea was to provide a common language focused on the usage to all construction professionals as a basis for their solution rationale, to communicate without technical a priori due to a solution-neutral expression, and to provide a generic way of building characterisation, independent from the building type. The approach is rather bottom-up, influenced by an industrial viewpoint of building construction.

A *Usage Function* is considered as the essentials functionalities required to perform certain tasks, independently from external elements (e.g. climate, environment, neighbourhood) that could hinder their achievement (Gobin 2003). It is defined as facilities to provide by the building that allow the user to perform activities. The referential is composed of seven kinds of *Usage Function* (Gobin 2006) that define a building (Table 8). Each one of them gives a different viewpoint on the same activity. It is a user-oriented approach; the primary concept is the activity performed by the user. The building is then specified regarding each activity according to the seven *usage functions* in terms of operating principle, usage performances, and operating performances (e.g. maintenance).

Table 8 – Seven usage functions referential (Gobin 2003)

Functions	Operating Principles	Usage Performances	Operating Performances
Space			
Atmosphere			
Protection			
Relation			
Site			
Goods & Tools			
Semiology			

The definition proposed by Gobin underlines a dependency between the concept of *Function* and the concept of *Activity* (Figure 20). An activity cannot be performed if the seven functions are not ensured by the building. The performance criteria are outputs used to assess the adequacy of a technical solution regarding each function related to activities.

Figure 20 – Conceptualization of Gobin's proposal

3.2.4. Software Engineering

In Software Engineering, the term *function* gathers several definitions. Nine definitions can be found in the Systems and Software Engineering standards (IEEE 2010):

1. "*An elementary unit of requirements and specifications defined and used for measurement purposes*"
2. "*A software module that perform a specific action, is invoked by the appearance of its name in an expression, may receive input values, and returns a single value*"
3. "*A task, action, or activity that must be accomplished to achieve a desired outcome*" (IEEE 1998)
4. "*The features or capabilities of an application as seen by the user*"

5. "*A defined objective or characteristic action of a system or component*"
6. "*A transformation of inputs to outputs, by means of some mechanisms, and subject to certain controls, that is identified by a function name and modelled by a box*" (IEEE Computer Society 1998)
7. "*An aspect of the intended behaviour of the system*"
8. "*A single-valued mapping*"
9. "*Part of an application that provides facilities for users to carry out their tasks*"

Definitions 3, 4, 5, 7, and 9 are of particular interest regarding our meaning of the "*What*" interrogative and previous definitions. In definition 3, *function* is assimilated to a task, an action, or an activity which strengthens the dependency link presented in construction (Section 3.4) between *function* and *activity*. Definition 4 relates the concept of *Function* to a capability of a system with a user perspective. This point will be further developed in Mechanical Engineering (Section 3.5). In definition 5, *function* is referred to a purpose of a system, something that somewhat defines the raison d'être of the system. This point will also be further developed in Mechanical Engineering (Section 3.5). Definition 6 reminds a mathematical understanding of the concept of *function*: $y = f(x)$, where y is the output, x is the input and f is the function. Definition 9 is close to the definition proposed by Gobin regarding *Usage Function*, a facility provided by a system to allow the users perform their tasks. In this sense, it is related to definition 4 (i.e. a capability) from the user perspective. These last definitions highlight the relationship and distinction to be made between the *Function* concept and the *Activity* concept. If a function allows an activity by the user, then they should be considered as different concepts.

3.2.5. Mechanical Engineering

In the Design research community, there are a lot of complex discussions on the concept of *Function*. Erden et al. present a review of functional modelling that underlines the various viewpoints on and co-existing meanings of *Function* (Erden et al. 2008). Even practitioners use the concept of *Function* without clear defined understanding of the concept (Eckert et al. 2010; Eckert 2013). This variety is acknowledged by the community (Vermaas 2011). The resulting ambiguity is considered as a flexibility that presents its own advantages to Engineering (Vermaas 2009). Far from these theoretical and methodological discussions on *Function*, this research seeks to find a suitable definition for Architectural Programming regarding the "*What*" interrogative.

In Value Management, *function* is defined as the "*effect of a product or of one of its constituents*" (AFNOR 1996), "*expressed in terms of purpose*" (Charpentier 2005). It is considered as a result described as a purpose, i.e. without consideration of specific technical solution. With Axiomatic Design, Suh also refers to *function* as a desired output (Suh 1998). In his approach, the customer needs are transformed into functional requirements (i.e. functions), design parameters, and then process variables (Suh 1990). In FBS, Gero with his different co-authors define a system's function as its teleology, i.e. what it is for (i.e. its purpose) (Kannengiesser & Gero 2011).

Pahl and Beitz define *function* as "*the general input/output relationship of a system whose purpose is to perform a task*" (Pahl & Beitz 1995) based on Rodenacker's widely accepted definition (Rodenacker 1970). The *function* transforms an input into an output through the system when performing a task it was built for. System Engineering proposes a different formulation: "*a task, action, or activity performed to achieve a desired outcome*" (AFIS 2004; ANSI/EIA 2003). Chandrasekaran and Josephson refer to these two meanings using the terms "*function as effect*" compared to "*function as what a device does*". The first one is said as *environment-centric* whereas the second is considered *device-centric* (Chandrasekaran & Josephson 2014).

Both definitions can be found in Software Engineering (Section 3.2.4) and refer to two aspects of a system, the purpose (i.e. its raison d'être), and the activity (i.e. how it performs). However, a third set of definitions gives another viewpoint on the *Function* concept in Design:

"Every object, through its appearance, informs us of what it is, and through its function advises us about what it can do" (Kim & Boradkar 1988)

In this quotation, Kim and Boradkar refer to the "*form follows function*" principle evocated in Architecture (Section 3.2.2). In this case, *function* is defined as what a system can do. The subtle value added in this definition resides in the "*can*" term indicating a capability as in definition 4 of Software Engineering (Section 3.2.4). This might be the subtlety necessary to make a distinction between requirements definition and design. "*Function as effect*" suggests the purpose whereas "*function as what a device does*" suggests the how to. In this research, the definition of function is closer to Pahl and Beitz's definition and the *environment-centric* "*function as effect*" rather than the *device-centric* "*function as what a device does*".

3.2.6. Proposed Definition

In this literature review of different disciplines, several definitions were used to define the concept of *Function*. It is acknowledge that several meanings co-exist in engineering (Vermaas 2011). Vermaas synthesizes the various interpretations of *Function* in three notions: *capacity-related*, *behaviour-related*, and *goal-related* (Vermaas 2013). Instead of adopting a generic definition that covers the three notions (Eisenbart 2014), a single meaning is required in this research. The notion corresponding to our meaning of the "*what*" interrogative is the *capacity-related* notion defined by Vermaas. The other notions refers to the "*why*" interrogative (*goal-related*) addressed in this research through the concept of **Goal** (Section 3.1) and the "*how*" interrogative (*behaviour-related*) addressed through the *Activity* concept (Section 3.3).

Function is the answer to the "*what it does*" or "*what it should be able to do*" question, resumed in the "*what*" interrogative. It is, therefore, considered as a potential requiring context and external event to be triggered. One object has several functions but they are not used at the same time. For example, a smart phone can have plenty of them; from "*call someone*" to "*assess your health*". Another good example is the multifunctional Swiss Army knife. You can only use one function at a time. A building, in its own complexity, has also a lot of functions and even more ranges. It has to protect people and goods from the unpredictable climate night and day; it has to provide resources to perform a set of activities.

In this research, a *function* as the answer to the "*what*" question is considered as objective and context independent. It is used to make explicit, structure, and abstract what the system (i.e. facility) should be able to do. It refers to the concept of *external function* in Value Engineering. We consider it as a prerequisite to the definition of the building. The "*how to*" on the other hand is totally subjective and depends on the context and means available. The "*how to?*" refers to the concept of *internal functions* in Value Management. The following definition, illustrated by Figure 21, is therefore proposed for the concept of *Function*.

Definition 2: A **Function** is an ability of action, granted by the facility to its clients allowing them to change their environment from a state A to a state B, maintain or avoid a state C, or optimize a state D.

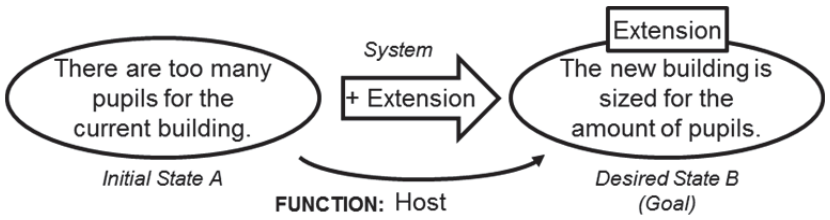


Figure 21 – Illustration of the Function Concept on the LTB Case Study

The "*desired state B*" in the proposed definition of **Function** corresponds to the previously defined concept of **Goal** as illustrated in Figure 21. The link between these two concepts can be formulated by the sentence: "*a **Goal** is achieved by one to several **Functions**, and a **Function** achieves one to several **Goals***" (Figure 22). The term "*achieve*" is used in the conceptual model as a generic term to simplify its representation. Using the five patterns of Section 3.2, the relationship modelled between **Goal** and **Function** covers: a **goal** is achieved by a **function**, a **goal** is ceased by a **function**, a **goal** is maintained by a **function**, a **goal** is avoided by a **function**, and a **goal** is optimized by a **function**.

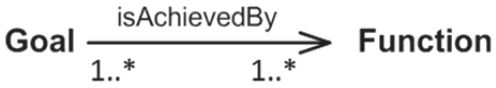


Figure 22 – Conceptual Model of the Goal–Function Relationship

3.3. Activity – How & When

In Architecture-Engineering-Construction, the concepts of *Activity* and *Space* are tightly related. Architectural programming is mainly concerned with the concept of *Activity* whereas Architecture is mainly focused on the concept of *Space*. *Activity* is the input of architectural programming, and *Space* its output. A space cannot be understood without the activities performed inside. However, this relationship between these two concepts induces a few confusions in their use and formulation (Section 3.3.1). This section aims to first define the concept of *Activity* in the frame of this research.

3.3.1. Architectural Programming: Activity vs. Space

Peña and Parshall define *activities* as "*organized units for performing a specific function*" (Pena & Parshall 2001). In their definition, *activities* describe how functions are delivered. They are what happen inside the building after its construction, during its use.

"Probably the most fundamental purpose of an architectural brief is to describe what takes place in the building to be designed (i.e. the activities of the users)." (Koutamanis 2013)

In his book, Koutamanis describes the concept of *Activity* and issues gravitating around it. From his viewpoint, *activities* give a concrete and meaningful dimension to goals, requirements, and constraints. Even if it is not necessary to fully describe them, the breaking down of a brief into *activities* is an appropriate abstract way to tell the architect what it is expected from them: "*shelter people doing certain things in a certain way*" (Koutamanis 2013). However these *activities* are described, it does not prevent the architect from changing them through his design but it provides him/her with a basis to design a "*good enough*" solution, not necessarily optimal or perfect (Simon 1996). The problem is that the description of *activities* is often replaced by information about capacity and performance which are of little help to understand how things will be done to ensure these capacity and performances. Solving such issues is a prerequisite to design (Koutamanis 2013).

3.3.2. System and Software Engineering: Activity

System and Software Engineering widely support the automation of *activities*. As a result, the definition of *Activity* concept is more universal than the definition of *Function* concept. *Activity* is defined as a unit of work (IIBA 2009), a set of cohesive tasks (IEEE 2008; IEEE 2010), or "*a set of actions performed as part of a process that consumes time and resources and whose performance is necessary to achieve, or contribute to, the realization of one or more outcomes*" (IEEE 2008; INCOSE 2010).

The concept of *Activity* is related to other concepts such as *Action*, *Task*, *Process*, and *Resources* (Figure 23). *Process* is defined as "*a set of interrelated or interacting activities which transforms input into outputs*" (AFNOR 2005; INCOSE 2010; IEEE 2008). *Task* is defined as a "*requirement, recommendation, or permissible action, intended to contribute to the achievement*

of one or more outcomes of a process" (ISO/IEC 2008a). In Goal-Oriented Requirement Engineering, Lamsweerde defines *Action* as "*an auxiliary operation associated with a state transition. […] Like any operation it is atomic*" (Lamsweerde 2009). By auxiliary, Lamsweerde considers that an *action* has no meaningful effect and does not require to be captured as precisely as an *operation*. *Operation* and *Task* concepts can be considered as equivalent.

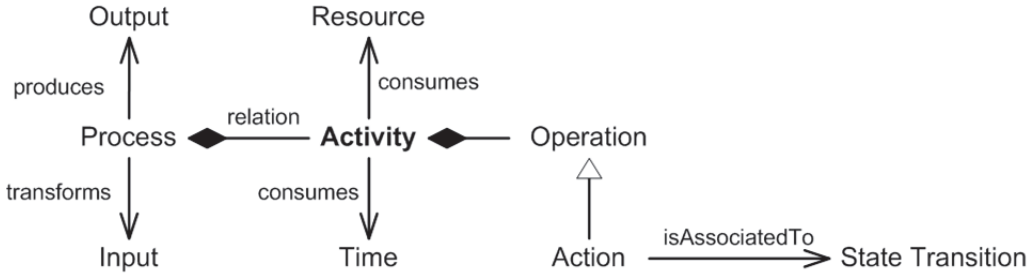


Figure 23 – Synthesis Conceptual Model of the Activity Concept in System and Software Engineering

3.3.3. Enterprise Architecture

Enterprise Architecture defines an *activity* (or Enterprise activity) as "*a set of partially ordered basic operations executed to perform things to be done within an enterprise. Activities are performed by the functional entities of the enterprise and transform an input into an output state*" (Vernadat 1996b). A set of partially ordered set of *activities* is defined as a *Business Process*. A *Business Process* is triggered by an *Event*. An *Activity* is called by a *Business Process* when all preconditions are met (Vernadat 1996a). Instead of defining the full set of concepts used in Enterprise Architecture, we propose to use Figure 24 as a synthesis.

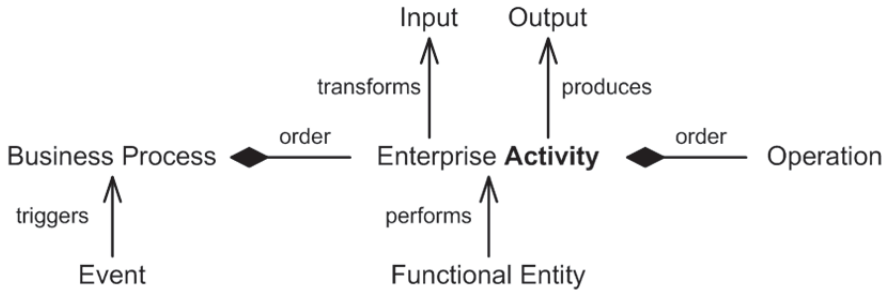


Figure 24 – Conceptual model of the Activity concept based on (Vernadat 1996b; Vernadat 1996a)

The concept of *Event* in Enterprise Architecture is the main concept that answer the "*when*" interrogative defined in Section 2. In this research, this concept is considered as part of the attributes characterizing the *Process* concept.

3.3.4. Proposed Definition

The concept of *Activity* is essential in architectural programming. However, it does not suffice to describe what people do inside a building. In architectural programming, an activity is usually described alone, without formal relationship with other activities. There is certain lack of structure in the presentation and description of activities. Moreover, this concept is too often neglected to the benefit of the *Space* concept. Both concepts answer to different questions. *Activity* refers to "*how*" a **function** is realized whereas Space refers to "*where*" the *activity* takes place. Spatial requirements are tightly related to the *activity*. They are specified according to it. Complementary concepts are required to fully describe an *activity* and to understand the spatial requirements. The aim of this research is not to make too complex the description of *activity* but to provide enough complements. As a result, only three concepts are kept from System and Software Engineering and Enterprise Architecture: *Activity*, *Process*, and *Operation*.

The concept of *Process* provides temporal information about the *activities* by ordering them. As seen in Section 3.3.3, a process is triggered by an *event*. The concept of *event* is not directly considered in this research but included in the characterization of a *process*. Its integration to the concept of *Process* allows us to somehow limit the complexity of the definition domain. *Process* is, therefore, used to answer the "*when*" interrogative.

Based on the definition from the various domains of engineering, the following definitions are proposed for the concept of *Activity*. *Activity* is considered as an encompassing concept (Figure 25). *Process* and *Operation* are specific kinds of *Activity*.

Definition 3: An **Activity** is a unit of work required to perform a function.

Definition 3.1: An **Operation** is an elementary activity.

Definition 3.2: A **Process** is a set of ordered activities.

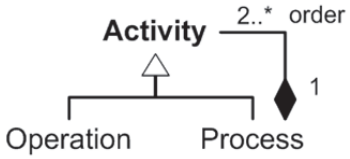


Figure 25 – Conceptual Model of the Activity Concept

Therefore, in this research, **Activity** is defined as the answer to the "*how*" interrogative. Using the previously defined concepts of **Function** and **Goal**, the "*how*" question stated in Section 2 can be reformulated that way: How would the system do the required **functions** to achieve the defined **goals**? **Functions** are realised through one (**operation**) to several (**process**) **activities**. However, an **Activity** optionally realises **Functions**. In the case of a **process** is required to realise a **function**, the composing **activities** are not sufficient to fully realise it. As a best practice, the relationship between Functions and Activities should follow the independence axiom from Axiomatic Design (Suh 1990) (i.e. "*Maintain the independence of the functional requirements*"). Figure 26 proposes the conceptual model representing the relationships between the main concepts of **Goal**, **Function**, and **Activity**.

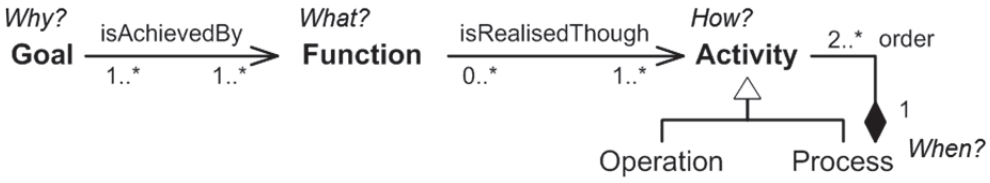


Figure 26 – Resulting Conceptual Model of the Goal, Function, and Activity Concepts

3.4. Resource – Who

"*Who*" is the fifth interrogative interpreted in this research through the question: "*Who would perform the how?*" Based on the previous definitions, the question becomes:

"*Who would perform the **activities** that realize the **functions** required to achieve the **goals**?*"

The answer of such a question is larger than what is implied by the term "*Who*". In English, its sense is limited to the designation of a person, a human being. In this research, the "*Who*" interrogative is used as a generic term to label the performer of an **activity**. The performer can be either a human being (i.e. user) or any kind of entity or artefact (e.g. a computer or an automaton) linked to the system. Our interpretation of the "*who*" interrogative is not limited to the performer, it also covers any artefact required to perform the **activity**. As a result, the concept of *Resource* was identified in Section 2 to answer this interrogative. In this section, the concept of *Resource* is

defined based on a literature review of its meaning in four disciplines: Architecture-Engineering-Construction, System and Software Engineering, and Enterprise Architecture and Operation Management in Industrial Engineering.

3.4.1. Architecture-Engineering-Construction

In the Industry Foundation Classes, the concept of *Resource* is defined through the artefact *IfcConstructionResource*: "*A resource represents "use of something" and does not necessarily correspond to a single item such as a person or vehicle, but represents a pool of items having limited availability such as general labour or an equipment fleet. A resource can represent either a generic resource pool (not having any task assignment) or a task-specific resource allocation (having a Task assignment)*" (buildingSMART 2013). In this definition, the concept of *Resource* is associated to human beings and material artefacts as a single or multiple entities (Figure 27). It supports our view of what a *resource* is regarding the "*who*" interrogative. However, its definition and interpretation through the IFC associate it to the "*how*" interrogative.

To go further, the *IfcConstructionResource* covers human resources of the construction project (e.g. paying client, architects, designers, or contractors) whereas in this research, the concept of *Resource* extends to human resources of the activities performed during the building usage (e.g. user clients and customer clients).

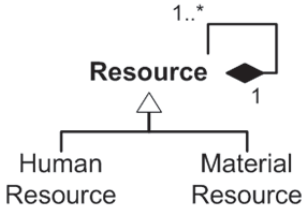


Figure 27 – Conceptual Model of the Resource Concept from IFC viewpoint

3.4.2. System and Software Engineering

The concept of *Resource* has several definitions across System and Software Engineering standards (IEEE 2010):

1. "*Skilled human resources (specific disciplines either individually or in crews or teams), equipment, services, supplies, commodities, materiel, budgets, or funds*" (Project Management Institute 1996).
2. "*An asset that is utilized or consumed during the execution of a process*" (INCOSE 2010; ISO/IEC 2008a; IEEE 2008).
3. "*A role (with respect to that action) in which the enterprise object fulfilling the role is essential to the action, requires allocation, or may become unavailable*" (ISO/IEC 2006).
4. "*An enterprise object which is essential to some behaviour and which requires allocation or may become unavailable*" (ISO/IEC 2006).
5. "*People, procedure, software, information, equipment, consumable, infrastructure, capital and operating funds, and time*" (ISO/IEC 2008b).

Resource is understood as any artefact necessary to perform something (i.e. any action). These *Resources*, due to their limitations in quantity and capacity, have to be managed. Their status can change over the time through wear and tear related to their use or through their consumption or transformation.

3.4.3. Enterprise Architecture

The definition of *Activity* in Enterprise Architecture introduces the concept of *Functional Entity* as the performer of an *Enterprise Activity*. A *functional entity* is defined as "*any active*

resource inside or outside an enterprise capable of executing basic functional operations of an activity and playing a given role in the course of a process" (Vernadat 1996b). This definition introduces an interesting notion about *resources*, the fact that it can be "*inside*" or "*outside*" an enterprise.

The concept of *Resource* is assimilated to the concept of *Mechanism* in the sense of IDEF0: "*enterprise object contributing to the realization of the functionality of activities of business processes*" (Vernadat 1996b). This definition is similar to the definition used in System and Software Engineering. In the same reference, Vernadat proposes another definition: "*a resource is an entity (human or technical) which can play a role in the realization of a certain class of tasks, when it is available*" (Vernadat 1996b). Therefore, the concept is not limited to active entities but also passive ones as long as they are required to perform an *activity*.

This distinction can be observed in the IDEF0 that defines three kinds of *resources*: *input*, *output*, *mechanism*, and *control*. Each kind has a different relationship with the *activity* concept. An *input* is transformed by *mechanisms* during an *activity* into an *output* governed by *controls*.

3.4.4. Operation Management

In the domain of Operation Management, the American Production and Inventory Control Society (APICS the Association for Operations Management) has been providing the terminology basis since 1957. In its reference dictionary, the APICS provides a rather broad definition of the concept of *resource*: "*anything that adds value to a product or service in its creation, production, or delivery*" (APICS 2008). This definition supports our synthesis about the definitions from System and Software Engineering (i.e. any artefact necessary to perform something) through the concept of *Value Added*.

3.4.5. Proposed Definition

Across this literature review on the use and definition of the *Resource* concept, we noticed that the concept of *Resource* is first used to designate the performer of **activities**. However, we also saw that resources are not limited to an active role to perform an activity. Therefore, in this research, we propose a rather generic definition of *Resource* that covers both active and passive artefacts involved in the performance of an elementary **activity** (i.e. operation).

Definition 4: A **Resource** is any entity (inside or outside the system) required to perform an operation.

Therefore, the concept of **Resource** is connected to the concept of **Activity** through the concept of operation (Figure 28). An operation requires (at least) one to several **resources** to be performed whereas a **resource** can be required by one to several operations. A **resource** is not necessarily a single artefact but can be composed of several **resources**. Two kinds of resources are considered for now: human resources and material resources. Grammatically speaking, both are considered to answer to the "*who*" interrogative. Based on the previous considerations, the full question associated to the "*who*" interrogative becomes: "*Who is required to perform the **activities** that realize the **functions** required to achieve the **goals**?*"

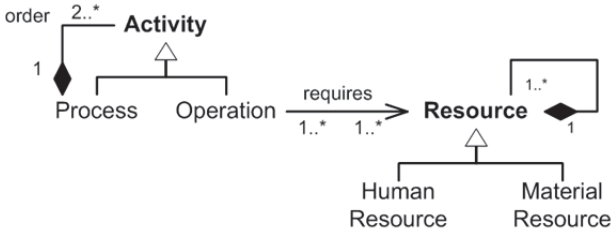


Figure 28 – Conceptual Model of the Activity–Resource Relationship

Based on the proposed definition of **resource**, one may say that another kind of resource could be considered for architectural programming: *spatial resources*. Far from arguing against such a consideration, we decided to separate the spatial notion from the intended definition of resources because of its importance in the architectural programming domain. The notion of *spatial resource* does not refer to the "*who*" interrogative but to the *where* interrogative. Therefore, instead of using the notion of *spatial resource*, we use the concept of *Space* as an answer to the "*where*" interrogative.

3.5. Space – Where

The concept of *Space* is proper to the Architecture-Engineering-Construction discipline. Other engineering domains do not define this concept as it is not required. The focus is, therefore, on Architectural Programming, Architecture, and Construction.

3.5.1. Architectural Programming

Based on the architectural programming deliverable, three distinct kinds of *space* have to be considered: *functional space*, *planned space*, and *designed space* (Figure 29). The first one refers to the *space* demanded by clients and formalized in the brief through the functional diagram. The functional space does not integrate quantities but functional relationship between spaces. Moreover, a *space* can be multi-functional if several *operations* or *activities* can or have to be performed in the same *space*. Therefore, the programmer defines a second kind of space which refers to the space indicated in the space table. The *planned spaces* are quantified by the programmer based on the quantity of people, resources, or work to be performed in the building. The third one is the *space* proposed by the architect in the project proposal (i.e. drawings). *Functional spaces* are implemented by *designed spaces*. However, the architect is free to make some changes in his design as long as the proposed drawing satisfies the clients' needs.

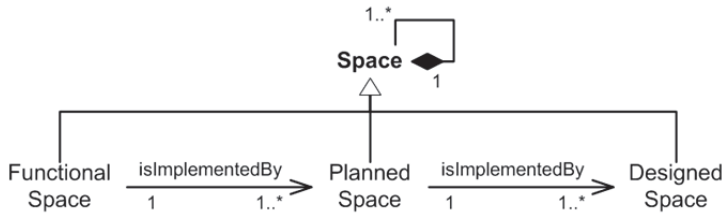


Figure 29 – Conceptual Model of the Space Concept

3.5.2. Architecture

The artistic, philosophical, social, and human aspects of architecture are difficult things to grasp for an Engineer. When dealing with *Space*, an Engineer mainly focuses on its physical ("hard") aspects described via formal and structural elements such as length, size, walls, surface, light, sound, etc. and with little concern about its users. There are two languages used by architects to talk about *space* (Dursun 2009). The first language describes a *space* as a static artefact, concrete and easy to formalize. The second language refers to the "soft" aspects not considered by Engineers. It focuses on the logic of *spaces* inscribed in the man to man or man to *space* relationships difficult to formalize (Dursun 2009). Table 9 provides a comparison of these two languages.

Hillier developed *Space Syntax* to represent the logic of *spaces* from a social viewpoint using patterns (Hillier & Hanson 1989; Hillier & Hanson 1997). In this research, we also intend to represent part of the logic but instead of focusing on social behaviour, we focus on "functional" behaviours. "Functional" behaviours refer to behaviours guided by the facility to provide through the building system. It includes the user and customer clients' behaviours as well as resources' behaviours (e.g. flow of information or materials). "Functional" behaviours are systematic and can be modelled using knowledge from Industrial Engineering (including Enterprise Architecture). In fact, the concept of "functional" behaviour corresponds to the concept of **Activity** in the limitations of this research whereas social behaviour is similar to usage. In the

same way, space layout research (Lobos & Donath 2010) also focuses on the physical properties and "functional" behaviour rather than on its social behaviour of spaces.

Table 9 – Two different languages to talk about Space (Dursun 2009)

Talking about Space via	
Physical properties	**Its logics**
quantified by measuring devices which do not depend on human agency	quantified by measuring devices which depend on human agency
describing spatial elements and their individual characteristics	describing spatial relations, their potentials and meanings
concrete	abstract
formal, dimensional, physical	rule or code based, logical, social
looking from outside	looking from inside
visible	invisible
easy to measure	difficult to measure
easy to talk about	difficult to talk about
discursive	non-discursive
static	dynamic

The concept of *Space* is not really "defined" in Architecture but rather "felt" or "perceived" (attached to its logic). The "definition" of *Space* depends on the architect, his understanding and own analytical perception of it (i.e. spatial awareness) (Dursun 2012a; Dursun 2012b). To illustrate such statement, we can take as a reference *A Pattern Language*. Alexander et al. give the following definition to the space concept: "*Space, when properly formed, is whole. Every part of it [...] is whole, in the sense that it is both an integral entity, in itself, and at the same time, joined to some other entities to form a larger whole*" (Alexander et al. 1977). Compared to definitions in other engineering domain, this definition remains conceptual, abstract, and almost philosophical.

3.5.3. Construction

Research in Building Information Modelling (BIM) focuses on the physical aspect of *space* through the Industry Foundation Classes (IFC) whereas architects focus on its logics. Regarding building product data models (before IFC), Björk defines of space as "*A volume bounded on all sides by enclosing structures, which forms the physical boundaries of the space*" (Björk 1992a). The latest definition provided in the IFC4 by buildingSMART evolved toward: "*A space represents an area or volume bounded actually or theoretically. Spaces are areas or volumes that provide for certain functions within a building*" (buildingSMART 2013). In both definitions, the only logic remaining in a *space* is related to its function. All social and human aspects of the *space* are concealed. Despite its reference in the definition, the function of a *space* is not really informed in the model. However, the *physical properties* of *spaces* are widely developed in the IFC through property sets that describe a *space* (e.g. Pset_SpaceOccupancyRequirements, Pset_SpaceCommon, or Pset_SpaceLightningRequirements).

3.5.4. Proposed Definition

Architects give more senses to *Space* than Engineers. According to Lefebvre, a *space* is a "*social product*" (Lefebvre 1974). It creates settings which organize users' lives, activities and relationships (Lawson 2001). This social dimension represents the logics behind the *space* concept through a man-space relationship (Dursun 2012b). Activity may be the central concept of architectural programming but *Space* is clearly the core of design in architecture (Dursun 2009). In this research, the social dimension of *Space* is reserved to another research study that would focus on the design phase. As a result, only the physical properties are included in the concept of *Space* while the (functional) logic behind the space layout is postponed to the concept of **Activity**.

Definition 5: A **Space** is the place (Figure 30) where activities are performed.

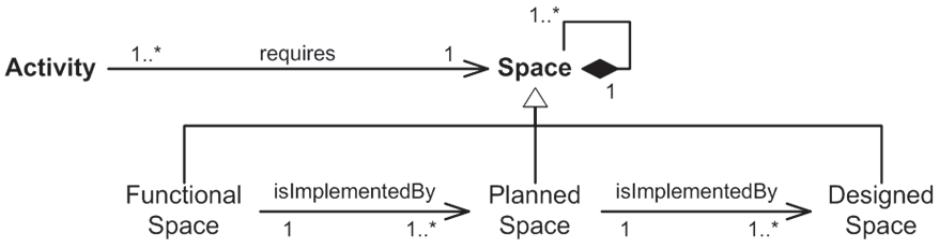


Figure 30 – Conceptual Model of the Activity-Space Relationship

3.6. Definition Domain

In this section, we analysed the various meanings of five concepts (i.e. constructs) used to describe an object of study in various engineering domains. Based on this multidisciplinary analysis, we proposed our own definitions adapted to our object of study, i.e. building requirements. Each concept is related to an interrogative that describes a specific viewpoint on the object of study (Figure 31).

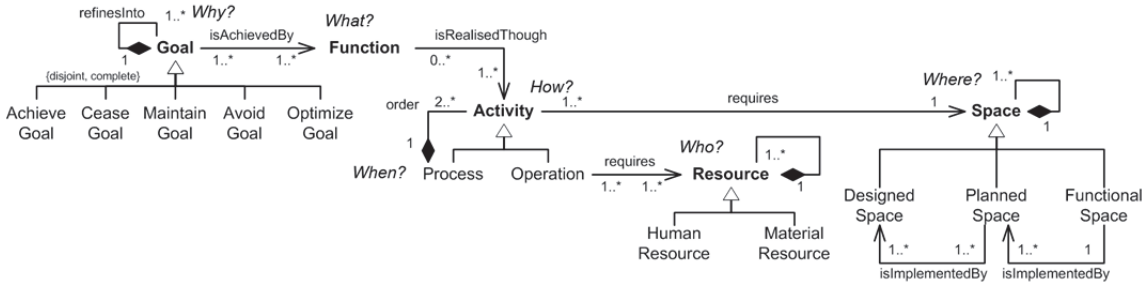


Figure 31 – The six interrogatives and their key concepts for Architectural Programming

The resulting taxonomy forms the definition domain required to define and describe the building requirements. This definition domain is structured around question-construct pairs that provide five different viewpoints on the object of study:

Why-Goal establishes the raison d'être (i.e. rationales and limits) of the facility, the to-be situation desired by the clients.

What-Function abstracts the (expected) ability provided by the facility to achieve, cease, maintain, avoid, or optimize the defined **goals**, regardless of the technical solutions.

How & When-Activity describes the way (i.e. solution) the facility grants the **functions**.

Who-Resource describes the means required to perform **activities**, and to implement solutions.

Where-Space describes the building itself, the place where the **activities** are performed.

4. Synthesis

In this chapter, we proposed a set of constructs to define and describe the definition domain of building requirements: **Goal**, **Function**, **Activity**, **Resource**, and **Space**. Each construct addresses one of the primitive six interrogatives used to structure the definition domain: *why*, *what*, *how*, *when*, *who*, and *where*. Each pair of question-construct constitutes a different viewpoint describing the object of study. The viewpoints are organized by level of abstraction regarding the building from a high-level business-oriented viewpoint (**Goals**) to a low-level building-oriented viewpoint (**Spaces**) (Figure 32).

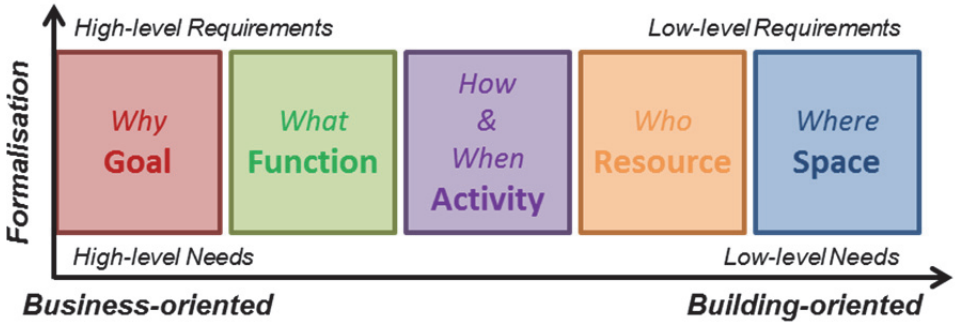
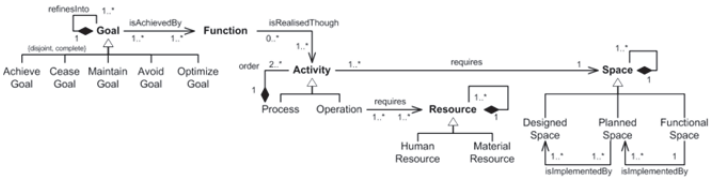


Figure 32 – Structuration of the Definition Domain Viewpoints

The proposed constructs are organised in taxonomy. Their relationships are established in a conceptual model to facilitate the identification of dependencies and their automation. By doing so, the decision making and information change management should be facilitated (e.g. dependencies, traceability). This structure of constructs represents the definition domain of the building requirements, first research output of this work (Table 10). Based on this definition domain, Chapter III proposes a set of modelling languages to formalise the shift from needs to requirements according to each defined viewpoint.

Table 10 – Research Output of Chapter II: the Definition Domain

		Research Activities
		Build
Research Outputs	**Constructs** ***Chapter II***	*Definition domain* 

Chapter III – MODEL A BUILDING AND ITS DEFINITION DOMAIN

Chapter III introduces a set of models to support the formalisation of the needs into requirements based on the building definition domain proposed in Chapter II. Besides Section 1 which introduces the chapter, this chapter is organised in two main parts (Figure 33). The first part is composed of Sections 2 and 3 and it presents conceptual models that structure the object of study (i.e. buildings) and the requirements definition. Section 2 starts with the modelling of the object of study (i.e. its perimeter). Section 3 continues with the modelling of the transitions from **Goals** to **Space** following the requirements definition process.

The second part, composed of Sections 4 to 6, presents operational models that support the requirements definition. Section 4 is based on a literature review of modelling languages across engineering used to graphically formalize the defined constructs and their relationships. These modelling languages support the multiple views required to describe buildings. Section 5 proposes a new design artefact to formalize the transition from the concepts of **Activity** and **Resource** to the concept of **Space**. Indeed, this transition is poorly (practically) supported in current approaches except by the programmers' experience. Section 6 introduces a graphical model (i.e. diagram) to support this formalisation with practical tools. Section 7 concludes this chapter.

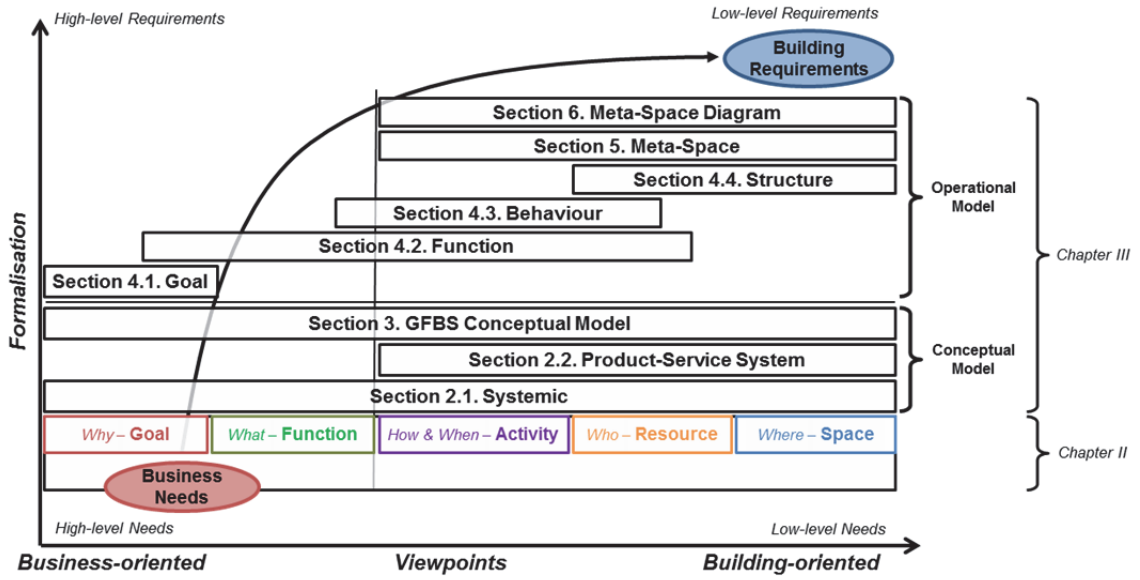


Figure 33 – Plan of Chapter III

1. Introduction

How to model the building and its definition domain according to the multiple (views) stakeholders involved?

In Chapter II, we proposed a definition domain that structures the transition from low-level to high-level information about the object of study through five viewpoints organized by a level of abstraction. The proposed constructs describe the building definition domain without defining the perimeter of the final object of study, i.e. the building. In Chapter III, we propose a set of models to state the (external and internal) boundaries of the object of study (Section 2), to articulate its description (from outside to inside) (Section 3 and 5), and to provide (semi-)formalized viewpoints on its definition domain (Section 4 and 6).

"A model *is a set of propositions or statements expressing relationships among constructs. In design activities, models represent situations as problem and solution statements."* (March & Smith 1995)

In the next sections, we propose two sets of models: a set of conceptual models and a set of operational models. The conceptual models provide a set of abstract models to represent the boundaries of the building and to complete its definition domain. They are introduced in Section 2 and 3.

In Section 2, the boundaries of the object of study are modelled through systemic and the *Product-Service System* paradigm. This last model allows understanding the "*bicephalous*" nature of buildings (i.e. its static and dynamic parts).

Section 3 proposes a modelling of the transitions between the defined constructs through Gero's *FBS* paradigm to distinguish the realization of **Functions** by a *Product* part, a *Service* part or a combination of both. Far from just modelling and structuring the transitions between constructs, the *FBS* paradigm completes the definition domain from Chapter II with additional constructs: *Behaviour*, *Property*, *Structure*, and *Characteristic*.

The operational models formalise the definition domain through practical models that can be used by the programmer to represent it through different viewpoints. They are introduced in Sections 4, 5, and 6.

Section 4 presents a multidisciplinary literature review of modelling languages that can be used to represent the definition domain through different, complementary, viewpoints. The modelling languages are introduced as a (semi-formal) way to facilitate the apprehension of the definition domain by the different stakeholders of buildings (e.g. paying, user, and customer clients, programmer, designer, etc.).

Based on conclusions from Section 4, Section 5 introduces a new design artefact to structure the transition from **Activities** to **Spaces**: the *Meta-Space*. This transition is currently ensured by the programmer (through his/her experience) in a single way. This means that once the definition of **Spaces** is done, **Activities** associated to it are difficult to trace back or change. The proposed design artefact structures and formalises the transition from **Activities** to **Spaces** but also from **Spaces** to **Activities**.

Based on this new design artefact, Section 6 introduces a new modelling tool, the *meta-space diagram*, to support the transitions between the main deliverables that manipulate **Activities** and **Spaces** constructs (i.e. the brief, the functional diagram, and the 2D drawings). Through its use, the programmer can ensure certain consistency between information contained in the various deliverables through different viewpoints.

2. Modelling the Object of Study: Building as a System

A building is already acknowledged as a multi-layered complex system (Bertelsen 2003). Its first level of complexity comes from its design, industrialisation and production (i.e. construction) regarding technical building services (Kubicki et al. 2006). This research focuses on another layer

of its complexity regarding its definition and description during the briefing process. This complexity is related to the "*bicephalous*" nature of buildings. Indeed, buildings are not stand alone artefacts but complex systems composed of a static part (i.e. **Spaces** and **Resources**) and a dynamic part (i.e. **Activities**).

In this section, this complexity is introduced through different conceptual models used to formalise the boundaries of the object of study. Each one of them provides a different, complementary, viewpoint on the object of study. Section 2.1 models the boundaries of the object of study through a systemic paradigm. Section 2.2 models the (internal) "*bicéphale*" nature of buildings through a Product-Service System paradigm. Section 2.3 introduces the alignment principle that models the relationships between the components of such complex system. Section 2.4 concludes this first level of conceptual formalisation.

2.1. Systemic

The concept of *System* is used to model complex systems in an abstract view that makes them easier to deal with. It provides a solution-neutral start to define a design problem. *System* is defined as a "*combination of interacting elements organized to achieve one or more stated purposes*" (ISO/IEC 2008a; INCOSE 2010). To define a *system*, three concepts have to be considered: requirements, scope, and architecture (Faisandier 2011). Requirements refer to the expected outcomes provided by the *system*, in terms of services and constraints. This is the main outcome of the requirements definition. Scope refers to the limit of the *system*, what is included in it and what are its interfaces with the outside world. Architecture refers to the clarification of the *system*'s operating and its physical structure (i.e. components organisation). This section clarifies the scope and architecture of the studied *system* using as support example: the LTB case study (introduced in Chapter I Section 4.4.1).

2.1.1. Requirements

When a town needs a new school because of its growing number of inhabitants and the overcrowding of the current school, it does not just need a new building. It requires new teachers to provide education in this new building, administrative staff to manage this new school, new furniture, may be a new IT system, specific equipment to perform new activities and so on. Moreover, it may not be necessary that these elements are brand new. They might be old stuff to renovate, recycle, or simply reuse.

So, Architectural Programming is not just about the requirements definition of a building but rather the requirements definition of a complex *system*. The main output is the brief that leads the architect team to design the building part of this *system* but the building itself is not enough to satisfy the real needs of the clients. A building is composed of **spaces**, **spaces** are required to perform **activities**, **activities** realize **functions**, **functions** achieve **goals**, **goals** state the clients' needs, and clients' needs depend on the context. As a result, in order to define the building requirements, the programmer needs to understand the context and how the clients' want to achieve their **goals**. The context, **goals**, and the way to achieve them make the *system* unique. This is why copy-paste of existing solutions cannot provide a good answer to clients' needs. This is one of the specificities of the Architecture-Engineering-Construction domain.

2.1.2. Scope

To define the scope of the studied *system*, the first thing to understand is its *environment*. The *environment* represents the as-is context (i.e. without the system). System Engineering provides this definition for the concept of *Environment*: "*the surroundings (natural or man-made) in which the system-of-interest is utilized and supported; or in which the system is being developed, produced or retired*" (INCOSE 2010).

In Software Engineering, the term *Environment* is often replaced by the term *Domain*, in particular in the Problem Frames approach (Jackson & Zave 1993). System Engineering's definition is implicitly limited to physical objects whereas the concept of *Domain* is considered broader, integrating intangible artefacts such as knowledge or know-how. This concept provides

additional elements to the definition of *Environment*: independency regarding the *system*, broader nature of the surroundings, and a distinction between as-is situation and to-be situation. Some of these elements are used to define the concept of *Environment* across Software Engineering standards (IEEE 2010). In this research, the term *Environment* is kept but its definition is expanded:

Definition 6: An **Environment** is a specific "as-is" context integrating all the *system*'s surroundings that will impact or be impacted by, directly or not, the *system*'s existence. The surroundings include any environmental factors or external elements (including circumstances, conditions, and knowledge) that exist independently from the system's existence.

The **environment** of a school building consists inter alia of the inhabitant of the town and their children, the neighbourhood and its "geology", the Ministry of Education (i.e. its local representative), knowledge in education and pedagogy, and various regulations and standards. In the System Engineering view, knowledge in education and pedagogy would have not been considered. Each one of these artefacts exists independently from the *system* to define. The existing artefacts are called "*external elements*" or "*interactive agent*" and composed the **environment** in Value Management (AFNOR 1996).

Definition 7: An **External Element** is an independent already existing artefact, physical or not, which behaviour or condition will impact or be impacted directly or indirectly by the system.

The scope of a *system* is delimited by all the artefacts required to satisfy the clients' needs. The artefacts that composed the *system* are unknown at first (system as a black box, Figure 34) then they are defined along the requirements definition and design process (system as a white box). The concepts of **Goal** and **Function** are associated to the concept of *System* as a black box. These concepts are defined independently from the solutions (i.e. how to). **Goals** represent the expected outcomes whereas **functions** represent the interfaces between the *system* and its **environment**. This perspective corresponds to an external functional analysis (or functional analysis of needs) in Value Management (AFNOR 1996). The following definition is considered in this research:

Definition 8: A **System** is the object of study seen first as a black box representing a collection of structured and organized (inter) dependent components (i.e. sub-systems) either tangible or intangible working together to produce an expected effect on its environment. To be dependent, each component has to be designed, hired, bought, or created in the frame of the system development.

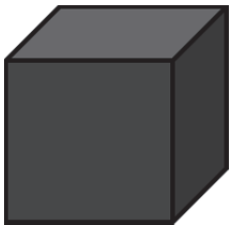


Figure 34 – System as a black box

2.1.3. Architecture

Architecture refers to the clarification of the **system**'s operating and its physical structure (i.e. components organisation). When dealing with its architecture, a **system** is considered as a white box (Figure 35) and all of its components can be seen and studied. The components of a **system** are called *internal elements* in opposition to **external elements** or *resources* as defined in Value Management taxonomy (AFNOR 1996). The concept of *Internal Element* still needs to be distinguished from the concept of **Resource**. In this research, a **resource** can be either internal or external to the **system**. Both concepts refer to two different, but complementary viewpoints. *Internal* and **External Element** refer to the systemic view whereas **Resource** refers to the concept of **Activity**.

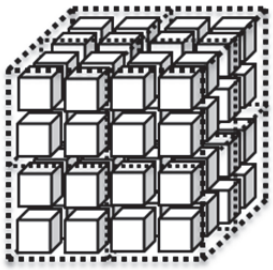


Figure 35 – System as a white box

Definition 9: An **Internal Element** is any artefact, physical or not, which existence is totally dependent on the system existence and which contributes to its proper operating.

In the school building example, a teacher is considered as a **resource** necessary to perform the teaching **activity**. The teacher can be a trained person hired by the National Education Ministry or a visiting teacher temporary paid by the same entity. The hired person is an **internal element** whereas the visiting teacher is an **external element** (Figure 36). The visiting teacher is independent from this specific school whereas the hired teacher is totally dependent on the creation of this school. To go further, the trained person and the hired person are the same person but only the trained person exist independently from the school whereas the hired person cannot exist if there is no school. So the trained person is considered as an **external element** whereas the hired person (i.e. teacher) is an **internal element**. As a short cut, we could consider the hired person as an internal resource and the stand-in teacher as an external resource. The transition from a trained person to a hired person entitled as teacher is part of the system. This transition is part of the interface between the system and its environment.

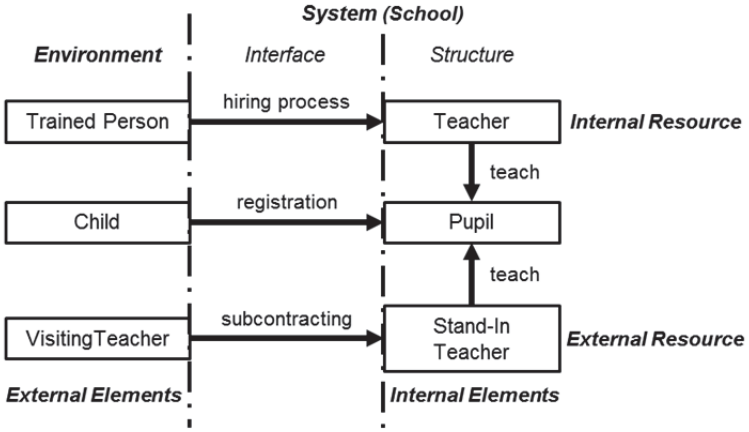


Figure 36 – Scope of the studied system illustrated on the school building example

The interest of such distinction has an importance when dealing with alternatives, innovation, and changes. It allows making a difference between manipulated artefacts regarding their degree of dependency on the system. If someone wants to shut down a department, people who work in the department will not cease to exist but their position and activities will disappear. It also transcribes the potential influence of the programmer in their definition. **External elements** can only be described "as-is" or estimated "to-be" whereas **internal elements** can be defined and designed as pleased.

This aspect will be more explained in the next section through the *Product-Service System* paradigm regarding resource management. "*Do we wash the laundry ourselves or do we subcontract?*" The choice between the two alternatives will have an impact in terms of spatial requirements (e.g. laundry room or loading dock) and organisation (e.g. washing machine management or laundry logistic).

2.1.4. Proposed Model

In this section, the concepts of **System** and systemic are used to model the object of study of this research. The concept of **System** allows dividing the briefing process in two parts. The first part is concerned with the **system**'s **environment** and the as-is situation to define the to-be situation. The focus is on the problem definition and the understanding of the domain of interest. **External elements** exist independently from the **system**. They have to be described whereas their relationships with the **system** have to be defined. The **system**, considered as a black box, is therefore defined in a solution-neutral and abstract way.

The second part is concerned with the **system**'s architecture, i.e. its composition and the organisation of its components. The **system** is considered as a white box and the focus is on the **system**'s operating (how to). **Internal elements** represent artefacts that do not exist from the beginning but they are required to be defined based on the **system** definition. The **system** is defined in terms of solutions using components to design, buy, hire, or create.

The systemic model introduces new constructs defined above and represented in Figure 37. Through these concepts, a distinction has been made between elements from the **environment** and elements from the **system**. The nature of the **system**'s elements was also extended from the single building artefact to some organisational artefacts. In the basic application of Functional Analysis, artefacts such as teachers, pupils would have been considered as **external elements** whereas they are integrated in the scope of the **system** in this research. Teachers are replaced by trained person whereas pupils are replaced by inhabitant's children. "*Teacher*" and "*pupil*" are considered as roles play through the building system and not physical artefacts. Then their relationship has to be defined in the **system** through its operation. This simple distinction implies the documentation of the roles and their integration in the **system**. This participates to a better understanding of the building system. By documenting the relationships between a teacher and pupils, one would better understand its complexity.

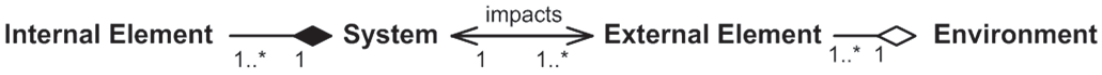


Figure 37 – Conceptual model of the System concept

The programming activity and technical solutions are different when dealing with **internal** or **external elements**. **External elements** are described based on existing information. The programmer can gather factual information on them. Time has an impact on them. **Internal elements** can only be defined by the programmer and will later be designed by the architect, bought by the paying clients, hired by the user clients, or built by the contractors. Their materialisation will not be effective before the end of the **system** development. They are subjected to changes from the clients whereas **external elements** are subjected to time.

In this research, the scope of the object of study is not limited to a physical artefact (i.e. the building) but enlarged to the service provided through it. Its integration in the scope introduces a

new complexity in the requirements definition of the **system**. Two parts of different nature can be distinguished: a physical part (i.e. the building) and an organizational part (i.e. the provided services). The object of study is then considered as an "*operating system*" where architectural programming aims ultimately to define the physical part. The dependency between its two parts and their mutual integration are developed in the next section.

2.2. Building as a Product-Service System

Architectural programming does not just aim to define a building but a whole "*operating system*" answering clients' needs. This "*operating system*" is composed of several elements of different nature. From a design perspective, the most interesting one is the building, but from a customer perspective, the most important one is the human activity performed inside it (i.e. the provided services).

In order to clarify the AEC specificities, the example of a public school is taken. When talking about public building in AEC, three kinds of clients are clearly distinguished: the paying client, the user client, and the customer client. In other engineering domains, as well as with other kinds of buildings, it is quite common that at least two of them are the same person. In the school example, the paying client is the Ministry of National Education who pays for a new school. The user client is the personnel (e.g. teachers, director, or janitor) required to provide (more or less directly) education to the customer clients identified as the pupils. This being said, the studied system in this research is not limited to the school building but to the whole "*operating system*" required to provide education to these pupils. As a result, the building as well as the user client (e.g. teachers) and his human-intensive activity (e.g. education) are part of the studied **system**. The user integration in the studied system is already considered in the design of result-oriented *Product-Service System*. In this section, the *Product-Service System (PSS)* paradigm is introduced as a support for modelling both parts of the studied **system**: the building and the service provided through it.

2.2.1. Introductory example

In order to explain why buildings could be seen as *PSS*, we first propose a concrete example based on the LTB case study. The main service provided by a school is education. The owner of the building is the same as the owner of the service i.e. the State. The State imposes directives to the superintendent and asks him to apply them in the future school. According to these directives and in association with his board and sample of teachers, the superintendent chooses a new leitmotiv associated to this school: "*Learn to be*". They want to encourage mutual aid, communication and team building. This leitmotiv would lead reflections about the future building.

One of the main issues with the former building concerned bully zones. After an analysis of the previous design (Figure 38), it appears that the only "*safe zone*" in the school is situated near the administrative and teachers area. Regarding the leitmotiv of the future school, things had to change. The research team guided the contracting owner and his task force on this issue amongst others. First step was to define the proper requirement on the system (i.e. the building as a black box) using a Functional Analysis approach. The job was not easy because the task force had a solution thinking process instead of a requirement-oriented mind-set. The corresponding high level requirement formulated was "*every pupil inside the school could be seen by an adult*". The second step was the generation of solution principles (i.e. programmatic concepts which is different from design solutions). The main difficulty to overcome by the research team consist of making the task force understand that they were not limited to single principles such as the building or additional expensive equipment.

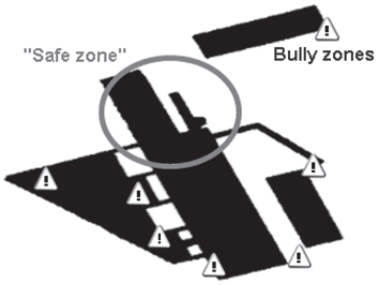


Figure 38 – Analysis of the former building plan view from above

Several solution principles are generated and retained by the superintendent's team to fulfil this main requirement:

- *Architecture and positioning.* The architect could propose a different layout and design shape of the spaces for watching improvement (like in prisons). Requirements to provide are about proximity between particular spaces.
- *Transparent materials.* The architect could favour the use of see-through materials to prevent bullying by lack of spaces privacy. Requirements to provide are about kind of materials to be used and visibility to and from particular spaces.
- *Teachers and rounds.* The superintendent could ask school's personnel to do some rounds scheduled by pair during the breaks. Independently from the design of the building, all the requirements concern only the organization and people activity.
- *CCTV system and guards.* Independently from the design by the architect, the superintendent could install a CCTV system and mobilise the janitor for supervision. Requirements would concern equipment, network and the janitor's office.
- *Hybrid solution principle.* The last possibility would be to combine positioning of the adults space around the school with a direct view on common spaces and circulation, the use of see-through material for their offices in order to allow a passive watch of the pupils by the adults. Requirements to provide would concern proximity of spaces and materials used for the walls.

Each one of these solution principles has an impact on different parts of the school: e.g. the building, the organization, the people or the equipment. Requirements on the building then differ and give orientations toward the proper solution. The role of the architect is to design buildings, not organisations. The choice between alternatives has to be made by the customer (i.e. the superintendent). Then, correspondent requirements will be given to the architect. It is part of his responsibility as contracting owner. Furthermore, the architect cannot make this decision instead of the superintendent as the cost of each solution principle not only impacts the building but also the long term use of the building, i.e. the organisation. His decision is led by the education service to be provided. It does not aim at watching pupils' acts or depriving them from privacy by any means as in a maximum-security prison. The point is on ensuring their well-being outside class hours to avoid stress and poor concentration due to bullying and intimidation.

This example presents different parts of a building by exploring a set of solution principles. The last principle focuses on the building as a way of ensuring the safety of pupils with as little impact as possible on other components of the building including day-to-day activities and processes. It underlines the fact that the building is not a standalone artefact but a component of a bigger system to be defined according to the service to be provided. As a consequence, the building as well as the services becomes a part of a "building system" to be defined by the customer. Other parts of this "building system" are: the people who work inside it, the equipment required to support the people's work, the energy to be supplied to the equipment and building, the information technology system which stores information about people and processes, and the business which organises and manages the whole system. Definition of each part should be done

during the requirements definition phase in an integrated process to ensure consistency and alignment between them.

Current practices in construction leave the customer focus on the building with few in depth discussions on the other parts and no proper formalisation of the solution principles. Interactions between the building systems' parts, their synergies, are not developed as it could be. They are defined later depending on the design of the building and most of the time after construction of the building. As a consequence, a couple of patches are required to adapt the building leading to extra costs and delays. PSS mind-set could contribute to improve the degree of integration and alignment of each components of a building system. Each one of them is related to a specific discipline.

2.2.2. Product-Service System

Product-Service System (PSS) is mainly presented in literature as a business model. It focuses on functionality or usages to provide to consumers instead of selling products (Meier et al. 2010). The idea is to sell a marketable mix of products and services that will jointly satisfy the consumers' needs (Goedkoop et al. 1999) and increase at the same time the market proposition (Mont 2000) by integrating services to traditional product functionality (Baines et al. 2007). Moritz provides an overview of their most prominent differences with a service design point of view (Table 11) based on (Bachmann 1998).

Table 11 – Overview of prominent differences between Product and Service (Moritz 2005)

PRODUCT	SERVICE
Produced	Performed
Material	Immaterial
Tangible	Intangible
Can be stored	Can't be stored
Usually without client	Interaction with client
Consumption after production	Consumption = production
Defects in manufacturing	Mistakes in behaviour

In this research, PSS is presented as a mind-set to structure a vision of building systems richer than a standalone artefact. This vision allows the integration of the service provided through the building in the scope of the studied system. There are mainly three kinds of PSSs considered in literature (Figure 39): product-oriented PSS, use-oriented PSS, and result-oriented PSS (Manzini & Vezzoli 2003; Yang et al. 2009; Tukker & Tischner 2006). In terms of business models, the main difference between each one could be synthesised into the degree of ownership sold to the consumer: property of the product, property of its use, and property of its results (Cook et al. 2006). It demonstrates a shift in business strategy from a product-oriented to a service-oriented focus (Tan et al. 2009) where the value of the product is transferred to its utility or purpose (i.e. the result) from the client's perspective (Vasantha et al. 2012). Other classifications of PSSs exist (Behrendt et al. 2003; Brezet et al. 2001; Zaring et al. 2001), more detailed, like Meijkamp's (Meijkamp 1998) or White's classifications (Mont 2000; Mont 2002) but we limit our mind-set analysis to the three main types of PSSs based on a literature review in this research.

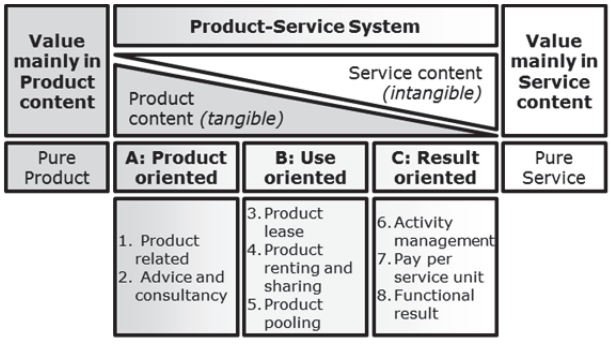


Figure 39 – Main and subcategories of PSS from (Tukker 2004)

- **Product-oriented PSS** – A product is enhanced with services as a support of its use along its life-cycle from the selling (e.g. financial facilities) to the disposal (e.g. recycling). The client owns the product and has the possibility to buy additional services to improve his experience. The product is little, or not, impacted by these *service additions*. Its lifetime is expected to be improved compared to a standard manufacturing product (Cook et al. 2006).
- **Use-oriented PSS** – A product is no longer sold to the client, it remains the property of the provider. The client benefits from its use over a determined period he paid for. The product is impacted in its design as the main purpose is to have high usage intensity to generate income (Yang et al. 2009). There is *service integration* to a usable/available product. This integration requires a re-engineering of an existing product to adapt some of its properties in terms of robustness and integrity.
- **Result-oriented PSS** – The product is no longer an issue for the client, only the quality of the result is. A functionality provided by a product is replaced by a pure service. There is *service substitution* (Sundin et al. 2009) to a buyable product. In this case, the providers' degree of expertise about the product's use is somehow highlighted or recognized which guarantees a higher efficiency. Consumer goods become professional equipment which requires high technical skills when manipulated.

For each type, a product is the starting point and services are added, integrated, or substituted. The final solution is composed of two kinds of artefacts: a product part and a service part. Each part of the system is capable of satisfying specific consumers' needs, complementarily or jointly (Manzini & Vezzoli 2003), through an anticipated integration from the early stages of development. A proper alignment between each part of the PSS provides synergies and added values for providers and consumers in a win-win situation. PSSs are expected to significantly reduce resource deployment and improve their efficiency (Cook et al. 2006). This mind-set is setup in the earliest phase of a project, during the requirements definition and conceptual design phase. Regarding the scoped system defined in Section 2.1, a building can be seen as such a system composed of a product part and a service part.

2.2.3. Buildings as Product-Service Systems

In this research, regarding the studied "*operating system*", any tangible artefact (i.e. **resource**) required to perform an **activity** is considered to be the product part. As a result, building (i.e. **Space**), furniture (e.g. tables, chairs) and equipment (e.g. HVAC components, plumbing fixtures) (i.e. material resources), and people (i.e. human resources) are considered as product parts of the "*operating system*".

The concept of *Service* has several meanings in AEC that differ from PSS. The first one concerns the technical building services that basically give life to the building (e.g. HVAC systems). Technical building services are composed of product parts of the operating system (e.g. heat pump, air vent, or conduit). The second one is more related to public services (e.g. education) referring to human-intensive **activities** performed by user clients to customer clients inside the building. In this sense, the customer client is assimilated to the consumer of the service.

Therefore, PSS is used in a broader sense, as a model to represent the defined "*operating system*" composed of interdependent product parts (e.g. building, furniture, equipment) and service parts (i.e. human-intensive activity). Regarding the public school example, the operating system has to provide education to pupils throughout the building, equipment, furniture, and personal.

With result-oriented PSS, products are substituted by services (e.g. laundry services) (Sundin et al. 2009). In this specific case, two kinds of service content can be distinguished: *core services* and *support services* (Yang et al. 2010) (Figure 40). *Core services* are customer-oriented (e.g. cleaning sheets) whereas *support services* are product-oriented (e.g. maintenance of the washing machines).

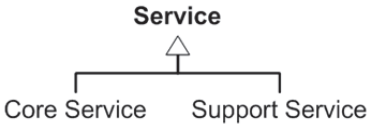


Figure 40 – Taxonomy of Services

This distinction is adopted for AEC in this research. However, the concept of *support service* is broadened to any secondary service that supports the *core service*. Regarding the school building example, any user (e.g. teachers) activity performed to provide value to the customers (e.g. pupils) is then considered as a service. The public service (i.e. education) corresponds to the *core service* of the "*operating system*". In the school example, the administrative, financial, and human resource services are considered as a support service. The technical building services (e.g. HVAC systems) can be considered as internal *support services* whereas cleaning and maintenance services (i.e. servicing) can be considered as external *support services*. During the architectural programming, all of these services have to be considered, especially in the context of sustainable development. Current literature on sustainability in AEC seems to focus more on the *support services* and issues regarding energy efficiency and environmental performances (Ding 2008) rather than the *core services*.

Definition 11.1: A **Core Service** is a customer-oriented service from which the customer will receive the desired value added.

Definition 11.2: A **Support Service** is a product-oriented service or a secondary service required to perform the core service. Its existence is dependent on the existence of the core service or the product.

Another research trail is introduced to contribute to sustainable development through a better alignment between the product part and the service part that compose a building. Both parts can contribute to the customers' needs satisfaction but there is a lack of theory to support the assignment of functions to the product part, the service part, or to an integrated combination of them.

2.2.4. Service-oriented PSS

In public construction, a building is a way to provide a service to the community and not an end in itself. The starting point of its definition is the service to provide. Therefore, the three types of PSSs presented earlier do not cope with buildings as all of them start from a product perspective. As a result, the concept of *Service-Oriented PSS* (Mauger et al. 2013) is proposed to support the view of the building system defended in this research.

Existing PSSs are about *service* integration whereas *Service-Oriented PSS* (SoPSS) is about *product* integration. The *service* is the starting point of the reasoning and represents the major part of the system. The *product* part of the building system has to support the *service* delivery. Integration of the *product* part to the definition of the *service* leads to improvements through a better alignment. The *product* design could have an important impact on the *service* delivery for this kind of PSS. This impact can be either good or bad, and *service* is quite sensitive to it. Poor *product* design leads to poor quality of *service* delivery. This poor design is often caused by a poor understanding of the *service* activities and a lack of completeness of its process descriptions. The design of the support *products* depends on the *service* to be provided and not on the business model as for the other kinds of PSS. The idea could be compared to Shimomura's description of the essence of *service* design, i.e. "*what to offer*" and "*how to satisfy customers*" (Shimomura et al. 2009) as the first step towards *product* definition/integration. Depending on the *service* to provide, derived **functions** could be implemented by humanware (i.e. human resources) or

hardware (i.e. material resources), i.e. the different kinds of components of the PSS. Therefore, the concepts of *Product* and *Service* are redefined in the context of the building system as:

Definition 10: A **Product** is a tangible or non-tangible resource or space required to perform an activity contributing to the provision of a service.

Definition 11: A **Service** is a human or computed/automated activity, performed by and using a set of tangible as well as non-tangible resources, for other humans or systems providing them value in different ways.

Table 12 provides a few illustrations of the concepts of product parts and service part of a building on the LTB case study.

Table 12 – Example of Products and Services based on the LTB Case Study

PSS	Type	Examples
Service part	Activity (Process)	Teaching
	Activity (Operation)	Learning, Assessing
	Activity (Process)	Recruitment
	Activity (Operation)	Interviewing, Hiring
Product part	Resource (Human Resource)	Teachers, Pupils, HR Manager
	Resource (Material Resource)	Tables, Chairs, Computers, Electricity, Network, Heating System
	Space	Classrooms, Offices

Regarding the definition domain (Figure 41), the **Service** part of a "*building system*" is described through the concept of **Activity**. The description of its **Product** part is covered by the concepts of **Resource** and **Space**.

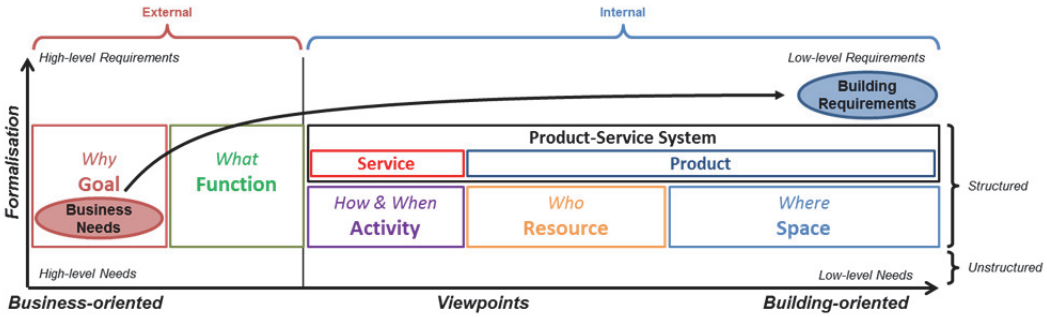


Figure 41 – PSS: First level of conceptual formalisation

Accuracy of the **service** definition improves the definition and alignment of the building but requires time and perspective. At present, "*building systems*" are defined with less integration between their **product** and **service** parts than potentially expected. The main input comes from the building and ratios instead of the **services** to provide. Sakao underlines in (Sakao & Lindahl 2012) that companies develop **services** after **product** realisation or even after release on the market. This is the same in construction. **Service**, material resources, and human resources are refined in later stages depending on the defined building. As a result, issues which appear after implementation might require changes or indicate a dysfunction in the **service** provision.

The proposed mind-set calls for deeper introspection by the customer on its **service** and "*building system*". It would help to anticipate issues with the architects' proposals and it would lead to more aligned proposals with **services** to provide. There is a need for a more detailed framework to guide the contracting owner through these questions. It should tackle the issue of product-service integration to provide synergies and improve the performance of the **service**.

2.3. Alignment Concept

In this section, the object of study is defined from a systemic viewpoint, and then it is refined through a Product-Service System perspective. The resulting **system**, qualified as "*bicephalous*", is composed of two main parts: a **product** part and a **service** part. The requirements definition of such **system** requires the understanding of both parts but also of their relationships and dependencies. The defended viewpoint in this research is that the **service** part should be defined first and conditioned the definition of the **product** part, especially concerning public services. Indeed, the main value added of a public building is not in the building itself but in the public service provided inside the building. The difference between this research and current architectural programming practices reside in the integration of the **service** part in the scope of the **system** to define. As a result, the **service** has to be more deeply investigated by the programmer and requires a greater introspection effort by the paying clients about the future building. The consequence of this effort takes the shape of an improved shared understanding of the future building. If the stakeholders do not know how the **services** will be provided, how could they be able to properly define the requirements about the building?

The principle of alignment applied in this research comes from Enterprise Engineering. This domain is concerned by Business–IT alignment and the design of enterprises in order to overcome current challenges (Zachman 1987). It is a multi-perspective approach that "*enables the achievement of organizational cohesion and integration*" (Greefhorst & Proper 2011). An enterprise is a complex dynamic system made of parallel and transversal processes (Vernadat 1996b). Enterprise architecture, through its panel of models, is an instrument to harness this complexity as a whole rather than step by step (Op't Land et al. 2009). The same principle is applied in this research. Regarding Architectural Programming, the aim is to align the building (**product**) part to the **service** part. Usually, the **service** is adapted to the building after construction with more or less changes on the building structure. These changes represent a consequent amount of money. In order to reduce the changes, the **service** part should be defined in priority with a certain degree of flexibility. Thereafter, the building can be defined in knowledge of consequences. Moreover, in the case of changes on the building part, the impacts on the **service** part can be traced and the decision making process can be supported by such elements.

This principle goes further in this research. The alignment is not limited to the building-service relationship but extended to all the components of the "*building system*" (i.e. "*operating system*"). Following the Service-oriented PSS paradigm, the building should be the last component defined. In the simplified view of the building system, all the other components are defined earlier in the process. Indeed, the building represents the most important and most expansive component of the **system** but also the one that contribute the less directly to the value added of the public services. In current modernisation of public services, all the formalities are simplified and available online. The interactions between customer clients and user clients are reduced to the minimum. Therefore, the building component is less and less perceived as necessarily "*big*" and can be rationalised. PSS is an alternative solution for sustainability. It brings new ways of improvement with a reduction of primary materials and physical artefacts, and an increasing of *dematerialisation* and *servitisation*. This reflection should be put in perspective of the opportunity and feasibility studies of every single public building. If solutions that do not involve the definition, design, and construction of a new building can be found, or at least that the size of a building can be reduced, then some substantial economies can be made.

2.4. In Summary

In this section, the object of study is proposed to be modelled as a complex system to deal with its "*bicephalous*" nature. This complex system is composed of a dynamic **Service** part and a static **Product** part modelled in the *Product-Service System* paradigm. The **Service** part is described through the **Activity** concept whereas the **Product** part is described through **Resources** and **Spaces** (Figure 42). The building which represents one of the **Product** parts depends on its definition to the **Service** part. The concept of Service-oriented PSS is introduced to model this

dependency and subjection. To conclude on the relationships between the two mains parts of this SoPSS, the principle of alignment is introduced.

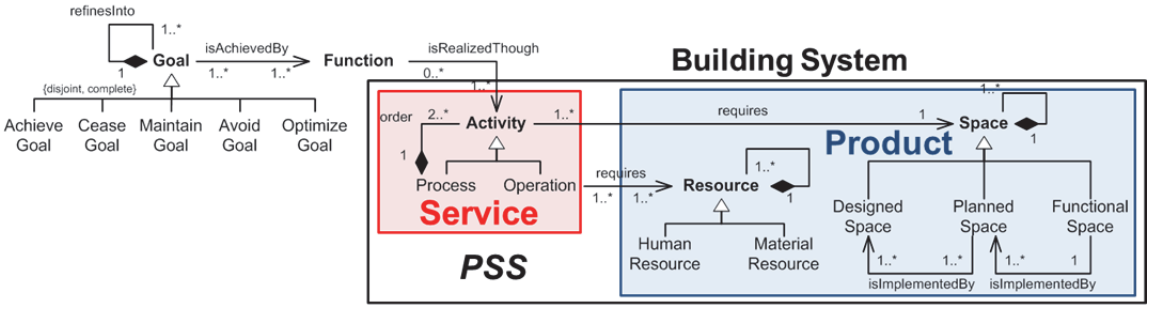


Figure 42 – Representation of the Coverage of PSS parts on the Defined Constructs

With this PSS paradigm comes a new issue relevant to the conceptual design phase of PSS development: the **function** allocation (Muller et al. 2007) to the **Product** part, the **Service** part, or, in our case, a combination of both. The next section proposes to use the *FBS* paradigm to model this allocation of **Functions** and to structure the transitions from **Goals** to **Spaces**.

3. A Model for Architectural Programming

The FBS paradigm developed by Gero models the design process using relationships between three main constructs: *Function*, *Behaviour*, and *Structure* (Gero 1990). This *FBS* paradigm is used as a basis to model and structure the transition from business needs (i.e. **Goals**), to building requirements (i.e. **Spaces**) and the allocation of **functions** to the different parts of our *SoPSS building system*.

In Section 3.1, we position our design model based on PSS and FBS paradigm regarding an (non-exhaustive) overview of the literature on design theory. Section 3.2 presents the FBS paradigm as developed by Gero and his collaborators through a literature review. Based on this literature review, Section 3.3 proposes an adaptation of the FBS constructs definition to our *SoPSS building system*. The resulting conceptual model of the briefing process (i.e. GFBS model) is presented in Section 3.4. Section 3.5 concludes this second level of conceptual formalisation of the definition domain.

3.1. Models from Design Theory

Models for design theory cover a large spectre of systems and disciplines. Most of them concerns physical artefacts with their origin in architecture. In architecture, the first one to appear is the building model through the Vitruvian model (Morgan 1914) based on Vitruvius writings (Vitruvii 25BC), the man-environment paradigm (Hillier & Leaman 1973), or the form-follows-function model (Whitford 2005; Le Corbusier 1923). The second model in architecture shifts the focus from the building product to its briefing and project process through the brief/project model , the analysis/synthesis model (Alexander 1971), and the define/solve problem (Simon 1973). From the Vitruvius approach, design models were developed in engineering disciplines after WWII (Le Masson et al. 2013). Most of them mainly focus on product and process systems such as Systematic Design (Pahl & Beitz 1995), the Theory of Technical Systems (Hubka et al. 1988), FBS (Gero 1990), Axiomatic Design (Suh 1998), or C-K Theory (Hatchuel & Weil 2003) amongst others.

The concept of service appeared more recently, in the early 1990s (Meyer et al. 2002). Compared to Moritz' seven qualities (Moritz 2005), three qualities are considered as basic for most of the services: activity components rather than things, simultaneous production-consumption, and the client involvement in the process (Karni & Kaner 2007). Even if these qualities can be discussed, it provides certain enlightenment about the gap with existing design theory models. Current industrial problems mainly concern the environment and the impact of mass production/consumption of artefacts (Tomiyama 1997; Umeda et al. 2000). One of the

identified solutions is the concept of Product-Service Systems (PSS): a business model change from product ownership toward utilisation and functionality selling (Mont 2000). The paradigm shift from product and service systems to product-service systems leads to a novel engineering called service engineering (Arai & Shimomura 2007) and new design models more adapted to such integrated systems. At the moment, there is a lack of PSS methodologies (Müller & Blessing 2007) and a great amount of research is still required (Vasantha et al. 2012).

Design in Architecture, Engineering and Construction (AEC) can benefit from this paradigm shift even if industrial problems are different. As seen in Section 2, buildings can be modelled as PSS. Existing design models in architecture refer to either the building **product** alone or its briefing and project process. Both models do not fully integrate this **service** dimension of buildings whereas PSS models are mainly business-oriented.

Based on a critical analysis of our definition domain and an extensive (but non-exhaustive) literature review on design methods, models, and PSS, Gero's FBS approach was identified as a relevant framework for the development of a requirements definition model through the construct of *Behaviour*. The *Behaviour* construct includes the concept of **product** *properties*, something not integrated yet in our definition domain (Figure 42). A **product** *property* represents a "*passive*" *Behaviour* that can realise a **Function**. It is an alternative to an **Activity** which can be considered as an "*active*" *Behaviour* of the *PSS building system*. Therefore, we propose to use the *Behaviour* construct to model and support the **function** allocation to the **product** part through *properties* or the **service** part through *activities* of our *building systems*. Moreover, the construct of *Structure* provides an additional layer of conceptualisation to our definition domain. In the next section, the original *FBS* paradigm is presented before introducing our interpretation and adaptation of it to our object of study.

3.2. FBS

FBS was first introduced by Gero to model the design as a process and capture the nature of the concepts manipulated during this process through the use of knowledge representation diagram (Gero 1990). The basis of this proposition is Gero's FBS paradigm applied to structure more precisely the requirements definition of *building systems* with the following definitions (Kannengiesser & Gero 2011):

- *Function* – Teleology of the artefact, what is it for.
- *Behaviour* – Attributes derivable from the structure, what it does.
- *Structure* – Components and their relationships of an artefact, what it consist of.

The *FBS* "ontology" is not limited to the description of the design objects. This framework is also used to model the design process (Gero & Kannengiesser 2006) as precise as possible. In 14 years, the FBS framework has evolved from an eight fundamental processes representation to a twenty steps model (Figure 43) detailing the eight original ones in a three level perception of the world (i.e. the situated world). The starting point was originally the concept of *Function* as the purpose of the system to design. Later, it changed to become the concept of *Requirement*, an input given to the designer by the customer that indicates the design problem. The last step of the design process remains the same, the documentation (i.e. the description of the design, e.g. CAD drawings and component lists) used for construction or manufacture.

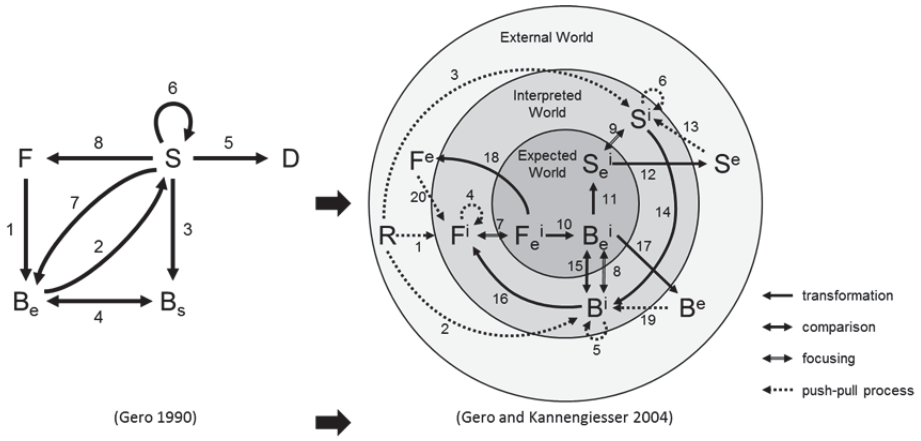


Figure 43 – Gero's FBS framework from (Gero 1990) *to* (Gero & Kannengiesser 2004)

The *FBS* framework was enriched on various occasions (Cascini & Frate 2011) by concepts such as *purpose* (Rosenman & Gero 1998; Rosenman & Gero 1999), *process* (Magner-Canet & Yannou 1998), *environment* (Deng et al. 2000), *resource* in FBS-PPR (Labrousse & Bernard 2003; Labrousse et al. 2004), *context* (Gero & Kannengiesser 2004), or *requirement* (i.e. constraint and performance) (Christophe et al. 2010; Kannengiesser & Gero 2011). Despite its apparent wide use, *FBS* still held quite a few ambiguities in the definition of its concepts (Dorst & Vermaas 2005). These concepts have evolved through time to become clearer (Table 13). Other approaches structure the design process around similar concepts to *FBS* such as RFLP for Requirements Function Logical Physical by Dassault System, Structure Behaviour Function (SBF) for programming language (Goel et al. 2009) or others proposals already detailed by Erden et al. (Erden et al. 2008). The basis of this proposition is Gero's *FBS* "ontology" applied to structure more precisely the requirements definition of building systems with the following agreed upon definitions based on (Kannengiesser & Gero 2011):

- *Function* – What the system should be able to achieve expressed in terms of purpose, what it is for.
- *Behaviour* – How the system is expected to perform its functions and when. It describes how the system and its components perform.
- *Structure* – Who performs the functions inside the system, which components, and their relationships. It describes the internal composition of the system, what it physically is, but also where it is in the construction.

Table 13 – Evolution of the FBS concept definitions

Reference	Function	Behaviour	Structure
(Gero 1990)	the design intentions or purposes	how the structure of an artefact achieves its function	the components which make up an artefact and their relationships
(Rosenman & Gero 1998)	the result of the artefact's behaviours	the artefact's actions or processes in given circumstances of the natural environment	the elements of the artefact, the material arrangement of these elements and their connectivity
(Gero & Kannengiesser 2004)	teleology of the artefact, what is it for	attributes derived or expected to be derived from the structure, what it does	components of the object and their relationship, what it is
(Dorst & Vermaas 2005)	the purpose of the design being designed	attributes derivable from structure or expected of structure	the elements of an artefact and their relationships
(Kannengiesser & Gero 2011)	teleology of the artefact, what is it for	attributes derivable from the structure, what it does	components and their relationships of an artefact, what it consist of

3.3. GFBS Constructs

The concept of *Requirements* is the input of the design process as modelled by Gero and Kannengiesser (Gero & Kannengiesser 2004). In our research, we define *requirements* as the output of the requirements definition. Following Goal-Oriented Requirements Engineering (GORE) principles (Lamsweerde 2009), the concept of *Requirement* is derived from the concept of **Goal**. A *requirement* is a **goal** refined into a statement about what the system will do (i.e. operationalization of the **goal**) (Letier & Lamsweerde 2002). **Goal** is, then, considered as the origin, the driving force, of the requirements definition. It provides the necessary rationales to make the designer understand the *Requirement* and to support the decision making process during the designer-customer interactions. The definition domain proposed in Chapter II structures the transition from **Goal** to *requirements* by adding intermediary constructs (i.e. **Function**, **Activity**, **Resource**, and **Space**). Based on the already established definitions and the relationships introduced in Gero's *FBS*, we propose our definitions of the GFBS constructs in this section.

3.3.1. Goal and Function

The Goal and Function constructs are not impacted by the FBS paradigm. Their definitions are just remembered before introducing the definitions of Behaviour and Structure.

Definition 1: A **Goal** is a state of affair or condition defined by the clients that the system (i.e. planned facility) must satisfy through cooperation of its components (i.e. sub-systems) and establishes the raison d'être of the system (i.e. answer to the "Why do we need it?" question)

Definition 2: A **Function** is an ability of action, granted by the facility to its clients allowing them to change their environment from a state A to a state B, maintain or avoid a state C, or optimize a state D.

Function and **Goal** are concepts related to the **system** as a black box (i.e. external part of it). They are mainly environment-oriented (Figure 44). Their combination provides answers to how things should be to satisfy the clients' needs. They provide limits and constraints on the end result and not on the manner or means.

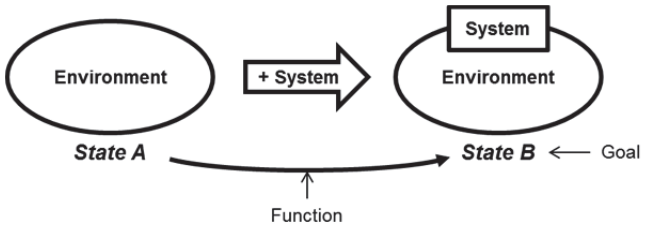


Figure 44 – Illustration of the Relationship between Goal and Function

3.3.2. Behaviour

The concept of *Behaviour* is considered in this research as an encompassing concept that is suitable for structuring the various constructs introduced in Chapter II as well as introducing a new complementary construct. *Behaviour*, first in this research, refers to the concepts of **Activity**, Process, and Operation (i.e. answer to the *how* interrogative). However the FBS definition of Behaviour introduces the concept of "*attribute derivable from the Structure*". Regarding the *PSS* paradigm, we, therefore, make a distinction between two kinds of *behaviours*: *behaviours* of the **Service** part (i.e. **Activity**) and *behaviours* of the **Product** part (i.e. attribute derivable from the Structure). In the Characteristics-Properties Modelling (CPM) approach, the product behaviours are referenced as "*properties*" (Weber 2005). Following Weber's definition, *properties* cannot be directly influenced by the designer (Weber 2005). Hubka & Eder refer to them as "*external*

properties" (Eder 2008) whereas in Axiomatic Design, they are defined as "*functional requirements*" (Suh 1998). *Properties* can be considered as *passive behaviours* whereas **activities** are *active behaviours*. The definition of *Behaviour* is restructured in order to reflect the various meanings encompassed in it for this research (Figure 45).

To illustrate the concept of *Behaviour*, Gero uses the window example (Gero 1990). The "light transmission" or "ventilation rates" expected from a window are considered as *behaviours* of the product, i.e. a *property* of the window following Weber's definition. Such *behaviours* are "natural" based on what is a window. It is called a *passive behaviour* of the window in this research. On the other hand, the action of "opening" or "closing" the window is not "natural" but "prompted". It refers to an activity performed by an external element, i.e. an *active behaviour* in this research.

Definition 12: A **Behaviour** is an activity (action or reaction) or a property of a system (or sub-system) under certain circumstances or triggering events, associated with or derived from its Structure (i.e. its components) explaining the "How and When does/can the system perform a Function?" question.

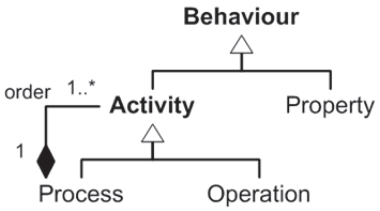


Figure 45 – Conceptual model of the Behaviour modelling construct

3.3.3. Structure

The concept of *Structure* is used in this research to regroup all the constructs that describe the components of a **system** and their organisation. It refers to the static part, i.e. the **product** part of PSS. The *structure* is described through *characteristic* attributes of the components, i.e. attributes that "*can be directly influenced or determined by the designer (e.g. material, shape, dimensions, etc.)*" (Weber 2005). They correspond to "*internal properties*" (Eder 2008) and "*design parameters*" (Suh 1998) in other design theories. The following definition is used in this research (Figure 46):

Definition 13: **Structure** refers to physical and virtual components of the system, their (static) description (i.e. characteristics), and their relationships describing what it is composed of, more generally answering the "Who (i.e. which resource) has to act in the system to perform a Function when needed, and Where is it performed?" question.

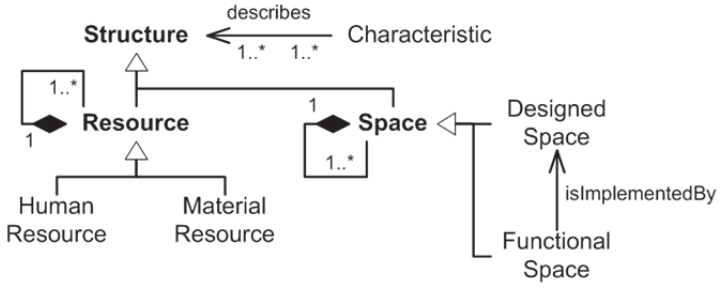


Figure 46 – Conceptual model of the Structure modelling construct

3.3.4. GFBS Constructs Conceptual Model

Based on the additional construct and definition, the conceptual model of the definition domain has to be updated to reflect the changes. **Goals** are still achieved by **Functions**. **Functions** are now realized through **Behaviours**. The **Behaviours** can be either an **Activity** (i.e. **behaviour** associated to the **Service** part), or a Property (i.e. **behaviour** associated to the **Product** part). **Behaviours** are assigned to a **Structure** element which can be a (human or material) **Resource**, or a **Space**. **Structure** elements are described by characteristics. These relationships are synthesized in Figure 47.

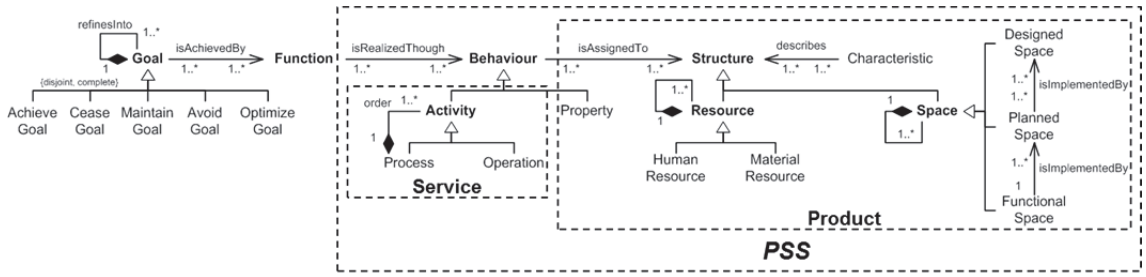


Figure 47 – Conceptual model of the GFBS modelling constructs

3.4. GFBS Model

Based on the defined ontology, GFBS and PSS modelling constructs are merged in a requirements definition model. GFBS structures the process whereas PSS structures the object of study. The **Goal** and **Function** constructs are associated to the global system (i.e. PSS) at a macro level. **Behaviour** is firstly associated to the global system (Sys) and then it is refined to its **Product** part (Pro) and **Service** part (Ser). To limit the complexity, the **Structure** concept is maintained on the full system. The GFBS design theory proposed in this section is structured around six steps (Figure 48) and it focuses on the system's design (not its components). These steps do not present back and forth. The back and forth loops have to be introduced in its instantiation in a briefing process. The Axiomatic Design principle of zigzagging (Suh 1998) should be kept in mind. Several design processes can be deduced from the proposed theory.

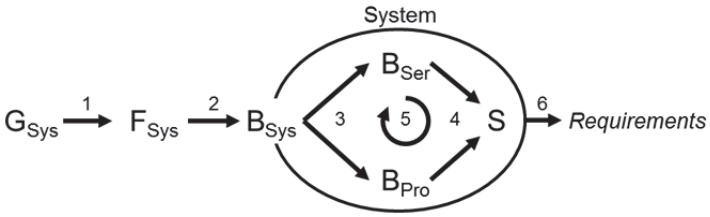


Figure 48 – GFBS conceptual model for PSS

In this research, design aims to find a solution to a current problem. The current problem is represented by the context of the system. This context is defined by unsatisfying facts about elements of the environment (i.e. external elements). Therefore, goals represent the desired state of affair or condition to achieve through the system's creation. **Goals** of the **System** (G_{Sys}) are the input of the design process; they are defined and refined with the client. **Goals** drive the decisions along the framework.

1. G_{Sys} ➔ F_{Sys} – **Function generation**: **functions** are abilities of action that contribute to the achievement of one or several **goals**, singularly or in combination with other **functions**. During this step, the designer transforms states of affairs or conditions of the environment to achieve or maintain into abilities of action by the **system**, independently from the technical solutions.
2. F_{Sys} ➔ B_{Sys} – **Function allocation**: at this step, the designer sketches how the **system** as a whole will behave to provide the required **functions**. It results in a description of

basic solution principles without details about the components (i.e. behavioural description of the system rather than structural description).

3. B_{Sys} ➔ $B_{Pro/Ser}$ – **Behavioural refinement**: based on the basic solution principles, the designer refines how the system will perform the **functions** in terms of product attributes (i.e. properties) (B_{Pro}) or service attribute (i.e. activities) (B_{Ser}). Product attributes (B_{Pro}) refer to expected behaviours of the product part of the system under specific conditions (i.e. passive behaviour) whereas service attributes (B_{Ser}) describe expected behaviours of the service part of the system (i.e. active behaviour). Decision to refine a system's behaviour (B_{Sys}) to either an active behaviour (B_{Ser}) or a passive behaviour (B_{Pro}) of its components is led by the defined system's goals (G_{Sys}).
4. $B_{Pro/Ser}$ ➔ S_{Sys} – **Structure synthesis**: expected behaviours are associated to elementary or combination of components of the system more or less independent. It is a nomenclature or a product breakdown structure listing all the required component of the system (i.e. structural description of the system).
5. **Requirements processing** or Behaviour-Structure balance: at this step, dependencies between behaviours and structure elements are checked for requirements consistency with the clients (e.g. limited number of structure elements can only ensure limited number of behaviours). A balance between them is performed to modify, reduce or delete requirements on the system according to the defined **goals** (Mauger & Kubicki 2013).
6. **Requirements specification**: this step consists of synthesizing resulting requirements about the system (i.e. globally), its components (i.e. locally), and their characteristics. It corresponds to the design description of the system as presented in Gero's model (Gero 1990).

Then, a new design phase starts: the design of the components. Based on these requirements, each component of the **system** (S_{Sys}) can be designed individually by the right designer (e.g. an architect for a building, or a cabinetmaker for furniture). This phase is modelled by Gero's FBS (Figure 49). A detailed illustration of GFBS conceptual model for architectural programming on the research case studies is proposed in Chapter IV to clarify the different steps.

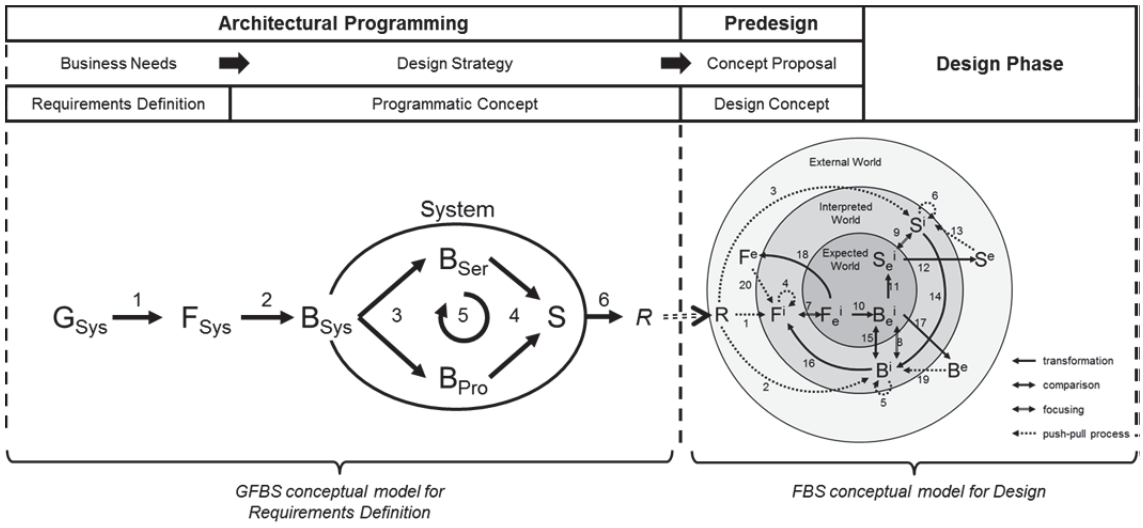


Figure 49 – Illustration of the continuity between GFBS and FBS conceptual models

3.5. In Summary

In this section, a FBS-based model was introduced to support the requirements definition of the buildings. The proposed model is developed based on Gero's FBS paradigm adapted to building systems modelled as PSS. The model consists of using the **Behaviour** concept as a

support for **Product** or **Service** distinction, assignment of **functions** and the extension of Gero's approach with the concept of **Goal** as a key element of decision support for the assignment. GFBS model provides a new conceptual layer to the definition domain proposed in Chapter II (Figure 50). This layer structures and formalises the transition from business needs to building requirements at the conceptual level. The next section tackles the operational formalisation of this definition domain.

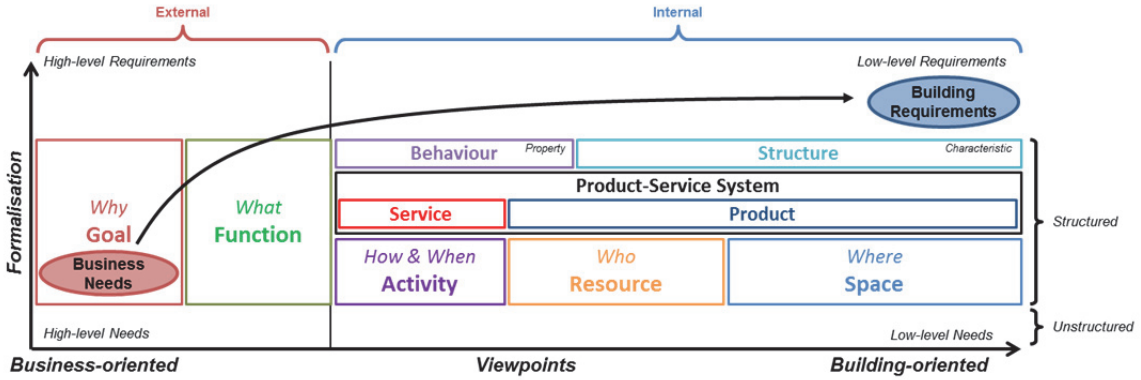


Figure 50 – GFBS: Second level of conceptual formalisation

4. Operational Models for a Multi-View Practical Formalisation

The briefing process generates a huge quantity of information from business needs to building requirements (Shen et al. 2004). In current practices, this amount of information is forwarded to architects through a brief, a textual document complemented with a few unstandardized diagrams. Digesting and handling such a huge quantity of information is quite a challenge in terms of consistency, completeness and redundancy.

In the first part of Chapter III, we proposed a set of conceptual models to structure the building definition domain through the PSS and FBS paradigms. The second part of this chapter focuses on operational (limited to semi-formal) models to practically support the requirements definition. Instead of designing a single all-inclusive model, we propose to use existing models to provide local views on the definition domain. Each existing model across engineering covers a set of constructs from the definition domain. As a result, the models provide the required multiple (complementary) viewpoints on the building definition domain.

The aim of this section is not to provide the state-of-the-art on modelling for each domain or a detailed critical analysis of the modelling languages but rather to demonstrate the interest of the discipline models to represent information on the *building system*. Any model from each domain could be used in practice as long as the building definition domain includes the information modelled. Only a short (abstract) description of the operational models is presented in this section. The reader will find illustrations of their application in the course of Chapter IV. This section is organised around the GFBS constructs: Section 4.1 starts with models focused on **Goal**, Section 4.2 follows with models focused on **Function**, Section 4.3 continues with models focused on **Behaviours**, and finally Section 4.4 ends with models focused on **Structure**. Section 4.5 concludes this section with an overview analysis of the presented models and their coverage of the definition domain.

4.1. Goal

In this research, the **Goal** construct is mainly taken from GORE in Software Engineering. The most common way to model **Goals** in GORE is goal trees. Each approach has its own modelling notation for almost the same constructs (Figure 51) and covers different phases of the software development process (Figure 52). The aim is not to analyse or compare them but just to provide a global idea of the constructs modelled by the different approaches.

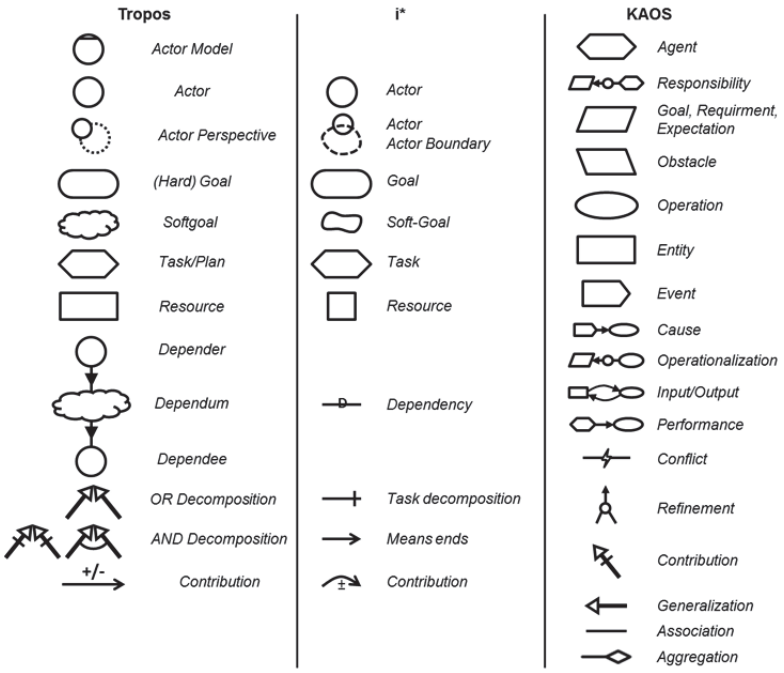


Figure 51 – Different GORE notations to model constructs

Comparison of the different frameworks and approaches are proposed across literature (Kavakli & Loucopoulos 2003; Anwer & Ikram 2006; Lapouchnian 2005). The aim in this research is not to favour one amongst the other but to introduce their value added in terms of graphical representation of information. The notation is not really important as long as it allows formalising valuable information for the clients, programmers, or architects.

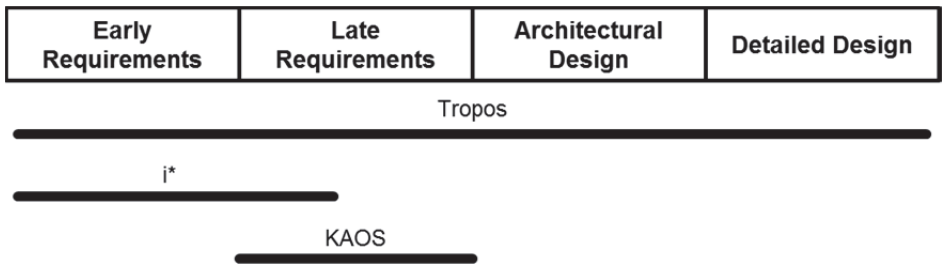


Figure 52 – Estimation of RE coverage by main GORE Techniques based on (Giunchiglia et al. 2001; Lapouchnian 2005)

Based on their notation and overview analysis, Figure 53 positions these three modelling languages on the definition domain. The coloured span represents the coverage of the modelling language with a darker colour on their main focus. The positioning is a rather subjective estimation of the coverage and is subject to discussions. However, the point is not to discuss the exact positioning but to provide elements of differentiation between modelling languages. Goal models do not include the concept of **Function** in their representation. Their focus is on **Goals** and their direct operationalization. In our research, the **Function** construct brings an abstraction layer (i.e. solution neutral expression of **goal** achievement) before dealing with the operationalization.

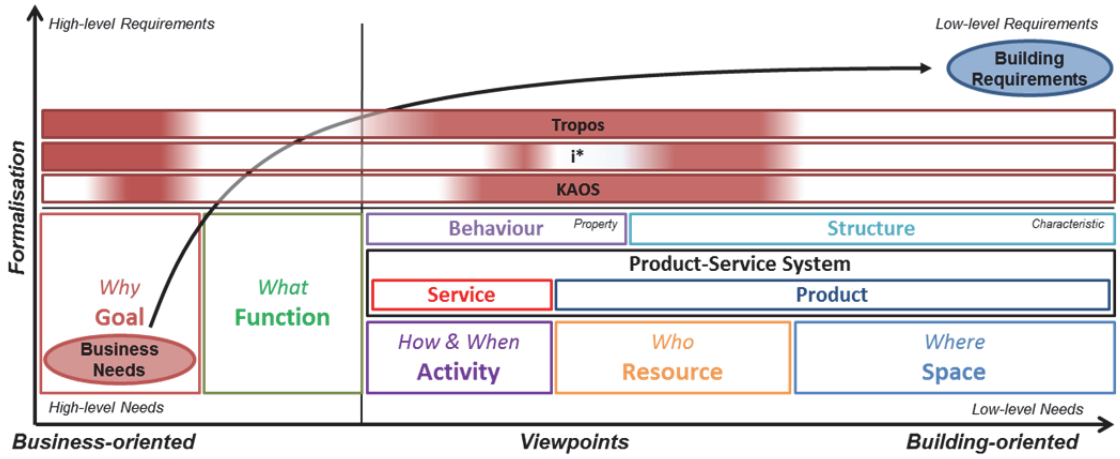


Figure 53 – Formalisation Coverage of GORE modelling languages

4.2. Function

Functional modelling corresponds to the activity of modelling systems based on their functionalities. It focuses on representing knowledge about *function* (Erden et al. 2008). The graphical framework plays an important part in order to benefit from shared understanding and automation. Its main use concerns the design phase of the system development. Each systematic design approach or method proposes its own definition and representation of the concept of *function* or functionality in a more or less operational way (Ross & Schoman 1977; Gero 1990; Pahl & Beitz 1995; Umeda & Tomiyama 1997; Suh 1998). In Engineering, as a general understanding, a *function* is defined as what a system does, i.e. an action performed by a product (system) or its components. *Functions* are defined on the basis of the clients' demands of the object to design.

In this research, functional modelling is limited to the graphical representation of the **Function** concept and the *building system operating*. However, a difference has to be clarified between the *building system operating* and the *building operating*. The *building system operating* refers to the description of the **service** part of the *building system* that requires **product** parts to be performed whereas the *building operating* refers to technical services of the building (i.e. **product** part). During the requirements definition, the building is not yet designed so as the technical services. Therefore, modelling them would be possible only after (or at least during) design of the building by the architects.

When considering a **system**, the concept of **Function** is divided in two types: *External Function* and *Internal Function*. *External functions* refer to what the **system**, considered as a black box, should be able to do (i.e. external description) whereas *internal functions* refer to how the **system** realises the *external function* (i.e. internal description). Mechanical Engineering provides graphical models that support and clarify this distinction. This section does not aim to provide the state-of-the-art of functional models across the literature as in (Erden et al. 2008; Eisenbart et al. 2012; Eisenbart et al. 2013) but rather to introduce a few models of particular interest regarding the building definition domain. The introduced models are mainly related to Functional Analysis and Value Management.

4.2.1. Interaction Diagram

The interaction diagram (i.e. "diagramme *pieuvre*" or octopus diagram in English) from the APTE method (de la Bretesche 2000; Société APTE 2007) models *external functions* as intermediaries between the **system** to define and its **environment** through *external elements* (Figure 54). It represents one of the first steps of the method and defines what the **system** should provide to its **environment**: "*the system should allow external element #1 to do something on*

external element #2". The **system** is considered as a black box and the **Functions** are defined in a solution-neutral way. This diagram is used to formalise the external description of the **system**.

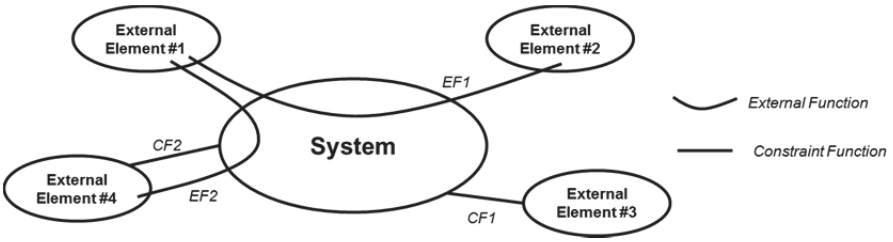


Figure 54 – Principle and convention of the interaction diagram

4.2.2. Function-Means Tree

Another interesting technique from APTE is the Technological Paths Tree (TPT) which is highly similar to the Function-Means Tree (FMT) (Tjalve 1978; Buur 1990; Bracewell & Sharpe 1996). These tree diagrams make the link between the external description of the **system** and alternative solutions to realise the **functions** (i.e. possible internal descriptions of the **system**). They are used for functional decomposition and to generate solution principles (Figure 55).

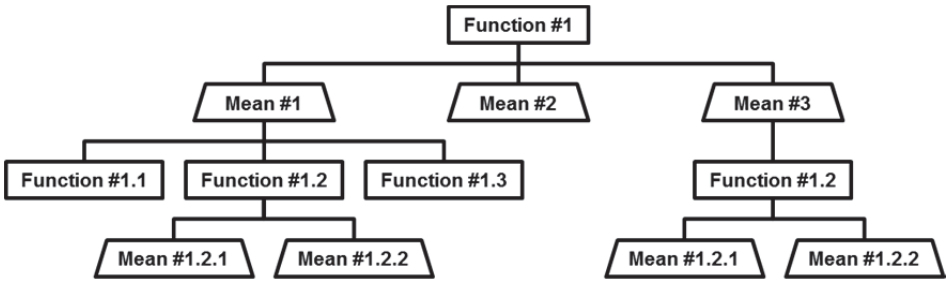


Figure 55 – Principle and convention of the Function-Means Tree

4.2.3. Function Analysis System Technique

Function Analysis System Technique (FAST) from Value Management (AFNOR 2000) models the *service functions* (i.e. *external functions*), the *technical functions* (i.e. *internal functions*), and the *solution principle* in a tree diagram (Figure 56). *Service functions* are realised through *technical functions* achieved by *solution principles*. The relationships formalise the link between the external description of the **system** and its internal description. Compared to the TPT of FMT, the FAST diagram only represents the selected solutions.

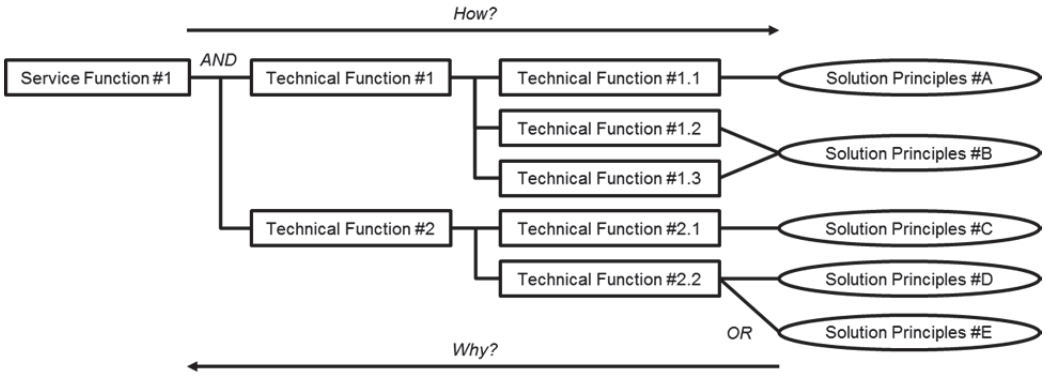


Figure 56 – Principle and conventions of the FAST diagram

4.2.4. Function Block Diagram

The last functional model introduced in this section on the **Function** construct is the Function Block Diagram (FBD). It models the interaction between the main *internal elements* of a **system**

and its *external elements* in the continuity of the FAST diagram (Maussang 2008) (Figure 57). This model closes the loop between the interaction diagram and FAST.

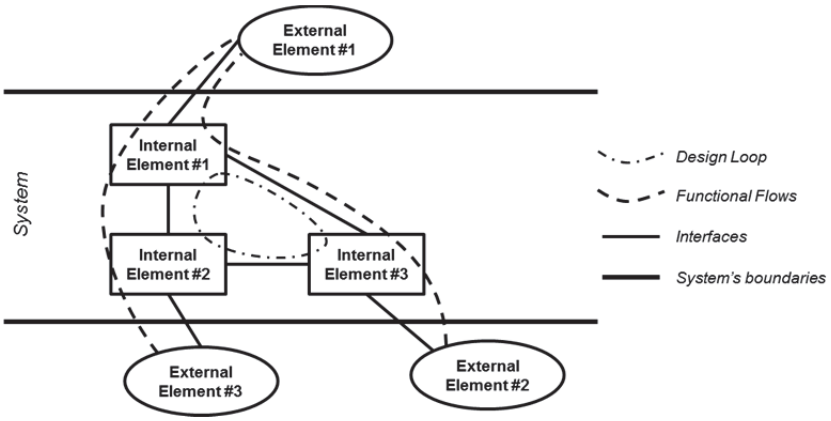


Figure 57 – Principle and convention of the Function Block Diagram

4.2.5. Function Models Positioning

There are plenty of other **Function** models in the literature (Eisenbart 2014) that could be analysed and positioned in this research. Only the introduced ones are positioned on the definition domain (Figure 58). The selected models are sufficient to show that function models can cover both the external description of a system (i.e. system as a black box) and its internal description (i.e. system as a white box).

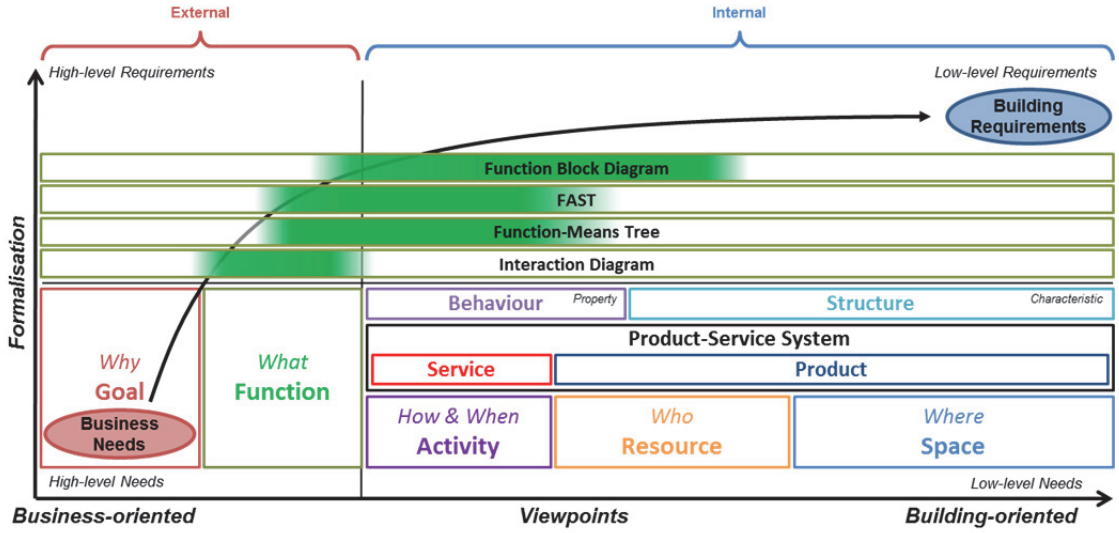


Figure 58 – Formalisation Coverage of Functional models

4.3. Behaviour

In order to model the Behaviour construct, we selected three main modelling languages from the literature: the Integrated Computer Aided Manufacturing Definition for Function Modelling (IDEF), the Business Process Model and Notation (BPMN), and the Unified Modelling Language (UML).

4.3.1. IDEF Modelling Language

Function modelling includes the modelling of the *internal functions* of a **system**. The *internal functions* represent an internal description of a system. In this research, the internal description of a system is attached to **Behaviours**. The IDEF modelling language is therefore considered as a "*behavioural modelling language*" as it is mainly focused on the internal description of a **system**.

The IDEF suite is divided into several techniques (Mayer, Crump, et al. 1995; National Institute of Standards and Technology 1993; Mayer, Menzel, et al. 1995; Benjamin et al. 1994):

- IDEF0 for function modelling,
- IDEF1 for information modelling,
- IDEF2 for dynamic modelling,
- IDEF3 for business process modelling,
- IDEF4 for object-oriented modelling,
- IDEF5 for domain ontologies,
- IDEF6 for design rationale used in enterprise system development,
- IDEF8 for human-system interaction,
- IDEF9 for business system constraints, and
- IDEF14 for computer and communication networks.

It has to be noticed that all of the IDEF techniques are not widely used or disseminated. In this research, we focus on the IDEF0 which is one of the most known. The IDEF0 is based on the Structured Analysis and Design Technique (SADT) developed by Ross in the early 1970s (Ross & Schoman 1977). As SADT, the IDEF0 is a top-down approach that represents the decomposition of a system (Figure 59). It models the data flow, system control, functional flow, and resources (i.e. internal description) required to achieve an external function (represented on the first A-0 level).

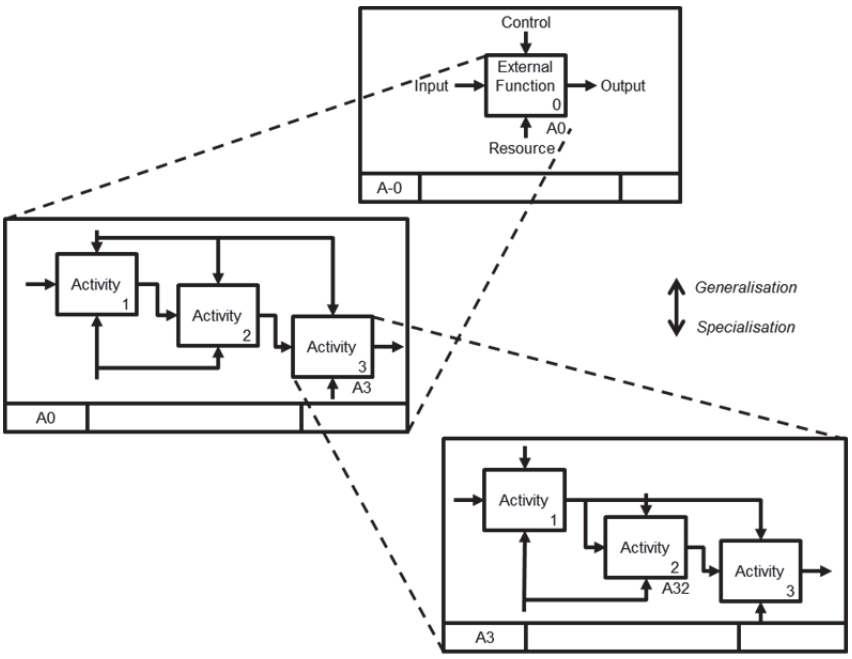


Figure 59 – Principle and convention of IDEF0/SADT

4.3.2. Business Process Model and Notation

BPMN is a standard that provides a graphical intuitive notation readily understandable by all business users (i.e. from technical to business users) (White 2004). It is used to communicate information through Business Process Diagrams (BPD). The full details of its notation will not be presented here (see (OMG 2011a) for more information). BPMN includes three main types of models in its latest version: process (flow of elements), collaborations (collection of participants), and choreography (coordination of participants). Each model provides additional information on the business process (Figure 60).

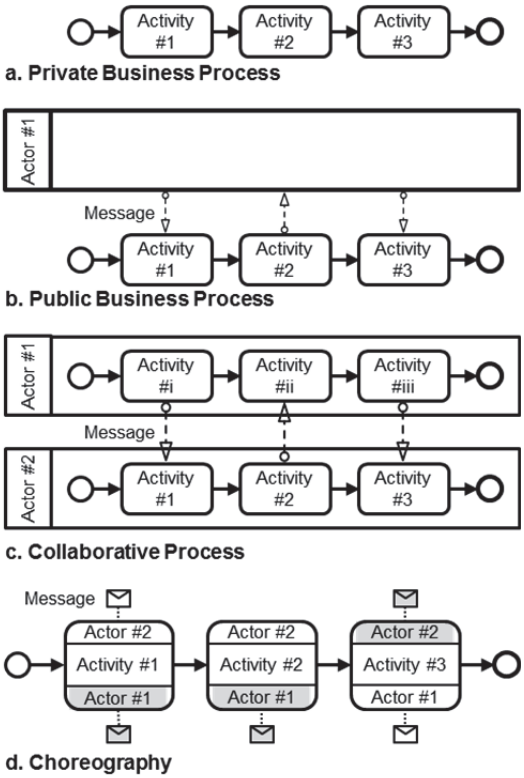


Figure 60 – Difference between the main Business Process Diagrams (OMG 2011a)

The concepts modelled in the BPMN are related to the main construct of **Activity** and they focus on the concept of Process. This type of model allows representing the relationship between constructs in a graphical format instead of staying with a descriptive text. BPMN provides models from the viewpoint of actors which is very interesting to check with the user clients if all of his activities are correctly represented. Despite the limitation to information and data objects, these models provide first elements of comprehension of the clients' activity to the architects.

BPMN was already introduced in the Architecture-Engineering-Construction domain to model the business processes and information exchange during construction project through Information Delivery Manual (IDM) for Building Information Modelling (BIM) users (Wix 2007; Weise et al. 2009; Berard & Karlshøj 2011). This provides a first clue about domain's acceptance of such models. The value added here is to use the modelling notation to represent business processes related to the object of study instead of business processes related to the project.

4.3.3. Unified Modelling Language

The Unified Modelling Language (UML) is defined as "*a standard graphical language for visualizing, specifying, constructing, and documenting the artefacts of a software-intensive system*" (Booch et al. 1998). It covers the modelling of abstract as well as concrete things, from business processes to software components. Based on its Meta-Object Facility, UML was adopted in other engineering domains such as System Engineering with SysML (OMG 2010) and used as a (recall) basis in Enterprise Architecture through UEML (Vernadat 2002). UML is indeed one of the most important object-oriented modelling languages.

UML, in its 2.4 version, proposes two types of diagrams to model **Structure** on one side and **Behaviour** on the other side (Figure 61). The static part of a **System** (i.e. **product** in the case of a PSS) is represented using **Structure** diagrams whereas its dynamic part (i.e. **service** part of a PSS) is visualised via **Behaviour** diagrams (OMG 2011b).

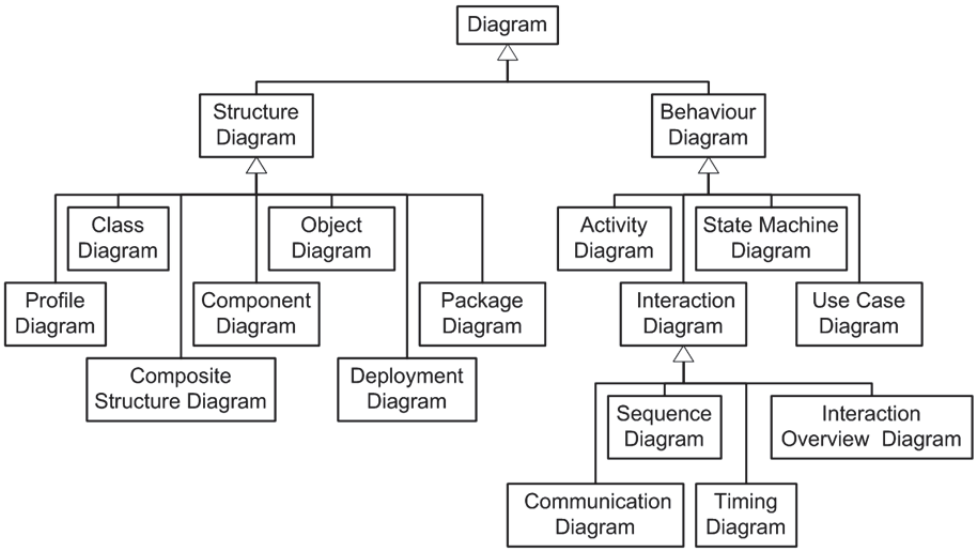


Figure 61 – UML 2.4 Taxonomy of Structure and Behaviour Diagram (OMG 2011b)

Each diagram provides a different but complementary viewpoint on the Software System. The idea is to use these models when it is necessary to visualize the briefing information in a structured way. However, it is acknowledge that all of the diagrams presented in Figure 61 are not of interest in the case of a building system. In this section, we focus on the behaviour diagrams: activity diagram, the sequence diagram, the use case diagram, and the state machine diagram (Figure 62). The structure diagrams are introduced and positioned in Section 4.4 on the Structure construct.

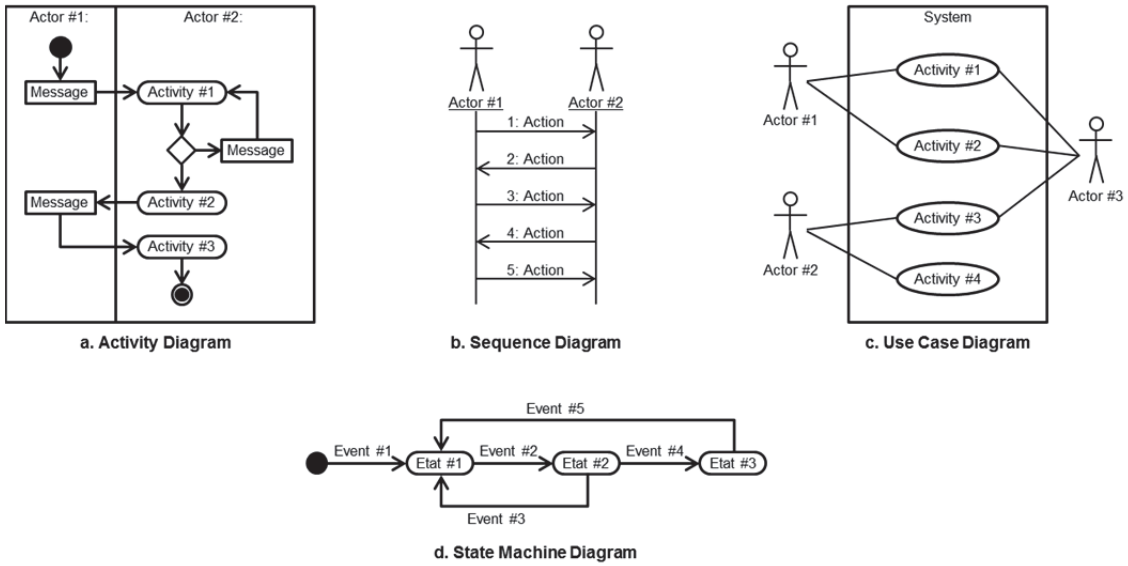


Figure 62 – Main UML Behaviour Diagrams: Principle and Convention

4.3.4. Behaviour Models Positioning

Behaviour models are legions. In this research, we introduce five different modelling languages to formalise the **Behaviour** construct. Based on their analysis, we position them on the definition domain in order to reflect the differences between the models (Figure 63). Each model is more or less complete regarding the formalisation of the **Behaviour** construct and, in some cases, **Function** or **Structure**.

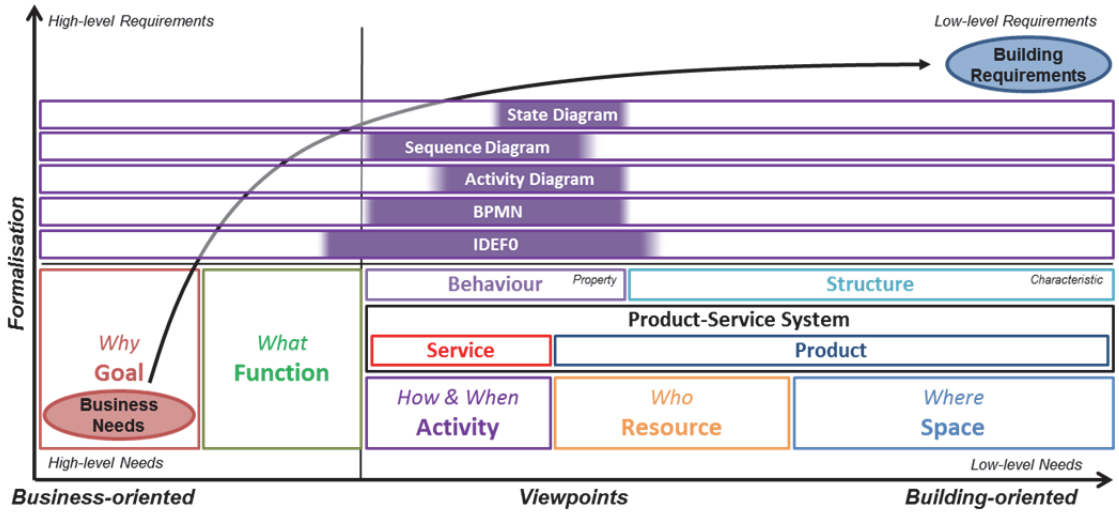


Figure 63 – Formalisation Coverage of Behaviour models

4.4. Structure

The structure section is divided in two parts. The first part (Sections 4.4.1 to 4.4.5) focuses on modelling language that represents Resources in general. It gathers the following modelling languages: UML Structure Diagrams (i.e. Class Diagram and Object Diagram), the Entity Relationship Diagram (ERD), and the Product Breakdown Structure (PBS).

The second part (Section 4.4.6 to 4.4.8) is dedicated to the Space construct as it is highly specific to the Architecture-Engineering-Construction domain. Indeed AEC has plenty of techniques used to model information about a building at different stage of its development. Information contained in a brief is generally modelled using such techniques so that architects can appropriate it. In this section, a small amount of these techniques are introduced as the state-of-the-art on the domain. All of the techniques are not necessarily used by architects but they provide interesting viewpoints on the same information. As a matter of fact, only a few of them are used during briefing or design phases. Most of them support the early stages of the design process but they can also benefit to the architectural programming when information is already there. The idea is to show how all of these models could interact, complete each other, and support the briefing process.

4.4.1. UML Structural Diagrams

The Class Diagram is used to represent the dynamic aspects of a system which include the structural (static) and the behavioural (dynamic) features. The structural features are represented through the attributes (i.e. characteristics) and relationship between classes whereas the behavioural features are described through the operations (i.e. properties). Figure 64 presents the principle and convention of the class diagrams.

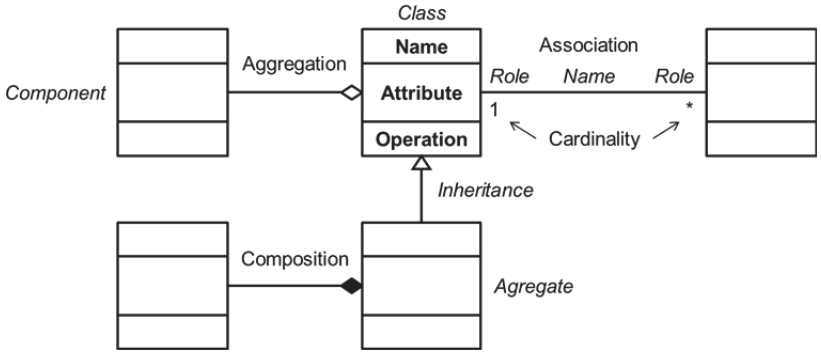


Figure 64 – Principle and Convention of the Class Diagram

The Object Diagram is limited to the representation of some particular instances, their attributes, and their relationships (not the behaviours). It is made to provide a local view of concrete information on a system at a specific time (i.e. a snapshot). An object diagram presents instances of a class diagram; its graphical convention is almost the same (Figure 65).

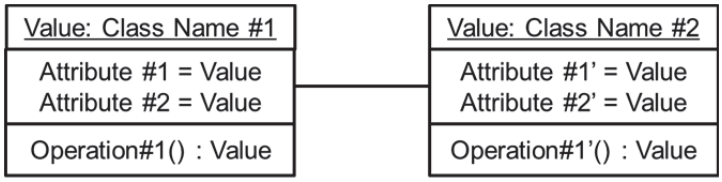


Figure 65 – Principle and Convention of the Object Diagram

4.4.2. Entity Relationship Diagram

Entity Relationship Diagram (ERD) is a data model developed by Chen in 1976 (Chen 1976). It focuses on the structural features of a system and only provides a static view of it (Figure 66). An **Entity Set #1** is related to several **Entity Set #2**. **Entity Set #2** is described by **Attribute #1**, **Attribute #2**, and **Attribute #3**.

"The entity-relationship model can be used as a basis for unification of different views of data: the network model, the relational model, and the entity set model." (Chen 1976)

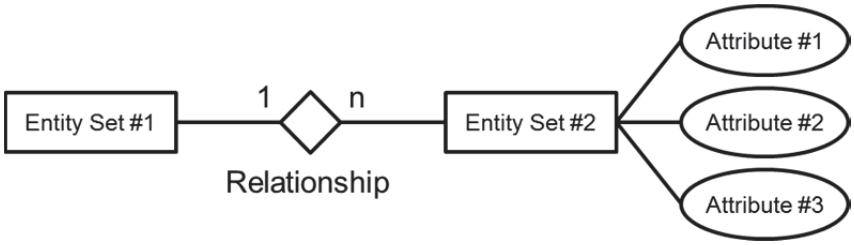


Figure 66 – Principle and Convention of the Entity-Relationship Diagram

4.4.3. Product Breakdown Structure

The last modelling language introduced to model the **Structure** construct is the Product Breakdown Structure (PBS). It is used to provide an exhaustive hierarchical tree structure of system decomposition from the main artefact to all of its components (Figure 67).

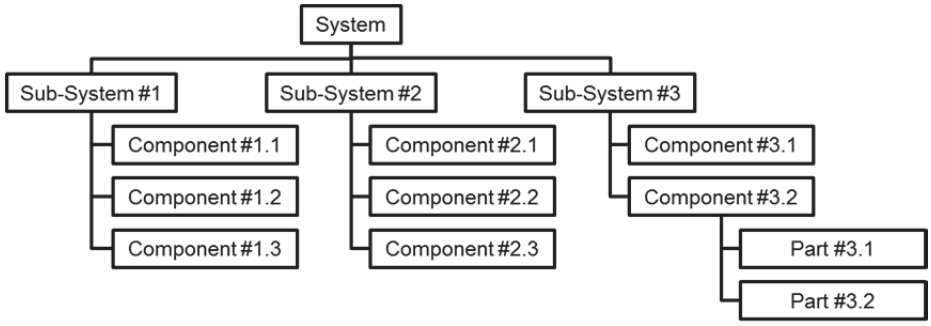


Figure 67 – Principle and Convention of a Product Breakdown Structure

4.4.4. Resource Models Positioning

In this section, we introduce four different modelling languages that formalise information about the **Resource** part of the **Structure** construct. These models can be applied on any kind of object (i.e. physical as well as virtual). In Figure 68, we position them on the definition domain and we tried to represent the degree of coverage regarding the **Resource** construct.

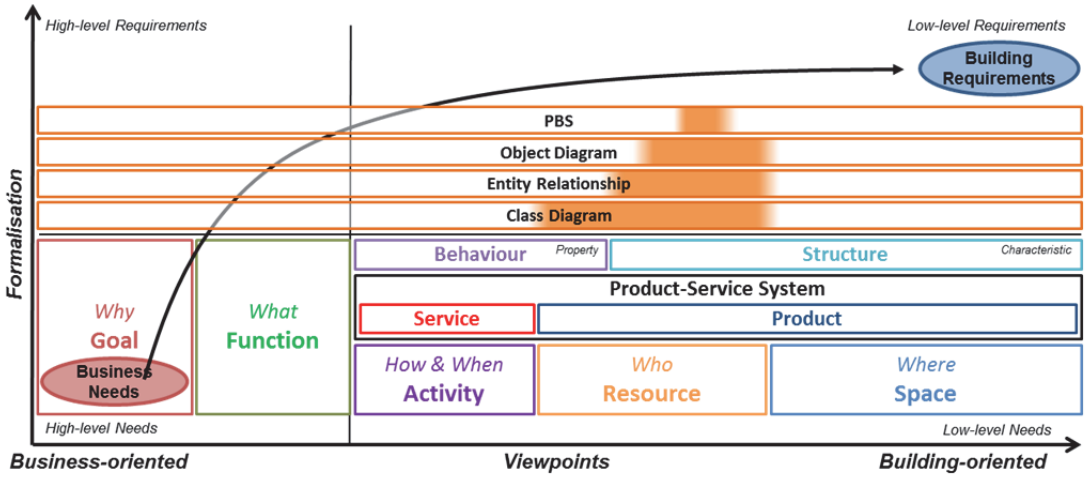


Figure 68 – Formalisation Coverage of Resource models

4.4.5. Functional Diagrams

In the briefing process, functional diagrams are used to model the human activities, i.e. operational requirements (Bisbrouck 2004). It graphically synthesizes the organisational and functional dependencies between the functional spaces based on the description of the human activities contained in the brief. A space table resumes its essential content into "numbers and words" (Lobos & Donath 2010). The functional diagram represents the dynamic part of human activities whereas space table resumes its static part.

A functional diagram is both a design artefact and a working method used by the programmer. Its main objective is to help the contracting owner, then the architect, deal with the functional and spatial organisation of the future building without enclosing themselves into architectural solutions too early in the project design. A functional diagram is a graphical synthesis (Figure 69) of a set of requirements. The requirements become easier to handle and to understand by the contracting owner and the architect than traditional textual brief. A functional diagram focuses on functional requirements and functional organisation of the spaces from the space table. The functional diagram is the first analysis/synthesis of the contracting owner's requirements by the programmer. This high-level synthesis is instantiated into the drawings by the architect who performs the second analysis/synthesis of these requirements.

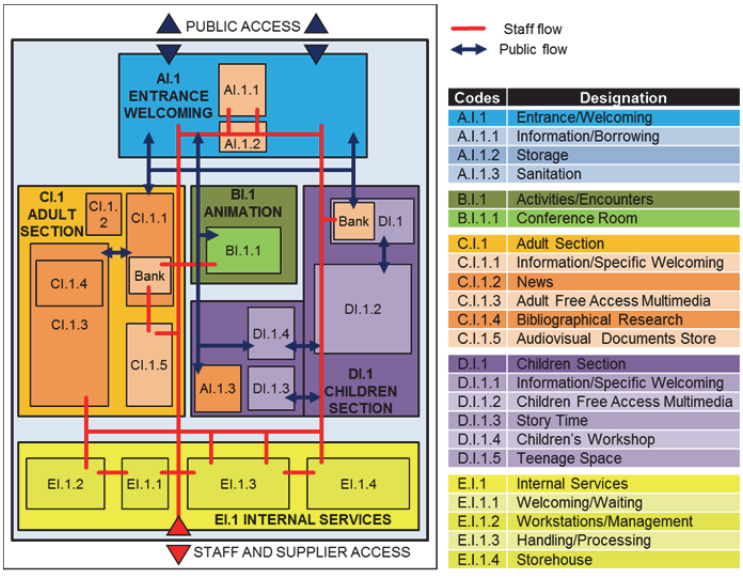


Figure 69 – Simplified functional diagram based on a multimedia library brief (Aubry & Guiguet Programmation, 2003)

The information required to define the functional diagram of a building is related to the services that the contracting owner aims to provide to its users. It is part of the functional requirements that define the building. These services are developed into processes and activities linked sequentially and temporally. Based on their more or less accurate definition, the programmer is able to define and refine the type of building (e.g. library, toy library, or multimedia library) and its spatial typology (e.g. adult section, administrative area) and to estimate the dimensions of each functional space. This static information (m², designation, characteristics) is gathered in the space table. The functional diagram illustrates the dynamic information between spaces and their functional organisation through a high-level drawing of the physical spaces with a concrete and an easy to read artefact. It is the first illustration and synthesis of the functional requirements by the programmer.

The functional diagram is composed of simple geometric figures, usually circular or rectangular forms, named according to the function they ensured (e.g. welcoming, information/borrowing) or the name of a required space for specific activities (e.g. storehouse, conference room). Spatial union, intersection, or exclusion can be expressed by figure superposition or grouping. Lines and arrows of different colours illustrate the flow of entities (e.g. staff or public flows) between the functional spaces regarding the functional organisation. They are sometimes augmented with icons or symbols to represent key characteristics (Certu 2006) (e.g. access points, specific resources) depending on the degree of information the programmer wants to provide. Using these geometric figures, the functional diagram gives substance to the organic structure of the building in a clear and simple way. They are usually drawn using vectorial 2D-drawing software applications.

The functional diagram does not replace the brief; it provides a visual synthesis support to it. The quantity of information the programmer could illustrate is limited by its own readability. Moreover, the synthesis processing is not well represented. It prevents from tracing back requirements hidden behind a line or geometric form. Detailed information and explanations are given by the brief. Several functional diagrams can be drawn if necessary to give additional global or local, synthetic or detailed viewpoints. Some programmers extract parts of the functional diagrams when developing a space or an activity in the brief as a supporting illustration.

There are a lot of different kinds of functional diagrams. The geometric figures, colours, icons, and symbols vary from a programmer to another or from one project to another. There is currently no standard formalism defining their use or describing their notation and content. Neither the geometric figures nor their layout prejudge what the architectural solution will be (Vanneyre 2011). On the one hand, it does not bridle the architect; it defines only the scope of his architectural solution space (Certu 2010). This context provides the architects a real freedom to express their creativity and ensure the diversity of the proposals (Figure 70, and Appendix D). On the other hand, despite the simplicity of the used geometric artefacts, a functional diagram lends itself to interpretation and misunderstanding (Kalay 2001). For the same building, there could be thousands of valid functional diagrams. This is the same with architect proposals which can be numerous and totally different although based on the same functional diagram.

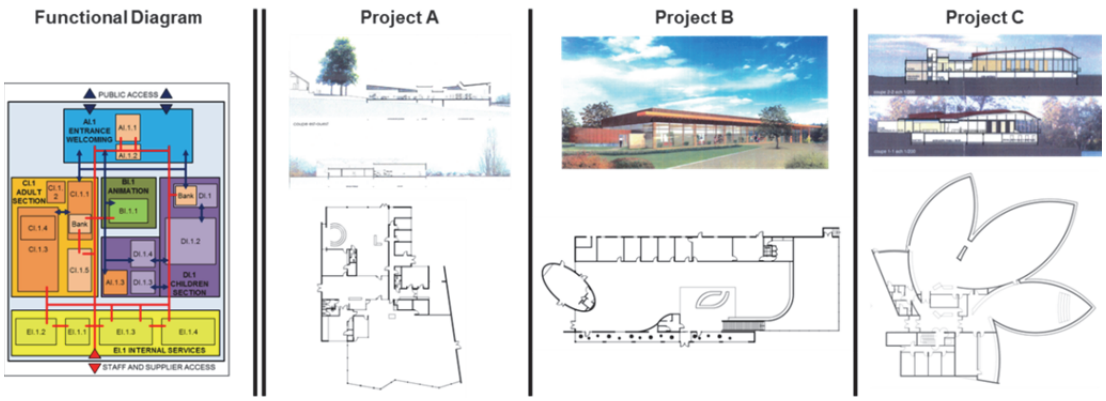


Figure 70 – One functional diagram, three design proposal

In France, the functional diagram is well used in the programmer community. This is partly due to the mandatory use of architectural programming for public construction/facility projects (JORF 2007). It is highly recommended, almost mandatory, to provide one in a brief (Certu 2010). It is an easy way to ensure consistency and an equal understanding of each involved stakeholder. The functional diagram is also called a Venn diagram, relation graph, or functional scheme (Certu 2010) even though a functional diagram is less formal.

In Luxembourg, architectural programming is not that well developed and mostly performed by architects before the design of proposals. It is often considered as a methodological competitive advantage to propose this kind of service. Functional diagrams are unknown or not used in the brief and most often the bill acts as the brief for public constructions. When analysing bills in Luxembourg, only the architect proposal is developed with the space table. There is no functional synthesis (Chambre des Députés 2011) apart from the text and space table, mainly used for cost estimation rather than functional analysis.

4.4.6. Layout & Space Adjacency Models

The following models are rather more concerned with the design process than the briefing process. Each one of them provides different viewpoints to support the special layout of buildings. The shapes used in these models are ovals, "potatoes" shape, or rectangles with curved corners as a reminder that the architects manipulate areas to perform activities and not rooms yet (Alread & Leslie 2006). Two main models are introduced in this research: bubble diagrams and zoning diagrams.

Bubble diagrams represent the adjacency relationships (lines) between functional spaces (bubbles) in a purely abstract way (Wurzer et al. 2012). It models information from adjacency matrix (Figure 71) that itself models information from the brief. Adjacencies usually represent direct access between two functional spaces. The line style and colour can be modified in order to indicate various degrees of relationships when necessary. The different names associated to this model are "balloon diagrams", "adjacency diagrams", "relationship diagram", or "bubble charts". They are used to represent the layout of the main spaces of a building and guide the architects in its design (Akinc 2005). Bubble diagrams are particularly used in research on computer-generated layout as a support to preliminary design (Fortin 1978; Merrell et al. 2010; Donia 1998). For further details, Lobos and Donath propose a literature review on this space layout problem in architecture (Lobos & Donath 2010).

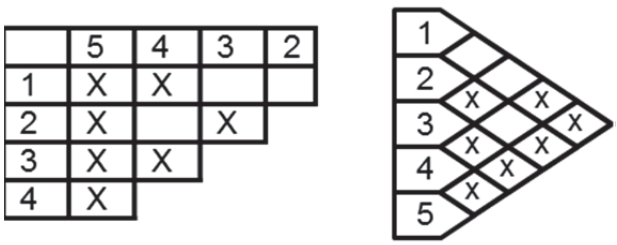


Figure 71 – Two kinds of adjacency matrix

A bubble diagram is a limited model in terms of information but more accurate and formal than a functional diagram. Architects synthesize information from the brief document in a bubble diagram very early in the design process to better understand the adjacency constraints on the required functional space (Lawson 2005). It quite heavily impacts the future layout of the building. However, the information is already in the brief and this kind of viewpoint can be useful for the programmer to directly validate the adjacency requirements with the clients. As a result, the architect would not have to create again such model is it is already in the brief documents.

In terms of formalism, a bubble diagram is halfway between a graph (aka circulation graph) (Euler 1736) and a Venn diagram (Chilakamarri et al. 1996). In Architecture, a graph only represents adjacencies whereas a Venn diagram only represents a layout and size proportions of

spaces. A bubble diagram usually represents both adjacency and size information in the same model (Figure 72).

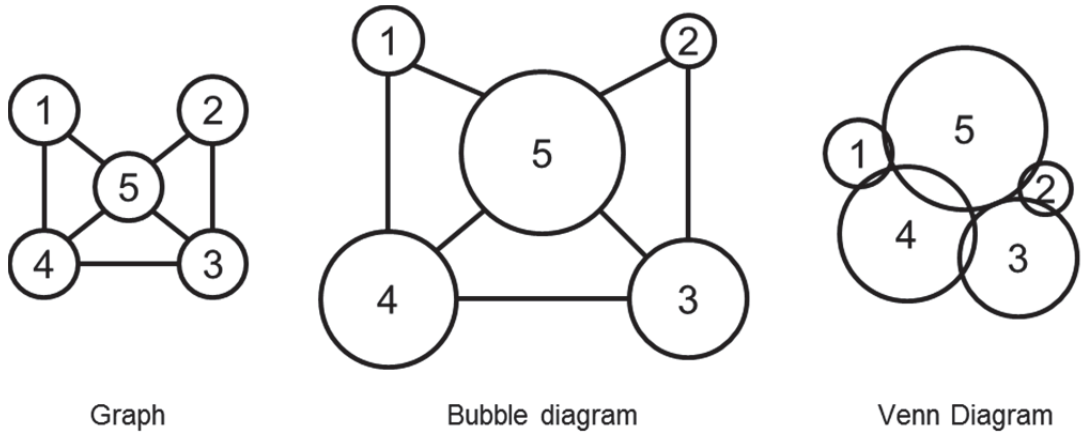


Figure 72 – Graphical difference between a graph, a bubble diagram, and a Venn diagram

By extension, each bubble representing a functional space ends in modelling a physical space during the design process (Ruch 1978) (Figure 73).

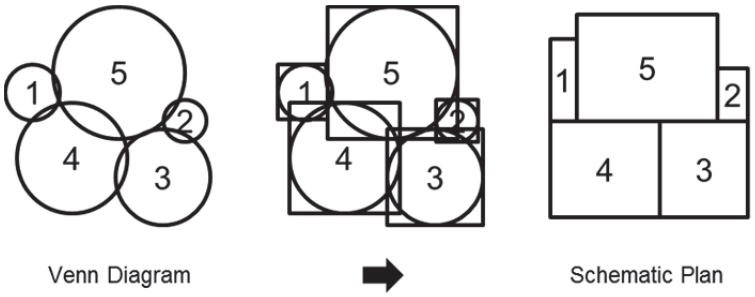


Figure 73 – From functional spaces in Venn diagram to physical spaces in Schematic Plan

Zoning diagrams (aka affinity diagram) provide additional layers of information to bubble diagrams (White 1986). They model local viewpoints on the bubble diagram to show potential groupings regarding specific requirements (e.g. noise level, privacy, servicing…). The bubble diagram represents circulations and flows of people or resources whereas zoning diagrams indicates conveniences or preferences regarding performances or constructability (Alread & Leslie 2006) (Figure 74). They represent different kinds of information priority or criticality.

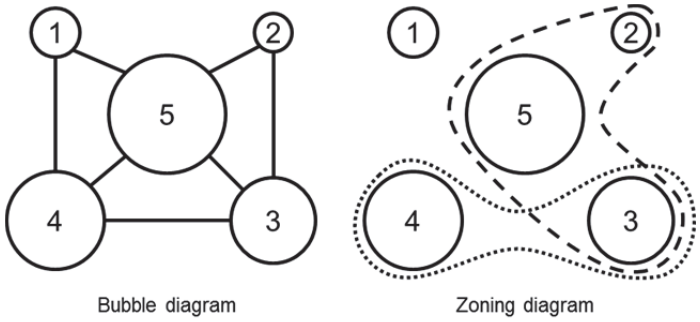


Figure 74 – Bubble diagram vs. zoning diagram

The functional diagram introduced earlier synthesizes information from both bubble diagram and zoning diagram in a less formal way (Figure 75). As a matter of fact, the functional diagram is a briefing model which programmers use to sum up information of the brief. Bubble diagrams and zoning diagrams are design models used by architects to start the design of the building

layout. Information used by models is the same and should be identical. The impression of similarity comes from the lack of distinction between activity, function, and space in architecture. Functional diagram refers to "*functions*", bubble diagram refers to "*activity*", and zoning diagram refers to "*space*". As a *function* is realised by *activities* performed in a *space*, all of the concept and information are graphically represented with the same artefact. It is only a matter of viewpoint on the system that makes a difference between the types of information the focus is on.

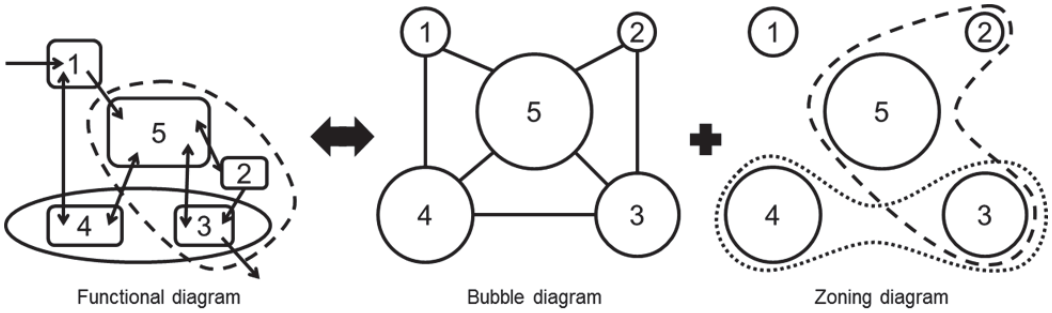


Figure 75 – From a single briefing model to several design models

4.4.7. Space Models Positioning

In the second part of the Structure models, we provide a literature review of the most common modelling languages used to formalise Spaces and related information. Most of the models are used by architects or designers during the design phase of a construction project. However, the information is already presented in the brief but little formalised. In this research, we believe that such models can be used upstream by the programmer to formalise the information contained in the brief in a solution-neutral way. The aim would be to limit the risk of misunderstanding of information by the architect. We still recognise that such models have to be made by architects to allow them making the brief content appropriate in their own way (both should be possible). Figure 76 presents a positioning of the introduced models on the proposed definition domain. The IFC data model was added to illustrate the kind of information it deals with.

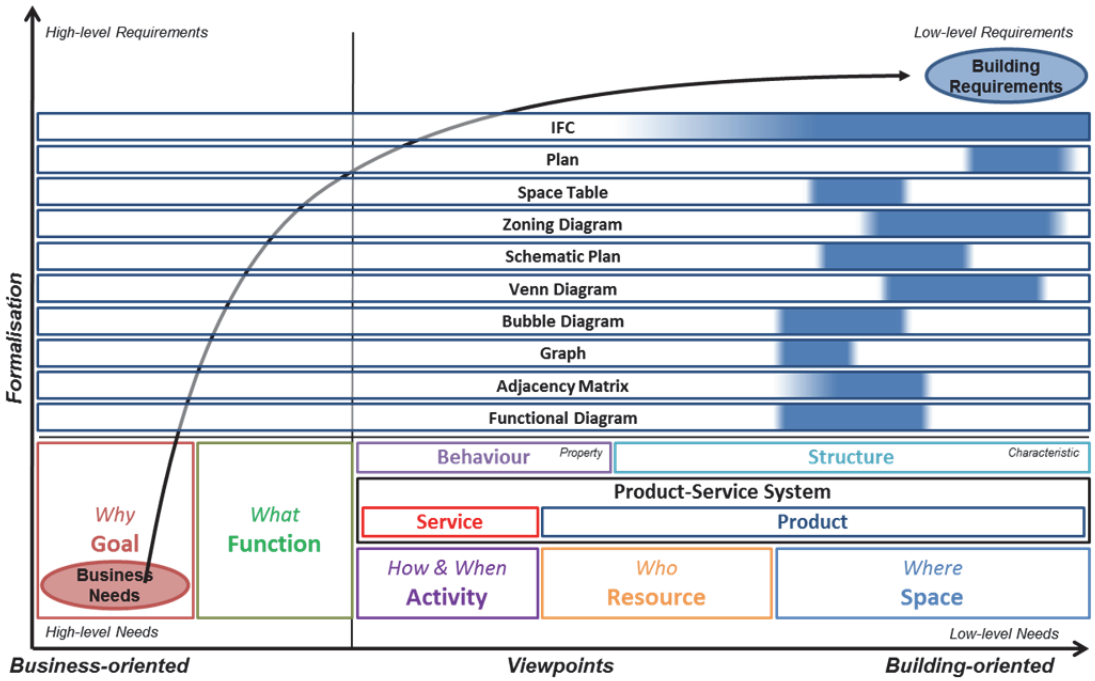


Figure 76 – Formalisation Coverage of Space models

4.5. Bridging Operational Models Together

The bridging principle consists of finding some links between models from different engineering domains. It is different from trying to design a transdisciplinary unified modelling notation. Researchers in one domain already have trouble to find such an unified and agreed solution, thus, finding a solution to such issue is totally out of the scope of this doctoral research.

In this section, various models are introduced, in a limited selection. As explained along the section, the point was not to give a comprehensive state-of-the-art of existing modelling languages but rather to introduce notations and languages to formalise the definition domain defined in Chapter II. Based on their analysis, we are able to position them on the proposed definition domain to represent the approximate coverage of each modelling language at a macro level (Figure 77 and Appendix E). As a result, we can see some overlaps between models except regarding the **Space** construct. Some models are focused on a specific construct, especially **Space** models, whereas some others formalise the transition from one construct to the next one. Moreover, we observe that one model is not enough to represent the full definition domain.

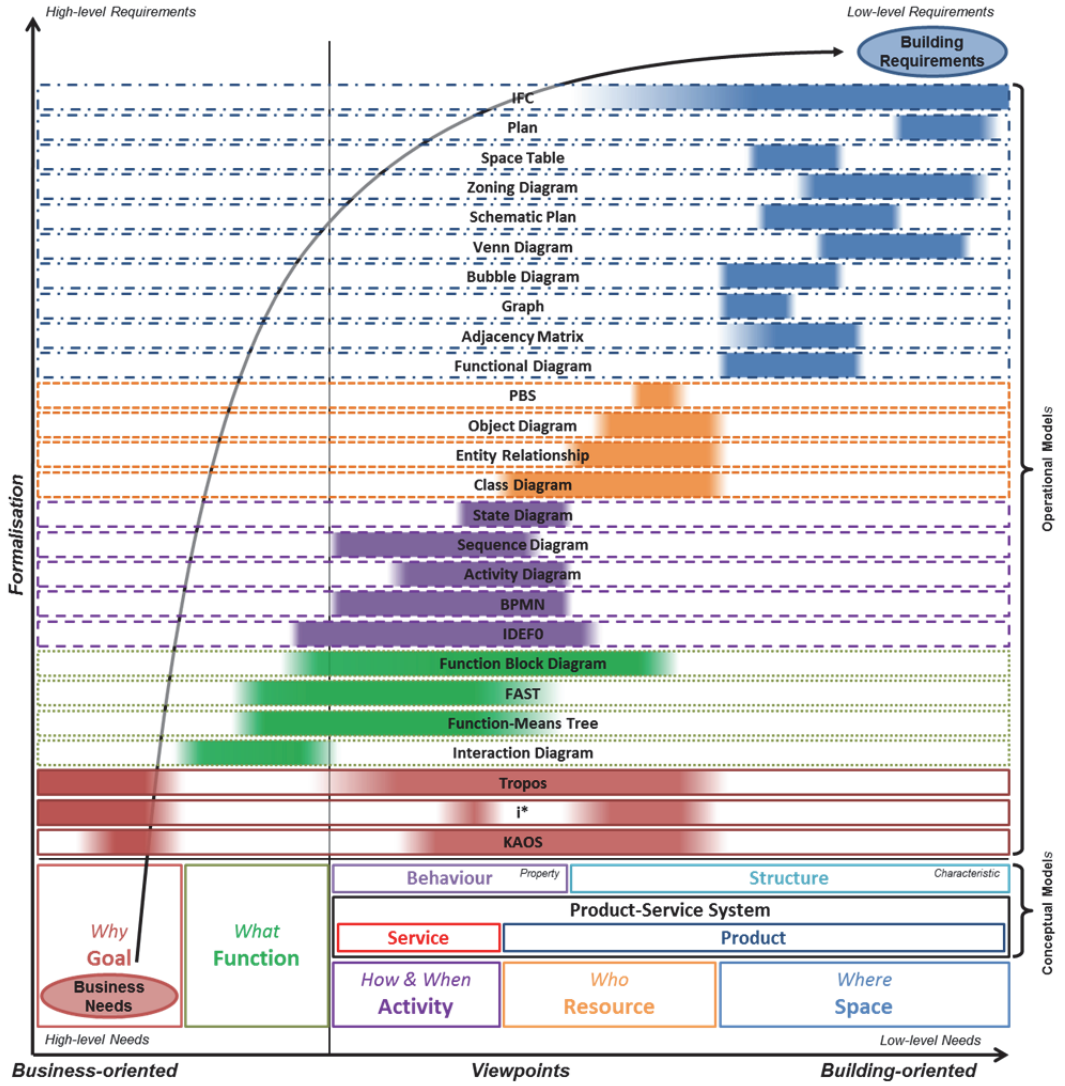


Figure 77 – Operational Models Coverage of the Definition Domain

The bridges proposed in this research are represented by the main constructs that structure the definition domain: **Goal**, **Function**, **Activity**, **Resource** and **Structure**. Each engineering domain provides modelling notations and languages that allow more or less complex representation of parts of them. They are considered as viewpoints on information about the same object of study.

Information is unique but graphical representations are legions. Instead of limiting oneself to a set of predefined models in a single engineering domain, this research fosters the use of any kind of models from any engineering domains as long as it provides valuable viewpoint on the information about the object of study. Therefore, the key point of this research is not the models (i.e. viewpoints) but the information and its structure (Figure 78).

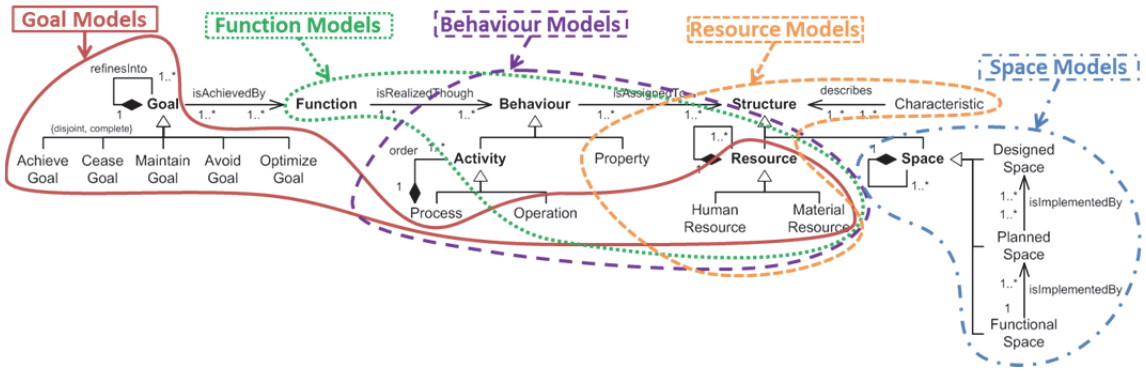


Figure 78 – Coverage of Model Viewpoints on the Conceptual Model of the Definition Domain

However, the definition domain is not fully covered by the introduced models. There is one transition that is neither modelled nor formalised: the transition from **Behaviour** to the **Space** part of the **Structure**. This transition is the part currently ensured by the programmers based on their experience. The links between the constructs are not formalised by them. The transition from high-level business needs to low-level building requirements can be represented by a "conceptual short-cut" from high-level business needs to low-level building needs before the formalisation of such needs into building requirements through the various space models. Figure 79 therefore represents current architectural programming practices (i.e. solution-oriented).

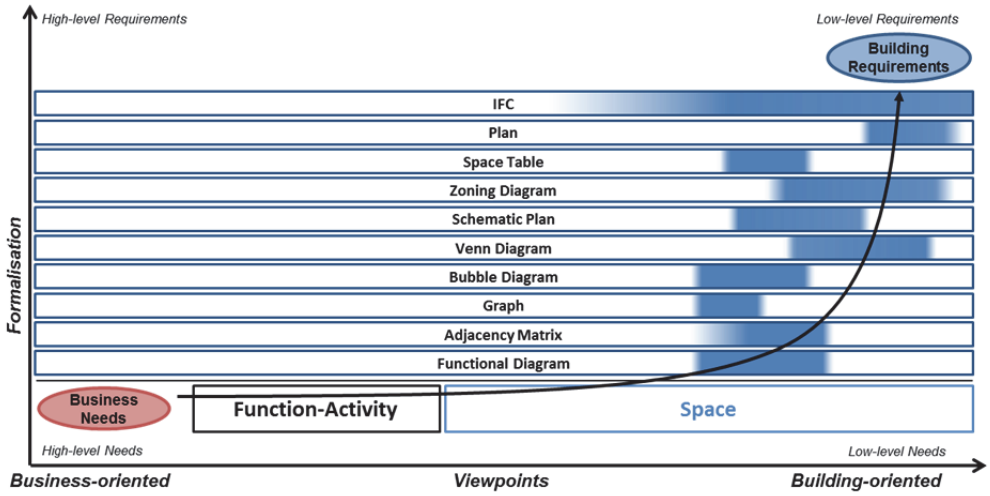


Figure 79 – Representation of Current Architectural Programming Definition Domain

4.6. *In Summary*

In this section, a small amount of graphical models were introduced from various engineering domains. Each model provides notations and rules to visualize information. Interactions between various constructs are formalised through different semi-formal models. They provide a large panel of potential viewpoints on the same **system** (i.e. of the same information). These viewpoints are easier to handle than a textual description.

"Graphical workflow representations are easier to understand and manipulate for humans."
(Simpson 2004)

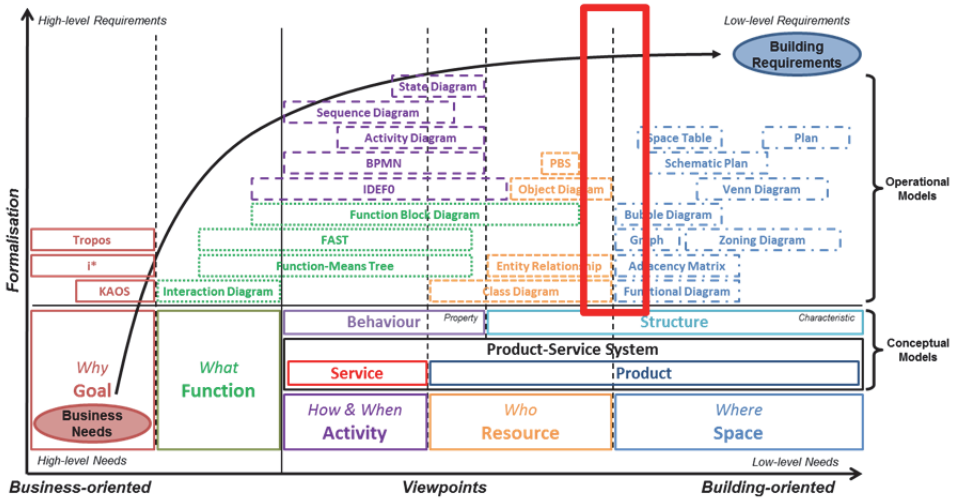


Figure 80 – Missing Overlap in Operational Models Coverage of the Definition Domain

Regarding the building definition domain, each operational model provides partial viewpoints on it. Most of them are interconnected and almost all constructs and transitions can be modelled through at least two different construct models except the concept of **Space** (Figure 80). This missing overlap is tackled in the next section through an original model: the meta-space diagram. It is designed to support the integration of the **Space** concept to the other constructs and to formalise the transition to and from **Space**. The addition of this model should provide a reasonable quantity of operational models to formalise and (practically) manipulate the full building definition domain.

5. From the brief to the design proposal

In the previous section, various operational models were introduced as a way to formalise information about the *building system*. Each model provides a viewpoint on specific kind of information (i.e. constructs). In current practices, this information is synthesized in the brief mainly in written format but also under the form of functional diagrams. During the design phase, architects have their own models to represent the information and to start their sketches. From those first models, they produce plans (drawings) using various dimensions (from 2 to n depending on the CAD-tools available).

Each time they pass from one representation model to another one, a little bit of information is lost. Major loss is on the first one where the information cannot be traced back to its origin (Figure 81). This is mainly due to the human-aspect of the process but also due to the lack of interoperability between models. Architects are humans. They cannot keep in mind all of the information about such a complex system to design. They are limited in terms of number of viewpoints to integrate. Later on, during the assessment of their design by the paying clients, the technical committee, or the jury, the information is once more interpreted and formalised in a different way. The consistency of the information along the process is not preserved and subjected to subjectivity and interpretation, not mentioning the lack of completeness in its consideration.

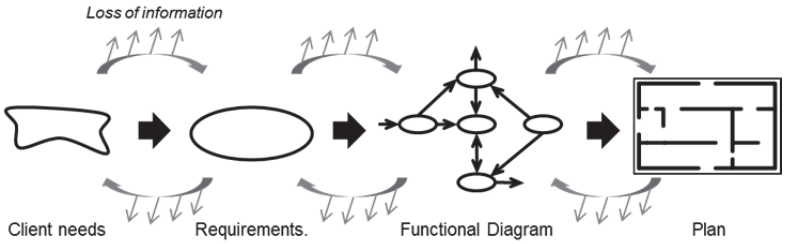


Figure 81 – Representation of information loss along the architectural programming from the clients' needs to the architects' plan

In this section, a design artefact is proposed to support traceability and integrity of the information during architectural programming, from the elicitation to the brief-project alignment.

5.1. Meta-Space – Principle

The main function of a building is to "*shelter people doing certain things in a certain way*" (Koutamanis 2013). As stated in Chapter II, activity is an appropriate abstract way to tell the architects what is expected from them in terms of design result. Each activity requires a space to be performed. The number of activities is quite huge regardless of the number of spaces in a building with limited budget. In order to evaluate the number of required spaces, an intermediary design artefact is introduced: the *meta-space*. It represents an abstraction of the concept of **activity** and **space** which makes it easier to analyse and manipulate through automation.

Definition 14: A **Meta-Space** is a virtual ideal space in which all the requirements regarding the proper performance of an elementary activity (i.e. operation) including the associated goals and functions are gathered.

A **Meta-Space** is associated to an operation as an aggregation of all the requirements linked to its operation. Why this operation is required to be performed? What does it contribute to? How and when does it happen? A **space** is defined by its characteristics (e.g. surface in square meters). A **meta-space** is an abstraction of the **Space** concept that integrates the upstream (but also the downstream) information that leads to its definition.

A **meta-space** is associated to a single elementary **activity**, i.e. an operation. If thousands of operations have to be performed in the future building, there will be thousands of corresponding **meta-spaces**. As spatial resources are limited, there is a need for number reduction. The reduction of **meta-space** number is realised through a requirements processing. Based on a set of business and logical rules, all the meta-spaces that have compatible requirements can be gathered in the same space. Figure 82 outlines the principle of **meta-space** based on **activities**, required **resources**, process and operational requirements taken from the book cycle in a library, i.e. the processing of new incoming documents. The example provided in this figure is further developed in Chapter IV.

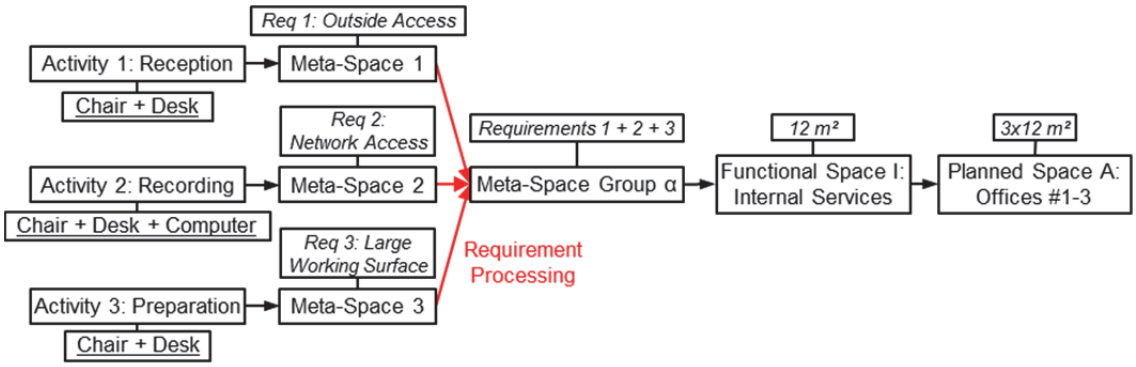


Figure 82 – Meta-Space principle illustrated on the book cycle

The main advantage of the **Meta-Space** concept is its abstractness. As it gathers all the information about the building to define, various viewpoints can be taken into account. It is the crossroad of all the possible viewpoints on the building regarding each previously defined concept. Instead of manipulating and grouping different concepts which require changing the referential of study, all of the viewpoints can be taken using the same referential. Based on such abstract concept, a new model is introduced in this research: the meta-space diagram. This diagram takes benefits of the abstractness of meta-spaces to model the different requirements in

the same model. It represents an interoperable model that allows transitions from one engineering model to another without loss of information.

5.2. Meta-Space – A Taxonomy

The **Meta-Space** concept is an intermediary artefact. To clarify its use and regarding its different value added along the briefing process, taxonomy is proposed before introduction of the Meta-Space diagram (i.e. model). The **meta-space** taxonomy is organised around two kinds of **meta-spaces** related to the main phase of usage (i.e. briefing meta-space and design meta-space). *Briefing meta-spaces* provide building requirements whereas *design meta-spaces* provide building specifications. In the first case, the building is not designed yet whereas in the second one the building is. The starting point is different so as the basis information. *Briefing meta-spaces* are defined by the programmer based on the clients' needs whereas *design meta-spaces* are defined by the architect based on its design proposal.

5.2.1. Briefing Meta-Spaces

Two types of *briefing meta-spaces* are distinguished in this proposal: *operational meta-spaces* and *functional meta-spaces*. Each one of them is related to a different part of the architectural programming deliverables. *Operational meta-spaces* are linked to the information contained in the brief, more specifically to the Operation concept. This type of meta-space is mainly related to a local viewpoint.

Definition 14.1: An **Operational Meta-Space** is an ideal virtual space associated to an operation (i.e. elementary activity). It is used as an intermediate between the definition of activities and the definition of functional spaces.

Functional meta-spaces are intermediary **meta-spaces** between the **operational meta-space** and the functional space. A *functional meta-space* is composed of one to several **operational meta-spaces**. Each *functional meta-space* is then associated to a functional space (Figure 83).

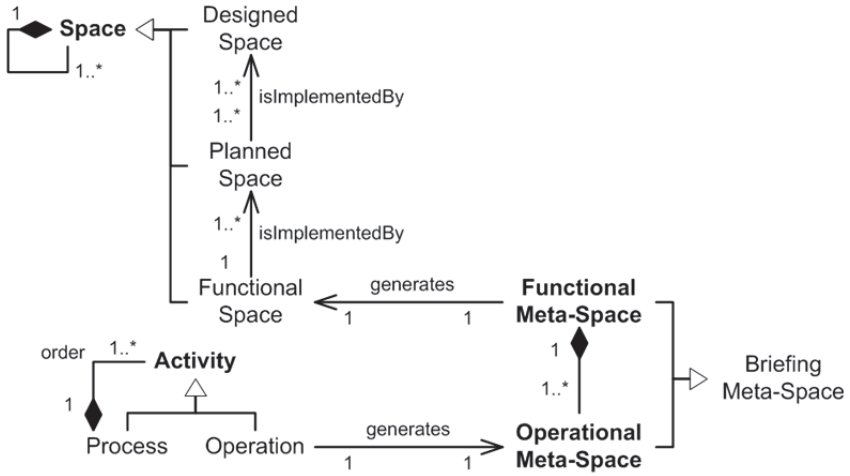


Figure 83 – Conceptual model of the Briefing Meta-Spaces

Functional spaces are modelled in the functional diagram drawn by the programmer based on information brief. *Functional meta-spaces* abstract functional spaces from the functional diagram. As a result, there are two ways of creating *functional meta-spaces*: starting from the brief and grouping operational meta-spaces, or abstracting the functional diagram. Information is richer when starting from the brief as more information is detailed in the operational meta-spaces. A lot of interpretation is required to document *functional meta-space* created from a functional diagram (see Section 5.3).

Definition 14.2: A **Functional Meta-Space** is an ideal virtual space associated to a functional space and composed of elementary operational meta-spaces. It is used as an intermediate between the textual part of the brief and the graphical functional diagram.

5.2.2. Design Meta-Space

Design meta-space is a **meta-space** associated to architect's deliverables. It abstracts design proposals produced by the design team. Information contained in *design meta-space* is related to building specifications from a design viewpoint (i.e. after design). It describes the designed physical object whereas information from *briefing meta-spaces* defines the system to design. A design meta-space is generated for each designed space contained in a 2D drawing (Figure 84).

Definition 14.3: A **Design Meta-Space** is an abstract model of a space designed by the architect and represented on a two dimensional drawing.

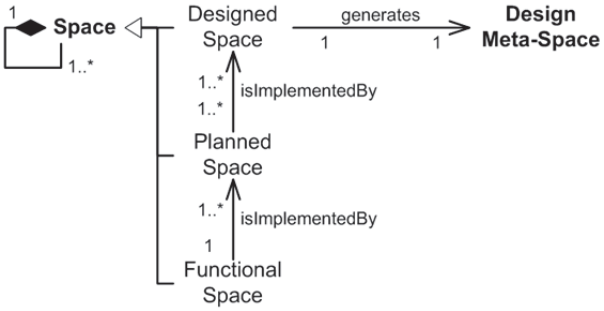


Figure 84 – Conceptual Model of the Design Meta-Spaces

5.3. Conclusion

In this section, three types of **meta-spaces** were introduced. Each one of them models different quantity of information at different moments of the architectural programming. These distinctions aim to support assessment and alignment of the different deliverables produced during the construction process. Indeed, each kind of **meta-space** models the same end artefact but with different quantities of information. This artefact is a room, or an office where some activities should be performed by people using resources. Design viewpoint and briefing viewpoint on this room are different in terms of information but in the end they have to match to each other to satisfy the clients.

The matching between the designed space, the functional space, and the **activities** performed in the building is currently ensured by the programmers. In this research, we propose to use the **meta-spaces** artefact to support it. The transition is realised through the functional meta-space (Figure 85) and supported by a set of meta-space diagrams (Section 6). Indeed, the functional meta-space is the only artefact related to constructs used in the main three deliverables.

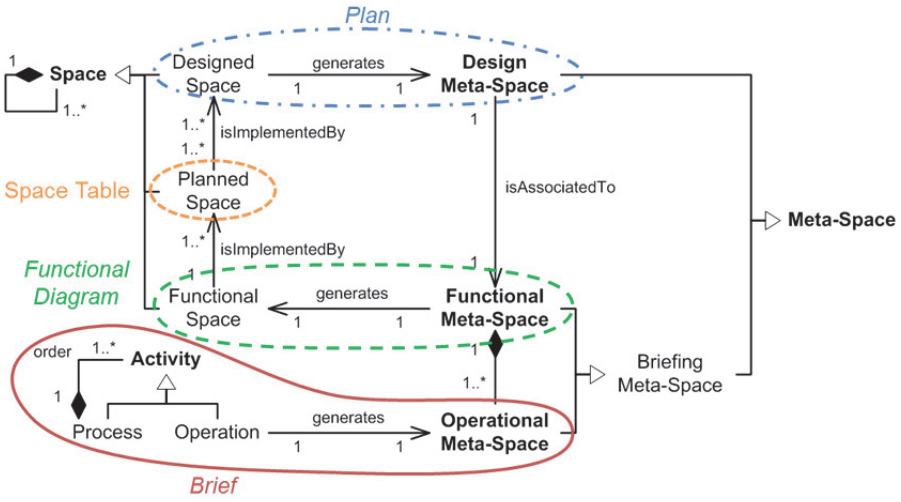


Figure 85 – Conceptual Model of the Meta-Space Artefacts and Related Main Deliverable

The introduction of these design artefacts change a little bit the conceptual model of the building definition domain (Figure 86 and Appendix F).

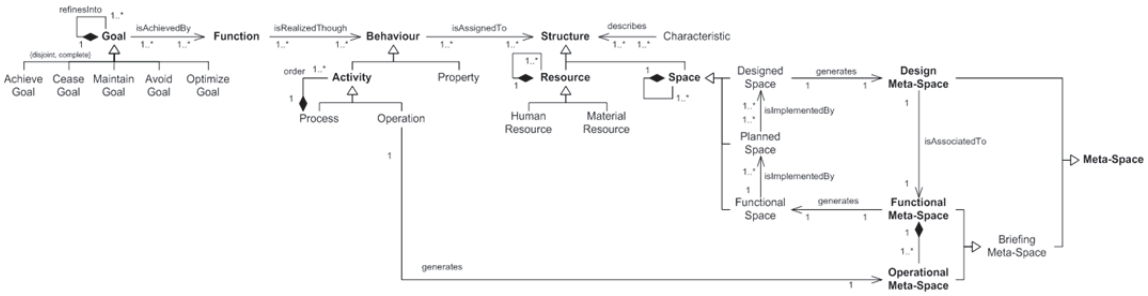


Figure 86 – Resulting Conceptual Model of the Building Definition Domain

6. Meta-Space Diagram – An Intermediary Transition Model

The defined taxonomy about **meta-spaces** is the basis of this section. As introduced, each kind of **meta-space** represents different information at different moment of architectural programming. In this section, the purpose of each kind of **meta-space** is presented through an operational model (modelling language): the meta-space diagram. This operational model takes benefit from the abstraction provided by the meta-space to support the transitions from one deliverable to another in both ways. This allows keeping traceability of information and their dependencies. Meta-space diagrams support the continuity of information along architectural programming.

6.1. Meta-Space Diagram

A meta-space diagram is an enhanced graph in the terminology of graph theory (Euler 1736; Grason 1971) and bubble diagrams (Séquin & Kalay 1998) but it is used in a different context and purpose. The layout is not the focus of this design artefact but it could later be used to work on layout improvement. The meta-space diagram synthesizes the grouping based on requirements into a graph but it keeps trace of the activities (Figure 87) through an associated data-base. The information model of the data-base could be based on the conceptual model of the definition domain. The grouping principle is same as in affinity diagrams (Alread & Leslie 2006) or zoning diagrams (White 1986) but it is extended with successive activities and dynamic behaviours related to flows of artefacts or human activities. Further details on the grouping criteria and illustrations can be found in Chapter IV Section 2.6.

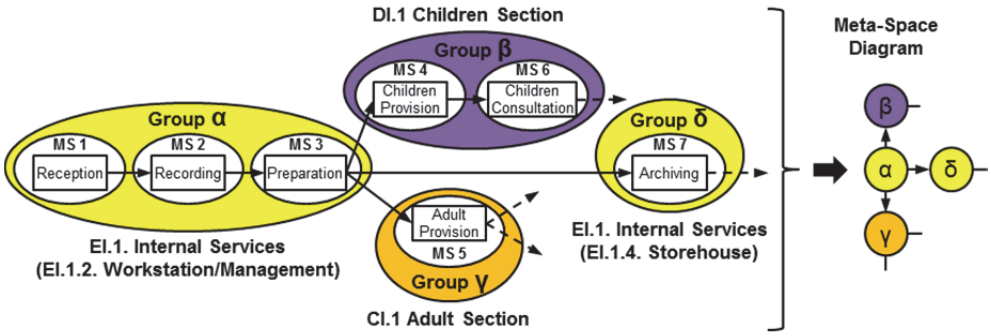


Figure 87 – Meta-Space Diagram Principle illustrated on a limited part of the book cycle

A simplistic view of architectural programming is proposed to introduce the different meta-space diagrams based on the previous taxonomy (Figure 88). Client needs are harvested orally, with little or without structure. Their needs define a fuzzy and messy solution space at first. From these needs, the programmer organises and cleans them to provide requirements (i.e. constraints for the architect). He formalises the functional requirements into a functional diagram illustrating functional spaces, their typology, and relations. The brief and its functional diagram provide constraints on the solution space and limit the design freedom of the architect. From them, the architect designs the building through drawings (plans) or 3D views.

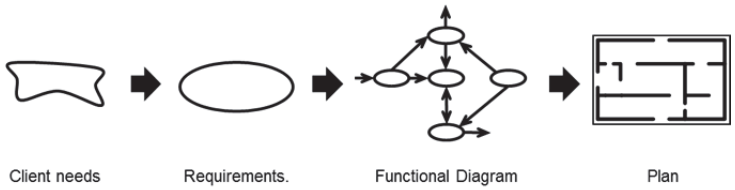


Figure 88 – From needs toward proposals, positioning of the functional diagram (already introduced in Figure 81)

6.2. Operational Meta-Space Diagram

The operational meta-model diagram is the first formalisation of the requirements before creation of the functional diagram. In a functional diagram, the functional spaces and their functional relationships are represented. The functional spaces are the result of the requirements processing. The operational meta-spaces are proposed as a prerequisite to the definition of these functional spaces (Figure 89). The operational meta-space diagram formalise the operational meta-spaces.

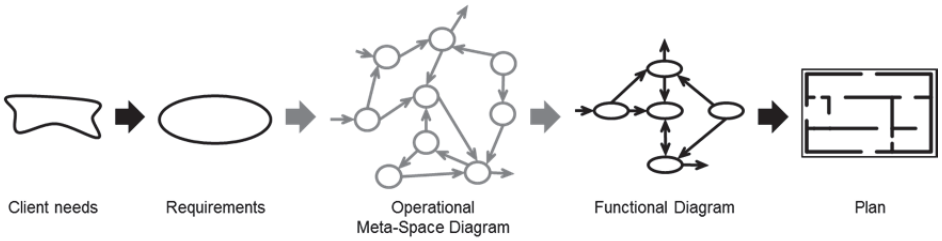


Figure 89 – Positioning of the Operational Meta-Space Diagram

In order to formalise the building definition domain, we propose to use a set of functional models. Therefore, the information contained in a brief can be modelled with several different and complementary viewpoints. The operational meta-space diagram abstracts the functional models that contain the concept of Operation (e.g. IDEF0, Activity Diagram, or State-Transition Diagram). The operational meta-space diagram gets rid of the formalism proper to functional models. It resumes them into simple graphs, i.e. interconnected bubbles.

In Figure 90, the final decomposition of an IDFE0 model is used to generate an operational meta-space diagram. The object flows (input and output) between operations are used to represent the edges whereas operations are transformed into bubbles (e.g. A61 becomes OMS61). The formalisation of the transition from modelling languages to the operational meta-space diagram is not presented in this prospective research but should be part of the perspective (industrialisation of the research contribution).

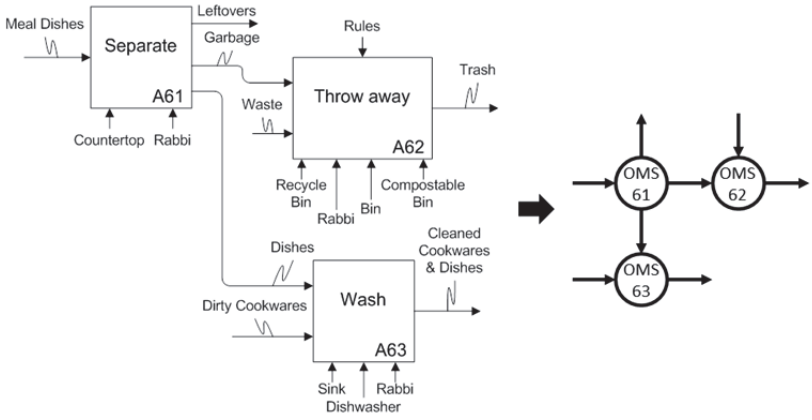


Figure 90 – From IDEF0 model to Operational Meta-Space Diagram (cleaning activity in a kitchen)

The difference with the Bubble diagram is that the information is stored in the building definition domain. Each operation and its requirements are modelled in an operational meta-space. If there are two different flows between two operations, they are synthesized into one link between two bubbles. In the example case, controls and mechanisms from the IDEF0 model are not represented into arrows in the operational model. Only input-output flows are represented. The information is stored behind the line (e.g. in the data-base) but they are not directly represented. Later on, several viewpoints can be taken on the same operational meta-space diagram. Each viewpoint can have a specific legend (e.g. coloured lines and bubbles depending on the information type) to represent a set of information.

Information contained in the functional model can be used on the operational meta-space diagram to provide different viewpoints. The same basis is used to represent information or viewpoints from other functional models. In the kitchen example, the difference between clean flows and dirty flows is of interest to avoid contamination of the food. This distinction can be represented in an operational meta-space diagram (Figure 91). The view is limited to a little amount of information at a time. Functional models tend to be quickly complex to understand and manage. The operational meta-space diagram is a way to simplify the multiple views on the same system.

Figure 91 – Quality of input-output flows during the cleaning activity

A larger example, the book life-cycle in a library is proposed in Figure 92. In this case, the BPMN model introduces a distinction in the processing of documents between adult and child contents. This distinction is kept in the operational meta-space diagram leading to two distinct operational meta-spaces for the same operations (i.e. putting away and on site consultation). Through the operational meta-space diagram, the consistency of requirements can be challenged. As a distinction is made between adult and child contents in their storage, is it enough to keep the

distinction only there or should the distinction be made when the book are returned? In the modelled example, the distinction is only made for the storage but there could have been two different operational spaces for returning adult books and child books.

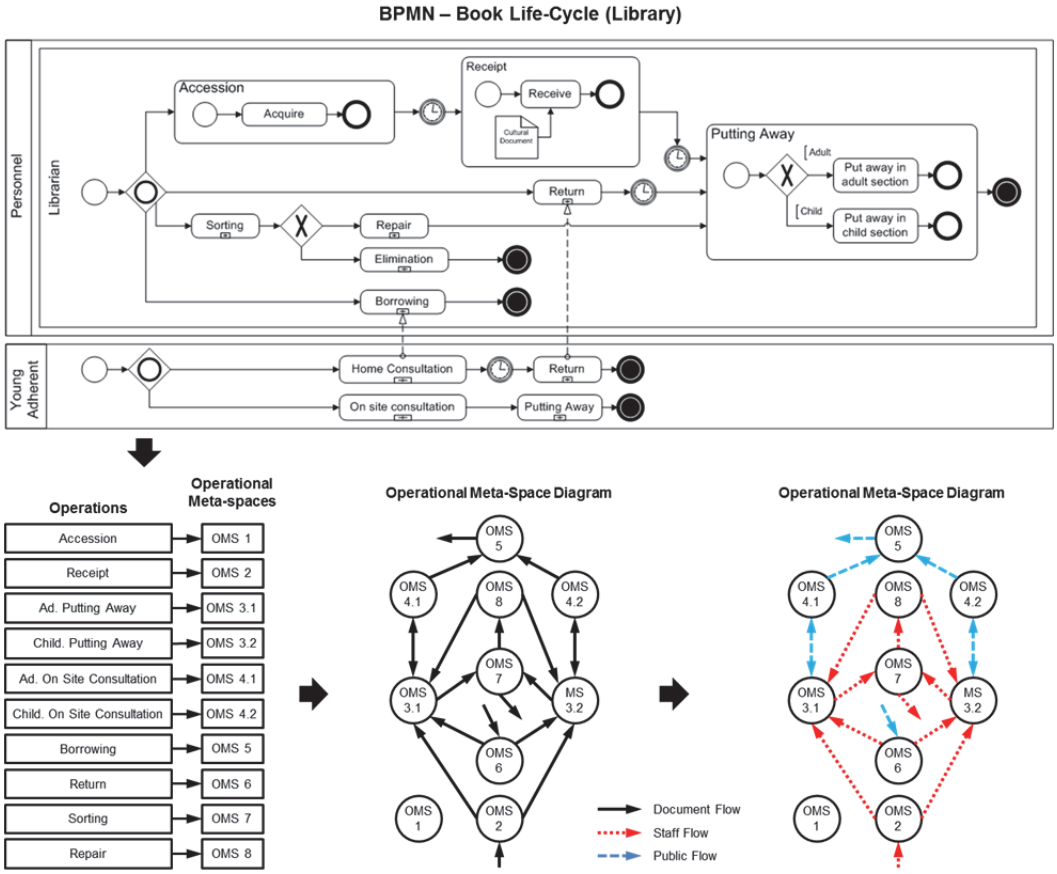


Figure 92 – From a BPMN functional model to an Operational Meta-Space Diagram

6.3. *Functional Meta-Space Diagram*

Among the different deliverables, there is continuity in the information from its raw version (i.e. client needs) to its first graphical expression (i.e. the functional diagram). The problem is that the requirement processing required to provide the functional diagram is obscure. There is a lot of information lost or hidden (i.e. implicit) in the process. The information behind the lines and spaces in a functional diagram is difficult to trace back in the brief. An intermediary deliverable in the briefing process is thereafter proposed to deal with it through the functional meta-space diagram (Figure 93).

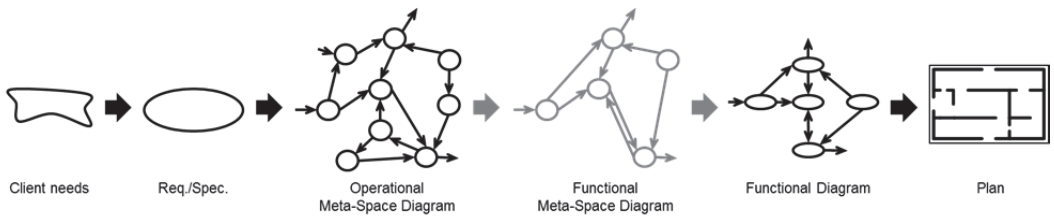


Figure 93 – Positioning of the Functional Meta-Space Diagram

The functional meta-space diagram is defined as an abstraction between the brief and the functional diagram. Its first use is to process requirements and design the functional diagram during the briefing process. The functional diagram synthesises organisational and functional requirements further developed in the brief. The functional meta-space diagram is developed to keep a trace of these requirements from the brief to the functional diagram (Figure 94) and from

the functional diagram to the brief in its second use (Mauger & Kubicki 2013). Chapter IV provides more details on the application of the functional meta-space diagram on case studies.

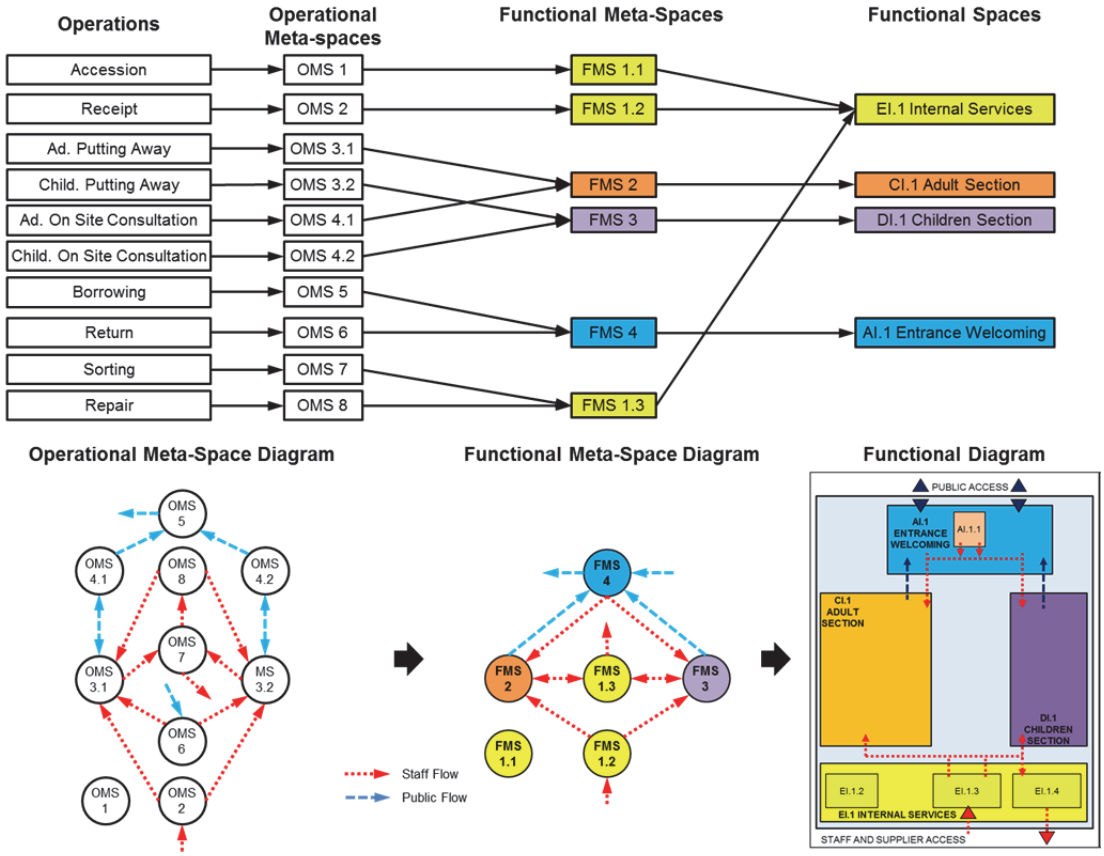


Figure 94 – From operations to functional spaces using meta-spaces artefacts and diagrams

The benefit of the graph form of the meta-space diagram is to allow multiple layouts based on the same functional diagram (Figure 95). The functional diagram could suggest a required layout by the programmers depending on the programmers' graphical convention. The meta-space diagram convention stays flexible regarding the layout and keeps trace of the requirements. As a result, based on the same functional diagram, a huge amount of functional meta-space diagrams are possible, just as for the design proposals.

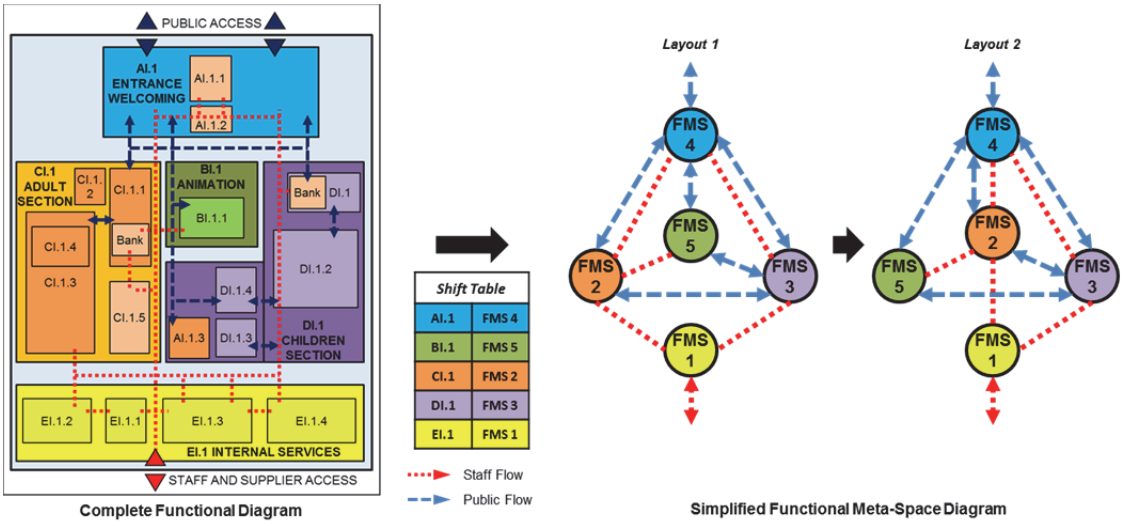


Figure 95 – From the complete Functional Diagram to the complete Functional Meta-Space Diagram with two different layouts based on the same content

6.4. Design Meta-Space Diagram

The design meta-space diagram is a design artefact used to abstract the design proposition of the architects. It allows the assessment and comparison of architecturally different propositions based on their functional aspects modelled using a functional meta-space diagram (Figure 96). This is of interest during the functional assessment of design proposals and the brief-project alignment. However, the design meta-space diagram contains less information than the functional diagram. The lines that connect the design meta-spaces only represent the access connections to the designed spaces. The rest of the information is hidden in the design and requires certain expertise to be deciphered. However, from time to time, the architects provide more information about their design by, for example, indicating their implementation of the functional spaces in their design (Figure 97).

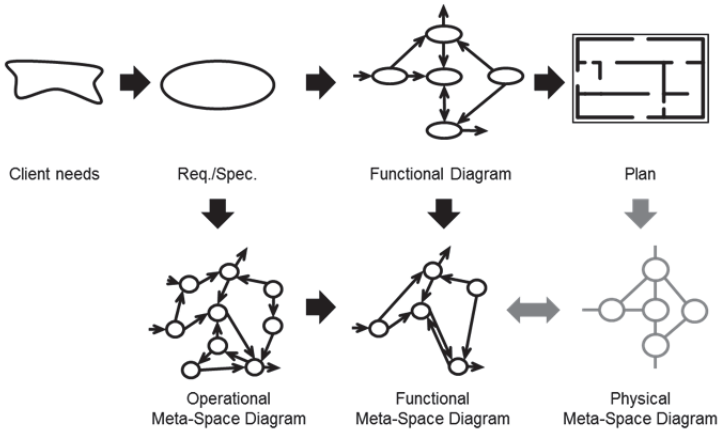


Figure 96 – Positioning of the physical meta-space diagram

Another interesting feature of the physical meta-space diagram is the capability to compare different architectural project on the same functional basis (Figure 97). The shapes of a building can be difficult to read by a layperson. Providing an abstraction of the buildings makes it easier to assess certain characteristics or properties, for example the circulations in a building.

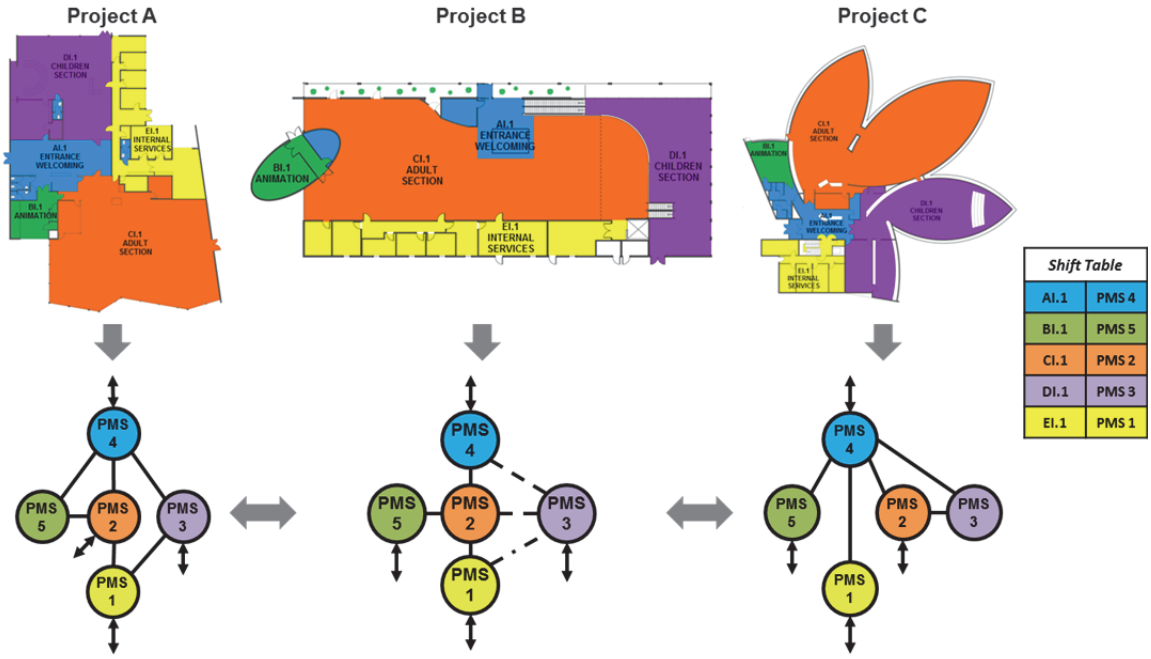


Figure 97 – Comparison of three architectural projects for the same brief (multimedia library)

6.5. *In Summary*

In this section, the **Meta-Space** artefacts were used to develop diagrams. These diagrams can support the briefing process by tracing information from the clients' needs to the design proposal. The processing of requirements is also supported through a variety of viewpoints using the same formalism. Its first use concerns the analysis and processing of requirements contained in the various functional models. Their traceability is, then, ensured through the building definition domain. A key element in the **Meta-Space** contribution is the ambiguous link between the concepts of **Activity** and **Space**. As an **activity** is performed in a **space**, models representing the concept of **Activity** can be directly translated into meta-space diagrams. Therefore, the meta-space diagram becomes an intermediary model among the other models used to graphically represent briefing information (Figure 98 and Appendix G). Information contained in an engineering model is not directly represented in a meta-space diagram but can be highlighted using colours or specific line styles.

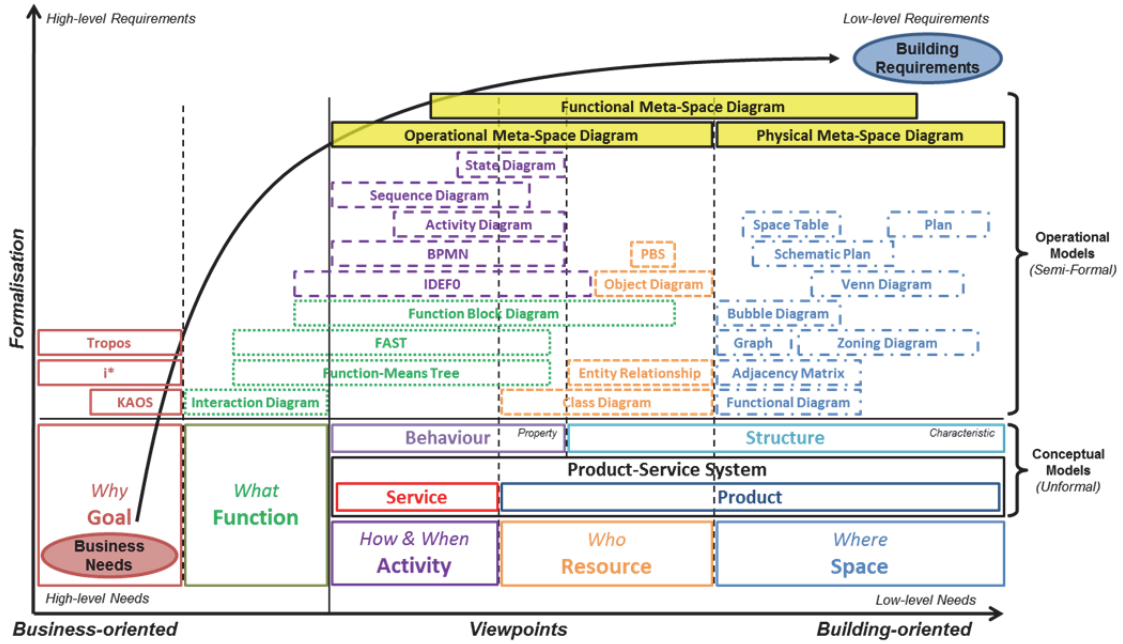


Figure 98 – Meta-Space Diagrams Coverage of the Definition Domain

7. Synthesis

In this chapter, we proposed two sets of models (Table 14). The conceptual models were developed based on two paradigms (i.e. PSS and FBS) to provide a structuration layer to the building definition domain. The PSS paradigm was used to model the object of study and its perimeter as a SoPSS. The FBS paradigm was used to model, structure, and complete the definition domain and the transitions between constructs through the GFBS design model.

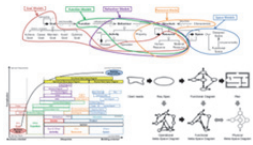
The operational models provided a formalisation layer to the definition domain based on existing modelling languages. In the first part, we proposed a literature overview of the existing modelling languages that could support the partial formalisation of the definition domain. Each model provides a specific viewpoint on the building definition domain with some overlaps between models except regarding the **Space** construct.

In order to create an overlap of the **space** models with the other operational models, we proposed a new design artefact (i.e. **meta-space**) and a corresponding graphical model: the meta-space diagram. These meta-space concepts and diagrams provide a support to ensure certain consistency and traceability between the deliverables. As a result, information formalised in other engineering models can be traced back in the architectural programming deliverables.

A smart combination of operational models should, therefore, support the full formalisation of the definition domain and a practical manipulation of the building definition domain through multiple viewpoints. Thus, the definition domain becomes the core element of architectural programming in which the programmers have to inform the clients. The operational models are intermediaries which support the information process by graphically formalising the needs into requirements.

To conclude this chapter, different models are introduced for different purposes. They are presented using a static view with little consistency between each other. The next chapter will provide a dynamic view that articulates and coordinates the different models regarding the briefing process. One of the scientific objectives is to cover the definition of a tool-supported framework that supports the whole architectural programming. Each model is a tool that can be used for different purposes according to the briefing step.

Table 14 – Research Output of Chapter III: Conceptual and Operational Models

		Research Activities
		Build
Research Outputs	**Constructs** ***Chapter II***	*Definition Domain* 
	Models ***Chapter III***	*Conceptual Model*  *Operational Model* 

Chapter IV – DEFINE A BUILDING

The previous chapters introduced a set of constructs and models to structure and describe the building definition domain. This chapter presents a method built on these design artefacts (Figure 99). It represents a possible instantiation of the GFBS design model (or theory) for the requirements definition of *building systems*. In this chapter, the reader will find contextual illustrations of the constructs and (a consequent amount of) models presented in an abstract way in previous chapters. To illustrate the method, the four case studies introduced in Chapter I Section 4.4 are used.

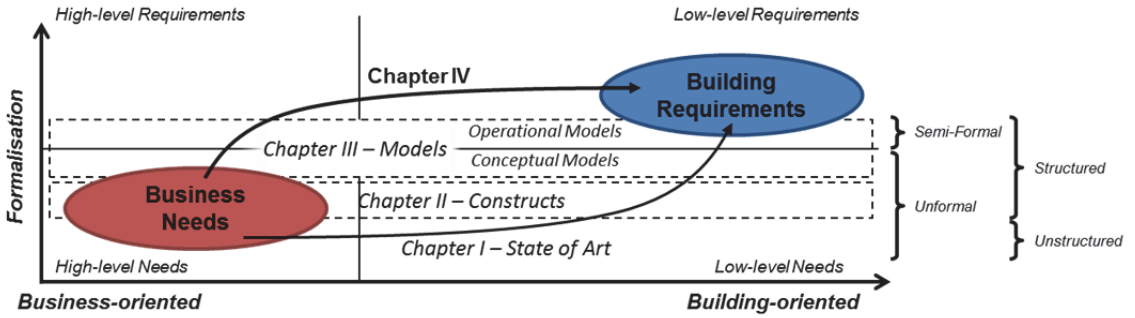


Figure 99 – Positioning of Chapter IV

Section 1 of this chapter proposes different modelling of the briefing process AS-IS (i.e. different viewpoints) as a reference to position my contributions. Section 2 presents the GFBS briefing framework from a theoretical but illustrated viewpoint. In Section 3, the GFBS briefing framework is instantiated by process model views and positioned regarding the AS-IS briefing process. Section 4 introduces a verification of the GFBS briefing framework through an assessment framework based on the same design artefacts. Section 5 concludes Chapter IV.

1. Introduction

The aim of the briefing process is to ensure the best possible quality of the future building (Certu 2010). There is currently no standard process or "formal" method to support it. The briefing process as performed by professional programmers is generally experience-based. It highly depends on the project's context (e.g. kind of building, stakeholders, etc.). However, as a first input for this section, an interpretation of the briefing process made in France is proposed in Figure 100. This process is based on documents from a professional training on architectural programming.

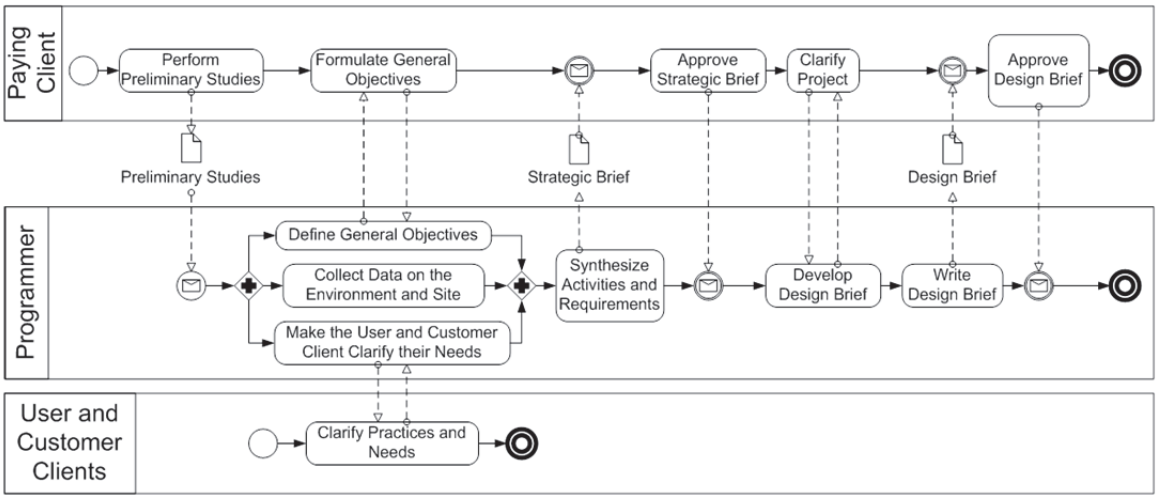


Figure 100 – BPMN modelling of a briefing process

The paying client performs the preliminary studies. The preliminary studies are communicated to the programmer. Based on this input, the programmer defines the general objectives with the paying client, collects data on the environment and site, and clarifies the user and customer clients' needs and practices with them. The programmer then synthesizes the main activities and requirements in a strategic brief. The strategic brief is submitted to the paying client who gives his approval. Based on the strategic brief, the programmer develops the design brief with the clients. Finally, the programmer submits the design brief to the paying client who has to approve it before contacting the designers.

This modelling of the briefing process represents the information exchange between the main actors of the briefing process. Most of these exchanges are realised through interviews or discussion groups (Heintz & Overgaard 2007), or more or less contractual paper-based documents (DGUHC 2000). The loops and iterations are not represented in the model but have to be considered to really model the reality.

In the frame of standardisation led by BuildingSMART, the Information Delivery Manual (IDM) standard focuses on the (object-based) information exchange processes and the realisation of an adapted Model View Definition (MVD). A mapping of the briefing process (Figure 101) is also proposed as a basis using a Building Information Modelling (BIM) process (Jerving 2011).

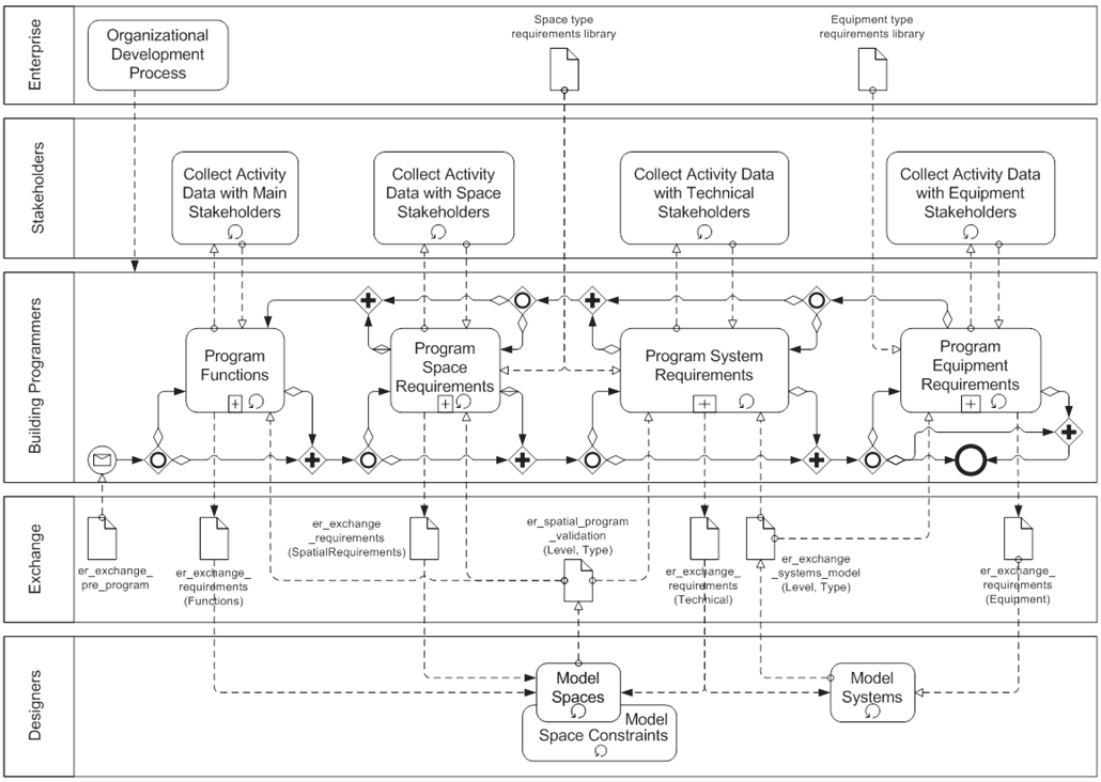


Figure 101 – IDM Building Programming process map from BuildingSmart (Jerving 2011)

In both cases, the needs and requirements are based on the activities perform by the user and customer clients (i.e. stakeholders) of the future building. The added values of this research are:

- the Service integration to the briefing process for thorough understanding and documentation of the building usages and requirements, and
- the interrelation of high level requirements (i.e. Goals and Functions) and low level requirements (i.e. Behaviour and Structure) as a support to decision making along the briefing process.

Information about the building part (i.e. product parts) and the service part are distinguished and put into relationship through the various contributions.

This research provides a rich theoretical basis for structuring and analysing information about the building system by making several distinctions between the product parts and service parts of the building system, between solution-neutral information and solution principles, between context specific and context independent information…

2. GFBS Briefing Framework

The GFBS briefing framework for architectural programming is articulated around 4 different states and decomposed into 8 steps (Figure 102). The first, the *observed state*, refers to the **environment** as it is before considering the *building system*. This state gathers facts about this **environment**. These facts describe the reality with its issues and components. In order to solve some of the issues regarding part of the components, the contracting owner (i.e. the paying client) asks for a *building system*. The second state defines this *building system* as a black box regarding the environment. It provides *expected* results to be attained by, with, or through the future *building system*. To attain such results, the building system needs to be described in its content. The third state focuses on this description. The *building system* is considered as a white box and all of its components are described. The resulting information is gathered to form the brief, i.e. the contracting owner's *demand* to the architect. Based on this information, the architect designs *proposals*, i.e. a building layout and an architecture that should fit the demand.

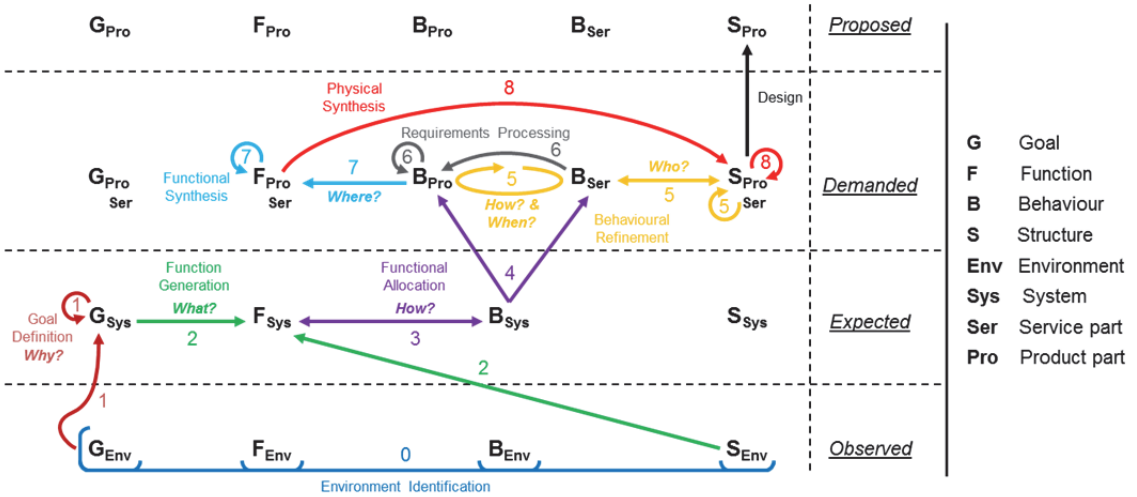


Figure 102 – GFBS Briefing Framework

To illustrate the framework, we used the data from the various case studies. Each case study provides a different set of heterogeneous data. As a result, the full coherence between the illustrations along the framework could not have been achieved. Figure 103 represents the coverage of each case study regarding the definition domain/framework. Only two parts of the framework were not covered by the case studies due to lack of data and resources.

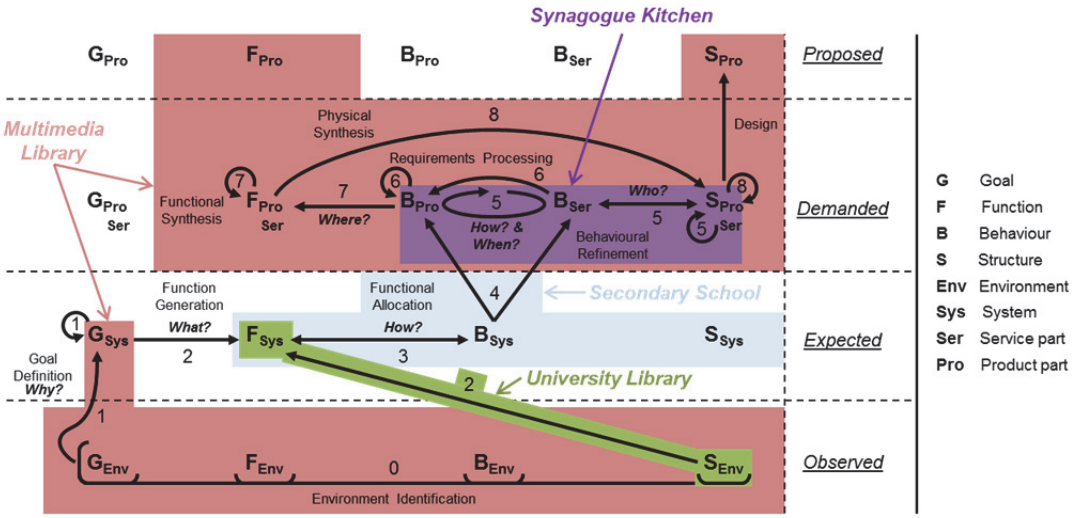


Figure 103 – Case Studies Coverage of the Framework

The following framework is presented as a waterfall model (a straight succession of stages to follow) to make its presentation easier and clearer. In reality, this framework is a recursive process similar to the zigzagging principle in the Axiomatic Design Framework (Suh 1998). Each concept is refined regarding the previous instantiated concept (Figure 104).

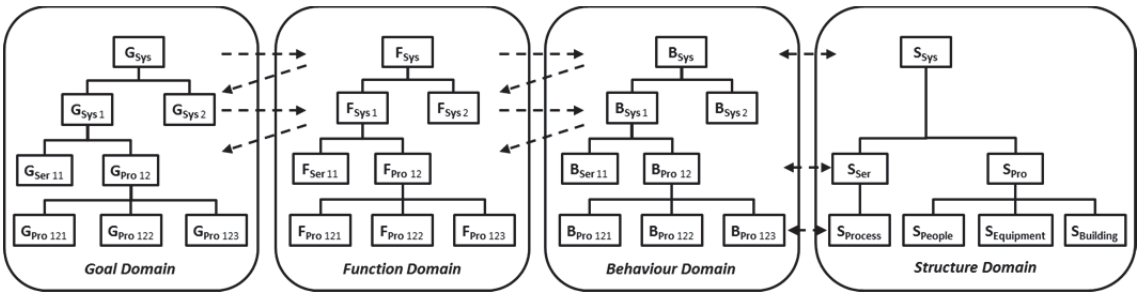


Figure 104 – Axiomatic Design refinement principle of the GFBS framework

2.1. Environment Identification (Step 0)

The first step of any architectural programming is to understand the context of the construction project. In this research, the focus is not on the project itself, but on the building system. Information about the project (e.g. who is involved, what kind of contract rules the project, etc.) is supposed to be known and is not included in this framework. The focus is on information about the future building system and its surroundings.

Objective

The objective of the Environment identification is to define and clarify the current (i.e. AS-IS) situation before the building system's influence. It represents the starting unsatisfying point that the building system will try to change. The object of the study is the Environment and its point is to document why (and which) things have to change.

Process

In this research, we consider that a building system is usually required to change a local unsatisfying state of affair. The first step in this framework consists in gathering information about the context and to clarify this state of affair, the AS-IS situation. This information can be structured in G_{Env}, F_{Env}, B_{Env}, and S_{Env} (Figure 105). Each one of them concerns already defined strategies (e.g. local policies), significant facts (i.e. local history), as well as observable and measurable behaviour associated to existing artefacts. This information is usually cropped during the earlier stage of the construction project development (i.e. preliminary studies). This step focuses on three different levels of studies: opportunities, localisation, and feasibility.

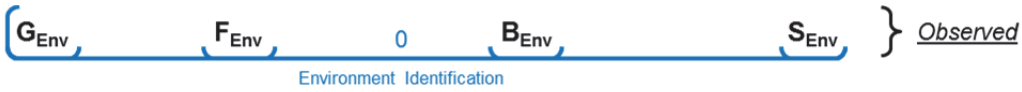


Figure 105 – Environment identification

Support

This step can be supported by requirements elicitation techniques (Table 15). Any investigation technique that allows answering and documenting who and what is part of this Environment, how they currently behave, and why things have to change, can support this step. Class diagrams and organisational charts can also be handy to structure information on the external elements (S_{Env}). The Current Reality Tree (CRT) from the Thinking Processes of the Theory of Constraints can also support the organisation of the AS-IS situation in a tree view (directed graph).

Table 15 – Requirements elicitation techniques (Coulin 2007)

Traditional Techniques	Cognitive Techniques	Group Techniques	Contextual Techniques
Interviews	Card Sorting	Brainstorming	Ethnography
Questionnaires	Laddering	Requirements Workshops	Observation
Task Analysis	Repertory Grids	Focus Group	Protocol Analysis
Domain Analysis		Creativity Sessions	Apprenticing
Introspection		Nominal Group Technique	Prototyping

Deliverable

The preliminary study synthesis document resumes this information about the context. The document should contain information on the following elements:

- Local context (G_{Env}, F_{Env}, B_{Env})
- Stakeholders (S_{Env})
- Real estate information (S_{Env})
- Local history (G_{Env}, F_{Env}, B_{Env})
- Details on inhabitants and neighbourhood (S_{Env})
- Living environment (B_{Env} ,S_{Env})

- Local area, sites, and topology (S_{Env}) (e.g. Figure 106)
- Socio-economic and political context (G_{Env}, F_{Env}, B_{Env})

This content is shortened in successive deliverables, i.e. the strategic brief and the design brief.

Illustration

This step is illustrated with information taken from the brief of the multimedia library Theodore Monod in Betton, France (introduced in Chapter I Section 4.4.1). The information in the next paragraph is illustrated on the local map of Betton (Figure 106). Figure 107 is an extract of the local urban plan of the city.

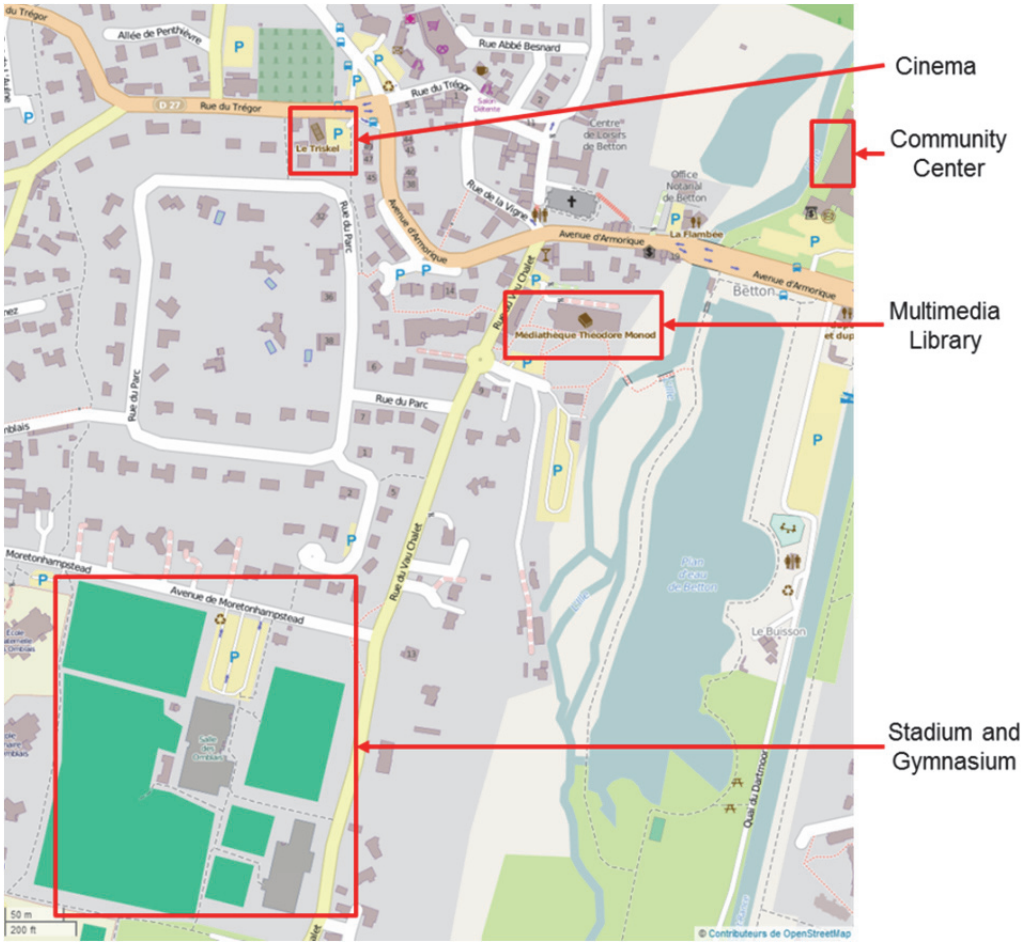


Figure 106 – Map of Betton positioning the community facilities and the multimedia library (2014)

The city hall of Betton wants to develop a true cultural policy to answer the population claims (G_{Env}). There is already a stadium (S_{Env}) for practicing outdoor sports (F_{Env}), a gymnasium (Salle des Omblais) (S_{Env}) for practicing indoor sports (F_{Env}), an auditorium (Le Triskel) (S_{Env}) for watching movies, as well as listening to music or attending shows (F_{Env}), and an old library (S_{Env}) mainly for reading books (F_{Env}). Finally, it appears that this population of 9,000 people (S_{Env}) globally has a high interest and level of involvement in cultural activities (B_{Env}) regarding the participation rates.

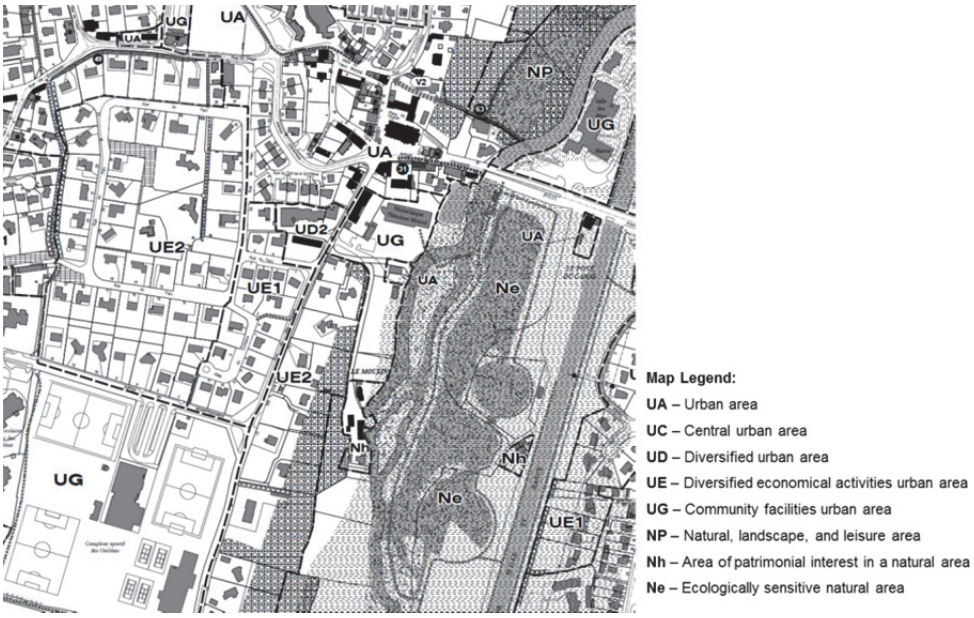


Figure 107 – Local Urban Plan of Betton around the multimedia library

The class diagram can be used to represent or structure the taxonomy of certain external elements (domain information). In the multimedia library, it is convenient to know the different kinds of cultural documents that can be distributed (Figure 108). Each kind of document has its own properties and characteristics that make it more or less easy to propose. For example, ancient books are sensitive and cannot be proposed without high constraints. Periodicals arrive every day and should be available quickly. The processing at their arrival should be as short as possible. It is part of the domain understanding to know the difference and to be able to advise the clients and to inform the architects. It is also a way of putting the quantity of cultural documents to store into perspective. Numerical documents accessible on the Internet are not placed on shelves. They can represent half of the collections if not more. At this stage, only the higher element is considered. In this example, "cultural documents" will represent the whole taxonomy of possible documents. The processing of each kind of document will be used in a later step. The aim is to simplify the study for the beginning and to add complexity step by step.

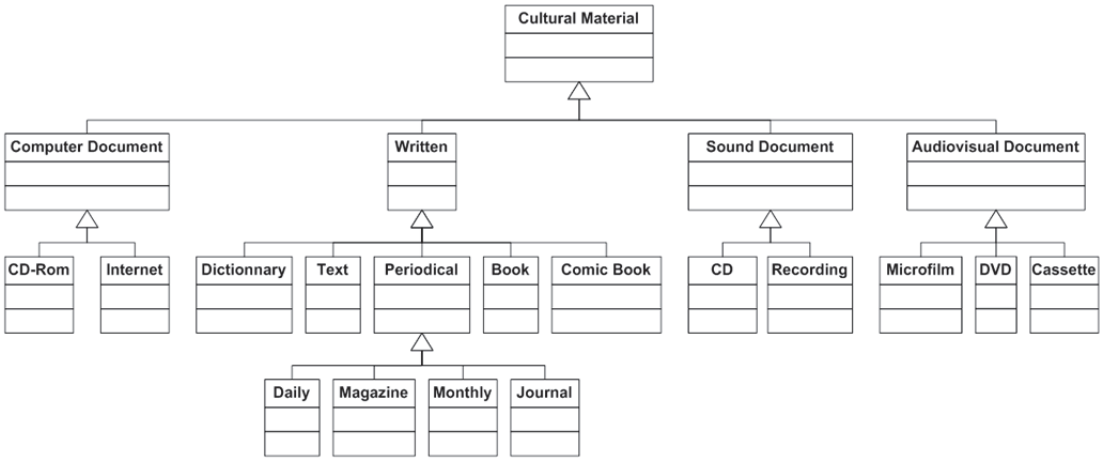


Figure 108 – Taxonomy of cultural documents that can be distributed in a multimedia library

2.2. Goal Definition (Step 1)

A definition of the main goals is a real challenge. Clients usually want to achieve as many objectives as possible with such an expensive project. There are different kinds of objectives: political, social, cultural, economic, urban, architectural, environmental, or technical. It is often a

key argument to support the funding and realisation of such projects. However as best practice, we advise that each building system should have one main goal to achieve and a few secondary goals of less importance. The main goal (G_{Sys}) provides an end to attain through the building system. The achievement of such a goal is not self-evident (as it requires a complex system).

Objective

The first objective of the goal definition is to clarify with all stakeholders the main goal to achieve through the building system that will condition its existence and life time (1.1).

The second objective is to refine the high level goal (G_{Sys}) into more concrete and precise goals (1.2).

Process

The main goal represents a state of affair to achieve through the building system. This state of affair is usually abstract and broad (i.e. high level information) (Koutamanis 2013). Its formulation implies an end that can be attained only through long term effort (Pena & Parshall 2001). The main goal is the raison d'être of the building system. If the main goal has been achieved, the building system does not have a reason to exist any longer. The definition of the main goal should integrate continuity over the time (in line with the building life cycle) which makes its formulation not so evident.

Based on the contextual information, the desired state of affair, i.e. the expected TO-BE situation (Figure 109) can be defined by the customer. The point is to define the main rationales behind the necessity of the building system (S_{Sys}), its raison d'être. G_{Sys} is derived (1.1) from information about the context (i.e. the Environment): G_{Env}, F_{Env}, B_{Env} and S_{Env}.

Thereafter, the main goal of the building system is refined (1.2) as much as necessary to be explicit. Based on the main goal, more precise goals are formulated as conditions for the achievement of the main goal. Each lower level of goals should offer necessary or sufficient conditions to consider the upper goal as achieved. Besides the main goal, there are usually several secondary goals to achieve. These secondary goals should also be refined into more concrete ones.

Figure 109 – Goal definition

Support

The "horned beast" from Functional Analysis can be used to represent the main goals with its main relative concepts.

The goal refinement can be structured using the GORE goal trees from i*, KAOS, TROPOS or any other approach. SMART criteria (Specific, Measurable, Attainable, Relevant, and Time-bound) (Doran 1981) can be followed to formulate goals. In the Theory of Constraints, the Thinking Process approach can also provide support to structure the tree view of goals and states through its Future Reality Tree (FRT) and its Prerequisite Tree (PRT).

Illustration

For example, the town council decided that the main goal was to develop a true cultural policy in Betton (G_{Sys}) (Figure 110).

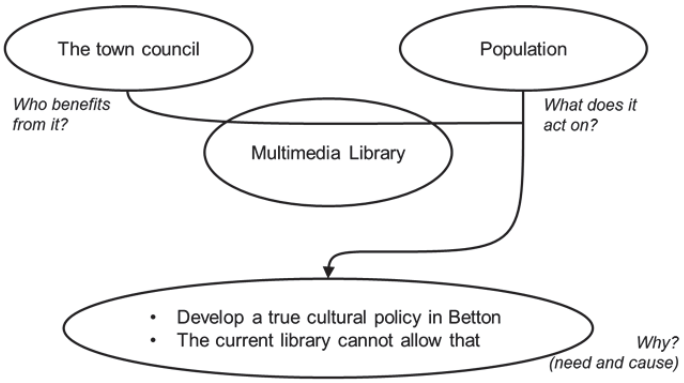


Figure 110 – "Horned beast" representing the main goal of the multimedia library

An illustration of the refinement step was not possible with either of the case studies. Indeed, both techniques supporting the goal refinement require certain commitment from the stakeholders of the case studies. Unfortunately, none of them provided such thing for various reasons, including the fact that most of these case studies were *post ex-facto*.

2.3. Function Generation (Step 2)

When a goal (G_{Sys}) cannot be further refined, a function has to be formulated to express what the building system should do to attain this goal in a solution-neutral way (i.e. independently from its components).

Objective

The objective of this step is to conceptualise the environments' change of states from AS-IS to TO-BE in a solution-neutral way.

Process

The generation of function is based on the initial and the desired state of the building system's environment. A function (F_{Sys}) should express the shift from an AS-IS (G_{Env}, F_{Env}, B_{Env}, S_{Env}) to a TO-BE (G_{Sys}) situation in a solution-neutral way. The building system is considered as a black box in the middle of its environment's external elements. Each function should link the system with at least one and up to two external elements. The system is required to facilitate or create interactions with the environment that will lead to a state change. Without the system, the environment is not able to change on its own. Therefore, from the refined G_{Sys} and required interactions between S_{Env} with or through the building system, functions to be achieved by the building system (F_{Sys}) are defined (2) (Figure 111).

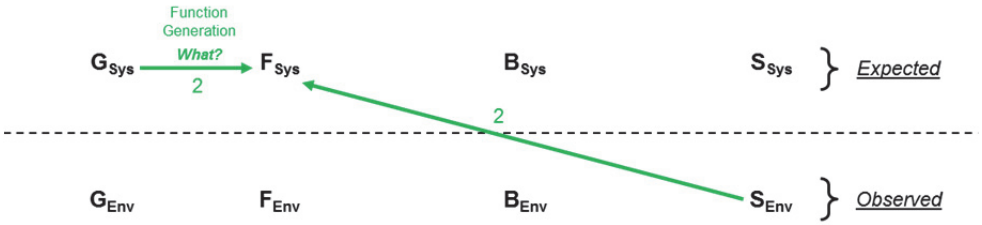


Figure 111 – Function Generation

The S_{Env} to consider are only the higher classes of a common taxonomy (as much as possible) in order to reduce the number of functions and to avoid particularisation at this step. In theory or as a best practice, a function should always be related to a refined goal. It is acknowledged that some functions can be induced by other functions. It is therefore possible to figure out more goals to achieve during the function generation. This is often the case if external elements or situations are too detailed. In the illustration, this case could have been encountered if the population (e.g. people from the university, people external to the university), documents (e.g. different kinds of documents) or information (e.g. different kinds of information) would have been more detailed.

Support

An interaction diagram (octopus diagram) and a functional table can support the generation of function (F_{Sys}) the building system should provide (Figure 112) based on the list of external elements (i.e. environment elements) that composed its environment.

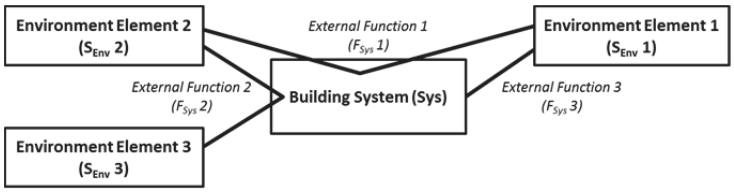


Figure 112 – Interaction diagram based on Functional Analysis method

Illustration

To illustrate this step, the University Library case study is used. This case study provided the most relevant information to build the illustration. Information is taken from the strategic brief and formalised using an interaction diagram (Figure 113). Unfortunately, this information was not sufficient to illustrate the link between the refined goals and the functions.

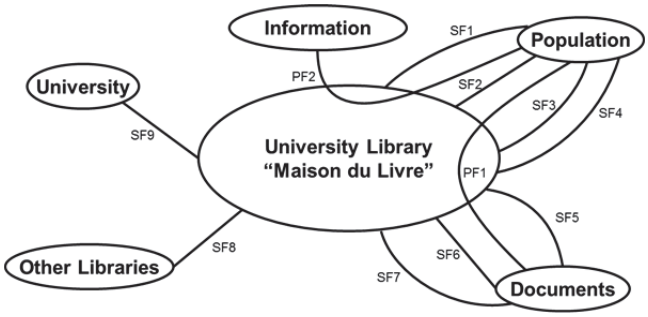


Figure 113 – Interaction diagram of the University Library based on the strategic brief

Table 16 provides a formulation of the building system's functions (F_{Sys}). The grammar and formulation of functions are not yet validated. They remain in a proposal state within this research. To protect the data of the case study, the criteria, levels, and flexibilities inserted in this table are only example values.

Table 16 – Functional table related to the University Library based on the strategic brief

Code	Principal Function	Criterion	Level	Flexibility
PF1	The System allows the population to access documents	Population number Documents number	10,000 1,200,000	+/- 1,000 +/- 10%
PF2	The System is able to distribute information to the population	Information kinds	Scientific Educational	None
Code	**Secondary Function**	**Criterion**	**Level**	**Flexibility**
SF1	The System allows the population to work	Place number	800	+/- 10%
SF2	The System is able to train the population	Full-time equivalent Trainings	5 15	+/- 1 +/- 5
SF3	The System allows the population to relax	Place number	200	+/- 10%
SF4	The System is able to advise the population	Full-time equivalent	3	None
SF5	The System is able to acquire documents	Document turnover	10%	+ 5%
SF6	The System is able to store documents	Storing shelves	12,000 ml	+/- 10%
SF7	The System is able to maintain documents	Full-time equivalent	2.5	None
SF8	The System is able to exchange with other libraries	Accessible documents	2,500,000	+/- 10%
SF9	The System is able to support the University	Full-time equivalent	2	+/- 0.5

As introduced in Chapter II, there is a link between functions regarding their taxonomy. An external function (i.e. Functions) is a principal function (PF) if it represents a relationship between two external elements through the system, or a secondary function (SF) if it only represents a relationship between the system and an external element. Secondary functions can be induced by a principal function (Figure 114). It means that if the principal function is no longer required, the secondary functions induced by it are also no longer required. This representation of dependencies supports the decision making process when costs are too high to satisfy one function.

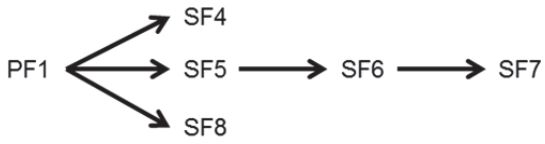


Figure 114 – Functions induction dependency

2.4. Functional Allocation (Steps 3 & 4)

Functional allocation is about definition of programmatic concepts (i.e. high level solution principle) that will ensure the implementation of functions.

Objective

The objective of the functional allocation is to define how functions will be implemented in terms of behaviours associated to main structure elements.

Process

After knowing what the building system (S_{Sys}) should do (F_{Sys}), it is necessary to define how it will be done but also to detail which part of it will do it (i.e. S_{Pro} or S_{Ser}) (Figure 115). A Function is performed (3) through Behaviour of the building system (B_{Sys}) seen as a black box. The function allocation to either S_{Pro} or S_{Ser} consists in refining the B_{Sys} (4) into Behaviours of the Product (B_{Pro}) and Service (B_{Ser}) parts. It corresponds to the definition of the design strategy (i.e. programmatic concepts) of the Building system (S_{Sys}).

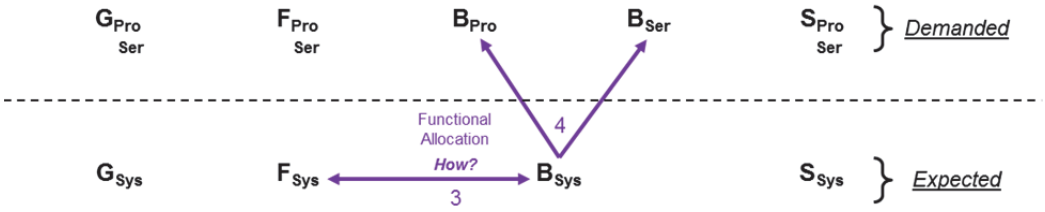


Figure 115 – Functional allocation

Support

Various techniques can support this step such as the Function Analysis System Technique (starting from external functions and considering internal functions as behaviours), the Technology Paths Tree (TPT) or the Function-Means Tree (FMT), the first levels of an SADT/IDEF0. The starting point is always a generic function (F_{Sys}) that is contextualised through behaviours (B_{Pro} or B_{Ser}) associated to either a product part (i.e. properties) or a service part (i.e. activities).

Illustration

To illustrate this step, we use the secondary school case study (introduced in Chapter I Section 4.4.1). The information used to illustrate the step is based on a group work session that was performed with a set of teachers and administrative representatives to achieve a specific goal: children feeling safe at school. The related function could be formulated as followed: “the building system is able to ensure the safety of children”. During the group work, several solution principles were proposed:

- *Architecture, relations and positioning.* The layout and design shape of the building system (S_{Sys}) improve observing the pupils (B_{Sys}) (like in prisons).
- *Materials.* The use of see-through materials for walls and windows (S_{Sys}) prevents bullying through lack of private spaces (B_{Sys}).
- *People.* School's personnel (S_{Sys}) do some rounds by pair scheduled during the breaks (B_{Sys}).
- *CCTV system.* A CCTV system (S_{Sys}) is deployed in the school to watch over pupils (B_{Sys}).

In the multimedia library case, the city hall decided to build a multimedia library system (S_{Sys}) to promote cultural activities (F_{Sys}) through services such as book loans or demonstrations training on new technologies (B_{Ser}) and thematic exhibitions (B_{Pro}). The whole will be hosted in a new building (S_{Pro}).

2.5. Behavioural Refinement (Step 5)

Programmatic concepts are high-level solution principles. In order to define requirements on the building part of the building system, each programmatic concept should be refined into technical solutions. Requirements on the building are based on the description of these technical solutions.

Objective

The objective of this step is to refine the programmatic concepts into technical solutions.

Process

The behavioural refinement (Figure 116) consists in describing the internal functioning (5.1) of the S_{Sys} necessary to fulfil the F_{Sys} allowing the achievement of the G_{Sys} in terms of its components' behaviour (B_{Ser} and B_{Pro}). These behaviours refer to processes and activities to be performed (5.2) by a set of resources (S_{Ser} and S_{Pro}). Each activity and resource requires certain conditions to be performed or used. These conditions constitute the requirements on the building part of the building system.

Figure 116 – Behavioural refinement

Support

The identified tools which are able to support the refinement of the behaviours include but are not limited to: organisational chart, SADT/IDEF0, state list, state-transition diagram, event table, BPMN, activity diagram, or sequence diagram.

The following list of tools can be used to structure the refinement of the structure: class diagram, component diagram, or object diagram.

Deliverable

The description of the technical solutions is formalised in the design brief (i.e. bill of specifications of the building).

Illustration

To illustrate the constructs manipulated during this step, we use the multimedia library example (Figure 117): *a citizen (S_{Env}), e.g. Ms Jane Smith (S_{Env}), can borrow a book (B_{Ser}: Borrowing process) only if she is registered (B_{Ser}: Registration process) in the library system (S_{Sys}) after paying the membership fees (B_{Ser}: Paying process) to a person (S_{Pro}), e.g. M. John Doe (S_{Env}), a personnel (e.g. assistant librarian) (S_{Ser}) in charge of the registrations (B_{Pro}), who will*

record (B_{Ser}: Registration process) her name and address (attributes of the S_{Env}) on a computer (S_{Pro}) connected to the internal network (S_{Pro}).

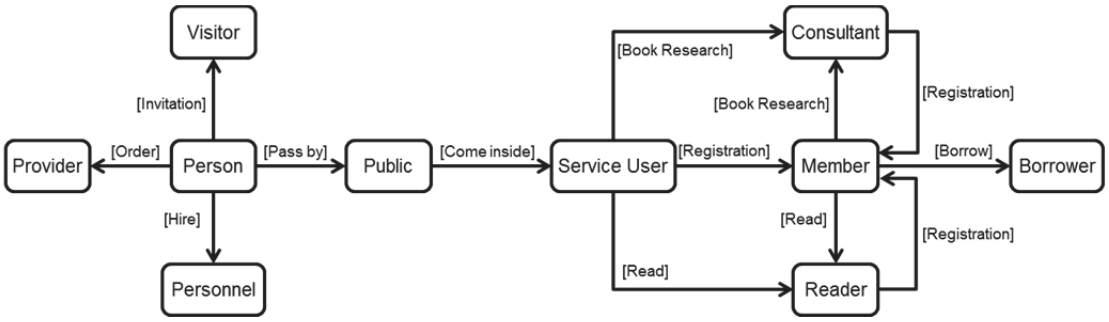


Figure 117 – State-Transition Diagram representing a person rights according to their status

To illustrate the process, we take the synagogue kitchen case study (Figure 118 and Appendix K) (introduced in Chapter I Section 4.4.3) which is richer in detailed information. The behavioural refinement on this case study consists in a description of the kitchen usage through use cases and scenarios. Based on these first elements, each identified use case is developed by using UML diagrams.

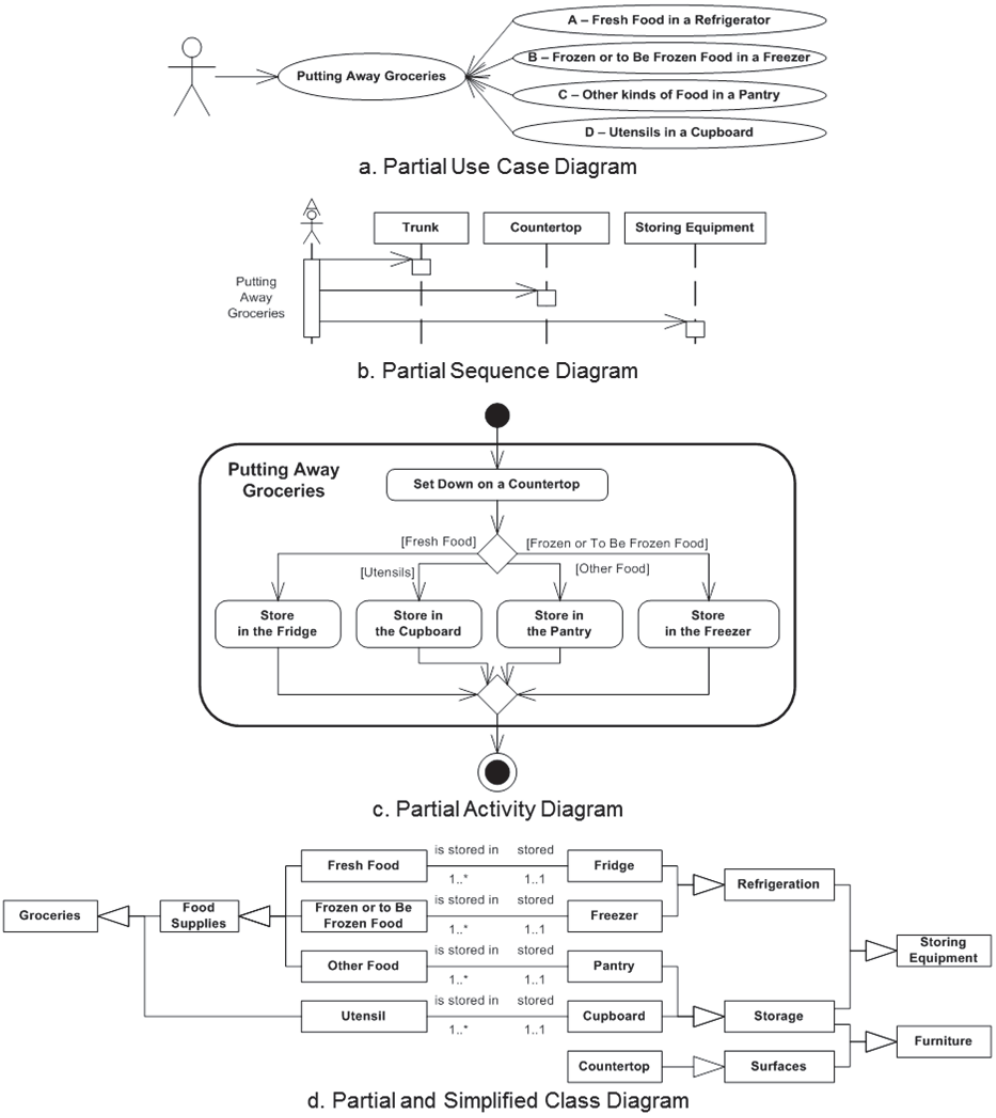


Figure 118 – Partial illustration of UML diagrams applied on the kitchen case study

Each UML diagram formalises a set of information about the kitchen usage, from general to details. The proposed sequence of diagrams could be followed during an interview with the user clients. Regarding the case study, such approach was performed only afterwards to assess the design proposal. The result of the assessment was that the original plan proposed by the architect was not adapted to kashrut rules (Mauger & Berry 2014). As a synthesis and to provide a complementary viewpoint on the kitchen, an IDEF0 modelling is also proposed in the Appendix. The complementary viewpoint represents the flows of input and output through the different activities partially illustrated in Figure 119.

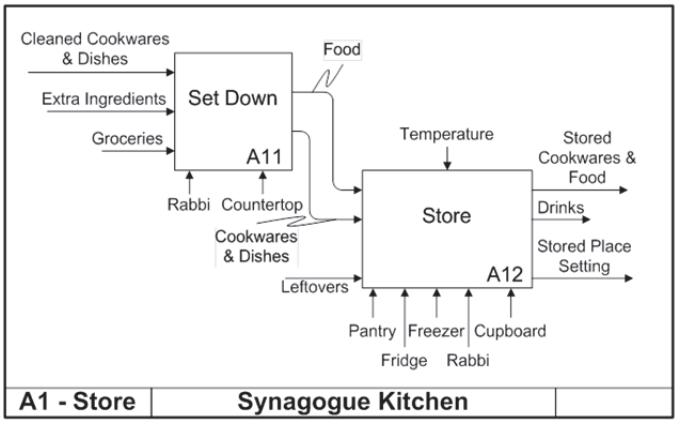


Figure 119 – Partial IDEF0 of the kitchen case study

2.6. Processing (Step 6)

The behavioural refinement generates a lot of information and viewpoints on the building system. This information is the basis of the requirements on the building part of the "building system".

Objective

The objective of the processing is to reduce the amount of requirements on the building part of the building system.

Process

The requirements processing (Figure 120) is performed in three steps. The main principle is that each operation is defined with a set of constraints and requirements. Therefore, an operational meta-space gathering all the necessary conditions is assigned to each operation (6.1). This operational meta-space is considered as part of the building's behaviour (B_{Pro}). As a result, there are as many operational meta-spaces as operations to perform. Based on information about the processes (B_{Ser} and B_{Sys}), the operational meta-spaces can be linked to each other into a graph (6.2). The amount of operational meta-spaces is reduced through their grouping based on their conditions and sets of practical and business rules (6.3) (e.g. both activities require the same resources at subsequent times).

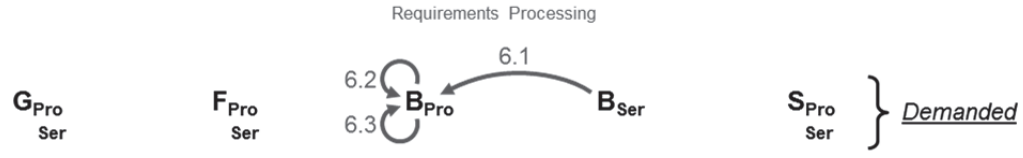


Figure 120 – Requirements processing

The grouping is a convergent process (Figure 121) that reduces the number of virtual ideal spaces and resources required to perform the B_{Sys}. It is guided by high-level building system goals to achieve (G_{Sys}). The usage and operating performance proposed to describe functions of a space by Gobin (Chapter II Section 3.2.3); the space property sets assigned to a space in the IFC

(Chapter II Section 3.5.3), the required physical resources, or the sequence of operations (and their coordination) can be used as grouping criteria. The grouping and addition rules of meta-spaces and their attributes are not defined in this research but require be exploring and formalising in the perspectives. The combination of these criteria can also be considered for the grouping which makes the task more challenging.

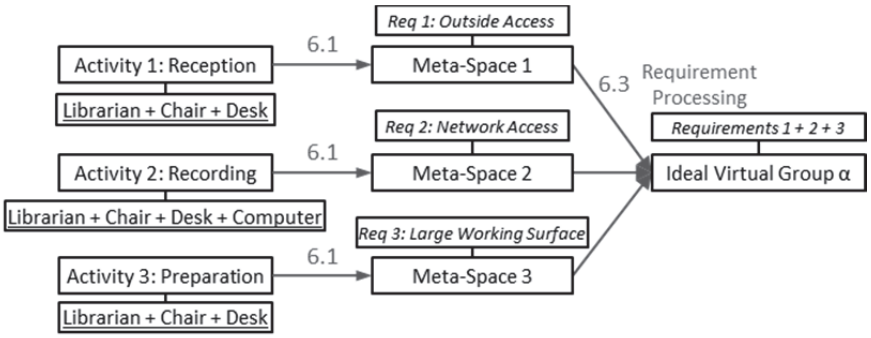


Figure 121 – Requirements processing principle in two steps
(already introduced in Chapter III Section 5.1 Figure 82)

Support

The generation of meta-space is supported by the operational meta-space artefact. The relationships between the meta-spaces are based on information and the conditions on activities. They can be modelled using a dependency model, a functional block diagram, a proximity matrix, or an operational meta-space diagram. The grouping or processing can be supported by a graphical representation of local information such as an affinity diagram, a zoning diagram, a Petri net, a Venn diagram, or an operational meta-space diagram. The best possible support remains an information system that stores all of this information and allows automatic processing.

Illustration

The processing step is illustrated for the multimedia library case study. The reception process (B_{Ser}) of new cultural material (S_{Pro}) consists in a lot of activities (B_{Ser}) (Figure 122). These activities are sequent and all require almost the same resources (e.g. a librarian, a chair, a desk, a computer) (S_{Pro}) to be performed. Their respective operational meta-spaces (B_{Pro}) have specific requirements to be considered (e.g. outside access, network access, and large working surface) (B_{Pro} & S_{Pro}) and relationships (Figure 123). In order to minimise (G_{Sys}) the number of transitions and flows (i.e. in efforts and time), all of them can be grouped in the same meta-space that gathers all of their requirements (Figure 124).

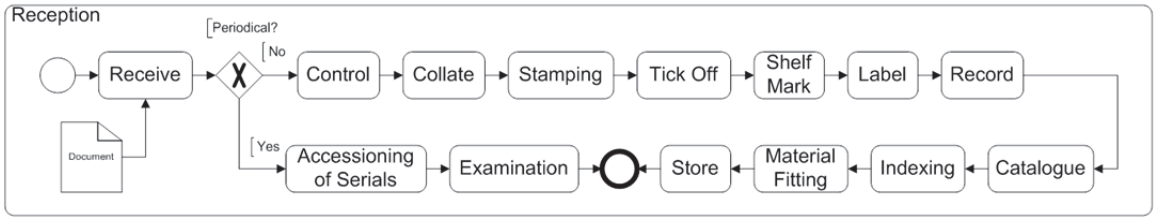


Figure 122 – Partial BPMN of the book cycle in a multimedia library

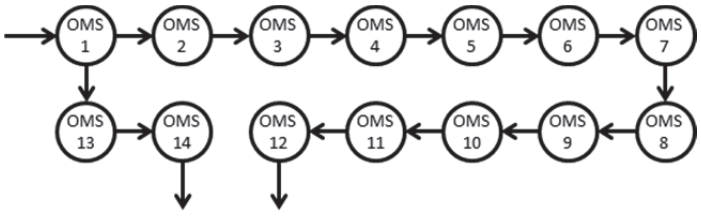


Figure 123 – Operational Meta-Space Graph (6.2) of the partial BPMN

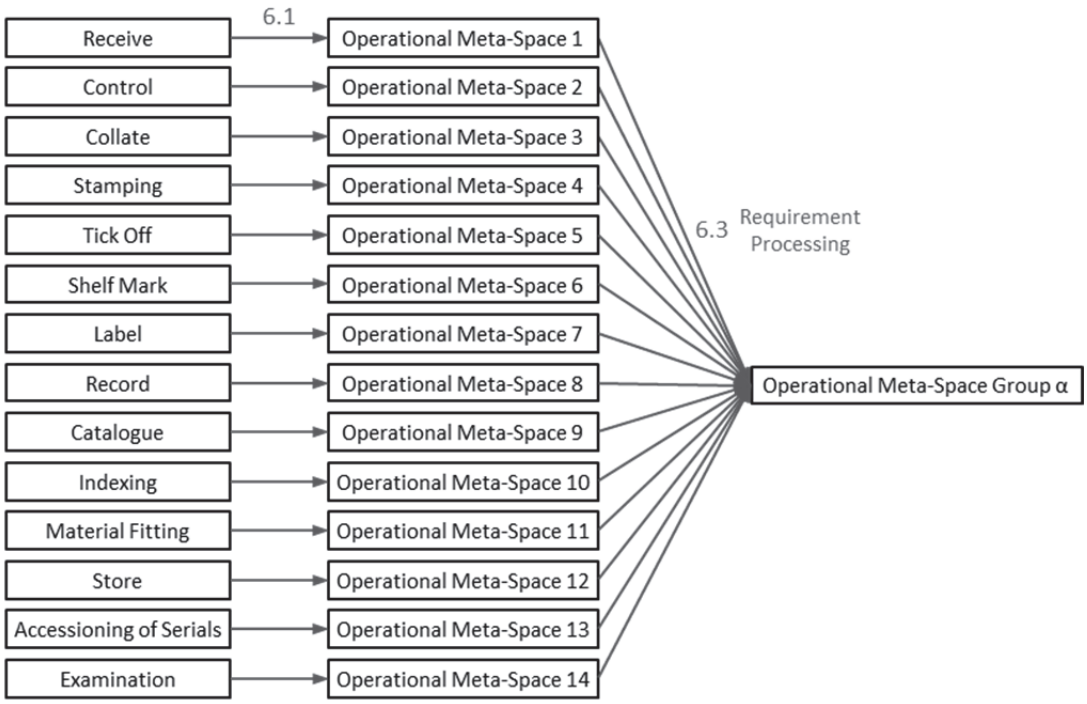


Figure 124 – Processing of the "Reception" Meta-Spaces (6.3)

2.7. Functional Synthesis (Step 7)

The functional synthesis consists in defining the typology and attributes of functional spaces demanded by the client regarding the functionality (F_{Pro}) associated to each operational meta-space group.

Objective

The objective of this step is to sum up the brief information into a set of physical spaces and spatial requirements.

Process

The functional synthesis is structured in two steps (Figure 125). The first step consists in defining a typology of space (7.1) based on the operational meta-space groups that reflects the various functions (F_{Pro}) ensured by the grouping. This leads to the definition of functional spaces. The description of functional spaces indicates the quality (B_{Pro}) and characteristics (S_{Pro}) of each required space without considering the quantity. Functional spaces are used to draw the functional diagram which graphically represents the functional relationships (B_{Pro}) between spaces (7.2).

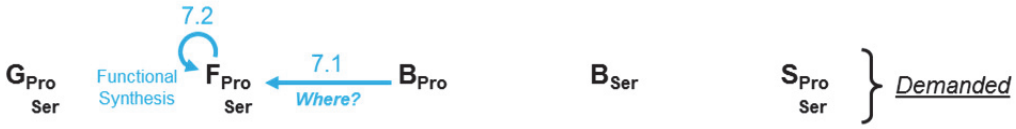


Figure 125 – Functional synthesis

Support

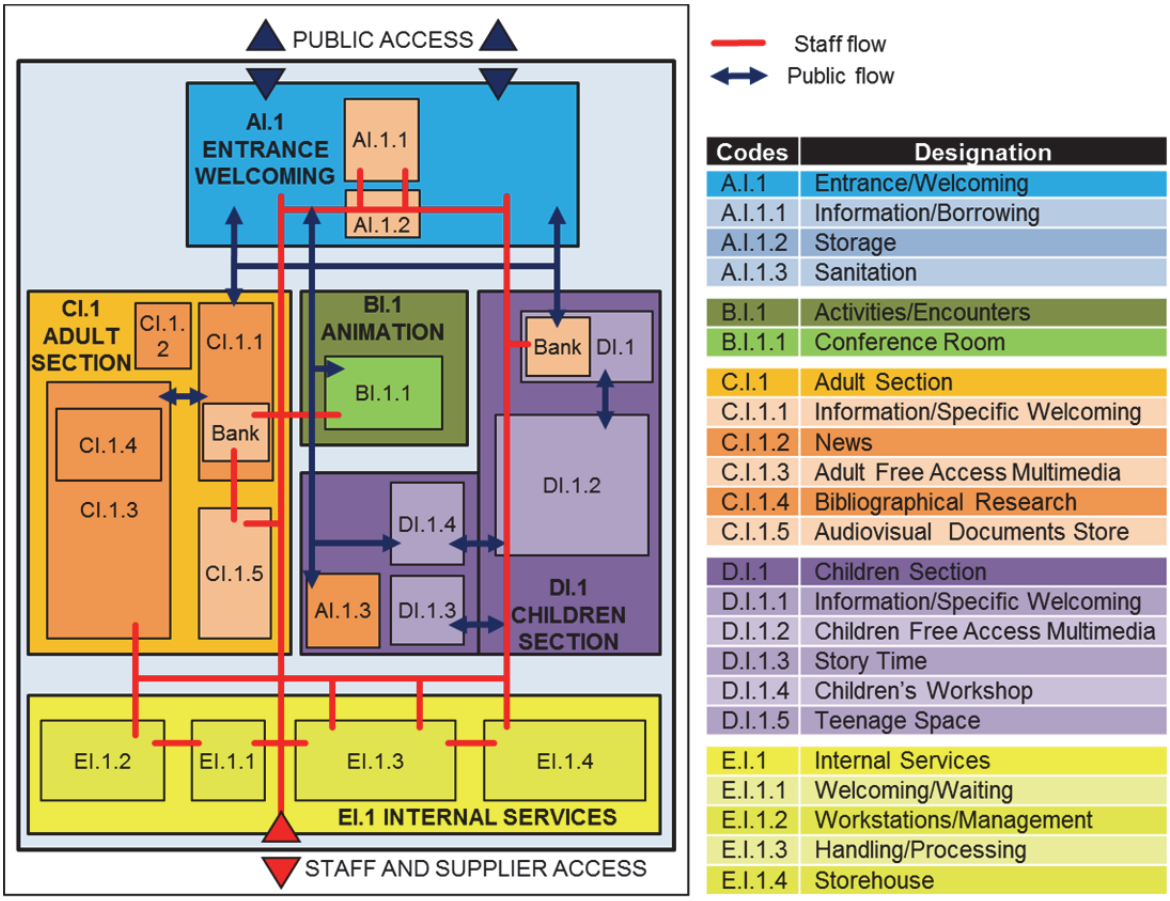
The functional synthesis in this research is mainly supported by the functional meta-space diagram as an intermediary toward the functional diagram. Other graphical models can be used to represent more functional viewpoints on the building such as: a circulation diagram, an affinity diagram, a zoning diagram, a proximity drawing, a general drawing, or a functional drawing. These models are mainly from the AEC domain but other models from mechanical engineering (e.g. substance-champ diagram) or industrial engineering (e.g. Petri Net), can be used too.

Illustration

The functional synthesis is illustrated based on the multimedia library example. The operational meta-space group (B_{Pro}) defined in Figure 125 develops functions (F_{Ser}) associated to a bigger set of functions related to the internal services of the library (F_{Sys}). As a result, the functional space associated to it (7.1) is called "Internal Services" (F_{Pro}) (Figure 126).

Figure 126 – From operational meta-space group to functional space

Figure 127 is the functional diagram of the multimedia library which represents the "Internal Services" functional space and its relationships with the other functional spaces.



Codes	Designation
A.I.1	Entrance/Welcoming
A.I.1.1	Information/Borrowing
A.I.1.2	Storage
A.I.1.3	Sanitation
B.I.1	Activities/Encounters
B.I.1.1	Conference Room
C.I.1	Adult Section
C.I.1.1	Information/Specific Welcoming
C.I.1.2	News
C.I.1.3	Adult Free Access Multimedia
C.I.1.4	Bibliographical Research
C.I.1.5	Audiovisual Documents Store
D.I.1	Children Section
D.I.1.1	Information/Specific Welcoming
D.I.1.2	Children Free Access Multimedia
D.I.1.3	Story Time
D.I.1.4	Children's Workshop
D.I.1.5	Teenage Space
E.I.1	Internal Services
E.I.1.1	Welcoming/Waiting
E.I.1.2	Workstations/Management
E.I.1.3	Handling/Processing
E.I.1.4	Storehouse

Figure 127 – Functional diagram of the multimedia library (reminder from Figure 69)

Figure 128 illustrates the entire functional synthesis process, from operations to functional spaces on the book cycle. The reception of documents introduced above is summed up in the list of operations. The traceability of functional information is ensured through the meta-space artefacts and diagrams.

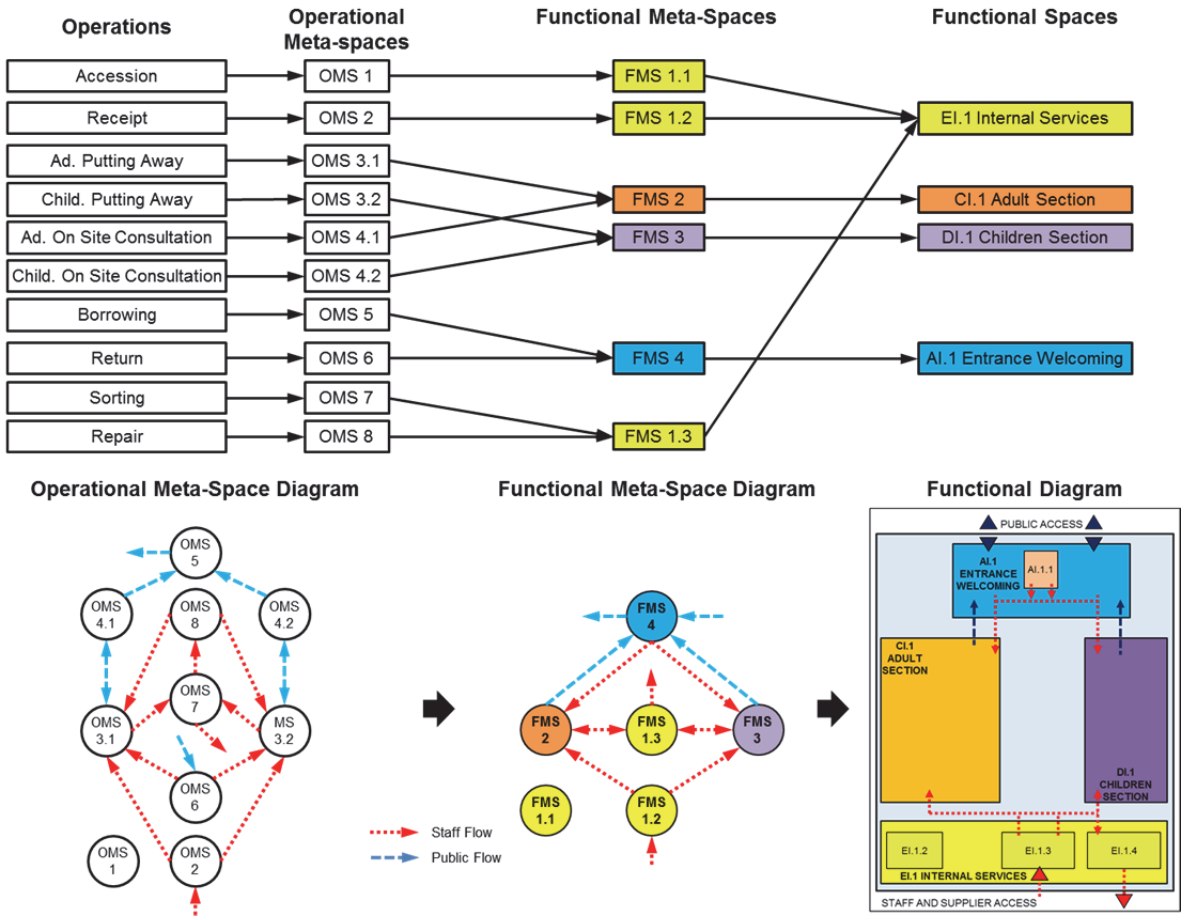


Figure 128 – Functional synthesis restricted to information about the book cycle (already introduced in Chapter III Section 6.3 Figure 94)

2.8. *Physical Synthesis (Step 8)*

The physical synthesis consists in formalising the physical spaces (S_{Pro}) demanded by the clients. This formalisation refers to their listing and quantification.

Objective

The objective of this synthesis is to evaluate the quantity of physical spaces required by the clients to perform all the previously described activities.

Process

Based on the functional spaces, a list of physical spaces (S_{Pro}) is produced that reflects the quantity of spaces required in the future building (8) (Figure 129). The quantity is calculated based on information about the activities (B_{Ser}) and a set of business and logical rules. In the end, there should be enough spaces to perform all the activities inside the building in the best possible conditions. The paying clients' budget moderates the quantity of demanded spaces.

Figure 129 – Physical Synthesis

Support

There is no real support for this step except the space table (table of surfaces). This space table is generally limited to a nomenclature of the demanded spaces.

Deliverable

The space table is annexed to the design brief delivered to the paying clients.

Illustration

Table 17 provides a space table based on the multimedia library. The number of columns depends from one brief to another. Depending on the construction project type, a few relevant characteristics are added such as floor loading, acoustic level, room height, etc.

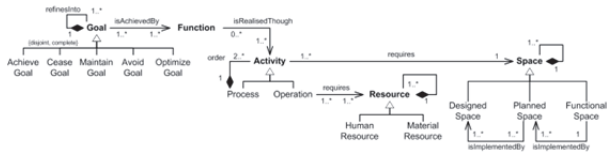
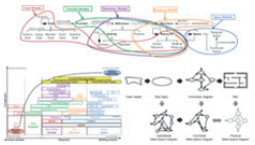
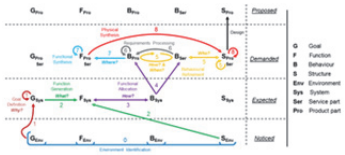
Table 17 – Space Table of the Multimedia Library

Codes	Designation	Unit Surface (m²)	Quantity	Total Surface (m²)
A.I.1	**Entrance/Welcoming**			**50**
A.I.1.1	Information/Borrowing	30	1	
A.I.1.2	Storage	10	1	
A.I.1.3	Sanitation	10	1	
B.I.1	**Activities/Encounters**			**50**
B.I.1.1	Conference Room	50	1	
C.I.1	**Adult Section**			**460**
C.I.1.1	Information/Specific Welcoming	30	1	
C.I.1.2	News	30	1	
C.I.1.3	Adult Free Access Multimedia	350	1	
C.I.1.4	Bibliographical Research	30	1	
C.I.1.5	Audio-visual Documents Store	20	1	
D.I.1	**Children Section**			**320**
D.I.1.1	Information/Specific Welcoming	30	1	
D.I.1.2	Children Free Access Multimedia	150	1	
D.I.1.3	Story Time	40	1	
D.I.1.4	Children Workshop	50	1	
D.I.1.5	Teenagers Space	50	1	
E.I.1	**Internal Services**			**180**
E.I.1.1	Welcoming/Waiting	10	1	
E.I.1.2	Workstations/Management	20	4	
E.I.1.3	Handling/Processing	50	1	
E.I.1.4	Storehouse	40	1	
	TOTAL			**1060 m²**

2.9. In Summary

In this section, a framework for the requirements definition of buildings based on the GFBS design model was proposed (Table 18). The framework was described and illustrated through four complementary case studies. The aim of this framework is to define the building requirements based on the business needs. Based on the building requirements, the architects can propose different building designs. The various modelling languages proposed as a support for each step of the framework provide different viewpoints on the same information in terms of use, understanding, or validation.

Table 18 – Research Output of Chapter IV 1/2

Research Outputs	Research Activities: Build
Constructs *Chapter II*	*Definition Domain* 
Models *Chapter III*	*Conceptual Model*  *Operational Model* 
Method *Chapter IV*	 *Frameworks*

The next section proposes different viewpoints on this GFBS framework through three process model views. These process model views represent instantiations of the GFBS framework in the frame of the design science methodology (March & Smith 1995).

3. GFBS Process Model Views

In Section 2, the GFBS framework was introduced as a theoretical one. In this section, three process model views are presented as instantiation of the GFBS framework. The first process view represents the dynamic links between the steps of the GFBS framework through a BPMN modelling. The second process view represents the information flows (including control loops) along the framework through an IDEF0 modelling. The last process view is the IDM process (Section 1) updated with information generated by the GFBS framework. It illustrates the documentation of the Service through its processes and the resulting additional information exchanges.

3.1. Process Model View

In this section, the GFBS framework is modelled using BPMN to introduce the various stakeholders of the briefing process (i.e. the paying, the user, and the customer clients referred here as the clients, and the programmer) and the dynamic relationships between its steps. The framework is structured around four main stages: external analysis, internal analysis, requirements processing, and requirements synthesis (Figure 130 and Appendix H). As a complement, the main documents introduced in the IDM process view (Section 1) are positioned on the BPMN process view.

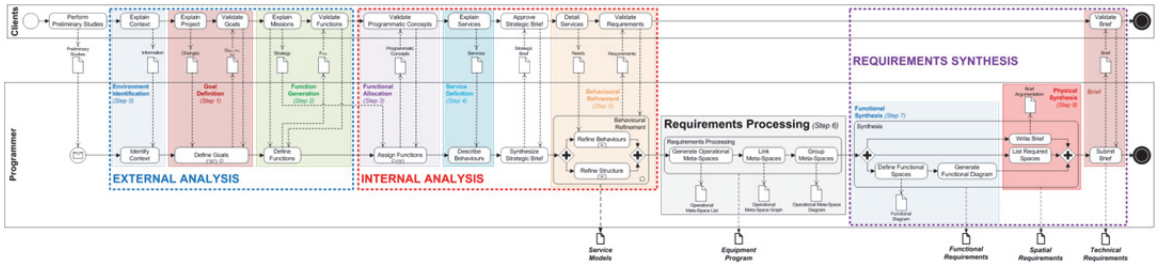


Figure 130 – BPMN process model view of the GFBS briefing

3.1.1. External Analysis (Steps 0 to 2)

An external analysis is performed in the Functional Analysis approach. The principle is to analyse the system to define starting from its future environment. The environment is composed of already existing artefacts that will interact directly (or indirectly) with the system afterwards. The system is considered as a black box, i.e. the focus is not on what composes the system but rather on the interface of the system with its future environment. As a result during this phase, only solution-neutral requirements should be produced. These requirements are represented by the concepts of Goal and Function (Figure 131).

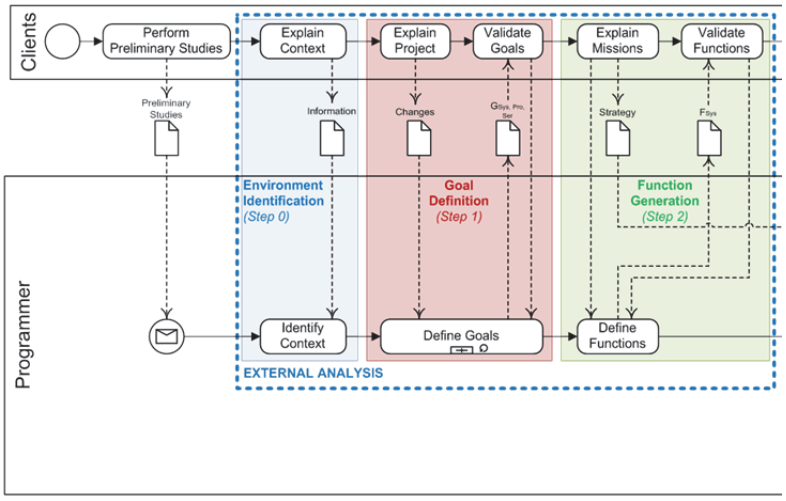


Figure 131 – Local process model view on External Analysis (Steps 0 to 2)

The first step in the GFBS framework is the **Environment Identification**. The clients explain the context to the programmer through interviews, documents, archives… any relevant support. The programmer has to clearly identify the context of the project, the stakeholders, their power struggles and hard lines on the project, the neighbourhood and their feelings about the project, the project's and stakeholders' history… The result is a set of statements and facts about the System's Environment (noted G_{Env}, F_{Env}, B_{Env}, S_{Env}). In Requirements Engineering, such information refers to the Domain Properties (Lamsweerde 2009), the Domain Assumptions (Jureta et al. 2008), or the Domain Knowledge (Zave & Jackson 1997).

The second step concerns the **Goals Definition**. Based on the explanation of the project by the clients, the programmer defines and refines the various goals to achieve through the system (noted G_{Sys}). The goals reflect changes which occur in the environment (i.e. the TO-BE situation). Their hierarchy and dependencies are clarified by the programmer with the stakeholders of the project. A validation of these goals is essential to ensure that the project aims in the right direction but also serves as a basis to support later decision-making processes. It is acknowledge that some changes can occur later in the briefing process due to refinement of the requirements.

The last step of the External Analysis is about the **Function Generation**. The concept of Function is an abstract concept that describes in a simple and straight way how to achieve each defined goal. Goals are specific to the context of the system, its environment, whereas functions are independent from this context. The Functions of the system (noted F_{Sys}) define in a solution-neutral way how goals are supposed to be achieved. The focus is not on the concrete realisation or description of the functions; this description is part of the Internal Analysis. However, each function has to be characterised in terms of performances regarding a set of criteria. These criteria are used to refine and to assess the goals' satisfaction. The next phases of the briefing process contribute to the refinement of these criteria.

3.1.2. *Internal Analysis (Steps 3 to 5)*

The external analysis supports the definition of the interface between the system to define and its environment. The output of the external analysis is a set of functions to realise through the system. The internal analysis focuses on the components of the system (Figure 132). In this research, the system is a building considered as a Product-Service System. It means that each function can be realised through the building itself or any other physical artefact that compose it, but also through services, i.e. agent-intensive activities. The programmer is in charge of organising the ideas and needs of the clients. Through this phase, he has to support the decision making of the clients by using the previously defined goals and functions and to document their choices.

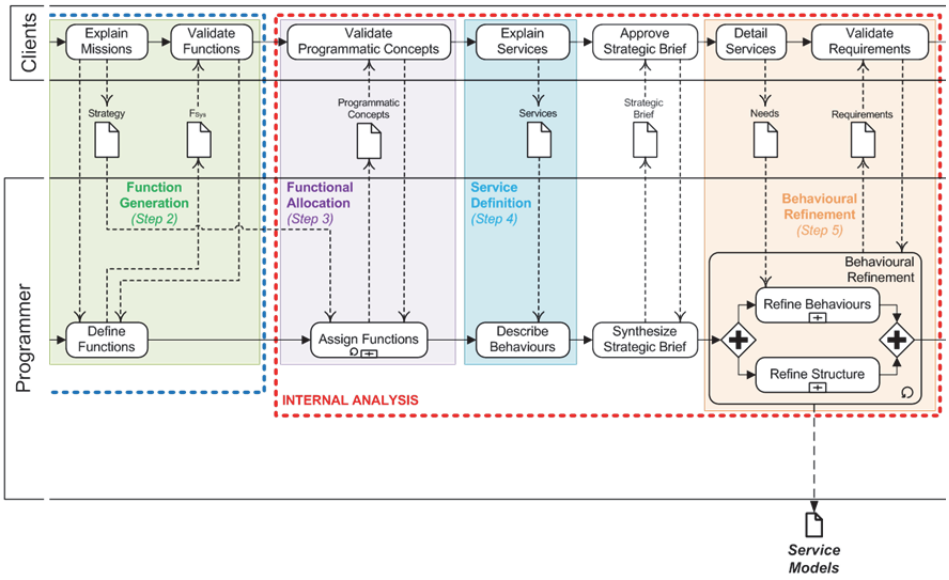


Figure 132 – Local process model view on Internal Analysis (Steps 3 to 5)

Therefore, the first step of the internal analysis is to **assign functions** to the product part, the service part, or to a combination of both. This results in the definition of programmatic concepts, i.e. high-level solution principles. Each programmatic concept is discussed and validated by the clients. The clients' strategy influences the assignment of functions. The strategy covers the clients' philosophy (e.g. motto or leitmotiv) and preferences about the project but also their business. An important issue put at stake during this step but not considered in this research concerns the global cost of the programmatic concepts.

Once the functions are assigned to a set of programmatic concepts, the clients explain to the programmer how things will be performed (i.e. the services). The services are described through processes and activities or properties (B_{Sys}) assigned to (product) components of the building system (S_{Sys}). At this stage, the description of the behaviours and structure remains of quite raw. Based on these first elements, the programmer is able to formalise the strategic brief to be approved by the clients. This strategic brief provides global indications about the building systems characteristics (e.g. spatial requirements, global costs…).

The last step consists in **refining the behaviours** previously sketched. Based on details about the services, processes and activities are refined into operations (B_{Ser}) which require sets of resources (S_{Pro}) (e.g. personnel, equipment, energy, or space) and an organisation (S_{Ser}) (e.g. positions, roles). Each physical component has characteristics (S_{Pro}) and properties (B_{Pro}) that can contribute to the implementation of certain functions. During this step, the building system is considered as an operating system. Each component has its own duty but the global organisation of the components has also to be managed (alignment principle). The main output of this consequent step is a set of requirements about the building system to be validated by the clients. At this step, the spatial requirements are considered from a local viewpoint, i.e. each operation

and equipment required certain space independent from each other which results in a huge amount of required square meters. The processing of these requirements is dealt with during the next phase. Regarding the IDM process, an additional deliverable is produced: the service models.

3.1.3. Requirements Processing (Step 6)

The internal analysis provides a huge quantity of raw requirements for the building system. The **requirements processing** aims to reduce the amount of required spaces by merging them based on the similarity or compatibility of their requirements (Figure 133). In order to manipulate the requirements, a design artefact is introduced: the meta-space. A meta-space regroups all the requirements about an operation to perform. The processing is therefore done on the meta-spaces rather than directly on operations or spaces. In current practices, the lists of activities and spaces are generated with few interactions, and then activities are allocated to standard spaces. In the GFBS framework, the idea is to generate the spaces based on the activities to perform and their requirements.

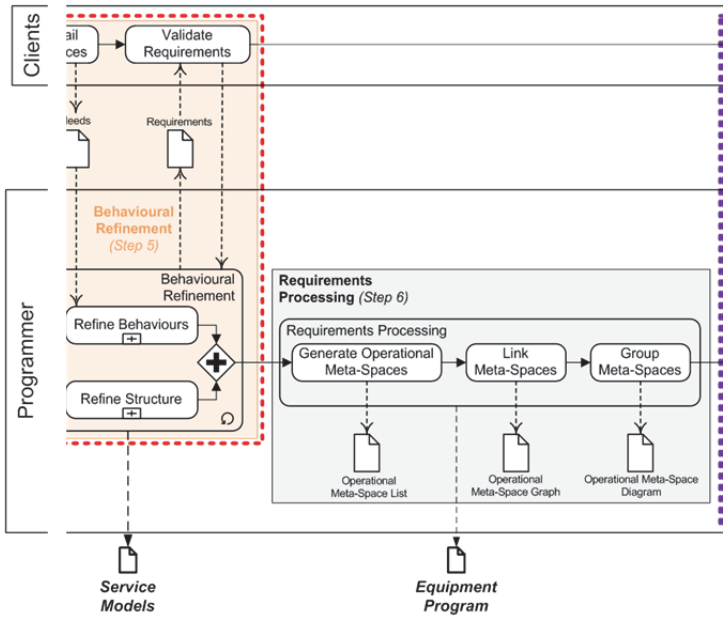


Figure 133 – Local process model view on the Requirements Processing (Step 6)

The first step of the requirements processing is about defining the operational meta-spaces (i.e. **Generate Operational Meta-Spaces**). For each operation, a meta-space is defined. This meta-space gathers all the requirements about the related operation.

The second step consists in **linking the meta-spaces** according to the dependencies and relationships between operation, from the services and processes (B_{Sys}). This results in a set of graphs (i.e. operational meta-space diagrams).

Then, the last step consists in **grouping the meta-spaces** according to their relationships and attributes (i.e. requirements). The grouping integrates reasoning about inter alia successive operations that can be perform in the same space to reduce flows of resources, compatibles ambiance, asynchronous operations, or similar resources to mutualise them. The Equipment Program is produced during this phase. It synthesises the quantity of resources required to perform the operations based on the grouping. This phase should be supported by automation in order to take full benefit of it. Automation provides a fast processing of the requirements and the original grouping that the programmer would not have thought about at first.

3.1.4. Requirements Synthesis (Steps 7 & 8)

The requirements synthesis consists in formalising the results of the requirements processing using diagrams and textual descriptions (Figure 134). The textual description is the main

deliverable (i.e. the brief). It develops the requirements about the building system and more particularly about the building part of the system. The diagrams and requirements about the other parts of the building system (e.g. service part and equipment) document the brief. They provide complementary information and views about the building part.

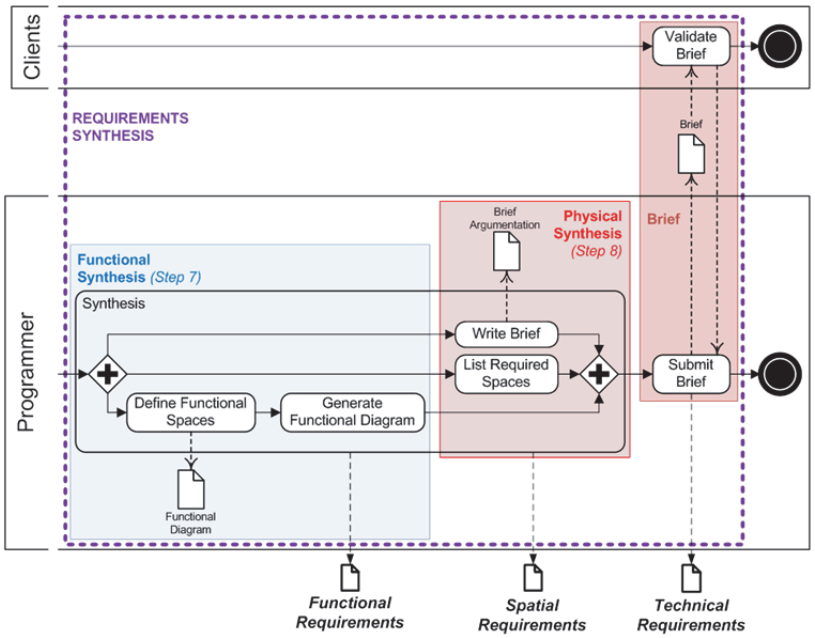


Figure 134 – Local process model view on the Requirements Synthesis (Steps 7 & 8)

The design brief is a contractual document. It has to be validated by the clients before its diffusion to the designers. Additional deliverables regarding the IDM process produced during the requirements synthesis are the spatial requirements (i.e. table of spaces), the functional requirements (i.e. functional diagram), and the technical requirements (i.e. servicing equipment of the building). All of these deliverables supply the numerical model of the building from an architectural programming viewpoint.

3.1.5. Synthesis

In this section, we propose a BPMN process model view of the GFBS framework. This first instantiation clusters the framework into four stages and introduces the dynamic relationships between the activities performed by the clients and the programmer and their necessary interactions. Indeed, the briefing process requires full support of the clients to be effective.

3.2. Information Flow Process Model View

In this section, the GFBS briefing framework is modelled using IDEF0. This viewpoint allows illustrating the various flows of information and the methods, tools, and techniques that can support each step of the framework. Compared to the BPMN representation, the IDEF0 model positions the main constructs following the framework and represents back and forth loops (as in Axiomatic Design) not represented in the framework.

Figure 135 represents a global view of the framework following the four main stages used to structure the framework in the BPMN process model view. The correspondence with the GFBS framework is represented by the number below the activity boxes. The detailed version of this figure can be found on an A3 sheet in Appendix I.

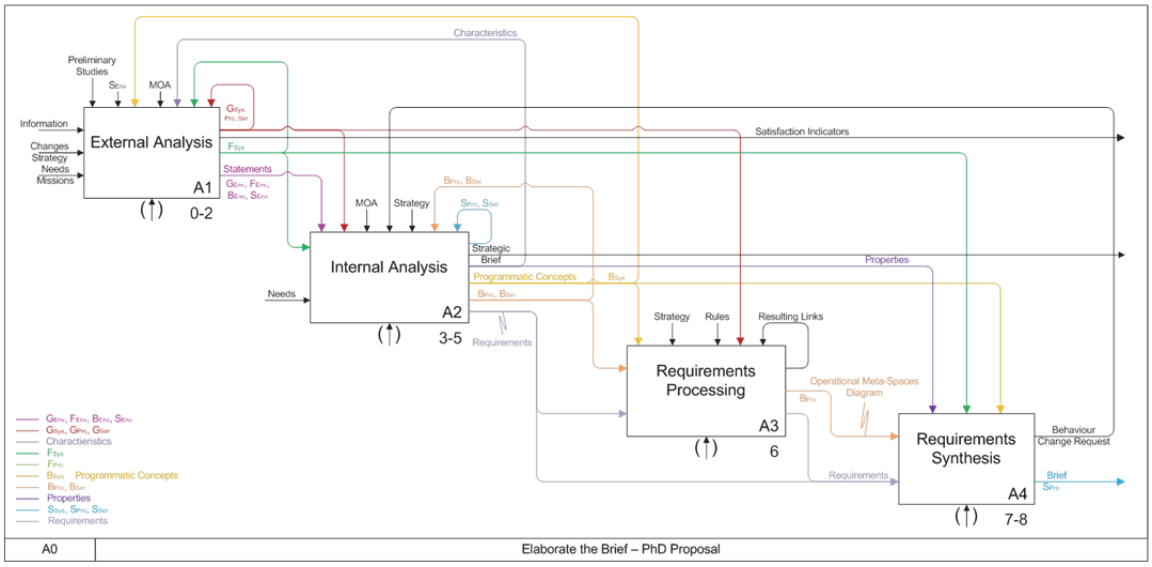


Figure 135 – IDEF0 Process Model View of the GFBS Framework

3.2.1. External Analysis

The external analysis produces a lot of information that is used as a control during the later stages of the briefing process (Figure 136). It is based on non-formalised information provided by the clients (i.e. needs, information, change strategy, or missions). The first information is about the environment (G_{Env}, F_{Env}, B_{Env}, and S_{Env}) which details the context of the project (A11_0). Based on this information, the programmer can support the clients in the formulation of their needs and the formalisation of their requirements. The second information focuses on goals. Goals guide the programmer and the clients during the next stages in terms of decision making (A12_1.1). All decisions should be made toward an achievement of goals. During the refinement process, the main goals can be modified or updated (A13_1.2). Finally, the last information produced during the external analysis is the functions of the building system. Based on the needs and missions proposed by the clients, and following the refined goals, the programmer abstracts what the building system should be able to do in a solution neutral way (A14_2). Each step of the external analysis requires a validation by the clients (noted MOA). The output of the external analysis is the context information, the refined goals, and the system's functions (F_{Sys}).

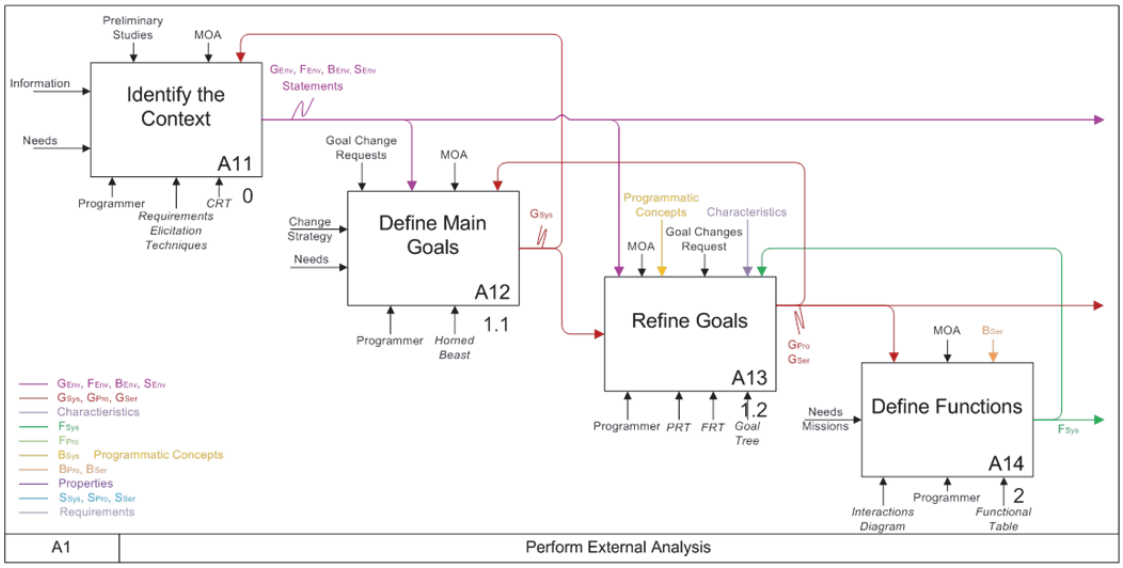


Figure 136 – External Analysis Information View

3.2.2. Internal Analysis

The main input of the internal analysis is based on the system's functions (F_{Sys}) (Figure 137). The principle is to find a way to ensure these functions (A21_3) through a combination of service and product parts of the building system. This method is called programmatic principle which describes the system's behaviours (B_{Sys}). The internal analysis is a refining process that starts with the system's function and results in service properties and structure characteristics (i.e. requirements). The macro level (global description of the programmatic principles) is sufficient to produce the strategic brief (A22_4). The lower levels (A23 and A24_5.1 and 5.2) are required to detail the operating system and to produce satisfaction indicators for the assessment of the design proposals. Each layer of refinement provides information that can challenge the upper layers in terms of consistency and comprehensiveness (control loops). Information from the external analysis is used as a control for each step.

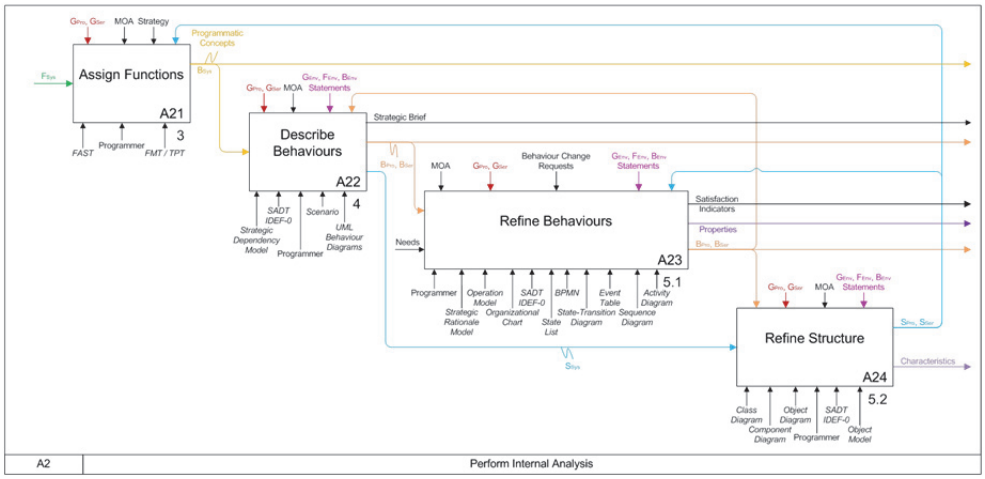


Figure 137 – Internal Analysis Information View

3.2.3. Requirements Processing

The refined behaviours of the product (B_{Pro}) and service (B_{Ser}) parts of the building system constitute the input of the requirements processing (Figure 138). Based on the operations, the operational meta-spaces are generated (A31_6.1). The requirements represent the full description of the operation meta-space (i.e. operation, required resources and linked processes, functions it contributes to, and related goals). These requirements are used to link operational meta-spaces (A32_6.2) and are processed along with them (A33_6.3). The requirements processing is a stage performed by the programmer alone, without control by the clients. It is part of his expertise. However, this expertise is mainly experience-based in the current situation. In this framework, this expertise should be supported by automation to handle the huge quantity of information about the building system. That way, all the decisions made (grouping) can be traced and changed later.

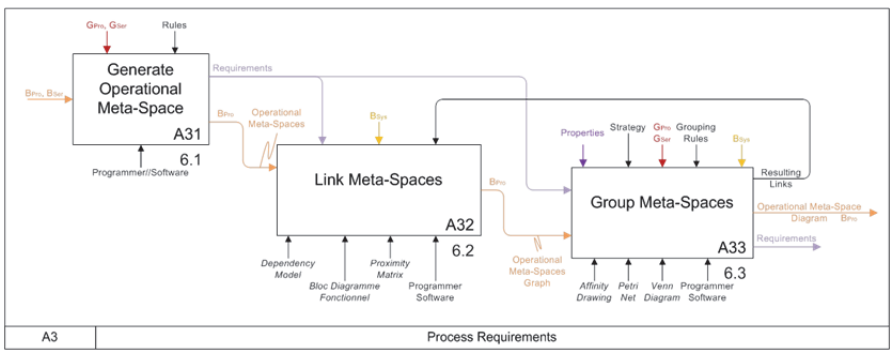


Figure 138 – Requirements Processing Information View

3.2.4. Requirements Synthesis○

The inputs of the requirements synthesis are the operational meta-space diagram (B_{Pro}) and all the requirements generated by the previous steps (Figure 139). The first step consists in generating a functional meta-space diagram based on the operational meta-space diagram and the system's functions (F_{Sys}) (A41_7.1). Based on this functional meta-space diagram (F_{Pro}), the programmer can generate a functional diagram (A42_7.2). The functional diagram (F_{Pro}) formalises one possible instantiation of a functional meta-space diagram amongst others. The functional diagram provides a global synthesis of building requirements to the clients. Based on the building requirements, the programmer lists and quantifies the required spaces (A43_8.1). The functional diagram is used as a control for this step. The last step is the brief writing (A44_8.2) which gathers all the building requirements, the functional diagram, and the space table.

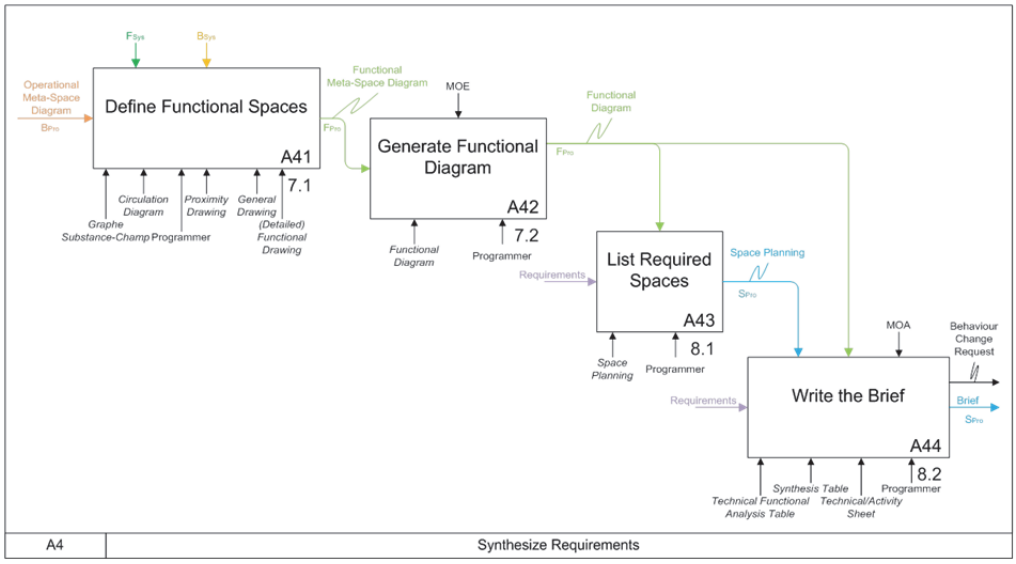


Figure 139 – Requirements Synthesis Information View

3.2.5. Synthesis

In this section, we propose a process model view to represent the flow of information during a potential instantiation of the GFBS briefing framework. This IDEF0 model also supports the positioning of the various tools and modelling techniques that can be used for each step of the framework.

3.3. Collaborative Model View

In the introduction, an IDM of the briefing process was presented as a basis, a starting point, from the literature. In the original proposition, all of the building requirements (i.e. product requirements) are based on the activities performed by the stakeholders. These stakeholders represent the user and customer clients of the building. The concept of Activity is therefore at the centre of the information generation during the briefing process (Koutamanis 2013). However, the concept of Activity is poorly developed in the literature.

In the GFBS framework, the concept of Activity is enriched through the concepts of Behaviour, Service, and Process on a global perspective and through the concept of Operation on a local perspective. Service and Process provide the dynamic relationships and dependencies between activities whereas Operation supports details about the resource requirements to perform the activities. This enrichment is represented in the IDM map by a new activity: "*Model the processes*" and its deliverable: "*the service models*" (Figure 140).

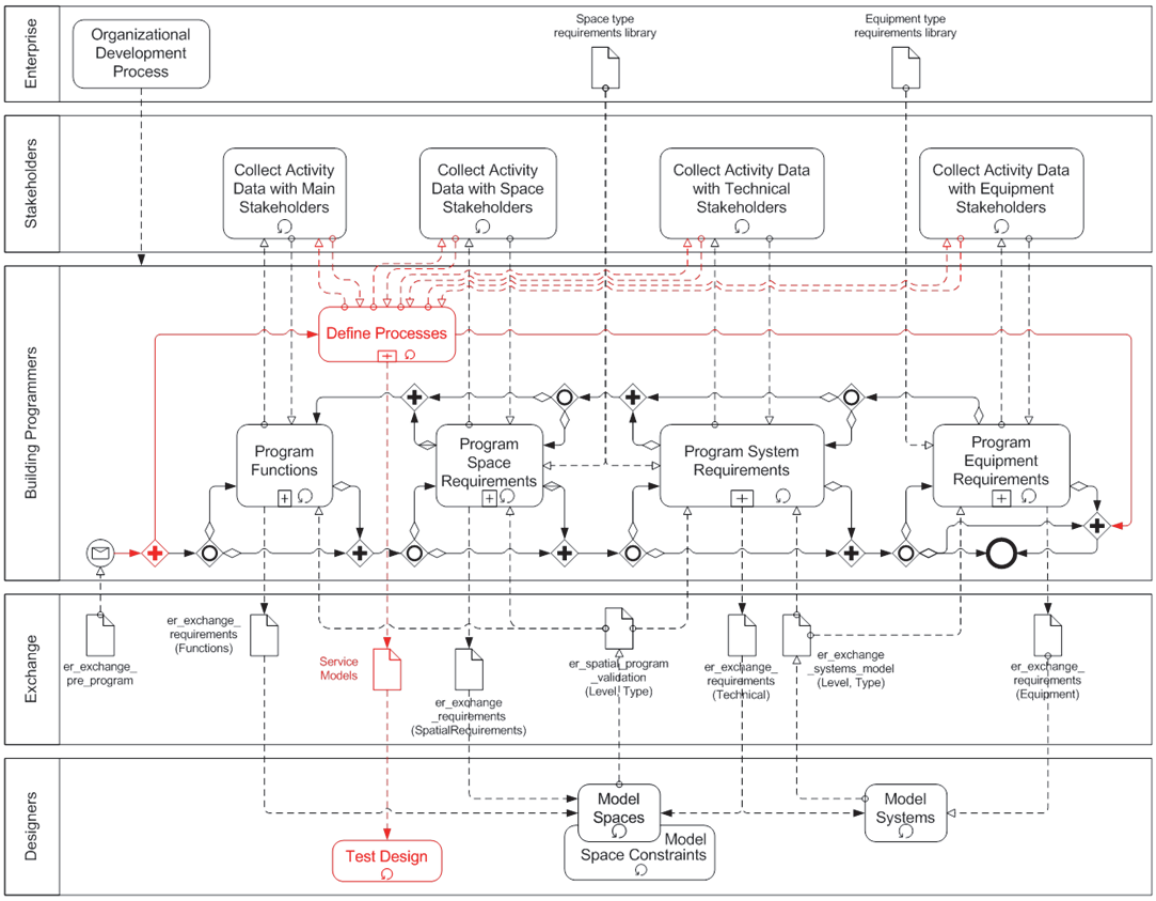


Figure 140 – IDM representation of the briefing process integrating GFBS framework

This activity aims to document and capitalise information about the building system. It is a formalisation step of the interactions and information exchange with the stakeholders. Information can be stored in an information system and then could be used later to assess the design proposal through the meta-space artefacts. Additional information can be stored in the numerical model of the building using the existing structure of Information Foundation Classes (IFC). Figure 141 illustrates an enrichment of the numerical model using taxonomy of services based on the IDM initial mapping.

The integration of the Service concept through the GFBS framework is an overlay of information formalised and provided to the architect. This formalisation is an additional task to perform based on information already generated during the briefing process. As a result, the required efforts remain limited. In current practices, this information is only stored in the programmer's memory. Its restitution is highly implicit in the various functional diagrams (Mauger & Kubicki 2013), affinity diagrams (Alread & Leslie 2006), zone diagrams (White 1986) and other proximity matrices (Fortin 1978) that complete the brief. This kind of information is a key element of the global understanding of the brief for the designers as well as to ensure the alignment between the design and the clients' demand (Allégret & Guilleux 2000). Its formalisation takes time but allows minimising a later waste of time and resources.

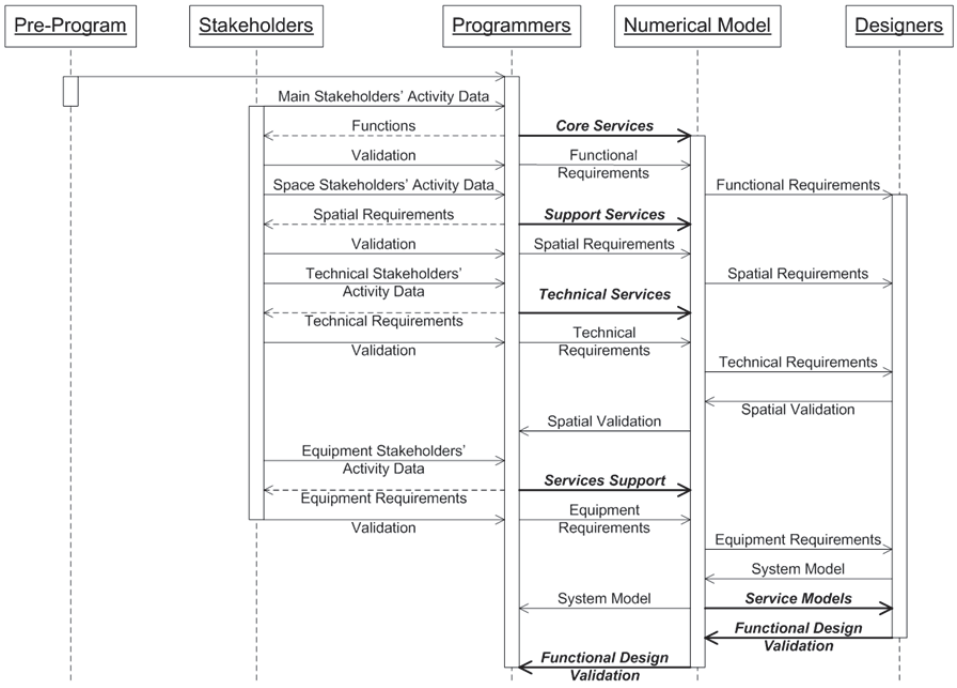


Figure 141 – Sequence Diagram representing the Services integration in the briefing process

3.4. In Summary

Through this section, we introduced three process model views of the GFBS framework. Each instantiation represented a complementary viewpoint on the framework. The BPMN process model view illustrated the succession of steps performed by the programmer and its main interactions with the clients. The IDEF0 process model view illustrated the information flows and control loops, and positioned the various modelling techniques on the GFBS briefing framework. Finally, the IDM process model view positioned the main contribution (i.e. service integration to architectural programming) in the context of BIM processes. These instantiations provide elements of verification of the proposed framework but they are not sufficient to validate it.

In the next section, we propose another way to verify the framework through the assessment of a building proposal. This verification consists in applying the same framework from the design proposal to the building requirements. The proposal assessment is considered as part of the briefing process.

4. GFBS Assessment Framework

The assessment of the building use is part of the selection process during the architectural design competition. Based on the brief information, the Technical Committee has to check whether the architects' proposals satisfy the future uses and to highlight potential issues (Bisbrouck 2006). In France, this Technical Committee is not legally mandatory but highly supported due to its proven necessity (JORF 1993; MIQCP 2012).

Despite its importance, the functional assessment regarding the use of buildings is not prevalent in the selection process. The functional assessment performed by the Technical Committee cannot anticipate the jury's conclusion of the selection process (MIQCP 2012). It remains an objective, factual assessment contrasting with the selection process. As a result, quite often, the key criterion of selection remains an aesthetic or poetic aspect of proposals despite the "user" or the Technical Committee's opinion (Bisbrouck 2006). The composition of the jury seems to partly explain this by the prevalence of architects (at least a third of the jury in France (JORF 1993)) compared to the user representatives (usually one or two persons).

In this section, we propose to use the GFBS framework to support the functional assessment of design proposals (Figure 142). This framework introduces a new state regarding the building information: the *analysed state*. The *analysed state* corresponds to the information created (interpreted) by the Technical Committee during the functional assessment. As a reminder, the *proposed state* is related to the architects and designers whereas the *demanded state* is related to the clients and programmer. These states represent three viewpoints on the building system.

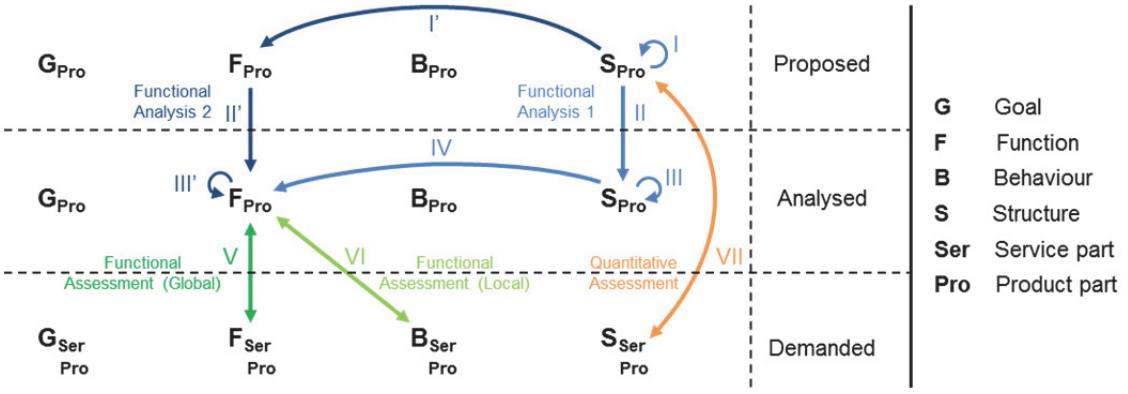


Figure 142 – GFBS Assessment Framework

The assessment framework is structured in three main stages: functional analysis, functional assessment, and quantitative assessment (Figure 143). Two variants of functional analysis and functional assessment are presented to reflect the interest of the design artefacts proposed in this research.

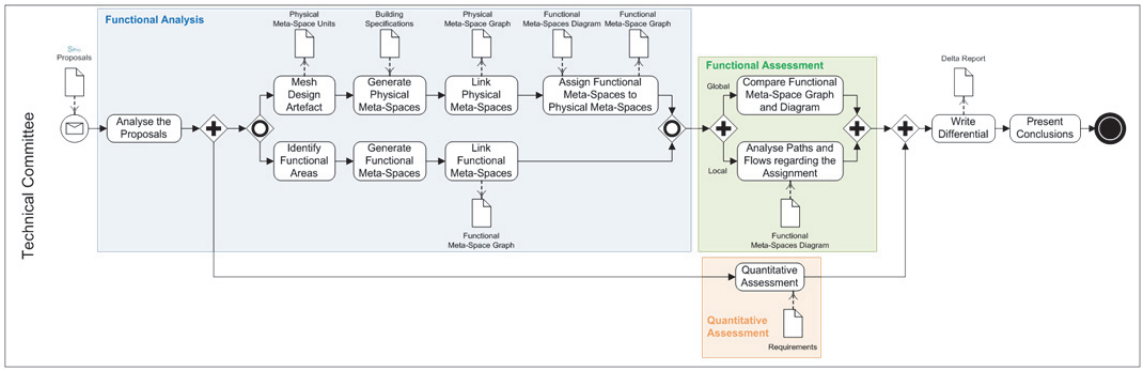


Figure 143 – BPMN model of the GFBS Assessment Framework

The GFBS assessment framework is illustrated based on the available data and scales of each case study used in this research (Figure 144).

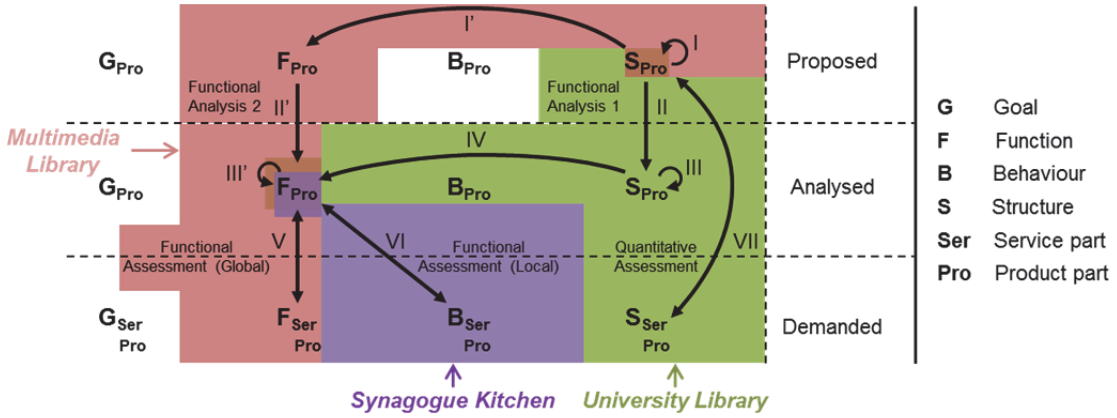


Figure 144 – Case Studies Coverage of the GFBS Assessment Framework

4.1. Functional Analysis

The first step consists in analysing the proposals (Figure 145). Depending on how far the designers went in describing their proposals according to the design brief, two analyses can be performed. The first possibility is that the designers did not match their proposal (S_{Pro}) with the functional spaces (F_{Pro}) defined in the brief. Therefore, the Technical Committee has to make an own analysis of the proposal to determine the physical position of the functional spaces. The second possibility is that the designers took time to explicit the functions of each space (F_{Sys}) in their proposal (S_{Pro}). Therefore, the Technical Committee then can directly create the functional meta-space graph from the proposal. Another possibility could be that the architects and designers described their proposal with more details such as how activities (B_{Pro} on the *proposed state*) should be performed in the building (like a user guide). However, none of the case studies present such a level of details.

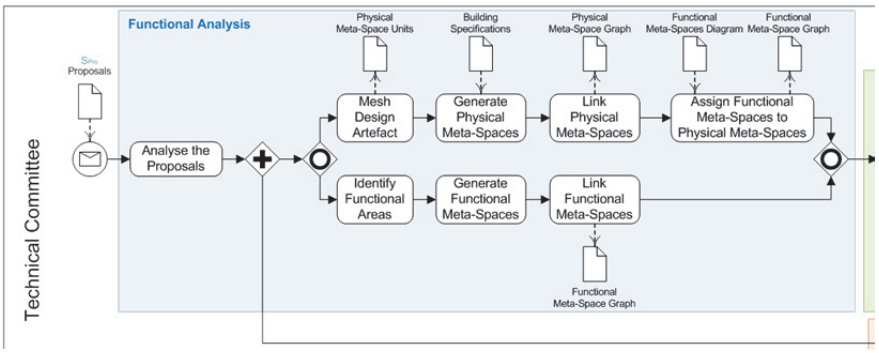


Figure 145 – Process Model View of the Functional Analysis

4.1.1. Functional Analysis 1 (Steps I to IV)

This section presents the first possibility of functional analysis illustrated on the university library case study. The functional analysis 1 is divided into four main steps (Figure 146):

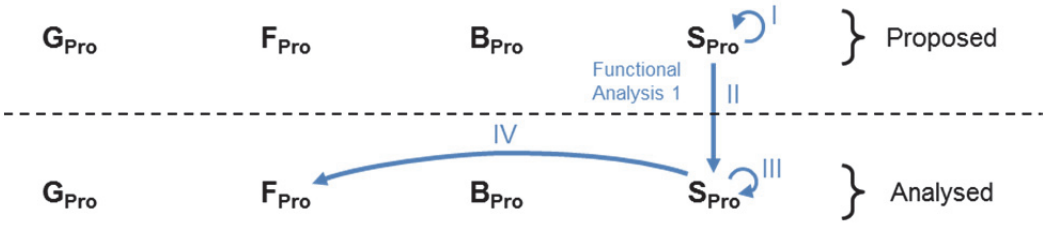


Figure 146 – Functional Analysis 1

The Technical Committee starts from "neutral" 2D drawings. To start with the meshing of the drawings is performed manually but may be computer-aided later (step I). The meshing consists in dividing the plan into small surfaces having a semblance of functional unicity. Manual meshing remains rough but can provide first interesting results for certain analyses. In the case of the University Library, such analysis was performed in order to initiate a flow analysis of the building based on its model (Figure 147).

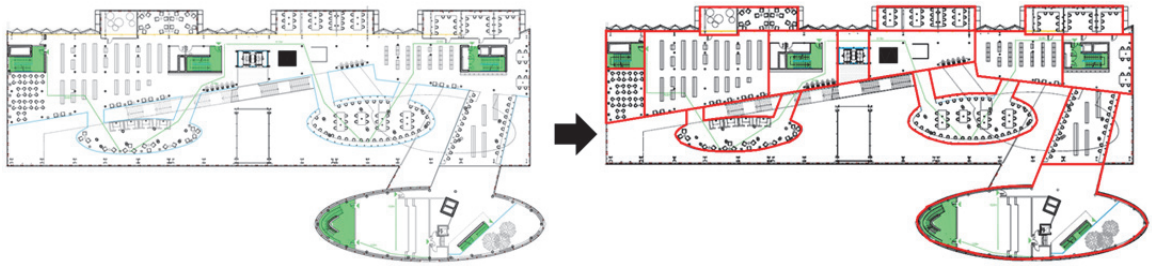


Figure 147 – Rough meshing of the University Library's 1st floor (I)

The second step consists in generating a physical meta-space for each surface delimited on the drawing (step II). Then, based on the possible direct accesses from one surface to another one, the corresponding physical meta-spaces are linked to each other (Figure 148).

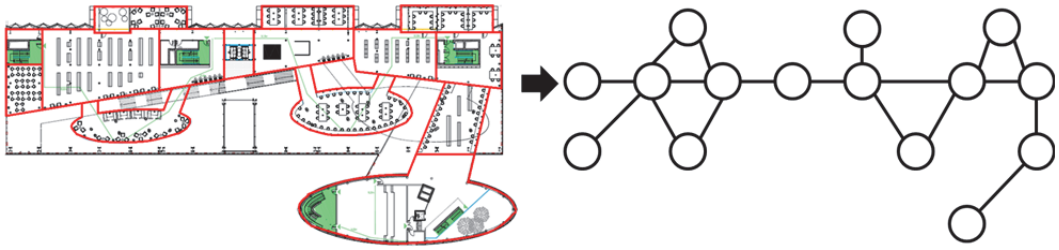


Figure 148 – Generation of a Physical Meta-Space Graph based on the meshed drawing (II)

Finally, based on the content of each area (step III), a function is assigned (step IV) to each physical meta-space in order to create a functional meta-space graph of the drawing (Figure 149).

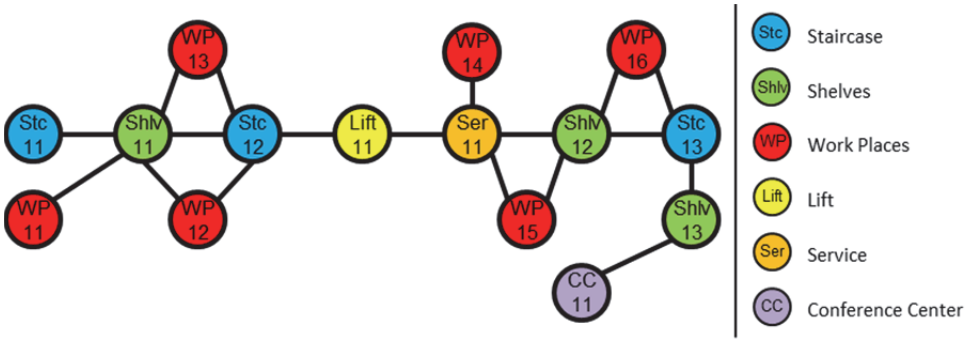


Figure 149 – Functional Meta-Space Graph of the 1st floor of the University library (III & IV)

The same principle was applied on the whole building with a focus on the shelves (i.e. not considering the work places and conference centre). The aim was to model the spaces related to the book cycle without considering a site consultation. Figure 150 illustrates how the whole building was partially modelled in a functional meta-space graph.

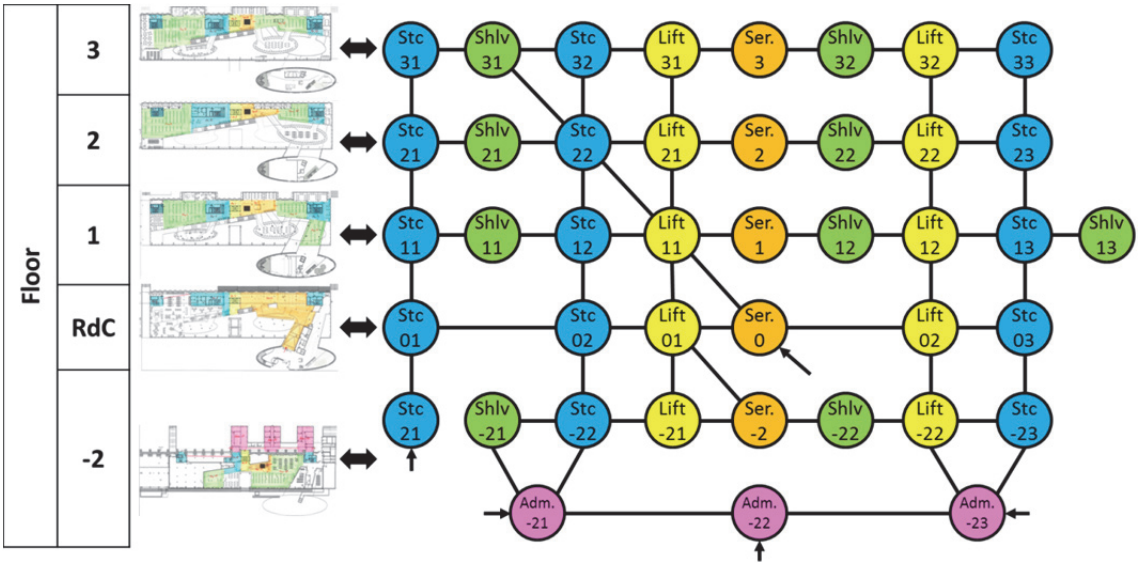


Figure 150 – Modelling of the shelf spaces of the University library

The meshing principle can also be performed on a smaller scale using the smallest unit of surface to mesh a space. For example, we took a storage surface and divided it into elementary surfaces in order to make a distinction between each space occupied by a shelf and circulation

spaces (Figure 151). The resulting physical meta-space graph can be used to assign book collections to each shelf using various criteria such as the minimum distance to cross in order to cover a full collection. In this case, the support of automation is essential to make this worthwhile, otherwise it is a too time consuming activity.

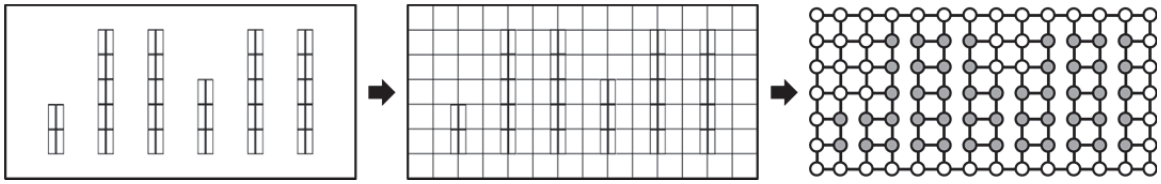


Figure 151 – Meshing and modelling of a storage space, each bubble represents 1 m²; white bubbles represent circulation spaces whereas grey bubbles represent storage spaces

Manual meshing has several limitations as well as the graph representation for human being:

- there can be as many possible meshing as people doing it,
- the meta-space diagram of a whole building is too complex to be represented which requires local modelling for specific purposes,
- and the number of connections between spaces can generate too many edges; some rules have to be defined to frame the meshing and modelling.

4.1.2. Functional Analysis 2 (Steps I' to III')

The second possibility of functional analysis is structured in three steps (Figure 152).

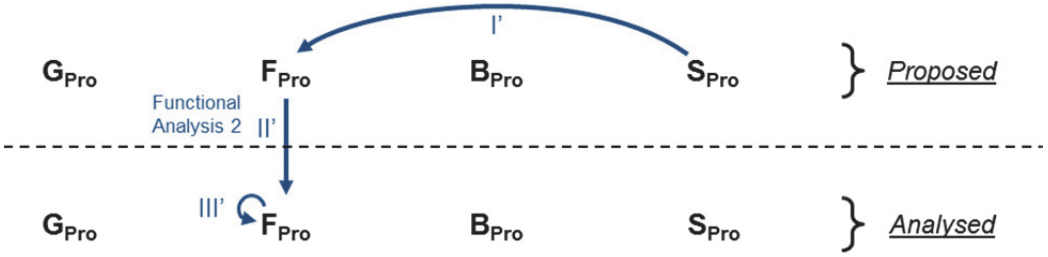


Figure 152 – Functional Analysis 2

In this situation, the designers have already indicated the function assigned to each space on their proposal (Figure 153). This was the case for the architectural design competition of the multimedia library. Therefore, the Technical Committee only has to identify the functional areas (step I'), to model the functional areas by functional meta-spaces (step II'), and to link the functional meta-spaces according to the accesses between functional areas (step III').

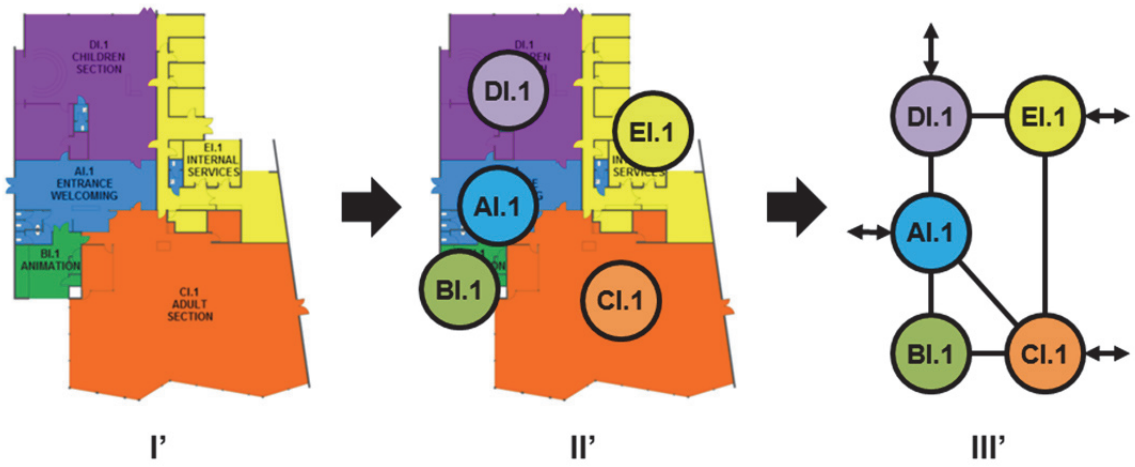


Figure 153 – Functional Analysis of the project A from the multimedia library

4.2. Functional Assessment

The functional assessment consists in an objective evaluation of the designers' proposals by the Technical Committee. The assessment is based on the design brief information and its various diagrams (e.g. functional diagram). The diagrams are supposed to synthesise all the functional spaces and relationships. Functional requirements are considered as must-have requirements. This assessment is a qualitative evaluation of the proposal. A quantitative assessment has also to be performed in parallel (Section 4.3).

The functional assessment can be performed on two different scales: a global or a local one (Figure 154). A global functional assessment consists in verifying the adequacy of the functional meta-space graph based on the 2D drawing analysis and the functional meta-space diagram based on the brief. A local functional assessment consists in checking all the processes and scenarios contained in the brief on the functional meta-space graph of the proposals.

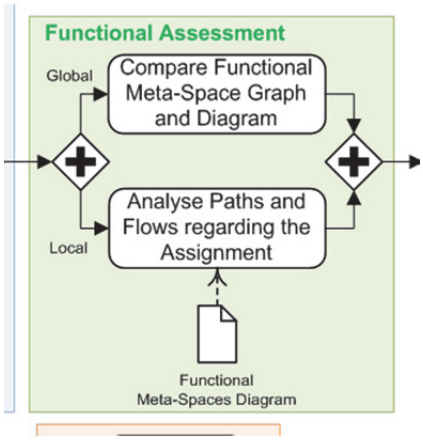


Figure 154 – Two Scales of Functional Assessment

4.2.1. Function Assessment (Global) (Step V)

A functional diagram and an architect's drawing are two very different graphical models, both using a distinct syntax. Their direct comparison is impossible. Moreover, their layout can be totally different. The evaluation consists therefore in comparing (step V) the functional meta-space diagram from the brief (F_{Pro} demanded) with the functional meta-space graph of the design proposal (F_{Pro} analysed), and thus to identify the differences (Figure 155). Further verifications can be performed based on scenarios and use cases to simulate the flows of people and resources (Section 4.2.2).

Figure 155 – Global Functional Assessment

The functional meta-space graph is based on the function assignment performed by the designers on their own proposal. Therefore, the Technical Committee has to compare this functional meta-space graph with the functional meta-space diagram that comes from the functional diagram synthesised by the programmer. In order to facilitate the comparison, the functional meta-space graph might require to be reorganised in terms of layout (Figure 156).

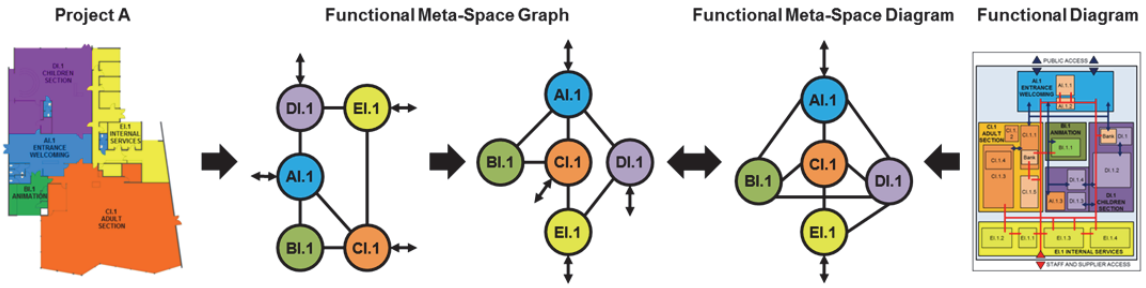


Figure 156 – Comparison between the Functional Meta-Space Graph and the Functional Meta-Space Diagram

In the illustration case study, it can be noticed that there are two missing edges in the project A proposal: a direct access from the Animation area (BI.1 in green) and the Children section (DI.1 in purple) and a direct access from the Adult section to the Children section. The additional arrows from the Adult and Children section with the outside represent emergency exits.

The functional diagram meta-space diagram has to be supported by a data-base containing the operational requirements related to each meta-space and link. In fact, missing links or meta-spaces lead to potential issues in the use of the building. These issues can be properly identified upstream, and then specific modifications can be suggested to architects. This would contribute to ensuring a functional quality of the detailed design after the winner selection.

The multimedia library case study provided enough information to make a comparison of the three design proposals based on the functional diagram. As a postulate, we consider that the functional diagram models fundamental functional requirements. Therefore, all the functional meta-spaces and their relationships should be found in the physical meta-space diagrams that modelled each design proposal. At first sight, the three proposals have different layouts that are not easy to analyse for non-architect people. Using the physical meta-space diagrams, the bubbles can be rearranged to correspond to the layout of the functional meta-space diagram.

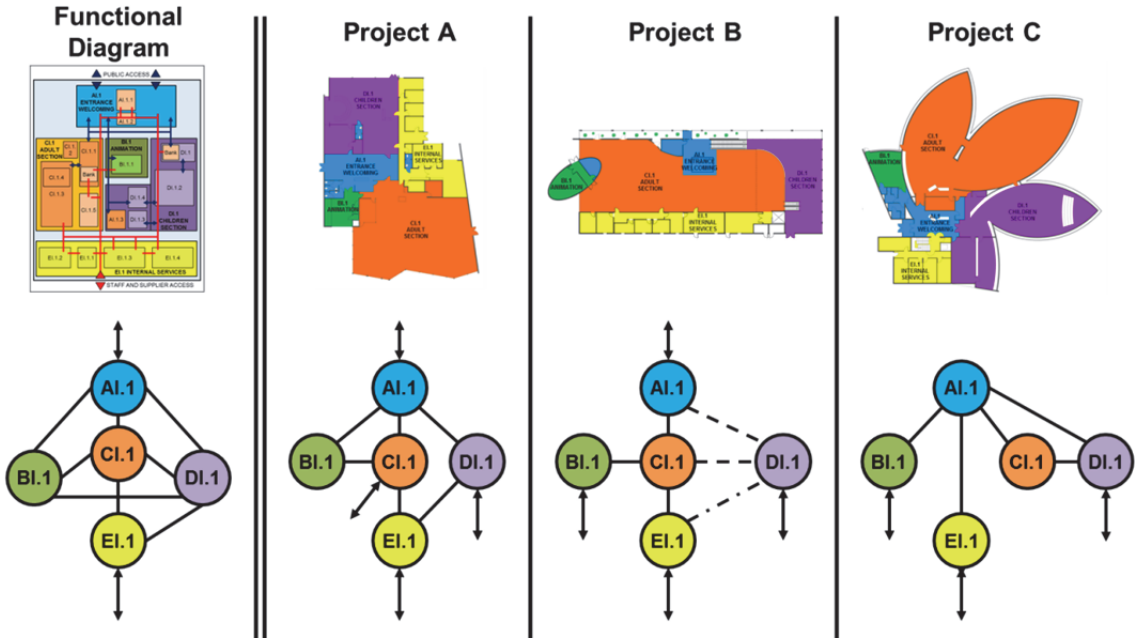


Figure 157 – Comparison of architectural projects based on the same functional diagram

Figure 157 represents the functional diagram and the three architects' proposal using functional meta-space diagram/graphs. The four design artefacts can therefore be compared. This assessment was done to prepare the interview with the library director. By doing it this way, a

validation of this assessment regarding the selected, built and used project was possible. The result of this high level assessment is resumed in four points:

- Overall: missing link between Animation and Children Section, required to cross either the Adult Section or the Entrance, addition of emergency exits
- Project A: closely matches the functional diagram, with the exception of the missing direct access from Adult Section to Children Section
- Project B: missing link between Entrance and Animation, two layers, basement dedicated to the Children Section, an access to the Animation from the outside is claimed
- Project C: Entrance as single distribution space, centralized design, two layers, basement dedicated to Internal Services

Based on the postulate that each link and functional space in the functional diagram is a necessary condition, the assessment does not need to focus on what matches, but only on differences. Then deltas are analysed with regard to the functional diagram information or information given by the architects. For example, for project B and C, the book cycle is complicated by the elevation in project B (dotted lines) and the mandatory passage through the entrance/welcoming in project C. The book cycle requires functional services to be easily accessible from both adult and children section. The elevation in project B regarding the children section will lead to difficulties in terms of human resources, as children cannot be on their own. It requires having one extra person to deal with it.

The Technical Committee Report does not provide so many details about the functional assessment of proposals. All of them were evaluated as satisfying. Project A was perfectly fine, without any negative comments. The elevation in Project B raised open questions in terms of use, noise and elevator, but without so many details. The pronounced separation of the Internal Services in Project C also raised open questions about its use and the surveillance of the other sections. Even if this functional assessment is not predominant in the selection process, it will be necessary to deepen the functional analysis later, in order to ensure that all the services and uses can be properly performed and that acceptable solutions or changes will be proposed. At the end of the selection process, Project B was chosen, regardless of the functional issue raised by the Technical Committee.

After the analysis of the technical report, an unstructured interview with the library director was held to confirm or deny observations made on Project B. The functional issue stated by both the meta-space diagram analysis and the Technical Committee was confirmed. No solutions were found regarding this issue during the design process. At each time human resource needs were scheduled, an extra person had to be added to deal with the Children Section.

4.2.2. Functional Assessment (Local) (Step VI)

The output of the functional analysis is a functional meta-space graph (F_{Pro} analysed). The functional meta-space graph is based on an assignment of functions from the functional meta-space diagram (F_{Pro} demanded). As a reminder, the functional meta-space diagram is an abstraction of operational meta-space design artefacts (B_{Pro} demanded) related to the system's internal behaviours (B_{Ser} and B_{Pro} demanded). Therefore, the Technical Committee has to analyse paths and flows regarding the assignment to assess the functional quality of the proposal based on the demanded system's behaviours (step VI) (Figure 158).

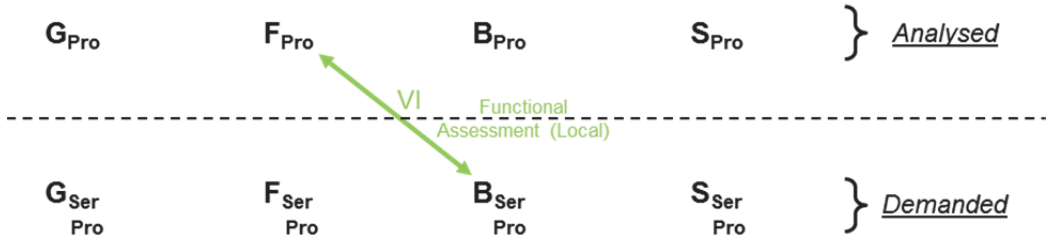


Figure 158 – Local Functional Assessment

Paths and flows can be analysed using the service models (B_{Ser} demanded) as a basis of use cases. Each use case has to be performed smoothly regarding the layout of the proposal. Figure 159 illustrates the book cycle limited to the "*borrow and return*" loop in the University library from two viewpoints. Figure 159.a represents the viewpoint of the books whereas Figure 159.b represents the viewpoints of the user (red lines) and customer (blue lines) clients (i.e. the viewpoint of the carrier). An important element to notice on this illustration is that the staircases are reserved for emergency situations and not for a day to day use.

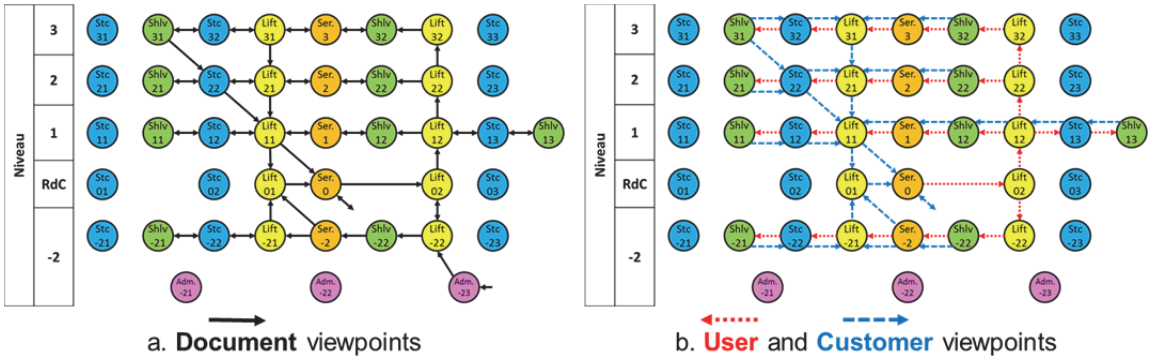


Figure 159 – Illustration of the "borrow and return" loop from the book cycle on the University library case study

The analysis started on the "*borrow and return*" loop can be refined based on the book collection and the yearly statistical data on the borrowings. Therefore, the allocation of the collection in the building can be carefully planned according to the clients' strategy. The allocation can favour the user clients' path (red lines) or the customer clients' path (blue lines). Such kind of analysis is possible depending on the degree of freedom left by the designers in the assignment of functions or in this case of the collections to the shelves in the building. The main constraint on this case study was the positioning of shelves independently from the assignment of collections. The first result of the analysis on this case study pointed out that the Dewey classification could not be applied on the floors from bottom to top in a logical way. Indeed, the distribution of shelves among the floors was unequal and discontinuous.

4.3. Quantitative Assessment

The quantitative assessment consists in verifying the quantities and characteristics of every component of the building system (step VII) (Figure 160). The design brief includes a space table that quantifies the number of spaces required in the future building. Usually, an equipment program is established during the briefing process, sometimes later, stating the quantity of required furniture and equipment. The Technical Committee is in charge of verifying that the proposals (S_{Pro} proposed) satisfy the numbers (S_{Pro} demanded). Such assessment can be supported by computer-aided tools, especially in the case of a BIM process. In terms of existing IT solutions for architectural programming, this part is almost systematically proposed (Ciscon et al. 2004; dRofus 2013; Fenves et al. 1995).

Figure 160 – Quantitative Assessment

A partial quantitative assessment is performed on the University library case study regarding shelves for the storage of books accessible to the public. Two assessments are performed. The first one is strictly based on the total quantity of shelves. The second one takes into account the distribution of shelves in the building in order to plan the distribution of the collection. The first assessment is positively concluding. The building disposes of 1,018 shelves to store around 160,000 documents. The collection from 2013 already counts around 142,000 documents. It allows an increase by only 15% of the collection before moving a part of the collection to the storehouse of the University library. Considering an average increase of the collection by 10% each year, it would have required 1 ½ years to reach this point. This conclusion leads to the decision to remove part of the current collection from direct access to the storehouse. The documents in question are unclassified documents regarding the Dewey classification (around 23,000 documents). Thereafter, the increase capacity reaches a more acceptable level of 35%.

The second assessment differs a little bit as the distribution of the shelves does not take into account the distribution of documents among the Dewey classification. The first issue concerns the Social Sciences class which is too big to be store in a single floor (#57,000 documents in 2013, 371 shelves). The maximum of shelves for a single floor is about 292 shelves on the 3rd floor. As a result, the Social Sciences class is divided into its 10 sub-classes and the user clients agree to shift the distribution from following the Dewey classification method to the creation of five thematic departments (poles). Each department occupies a single floor. The final result of the allocation (Figure 161) reduces the increase capacity from an average of 35% to 30%. Details of this case study are available in the Appendix J.

All Inclusive			Stockage max (-> %)								
			Niveau -2	Niveau -1	Niveau 0	Niveau 1	Niveau 2	Niveau 3	Total		
Classification DEWEY		Etagères	246	0	0	258	222	292	1,018	Reste	Pole
Généralités	000	31	100%						100%	0	1
Philosophie et psychologie	100	66						100%	100%	0	4
Religions	200	9	100%						100%	0	5
Sciences sociales	300	59				100%			100%	0	3
Statistiques	310	1						100%	100%	0	4
Science Politique	320	23				100%			100%	0	3
Economie Politique	330	70				100%			100%	0	3
Droit	340	96				100%			100%	0	3
Administration Publique	350	3				100%			100%	0	3
Services Sociaux; Associations	360	19						100%	100%	0	4
Education	370	201						100%	100%	0	4
Commerce, Communications, Transports	380	5				100%			100%	0	3
Coutumes et Folklore	390	2				100%			100%	0	3
Langues	400	47					100%		100%	0	2
Sciences	500	52	100%						100%	0	1
Technique	600	59	100%						100%	0	1
Arts, loisirs et sports	700	27	100%						100%	0	5
Littérature	800	162					100%		100%	0	2
Géographie et Histoire	900	57	100%						100%	0	5
TOTAL Espace restant à allouer		989	11	0	0	0	13	5	29.0	0	

Figure 161 – Dewey classification distribution in the University Library

4.4. In Summary

Based on the functional and quantitative assessment, the Technical Committee concludes with the writing of a "*delta report*" documenting the differences between the demand (i.e. the design brief), and the offers (i.e. designers' proposals). The conclusions of his objective assessment are presented to the paying clients and the jury of the architectural design competition.

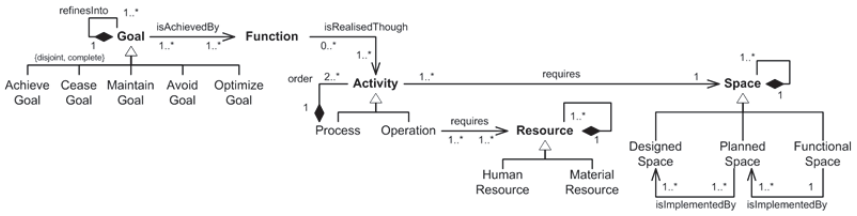
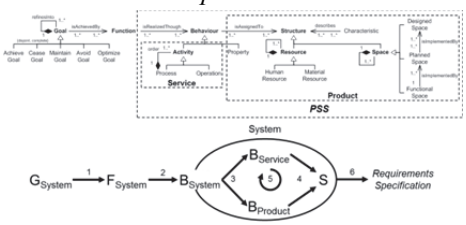
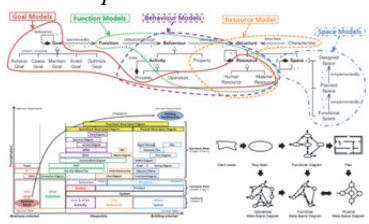
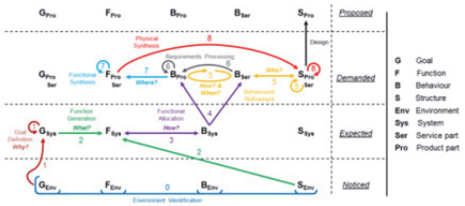
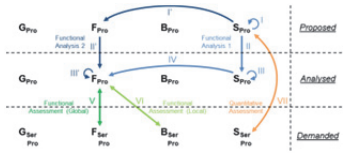
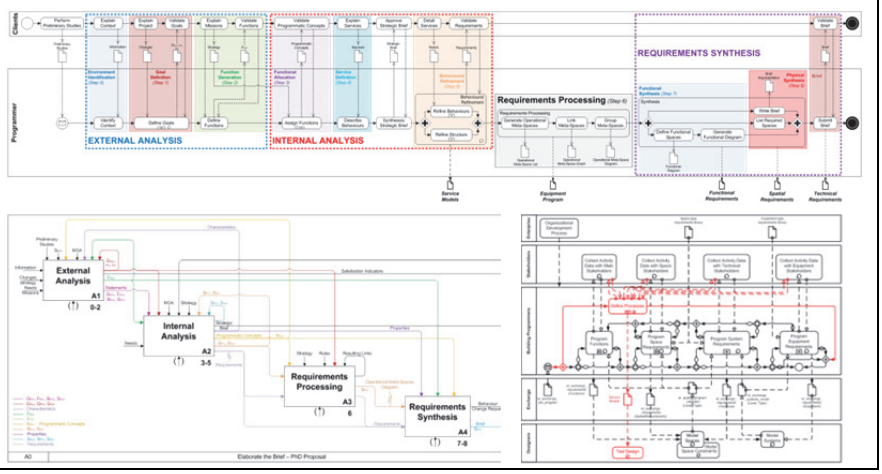
5. Synthesis

In this chapter, all of the design artefacts presented in the previous chapters were put together in an example of framework illustrated on case studies of various dimensions (Table 19). Four case studies were considered during the research project. Some were used as input for the design of the artefacts whereas the others were used to illustrate them. The aim of this chapter was to illustrate a theoretical briefing process using the proposed constructs and models. The resulting

GFBS framework is first introduced from a theoretical viewpoint, without formal representation of the method. Then, the framework is instantiated in different process model views: a user viewpoint (i.e. programmer) through a BPMN model, an information flow viewpoint through an IDEF0 model, and an updated information exchange process viewpoint through an IDM representation.

The combination of these representations of the same framework provides some elements about its consistency and applicability. It also underlines certain interest in the added value of each design artefact (i.e. constructs, models, and method). The next step would be to apply such a framework on a case study in real condition to confront it with reality. However, the various viewpoints also demonstrate the importance of developing a computer support for such a framework. The quantity of information generated at each step and the interoperation of the multiple tools and techniques make this matter essential.

Table 19 – Research Output of Chapter IV 2/2

Research Outputs	Research Activities: Build
Constructs *Chapter II*	*Definition Domain* 
Models *Chapter III*	*Conceptual Model*  *Operational Model* 
Method *Chapter IV*	 *Frameworks* 
Instantiation *Chapter IV*	*Process Model Views* 

Chapter V – CONCLUSION

This chapter concludes my research work in two sections. Each section analyses the contributions and findings based on the elements advanced in the Introduction chapter. Section 1 focuses on the academic side of this research whereas Section 2 focuses on the industrial side. Both sections conclude with some associated perspectives.

1. Research Synthesis

The research synthesis is divided into three parts. Section 1.1 analyses the research problem through the findings made all along this dissertation. Section 1.2 reviews the research questions and synthesises how they were addressed in the dissertation. Section 1.3 proposes a set of research perspectives to further develop these first findings.

1.1. Research Problem

The research problem addressed in this work focused on the definition of building requirements during the briefing process of construction projects. Building requirements are not defined out of certain purposes but should result from an analysis-synthesis of clients' needs by an expert: the programmer. Through a literature review, we saw in Chapter I that current practices, guides, and researches poorly document and formalise the transition from (high-level) informal business needs to (low-level) precise building requirements. We concluded that architectural programming is mainly experience-based, solution-oriented, and almost entirely manually performed.

In Chapter III, we confirmed this statement through a review of current practical models used in AEC to support this transition. The transition is poorly formalised in the early steps of the briefing process with quite an obscure and fuzzy shift from goal and objectives to a mix of functions, activities, and spaces. Furthermore, we highlight in this chapter that this transition is not formally documented and, thus, documentation and traceability of the transition is solely handled by the programmer and his (human) memory. Finally, Chapter II underlined that the distinction between modelling constructs like functions, activities, and spaces is not self-evident (or considered necessary) in architectural programming.

The transition from high-level business needs to low-level building requirements is structured, traced and formalised in other engineering domains. This statement was supported by literature overviews on Mechanical Engineering, Industrial Engineering, and Software Engineering on constructs required to define an object of study presented in Chapter II, and in Chapter III on modelling languages that formalise and interconnect them.

1.2. Research Questions & Contributions

The aim of this research was to develop an architectural programming reasoning based on engineering knowledge about requirements definition. *To reach this aim, we posed three research questions:*

- *(RQ1) which constructs are required to describe the building and its definition domain,*
- *(RQ2) how to model the building and its definition domain according to the multiple (views) stakeholders involved,*
- *and (RQ3) how to derive building requirements from business needs.*

The Research question (RQ1) was addressed through a literature review of constructs required to describe an object of study in engineering. The set of constructs (i.e. Goal, Function, Activity, Resource and Space) provided a structured building definition domain. This **building definition domain** was built in four steps. The first step was developed in Chapter II. It provided a first structure for the building definition domain. The second step appeared in Chapter III Section 2 when modelling the object of study with a distinction between two internal parts of building systems: product (static, passive) and service (dynamic, active). The third step was a result of this distinction between the product part and the service part but regarding function allocation (Chapter III Section 3). This allocation was supported by the concepts of Behaviour and Structure that applied to the product and the service parts. The resulting building definition domain represents an appropriate conceptual structure to define and describe a building from high-level to low-level needs. The last step was presented in Chapter II Section 5 through the addition of an abstract intermediary construct to support and formalise the transition from Activity (i.e. Operation) to Space: the **meta-space**.

Research question (RQ2) was addressed in three parts in Chapter III. The first part focused on the **conceptual modelling** of the building definition domain through a combination of two paradigms: PSS and FBS. The **PSS paradigm** was used to model the perimeter and complexity of our object of study: building systems. The **FBS paradigm** was used to model the transition from external analysis constructs (i.e. Goal and Function) to internal analysis constructs (i.e. Activity, Resource, and Space). The second part focused on the **operational modelling** of the building definition domain, i.e. the formalisation of needs into requirements. The proposed operational models were based on a multidisciplinary literature overview of existing modelling languages. However, existing modelling languages were not sufficient to cover all the required viewpoints on the building definition domain. The third step consisted in proposing a new model based on the meta-space construct to model the transitions between architectural programming deliverables taking benefit of the Activity-Space relationship: the **meta-space diagram**.

Research question (RQ3) was addressed through a conceptual design method for architectural programming: the **GFBS briefing framework**. This framework linked the **constructs** from Chapter II and the **models** from Chapter III in a theoretical **method** which can support the derivation of low-level building requirements from high-level business needs. The successive derivations were illustrated through four complementary case studies. In order to go further than just the theory, the GFBS briefing framework was instantiated in process model views. Through these views, we could see how current architectural programming practices could change to integrate such approach.

1.3. Research Perspectives

From an academic point of view, we identified a set of interesting perspective to explore mainly in AEC based on the limits and contributions of this thesis. Most of these perspectives are associated to the way we have conducted this thesis. Rather than to focus on some very specific research questions associated with specific elements of the briefing process, we have considered a broader scope by exploring the existence of a global framework supporting the whole process. By doing this, this is clear that many components of this framework should deserve a more detailed analysis.

1. Part of our approach was based on a multidisciplinary analysis where we have compared modelling constructs and languages existing in various engineering fields. Our first analysis should be completed by a more detailed ontological-based semantic analysis where the convergence identified among different constructs coming from different fields should be confirmed.
2. The meta-space diagrams are close to graphs used in layout generation and optimisation. The interaction with existing models, algorithms, and simulation software based on the information contained in the meta-space design artefact is another perspective to explore.
3. Parametric modelling and design is also another research perspective. The meta-space can provide simplified abstract views on building requirements (including its product and service parts) that can be used to add constraints on a parametric building model.
4. The influence of multiple engineering models on architectural design process is another research perspective to explore. How could these models be taught to architecture students and how could they change their perception/understanding of the building requirements. A practical aim of this contribution is to reach equally functional design proposals during the architectural design competition. If they were able to check the building requirements all along the design process through these models, what impact would it has.
5. Another perspective concerns the knowledge management for architectural programming to explore in terms of usage and data. The point of this research was to gather and formalise information about the building system during architectural programming and to be able to reuse it in later phases. The next stage would be to capitalise this information for a next construction project.

6. The last suggested perspective concerns the interactions (mapping) between the proposed building definition domain and the existing IFC and BIM models. Based on the high level information contained in the building definition domain, it should be possible to initiate a checking and reasoning process on the BIM models.

2. Industrial Synthesis

Despite the conceptual character of this research work, some industrial aspects had been integrated all along the research. First of all, the industrial scope limited to public construction projects provided a set of specific difficulties which we tried to face. The briefing process has to be done by the paying clients (or by the programmer under their responsibility) without intervention of the design team. Therefore, the contributions were developed based on the paying clients' knowledge, i.e. their business needs rather than on the building and the architecture. The brief is the essential communication document, which evolves through the briefing process. As a result, the contributions were developed to be flexible enough to integrate and trace changes and evolutions in the brief. The public construction briefing process does not end with the brief but continues through the assessment of design proposals. Ex-post facto case studies were therefore used to be able to analyse how the assessments of proposals could be integrated in a briefing framework.

The industrial synthesis section is structured into three parts. Section 2.1 provides an analysis of the industrial goal stated in the Introduction chapter through the contributions proposed all along the dissertation. Section 2.2 comes back on the industrial objectives and provides an analysis of their influence and integration in the development of the contributions. Section 2.3 suggests some potential transfers from research to industry based on the findings. Section 2.4 concludes on the industrial perspectives regarding what has been done in this research work.

2.1. Industrial Goal

The main practical goal of this research work was to *frame* and to *support* the *functional part* of the public construction briefing process. The *frame* was provided by the **building definition domain** and the **GFBS framework**. The *support* was proposed through the usage of **multiple engineering modelling languages** all along the framework and the development of a new design artefact and a modelling diagram: the **meta-space** and its associated meta-**space diagram**. The *functional part* of the brief refers to the activities performed in the building. The modelling of the **service part** of building systems and its relationship with the **product part** in the building definition domain and the **GFBS framework** (more specifically the functional allocation) was used to formalise this *functional part* of the briefing process.

2.2. Industrial Objectives

In the Introduction chapter, we defined a set of industrial objectives to guide the development of the constructs, models, and methods. In this section, we propose a review of these objectives regarding the various contributions.

Objective 1 – To help the clients *express* their requirements, and not solutions, we separated the first stages of the framework into two parts: an **external analysis** focused on the building system's environment and perimeter, and an **internal analysis** focused on the building system's components. Step by step, the clients are guided toward what they really want in terms of requirements. **Modelling languages** from multiple engineering domains provided a set of graphical tools and techniques to represent these requirements in a graphical way which is easier to *share* and *understand*. Even if they are not enough, the **multiple views** provided by the proposed models allow achieving a certain degree of *comprehensiveness*. The *relevancy* and *reliability* of requirements were not addressed in the contribution.

Objective 2 – To *support analysis* and *processing* of the requirements, we proposed a new design artefact, the **meta-space**, and a new modelling language, the **meta-space diagram**. The meta-space is an abstract concept that supports multiple views through the requirements it can model. However, its usage requires a computer support to be fully effective. The *consistency* of

requirements is partially tackled through the **multiple views** provided by **engineering modelling languages**. To better address *consistency*, a computer support is also required. The *maturity* of requirements was not considered in the contribution except in the **process model views** (i.e. IDEF0) through the control loops that can imply changes. Further research has to be done to really address this point. The *traceability* and *dependencies* between requirements can be observed through the **conceptual model** of the **building definition domain**. The **meta-space diagram** was designed to keep *trace* of higher level requirements and to represent the *dependencies* between **meta-spaces**. For example, the *dependencies* can be a sequence between operations in a process, or between flows of resources.

Objective 3 – Regarding the *specification* of requirements, we proposed to use the **multiple engineering modelling languages** that allow a graphical formalisation of needs for the interaction with clients (i.e. *validation*). The *usage* and *checking* by architects and designers could be supported by the **meta-space diagrams** which support the transitions and comparison between the brief, the functional diagram, and the drawings. The use of a **graph convention** to represent the meta-space diagrams tended toward *clarity*. The **multiple views** proposed by meta-space diagrams and its **synthetic graph aspect** were expected to provide *utility*. However, *clarity* and *utility* of the meta-space diagrams were just illustrated on case studies but not properly demonstrated. The integration to design state-of-the-art computer-aided solutions was only scratched during the research work and requires further investigations.

Objective 4 – The *verification* of the requirements implementation was more directly addressed in this research through the **GFBS assessment framework**. The **meta-space diagrams** partly support the *objective* assessment of proposals. In this dissertation, we proposed two ways to define a functional meta-space diagram based on project proposals. The first way was to use architects and designers information. The second way was to (manually) analyse the propositions. In both case, the objectivity of the analysis can be contested. We believe that the development of a computer support would improve the objectivity of this analysis. The *compliance* between the proposals and the brief was only illustrated on the multimedia library, showing how **meta-space diagrams** could support the identification of *compliances* and *non-compliances* through the **functional diagram**. The major hypothesis was to consider that the functional diagram synthesises all essential functional requirements which were not verified in the case study.

It is acknowledged that these objectives were not completely achieved but that was not our main goal. This research work started with a broad original scope that aimed to identify research trails of interest for the improvement of both the architectural programming practice and research through knowledge transfer from multiple engineering domains to AEC. We saw through their review how industrial objectives influenced or had been taken into account in the development of the contributions. Further researches are definitely required.

2.3. Research Transfer to Industry

The research findings made in this work could be of interest for:

- Architectural programming trainers/teachers who are looking for a theoretical framework supported by various practical modelling tools and techniques to teach architectural programming to architecture students,
- Programmers with little experience who are looking for a structured framework to begin with,
- Senior programmers who wish to update their practices with engineering models, techniques and supporting tools,
- Architects and designers who would like to thoroughly understand the clients' use of a future building before designing it,
- Consultant engineers in technical building services who would like to optimise their calculations through usage simulations.

2.4. Industrial Perspectives

My research study focused on conceptual contributions to architectural programming. The next steps should be an industrialisation of these contributions.

1. The extension of the building definition domain to the "non-functional" part of the brief should be considered as a perspective. A building system is not defined only by its physical properties and functional behaviours but also by social behaviours and various other qualities (e.g. semiology of spaces, spatial awareness, or aesthetic). However, the utility of modelling such information has also to be verified as the assessment of such requirements that cannot be automated through the meta-spaces.
2. A practical application from A to Z on a case study would be of interest to formalise the various steps of the theoretical framework. The case study does not need to be an ongoing one but requires at least the involvement of its stakeholders. Therefore, we could evaluate how the framework could have been applied and what would have changed all along the briefing and design phases.
3. Moreover, concerning the meta-space design artefact, the grouping rules are still to be formalised as well as the characteristics and properties of the building definition domain based on practical information. This research focused on a conceptual contribution and left the details such as an industrialisation of the findings aside.
4. A computer support and an automation of the framework could also be a research topic to explore. The first two elements would be the development of a database or information system centralising the data on the building definition domain then the development of a software support to manage the modelling languages and their interactions with such a database. This perspective has to be combined with a full practical application (2) and a study of practitioners' acceptance (6).
5. The integration or interaction with current BIM workflows should be investigated through:
    a. An improvement of BIM uses for building programming (information exchanges, IDM proposed in Chap IV),
    b. An improvement of existing programming software features (e.g. Revit, dRofus, or Trelligence)
    c. The definition of new checking rules tracing back high-level information for proposals comparison/assessment with the brief
6. Finally, the validation and acceptance of such contributions and findings by experienced and junior professionals (i.e. programmers, architects, and designers) should be evaluated. The complexity of the multiple engineering modelling languages and proposed design artefacts requires certain background knowledge or competencies that might be too difficult to acquire or acknowledge.

REFERENCES

Abdul-Kadir, M. & Price, A., 1995. Conceptual phase of construction projects. International Journal of Project Management, 13(6), pp. 387–393.

AFIS, 2004. Glossaire de Base de l'Ingénierie de Systèmes - Version Expérimentale, pp. 1–48.

AFNOR, 2005. ISO 9000-2005: Systèmes de management de la qualité - Principes essentiels et vocabulaire, p. 30.

AFNOR, 2000. NF EN 12973-2000: Management par la Valeur, p. 56.

AFNOR, 1996. NF EN 1325 - Vocabulaire du management de la valeur, de l'analyse de la valeur et de l'analyse fonctionnelle - Partie 1 : Analyse de la valeur et analyse fonctionnelle, p. 24.

Aguilar-Savén, R.S., 2004. Business process modelling: Review and framework. International Journal of Production Economics, 90(2), pp. 129–149.

AIA, 1988. The Handbook of Architectural Design Competitions, p. 41.

Akao, Y., 1993. Quality Function Deployment: Prendre en compte les besoins du client dans la conception du produit, AFNOR.

Akin, Ö. et al., 1995. SEED-Pro - Computer-Assisted Architectural Programming in SEED. Journal of Architectural Engineering, pp. 153–161.

Akin, Ö., Sen, R. & Donia, M.A., 1994. SEED-Pro: A framework for computer supported architectural programming. In Symposium on Architectural Computing: Technology in Transition, International Conference on Interdisciplinary Research and The 2nd Orwellian Symposium. Carlsbad-Karlovy Vary, Czech Republic, pp. 1–5.

Akinc, G., 2005. Architectural Programming for Achieving Value-Added Design. Middle East Technical University.

Alexander, C., 1971. The State of the Art in Design Methods. DMG Newsletter, 5(3), pp. 309–316.

Alexander, C., Ishikawa, S. & Silverstein, M., 1977. A Pattern Language, Oxford University Press, USA.

Allégret, J. & Guilleux, Y., 2000. Étude pour le recensement des dysfonctionnements récurrents dans les constructions publiques, p. 100.

Allégret, J., Mercier, N. & Zetlaoui-Léger, J., 2005a. L'exercice de la programmation architecturale et urbaine en France. In La Fabrication de la ville, pp. 87–101.

Allégret, J., Mercier, N. & Zetlaoui-Léger, J., 2005b. L'exercice de la programmation architecturale et urbaine en France - Note de Synthèse, pp. 1–40.

Alread, J. & Leslie, T., 2006. Design-Tech_Building Science for Architects, Architectural Press.

ANSI/EIA, 2003. ANSI/EIA 632-2003: Processes for Engineering a System.

Antón, A., McCracken, W. & Potts, C., 1994. Goal decomposition and scenario analysis in business process reengineering. Advanced Information Systems Engineering, 811, pp. 94–104.

Anton, A.I., 2003. Successful software projects need requirements planning. IEEE Software, 20(3), pp. 44–46.

Anwer, S. & Ikram, N., 2006. Goal Oriented Requirement Engineering: A Critical Study of Techniques. In Proceedings of the XIII Asia Pacific Software Engineering Conference (APSEC'06). IEEE Computer Society, p. 7.

Aoussat, A. & Le Coq, M., 1998. Méthodes globales de conception de produits. In M. Tollenaere, ed. Conception de produits mécaniques : méthodes, modèles et outils. Paris, France, pp. 53–75.

APICS, 2008. APICS Dictionary 3rd ed. J. H. Blackstone, ed., MGCM.

AQC, 2013. Sycodés 2013 - Les Indicateurs d'Evolution de la Qualité des Constructions, p. 72.

Arai, T. & Shimomura, Y., 2007. Proposal of Service CAD System - A Tool for Service Engineering. CIRP Annals - Manufacturing Technology, 53(1), pp. 397–400.

Bachmann, M., 1998. Design-Management als strategischer Erfolgsfaktor im Dienstleistungssektor: Ein Darstellungsversuch am Beispiel des Bankensektors, p. 160.

Baines, T.S. et al., 2007. State-of-the-art in Product-Service Systems. In P. G. Maropoulos, ed. Proceedings of the Institution of Mechanical Engineers - Part B: Journal of Engineering Manufacture. SAGE Publications, pp. 1543–1552.

Barrett, P., Hudson, J. & Stanley, C., 1999. Good practice in briefing : the limits of rationality. Automation in Construction, 8, pp. 633–642.

Barrett, P. & Stanley, C., 1999. Better construction briefing, Malden, Mass: Blackwell Science.

De Beauvais, C.-A., 1995. Programmes d'architecture. Techniques de l'Ingénieur, C4005, pp. 1–14.

Behrendt, S. et al., 2003. Eco-Service Development. Reinventing Supply and Demand in the European Union, Greenleaf Publishing.

Benjamin, P.C. et al., 1994. IDEF5 Method Report, p. 187.

Berard, O. & Karlshøj, J., 2011. Information delivery manuals to integrate building product information into design. In Proceedings of the CIB W78-W102 2011 International Conference, 26-28 October. Sofia Antipolis, France, pp. 26–28.

Berrada, T., 2006. Architectes et commanditaires : Cas particuliers du XVIe au XXe siècle, L'Harmattan.

Berry, D.M., 2003. More requirements engineering adventures with building contractors. Requirements Engineering, 8(2), pp. 142–146.

Berry, D.M., 2002. Requirements Engineering Lessons from House Building. In Schloß Dagstuhl Seminar on Supporting Customer-Supplier Relationships (CSR). p. 92.

Berry, D.M., 2000. Software and House Requirements Engineering : Lessons Learned in Combating Requirements Creep. Requirements Engineering, (1998), pp. 242–244.

Bertelsen, S., 2003. Construction as a complex system. In Proceedings of the 11th International Group for Lean Construction (IGLC-11). Blacksburg, VA, USA, p. 13.

Bisbrouck, M.-F., 2006. Architectural Competition for EPFL Library - Viewpoint of Technical Panel. LIBER quarterly, 16, pp. 222–235.

Bisbrouck, M.-F., 2004. From Concept to Commissioning_The Library - Scheduling, Programming, Phasing. LIBER quarterly, 14, pp. 218–231.

Bisbrouck, M.-F. & Renoult, D., 1993. Construire une bibliothèque universitaire - De la conception à la réalisation, Cercle de la Librairie.

Björk, B.-C., 1992a. A conceptual model of spaces, space boundaries and enclosing structures. Automation in Construction, 1(3), pp. 193–214.

Björk, B.-C., 1992b. A Unified Approach for Modelling Construction Information. Building and Environment, 27(2), pp. 173–194.

Blessing, L.T.M., 1996. Comparison of design models proposed in prescriptive literature. In J. Perrin & D. Vinck, eds. The role of design in the shaping of technology, COST A3/ COST A4 International research workshop. Lyon, France, pp. 1–19.

Blessing, L.T.M. & Chakrabarti, A., 2009. DRM, a Design Research Methodology, Springer.

Boehm, B.W. & Papaccio, P.N., 1988. Understanding and controlling software costs. IEEE Transactions on Software Engineering, 14(10), pp. 1462–1477.

Bogers, T., Meel, J.J. Van & Voordt, T.J.M. Van Der, 2008. Architects about briefing: Recommendations to improve communication between clients and architects. Facilities, 26(3/4), pp. 109–116.

Booch, G., Rumbaugh, J. & Jacobson, I., 1998. The Unified Modeling Language User Guide First Edit., Addison Wesley.

Bousbaci, R., 2004. Les modèles théoriques de l'architecture: de l'exaltation du faire à la réhabilitation de l'agir dans le bâtir. Université de Montréal.

Bracewell, R. & Sharpe, J., 1996. Functional descriptions used in computer support for qualitative scheme generation-'Schemebuilder'. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 10(04), pp. 333–345.

Breaux, T.D., Antón, A.I. & Doyle, J., 2008. Semantic Parameterization - A Process for Modeling Domain Descriptions. ACM Transactions on Software Engineering and Methodology, 18(2), pp. 1–27.

Brezet, J. et al., 2001. The Design of Eco-efficient Services: Method, Tools and Review of the Case Study Based'Designing Eco-efficient Services' Project, p. 46.

Broadbent, G., 1984. The Development of Design Methods. In N. Cross, ed. Development in Design Methodology. Toronto, Canada: Wiley, pp. 337–346.

buildingSMART, 2013. IFC4 Official Release. Available at: http://www.buildingsmart-tech.org/ifc/IFC4/final/html/index.htm [Accessed September 18, 2013].

Buur, J., 1990. A Theoretical Approach to Mechatronics Design. Technical University of Denmark.

Cabanieu, J., 1994. France - Competitions and Architectural Excellence. Places, 9(2), pp. 40–41.

Cascini, G. & Frate, L. Del, 2011. Beyond the Design Perspective of Gero's FBS Framework. In Proceedings of the 4th International Conference on Design Computing and Cognition (DCC'10), 12-14 July 2010. Stuttgart, Germany, pp. 77–96.

Certu, 2006. FICHE 26 : Le schéma fonctionnel, pp. 1–4.

Certu, 2010. Pour des bâtiments durables - Guide et outils de programmation, La Documentation Française.

Chambre des Députés, 2011. Projet de Loi relatif à la Bibliothèque universitaire de Belval, Luxembourg: Legitech.lu.

Chandrasekaran, B. & Josephson, J.R., 2014. Function in Device Representation. Engineering with Computers, 16(3-4), pp. 162–177.

Charpentier, F., 2005. Analyse fonctionnelle. Quels outils ? Technologie et formation, 35(117 & 118), pp. 10–15 & 24–35.

Chen, P.P.-C., 1976. The Entity-Relationship Unified View of Data Model - Toward a Unified View of Data. ACM Transactions on Database Systems, 1(1), pp. 9–36.

Cherry, E., 1999. Programming for Design: From Theory to Practice, John Wiley & Sons.

Chilakamarri, K., Hamburger, P. & Pippert, R., 1996. Venn diagrams and planar graphs. Geometriae Dedicata, 62(1), pp. 73–91.

Christophe, F., Bernard, A. & Coatanéa, E., 2010. RFBS: A model for knowledge representation of conceptual design. CIRP Annals - Manufacturing Technology, 59, pp. 155–158.

Chung, J.K.H., Kumaraswamy, M.M. & Palaneeswaran, E., 2009. Improving megaproject briefing through enhanced collaboration with ICT. Automation in Construction, 18(7), pp. 966–974.

Chynoweth, P., 2006. The Built Environment Interdiscipline: A Theoretical Model for Decision Makers in Research and Teaching. In Proceedings of the CIB working commission (W089) Building Education and Research Conference 2006 (BEAR 2006). Hong Kong, p. 12.

Ciscon, L.A. et al., 2002. Method and System for Leveraging Functional Knowledge in an Engineering Project, p. 24.

Ciscon, L.A. et al., 2004. Method and System for Leveraging Functional Knowledge in an Engineering Project, p. 50.

Clift, M., 1996. Building quality assessment (BQA) for offices. Structural Survey, 14(2), pp. 22–25.

Conan, M., 1997. L'invention des lieux, Theetete Eds.

Conan, M., 1988. La Programmation Générative, CSTB.

Conan, M., 1989. Secteur expérimental pour une programmation innovante de l'habitat des personnes agées. Méthode de programmation générative, Paris.

Cook, M.B., Bhamra, T. a. & Lemon, M., 2006. The transfer and application of Product Service Systems: from academia to UK manufacturing firms. Journal of Cleaner Production, 14(17), pp. 1455–1465.

Cooper, R. & Press, M., 1995. The Design Agenda - A Guide to Successful Design Management, John Wiley & Sons.

Le Corbusier, 1923. Towards A New Architecture, p. 138.

Coulin, C.R., 2007. A situational approach and intelligent tool for collaborative requirements elicitation. University of Technology, Sydney.

Cox III, J.F. et al., 2012. The Theory of Constraints International Certification Organization Dictionary 2nd ed. J. F. I. Cox et al., eds., McGraw-Hill Osborne Media.

Cross, N., 1993. Science and Design Methodology: A Review. Research in Engineering Design, 5(2), pp. 63–69.

Cross, N., 1981. The coming of post-industrial design. Design Studies, 2(1), pp. 3–7.

CRP Henri Tudor, 2012. La programmation architecturale au service du projet de construction. www.tudor.lu. Available at: http://www.tudor.lu/fr/event/la-programmation-architecturale-au-service-du-projet-de-construction.

Dardenne, A., Lamsweerde, A. Van & Fickas, S., 1993. Goal-directed Requirements Acquisition. Science of Computer Programming, 20, pp. 3–50.

Davis, G., 1978. A Process for Adapting Existing Buildings for New Office Uses. In W. F. E. Preiser, ed. Facility Programming. Stroudsburg, Pennsylvania: Hutchinson & Ross.

Davis, G., 1982. The Relationship of Evaluation to Facility Programming. In Symposium of Evaluation of Occupied Designed Environments. Georgia Institute of Technology, Atlanta.

Davis, G., 1972. Using Interviews of Present Office Workers in Planning New Offices. In Proceedings of the Environmental Design Research Association Conference. Los Angeles, USA, p. 9.

Davis, G. & Szigoti, F., 1985. Using a Joint Planning Process in Adaptive Reuse. In W. F. E. Preiser, ed. Programming the Built Environment. New York: Van Nostrand Reinhlod, p. 166.

Deng, Y., Britton, G. & Tor, S., 2000. Constraint-based functional design verification for conceptual design. Computer-Aided Design, 32, pp. 889–899.

Dettmer, W.H., 2007. The Logical Thinking Process: A Systems Approach to Complex Problem Solving, Amer Society for Quality.

DGUHC, 2000. Rédiger le préprogramme et le programme, Paris, France: Ministère de l'Equipement, des Transports, du Logement, du Tourisme et de la Mer - Direction Générale de l'Urbanisme de l'Habitat et de la Construction.

DGUHC & Certu, 1999. Choisir un programmateur et encadrer sa mission, Ministère de l'Equipement, des Transports, du Logement, du Tourisme et de la Mer - Direction Générale de l'Urbanisme de l'Habitat et de la Construction.

Dimeglio, P., 2001. La programmation générative et participative en architecture et en urbanisme : proposition pour l'élaboration du projet urbain de l'Ile. Urbanisme, (320), pp. 26–31.

Dimeglio, P., Zetlaoui-Léger, J. & Trichet, F., 2005. Contribution à l'élaboration d'un Charte de programmation concertée et participative, Paris.

Ding, G.K.C., 2008. Sustainable construction–The role of environmental assessment tools. Journal of Environmental Management, 86(3), pp. 451–464.

Donia, M.A., 1998. Computational Modeling of Design Requirements for Buildings. Carnegie Mellon University.

Doran, G.T., 1981. There's a S.M.A.R.T. way to write management's goals and objectives. Management Review, 70(11), pp. 35–36.

Dorst, K. & Vermaas, P.E., 2005. John Gero's Function-Behaviour-Structure model of designing: a critical analysis. Research in Engineering Design, 16(1-2), pp. 17–26.

dRofus, 2013. dRofus Users' Guide, p. 104.

Duerk, D.P., 1993. Architectural programming - Information management for design, New York: Van Nostrand Reinhlod.

Dursun, P., 2012a. An Analytical Tool for Thinking and Talking About Space. In M. Green, J. Reyes, & A. Castro, eds. Proceedings of the 8th International Space Syntax Symposium. Santiago de Chile, pp. 1–16.

Dursun, P., 2009. Architects are Talking About Space. In D. Koch, L. Marcus, & J. Steen, eds. Proceedings of the 7th International Space Syntax Symposium. Stockholm, pp. 1–8.

Dursun, P., 2012b. Dialogue on space: Spacial codes and language of space. ITU Journal of Faculty of Architecture, 9(1), pp. 104–119.

Eckert, C.M., 2013. That Which is not Form. The Practical Challenges in Using Functional Concepts in Design. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 27(3), pp. 217–231.

Eckert, C.M., Alink, T. & Albers, A., 2010. Issue Driven Analysis of an Existing Product at Different Levels of Abstraction. In Proceedings of the 11th International Design Conference (Design 2010). Dubrovnik, Croatia, p. 10.

Eder, W.E., 2008. Requirements to Properties - Iterative Problem Solving. In Proceedings of the AEDS 2008 Workshop. Pilsen, Czech Republic, pp. 49–66.

Egan, J., 1998. Rethinking construction, London.

Eisenbart, B., 2014. Supporting Interdisciplinary System Development Through Integrated Function Modelling. Université du Luxembourg.

Eisenbart, B., Blessing, L.T.M. & Gericke, K., 2012. Functional modelling perspectives across disciplines: a literature review. In Proceedings of the 12th Design Conference (Design 2012). Dubrovnik, Croatia, pp. 847–858.

Eisenbart, B., Gericke, K. & Blessing, L.T.M., 2013. An Analysis of Functional Modelling Approaches Across Disciplines. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 27(3), p. 26.

El-Diraby, T., Lima, C. & Feis, B., 2005. Domain taxonomy for construction concepts: toward a formal ontology for construction knowledge. Journal of computing in civil engineering, (October), pp. 394–406.

El-Gohary, N.M. & El-Diraby, T.E., 2011. Merging Architectural, Engineering, and Construction (AEC) Ontologies. Journal of Computing in Civil Engineering, 25(2), pp. 109–128.

Erden, M.S. et al., 2008. A review of function modeling: Approaches and applications. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 22(02), pp. 147–169.

Euler, L., 1736. Solutio problematis ad geometriam situs pertinentis. Commentarii academiae scientiarum Petropolitanae, 8, pp. 128–140.

Evbuomwan, N.F.O. & Anumba, C.J., 1996. Towards an Integrated Engineering Design Environment. In B. Kumar & A. Retik, eds. Information Representation and Delivery in Civil and Structural Engineering. Edinburg: Civil-Comp Press, pp. 127–134.

Faatz, S., 2009. Architectural programming - Providing essential knowledge of project participants needs in the pre-design phase. Organization, Technology & Management in Construction: An International Journal, 1(2), pp. 80–85.

Faisandier, A., 2011. Les bases de l'ingénierie de système, pp. 1–104.

Farbstein, J., 1976. Assumptions in environmental programming. In Proceedings of the 7th International Conference of the Environmental Design Research Association. Vancouver, Canada, pp. 21–26.

Feng, P.P. & Tommelein, I.D., 2008. Causes of rework in California hospital design and permitting: Augmenting an existing taxonomy. In Proceedings of the 17th Annual Conference of the IGLC (IGLC17). Taipei, Taiwan, pp. 407– 416.

Feng, P.P., Tommelein, I.D. & Ballard, G., 2009. Modeling the Effect of an Alternative Review Process: Case Study of a State Permitting Agency. In Building a Sustainable Future. ASCE, pp. 866–875.

Fenves, S.J. et al., 1995. Conceptual Structural Design in SEED. Journal of Architectural Engineering, pp.179–186.

Flemming, U. & Snyder, J., 1997. Building and Databases: the SEED Experience. In Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen. Weimar, p. 9.

Fortin, G., 1978. BUBBLE: Relationship Diagrams using Iterative Vector Approximation. In Proceedings of the 15th Conference on Design Automation. Las Vegas, USA, pp. 145–151.

Frederick, C., 1923. Household Engineering - Scientific Management in the Home.

Gans, H., 1962. The Urban Villagers, The Free Press.

Gericke, K., Qureshi, A.J. & Blessing, L.T.M., 2013. Analyzing transdisciplinary design processes in industry–an overview. In Proceedings of the ASME 2013 International Design Engineering Technical Conference & Computers and Information in Engineering Conference (IDETC/CIE 2013). Portland, Oregon, USA, p. 10.

Gero, J.S., 1990. Design prototypes: a knowledge representation schema for design. AI magazine, 11(4), pp. 26–36.

Gero, J.S. & Kannengiesser, U., 2006. A function–behavior–structure ontology of processes. In Proceedings of the 2nd International Conference on Design Computing and Cognition (DCC'06). Eindhoven, pp. 407–422.

Gero, J.S. & Kannengiesser, U., 2004. The situated function-behaviour-structure framework. Design Studies, 25(4), pp. 373–391.

Giunchiglia, F., Mylopoulos, J. & Perini, A., 2001. The Tropos Software Development Methodology: Processes, Models and Diagrams, p. 10.

Glinz, M., 2011. A Glossary of requirements engineering terminology, International Requirements Engineering Board.

Gobin, C., 2003. Analyse fonctionnelle et construction. Techniques de l'Ingénieur, C3052, pp. 1–15.

Gobin, C., 2006. Développement durable en BTP - Fonctions d'usage. Techniques de l'Ingénieur, C3057, pp. 1–12.

Gobin, C., 2001. L'ingénierie concourante - Un nouveau professionnalisme. Techniques de l'Ingénieur, C3050, pp. 1–14.

Goedkoop, M.J. et al., 1999. Product Service systems, Ecological and Economic Basics, (March), p. 132.

Goel, A.K., Rugaber, S. & Vattam, S.S., 2009. Structure , Behavior and Function of Complex Systems : The SBF Modeling Language. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 23(1), pp. 23–35.

Goldratt, E.M. & Cox, J., 1984. The Goal: A Process of Ongoing Improvement, North River Press.

Grandjean, A., 1980. La simulation du "Futur Vécu" - Une méthode d'analyse "dynamique" des projets, p. 32.

Grason, J., 1971. An Approach to Computerized Space Planning using Graph Theory. In Proceedings of the 8th Design Automation Workshop (DAC '71). pp. 170–179.

Greefhorst, D. & Proper, E., 2011. Architecture Principles - The Cornerstones of Enterprise Architecture, Springer-Verlag Berlin Heidelberg.

Green, S.D. & Simister, S.J., 1999. Modelling client business processes as an aid to strategic briefing. Construction Management & Economics, 17(1), pp. 63–76.

Guerriero, A. & Fry, C., 2014. La programmation architecturale au Luxembourg - Enquête et perspectives. In S. Kubicki, G. Halin, & J.-C. Bignon, eds. Proceedings du Séminaire de Conception Architecturale Numérique, 18-20 Juin 2014. Luxembourg: PUN - Editions Universitaire de Lorraine, pp. 103–112.

Hadorn, G.H. et al., 2008. Handbook of Transdisciplinary, Springer-Verlag New York Inc.

Hall, E.T., 1987. Au-delà de la culture, Seuil.

Harbouche, L., Schaefers, J. & Schwartz, T., 2008. Supply chain collaboration in construction: How to induce the paradigm shift for more value. In Proceedings of the International Conference on Information Systems, Logistics and Supply Chain (ILS 2008). Madison, USA, p. 6.

Hassanain, M. a. & Juaim, M.N., 2013. Modeling Knowledge for Architectural Programming. Journal of Architectural Engineering, 19(2), pp. 101–111.

Hatchuel, A. & Weil, B., 2003. A News Approach of Innovative Design - An Introduction to C-K Theory. In Proceedings of the International Conference on Engineering Design (ICED 03). Stockholm, pp. 1–15.

Heintz, J.L. & Overgaard, F., 2007. From Program to Design, How Architects' Use Briefing Documents. In Proceedings of the Nordes 2007 - Design Inquiries. Stockholm, Sweden, p. 8.

Hershberger, R.G., 1999. Architectural programming and predesign manager Har/Cdr., New York: McGraw-Hill.

Hillier, B. & Hanson, J., 1997. The Reasoning Art or, The Need for an Analytical Theory of Architecture. In Proceedings of the Space Syntax 1st International Sympymposium. London, England, p. 6.

Hillier, B. & Hanson, J., 1989. The Social Logic of Space Reprint ed., Cambridge University Press.

Hillier, B., Hanson, J. & Peponi, J., 1976. What do we mean by building function, pp. 61–72.

Hillier, B. & Leaman, A., 1973. The Man-Environment Paradigm and its Paradoxes. Architectural Design, 8(78), pp. 507–511.

Howard, H. et al., 1989. Computer integration: reducing fragmentation in AEC industry. Journal of computing in civil engineering, 3(1), pp. 18–32.

Hsu, W., 2000. Conceptual design: issues and challenges. Computer-Aided Design, 32, pp. 849–850.

Hubka, V., Andreasen, M.M. & Eder, W.E., 1988. Practical studies in systematic design, Butterworths.

Hunter, K. & Kelly, J., 2003. The path to the application of value management in the UK public service sector. In Proceedings of the PRoBE Conference. Glasgow, pp. 3–15.

Huovila, P. & Hyvarinen, J., 2012. Deliverable D12_Final Report, pp. 1–77.

IABG, 1997. The V-Model - Development Standard for IT-Systems of the Federal Republic of Germany.

IEEE, 2005. IEEE Std 1220-2005: IEEE Standard for Application and Management of the Systems Engineering Process, p. 97.

IEEE, 1998. IEEE Std 1233-1998: IEEE Guide for Developing System Requirements Specifications, p. 30.

IEEE, 2008. ISO/IEC 12207: Systems and Software Engineering - Software Life Cycle Processes, 8, p. 138.

IEEE, 2010. ISO/IEC/IEEE 24765-2010: Systems and Software Engineering - Vocabulary, p. 418.

IEEE Computer Society, 1998. IEEE Std 1320.1:1998: Standard for Functional Modeling Language — Syntax and Semantics for IDEF0, 1998, p. 115.

IEEE Computer Society, 2005. ISO/IEC TR 19759-2005: Software Engineering - Guide to the Software Engineering Body of Knowledge (SWEBOK), p. 204.

IIBA, 2009. A Guide to Business Analysis Body of Knowledge 2nd ed. K. Brennan, ed., International Institute of Business Analysis.

INCOSE, 2010. Systems Engineering handbook_A guide for system life cycle processes and activities, p. 185.

ISO, 1994. ISO 9699: Normes de performance dans le bâtiment - Liste de contrôle consultative - Contenu d'un programme de conception dans l'industrie du bâtiment, 1994, p. 20.

ISO/IEC, 2001. DTR 9126-4: Software Engineering - Product quality - Part 4_Quality in use metrics, p. 71.

ISO/IEC, 2008a. ISO/IEC 15288-2008: Systems and Software Engineering - System Life Cycle Processes, 8, p. 84.

ISO/IEC, 2006. ISO/IEC 15414-2006: Information Technology - Open Distributed Processing - Reference Model - Entreprise Language, p. 41.

ISO/IEC, 2008b. ISO/IEC 38500-2008: Corporate Governance of Information Technology, p. 15.

Jackson, M. & Zave, P., 1993. Domain Descriptions. In Proceedings of the 1st International IEEE Symposium on Requirements Engineering (RE'93). San Diego, USA, pp. 56–64.

Jerving, R., 2011. Process Map [Building programming], pp. 1–5. Available at: http://iug.buildingsmart.org/idms/information-delivery-manuals/idm-for-building-programming/IDM_process_map_building_programming_161111.pdf/view.

Jong, C.W. de & Mattie, E., 1995. Architectural Competitions, Taschen GmbH.

JORF, 1973. Décret n°73-207 du 28 février 1973 relatif aux conditions de rémunération des missions d'ingénierie et d'architecture remplies pour le compte des collectivités publiques par des prestataires de droit privé. Journal Officiel de la République Française, p. 6.

JORF, 1993. Décret n°93-1269 du 29 novembre 1993 relatif aux concours d'architecture et d'ingénierie organisés par les maîtres d'ouvrage publics. Journal Officiel de la République Française.

JORF, 1985. Loi n° 85-704 du 12 juillet 1985 relative à la maîtrise d'ouvrage publique et à ses rapports avec la maîtrise d'œuvre privée (loi MOP). Journal Officiel de la République Française, pp. 7914–7917.

JORF, 2010. Loi n°85-704 du 12 juillet 1985 relative à la maîtrise d'ouvrage publique et à ses rapports avec la maîtrise d'oeuvre privée, version consolidée au 09 décembre 2010. Journal Officiel de la République Française.

JORF, 2007. Loi n°85-704 du 12 juillet 1985 relative à la maîtrise d'ouvrage publique et à ses rapports avec la maîtrise d'oeuvre privée, version consolidée au 22 février 2007. Journal Officiel de la République Française, p. 10.

Juaim, M.N. & Hassanain, M. a., 2011. Assessment of factors influencing the development and implementation of the architectural program. Structural Survey, 29(4), pp. 320–336.

Jureta, I.J., Mylopoulos, J. & Faulkner, S., 2008. Revisiting the Core Ontology and Problem in Requirements Engineering. In Proceedings of the 16th IEEE International Requirements Engineering (RE'08). Catalunya, pp. 71–80.

Kalay, Y.E., 2001. Enhancing multi-disciplinary collaboration through semantically rich representation. Automation in Construction, 10, pp. 741–755.

Kamara, J.M. & Anumba, C.J., 2001. ClientPro: a prototype software for client requirements processing in construction. Advances in Engineering Software, 32(2), pp. 141–158.

Kamara, J.M., Anumba, C.J. & Evbuomwan, N.F.O., 2001. Assessing the suitability of current briefing practices in construction within a concurrent engineering framework. International Journal of Project Management, 19(6), pp. 337–351.

Kamara, J.M., Anumba, C.J. & Evbuomwan, N.F.O., 2002. Capturing client requirements in construction projects, Thomas Telford Ltd.

Kamara, J.M., Anumba, C.J. & Evbuomwan, N.F.O., 1999. Client requirements processing in construction: a new approach using QFD. Journal of architectural engineering, 5(March), p. 8.

Kannengiesser, U. & Gero, J.S., 2011. A Framework for Constructive Design Rationale. In Proceedings of the 4th International Conference on Design Computing and Cognition (DCC'10), 12-14 July 2010. Stuttgart, Germany, pp. 135–153.

Karni, R. & Kaner, M., 2007. An engineering tool for the conceptual design of service systems. Advances in Services Innovations, pp. 65–83.

Kavakli, E. & Loucopoulos, P., 2003. Goal Driven Requirements Engineering - Evaluation of Current Methods. In Proceedings of the 8th CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'03). Velden, Austria, pp. 1–11.

Kelly, J., Morledge, R. & Wilkinson, S., 2002. Best value in construction, Wiley-Blackwell.

Kim, D. & Boradkar, P., 1988. Sensibility Design, p. 8.

Kiviniemi, A. et al., 2004. PREMISS-Requirements Management Interface to Building Product Models: Problem Definition and Research Issues. CIFE Work ing Paper, (October), p. 62.

Kiviniemi, A., 2005. Requirements management interface to building product models, Stanford University Stanford, CA, USA.

Kossiakoff, A. et al., 2011. Systems Engineering - Principles and Practice 2nd ed. A. P. Sage, ed., Wiley-Interscience.

Koutamanis, A., 2013. Computer-Mediated Briefing for Architects, IGI Global.

Kubicki, S., Bignon, J.-C. & Halin, G., 2006. Assistance to building construction coordination–towards a multi-view cooperative platform. In Information Technology in Construction. pp. 565–586.

Kumlin, R.R., 1995. Architectural Programming - Creative Techniques for Design Professionals, McGraw-Hill Professional.

De la Bretesche, B., 2000. La méthode APTE Analyse de la Valeur Analyse Fonctionnelle, Paris, France: Pétrelle.

Labrousse, M. & Bernard, A., 2003. Modèle FBS enrichi pour la modélisation des processus d'entreprise. In Proceedings of the 3rd International Conference Integrated Design and Production (CPI 2003). Meknes, Morocco, pp. 1–16.

Labrousse, M., Perry, N. & Bernard, A., 2004. Modèle FBS-PPR - des objets d'entreprise à la gestion dynamique des connaissances industrielles. In B. Eynard et al., eds. Gestion dynamique des connaissances industrielles. Lavoisier, pp. 1–18.

Lamsweerde, A. Van, 1996. Divergent views in goal-driven requirements engineering. In Joint Proceedings of the Sigsoft 96 Workshops - Specifications 96. ACM Press, pp. 252–256.

Lamsweerde, A. Van, 2001. Goal-Oriented Requirements Engineering - A Guided Tour. In Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE'01). Toronto, p. 14.

Lamsweerde, A. Van, 2000. Requirements Engineering in the Year 00: a Research Perspective. In Proceedings of the 2000 International Conference on Software Engineering (ICSE 2000 the New Millennium). Acm, pp. 5–19.

Lamsweerde, A. Van, 2009. Requirements Engineering: From System Goals to UML Models to Software Specifications, John Wiley & Sons Ltd.

Lang, J. & Moleski, W., 2010. Functionalism Revisited: Architectural Theory and Practice and the Behavioural Sciences, Ashgate Publishing Limited.

Lapouchnian, A., 2005. Goal-oriented requirements engineering: An overview of the current research. University of Toronto, p. 32.

Latham, M., 1994. Constructing the Team, London.

Lawson, B., 2005. How designer think Fourth Edi., Elsevier.

Lawson, B., 2001. The Language of Space, Architectural Press.

Lefebvre, H., 1974. La Production de l'Espace, Economica.

Letier, E. & Lamsweerde, A. Van, 2002. Agent-based Tactics for Goal-oriented Requirements Elaboration. In Proceedings of the 24th International Conference on Software Engineering (ICSE'02). Orlando: IEEE Press, pp. 83–93.

Lobos, D. & Donath, D., 2010. The problem of space layout in architecture: A survey and reflections. Arquitetura Revista, 6(2), pp. 136–161.

Love, P.E.D. & Li, H., 2000. Quantifying the causes and costs of rework in construction. Construction Management & Economics, 18, pp. 479–490.

Luxembourg Building Ministry, 2006. Modular concept for Secondary School.

Macmillan, S. et al., 2001. Development and verification of a generic framework for conceptual design. Design Studies, 22(2), pp. 169–191.

Magner-Canet, S. & Yannou, B., 1998. IfNot: A Function-Based Approach for Both Proactive and Reactive Integrations of New Productive Technologies. In Proceedings of the 11th FLAIRS Conference. pp. 344–350.

Manzini, E. & Vezzoli, C., 2003. A strategic design approach to develop sustainable product service systems: examples taken from the "environmentally friendly innovation" Italian prize. Journal of Cleaner Production, 11(8), pp. 851–857.

March, S.T. & Smith, G.F., 1995. Design and natural science research on information technology. Decision Support Systems, 15, pp. 251–266.

Maslow, A.H., 1943. A theory of human motivation. Psychological review, (13), pp. 370–396.

Le Masson, P., Dorst, K. & Subrahmanian, E., 2013. Design theory: history, state of the art and advancements. Research in Engineering Design, 24(2), pp. 97–103.

Mauger, C. et al., 2010. Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: The elicitation process. In Proceedings of the 2010 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM 2010). Macao: IEEE, pp. 310–314.

Mauger, C. et al., 2013. Transdisciplinary Research - Buildings as Service-Oriented Product-Service Systems. In L. T. M. Blessing, A. J. Qureshi, & K. Gericke, eds. Proceedings of The Future of Transdisciplinary Design, 2013 (TFTD13). Luxembourg, p. 219.

Mauger, C. & Berry, D.M., 2014. Lessons Learned from and for Requirements Engineering and Building Construction : A Case Study of Requirements Engineering for a Synagogue Kitchen with Use Cases and Scenarios. In Proceedings of the Software Summit SwSTE 2014 The IEEE Computer Society International Software Conference, 11-12 June 2014. Tel-Aviv, Israel: IEEE Computer Society CPS, pp. 67–76.

Mauger, C. & Berry, D.M., 2013. Requirements Engineering and Building Construction : Requirements Engineering for a Synagogue Kitchen with Use Cases and Scenarios. In Poster session REFSQ2013. Essen, Germany, p. 15.

Mauger, C. & Kubicki, S., 2013. A Conceptual Model for Building Requirements Processing. In Proceedings of the 11th IPGRC. Manchester, UK, pp. 318–330.

Maussang, N., 2008. Méthodologie de conception des systèmes produits-services. Institut polytechnique de Grenoble.

Mayer, R.J., Crump, J.W., et al., 1995. Information Integration for Concurrent Engineering (IICE) - Compendium of Methods Report, p. 149.

Mayer, R.J., Menzel, C.P., et al., 1995. Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report, p. 236.

McLaughlin, H., 1976. Programming. In R. Class & R. Koehler, eds. Current Techniques in Architectural Practice. Washington DC: American Institute of Architects, pp. 121–132.

McLaughlin, H., 1979. Programming, Predilections, and Design. Architectural Record, (January), pp. 69–71.

Meier, H., Roy, R. & Seliger, G., 2010. Industrial Product-Service Systems — IPS 2. CIRP Annals - Manufacturing Technology, 59, pp. 607–627.

Meijkamp, R., 1998. Changing Consumer Behaviour Through Eco-Efficient Services_An Empirical Study of Car Sharing in the Netherlands. Business Strategy and the Environment, (7), pp. 234–244.

Merrell, P., Schkufza, E. & Koltun, V., 2010. Computer-Generated Residential Building Layouts. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2010, 29(6), p. 11.

Meyer, S. et al., 2002. The service concept : the missing link in service design research ? Journal of Operations Management, 20, pp. 121–134.

Michl, J., 1997. Form Follows WHAT - The Modernist Notion of Function as a Carte Blanche, p. 26.

Miles, L.D., 1972. Techniques of Value Analysis and Engineering, McGraw-Hill.

MIQCP, 2008. Guide de sensibilisation à la Programmation J. Cabanieu et al., eds., Paris, France: Mission Interministérielle pour la Qualité des Construction Publiques.

MIQCP, 2012. Le concours de maîtrise d'œuvre : dispositions réglementaires et modalités pratiques d'organisation C. Romon & G. Lamour, eds., Paris, France: Mission Interministérielle pour la Qualité des Construction Publiques.

MIQCP, 2001. Programmation des constructions publiques 2nd Ed., Paris, France: Le Moniteur.

Miyatake, Y. & Kangari, R., 1993. Experiencing computer integrated construction. Journal of Construction Engineering and Management, 119(2), pp. 307–322.

Moleski, W.H. & Goodrich, R.J., 1972. The Analysis of Behavioural Requirements in Office Settings. In W. J. Mitchell, ed. Proceedings of the 3rd Environmental Design Research Association Conference (EDRA 3). Los Angeles, pp. 1–7.

Mont, O., 2002. Clarifying the concept of product-service system. Journal of Cleaner Production, 10, pp. 237–245.

Mont, O., 2000. Product-Service Systems. Final Report, p. 85.

Morgan, M.H., 1914. The Ten Books on Architecture, Cambridge: Harvard University Press.

Moritz, S., 2005. Service Design - Practical access to an evolving field, London.

Müller, P. & Blessing, L.T.M., 2007. Development of Product-Service-Systems–Comparison of Product and Service Development Process Models. In Proceedings of the 16th International Conference on Engineering Design (ICED'07). Paris, France, pp. 1–12.

Muller, P., Schmidt-Kretschmer, M. & Blessing, L.T.M., 2007. Function Allocation in Product-Service Systems - Are There Analogies between PSS and Mechatronics? In Proceedings of the AEDS 2007 Workshop. Pilsen, Czech Republic, pp. 47–55.

National Institute of Standards and Technology, 1993. Integration Definition for Function Modeling (IDEF0), p. 128.

Neufert, E., 1936. Bauentwurfslehre 1st Ed., Bauwelt-Verlag.

Neufert, E., 1996. Les éléments des projets de construction 7th Ed., Dunod/Le Moniteur.

Neufert, E. & Neufert, P., 2012. Architects' Data 4th Ed., Wiley-Blackwell.

Neveu, M.J., 2005. Architectural Lessons of Carlo Lodoli (1690-1761): Indole of Material and of Self. McGill University, Montréal.

Newcombe, R., 2003. From client to project stakeholders: a stakeholder mapping approach. Construction Management & Economics, 21(8), pp. 841–848.

Nykänen, E. et al., 2008. Hospitool. Käyttäjälähtöinen sairaalatila, 2455, p. 71.

Oliveira, L.A. de, Maizia, M. & Melhado, S.B., 2008. Influence of the performance and buildability requirements on the building quality: Comparison between the Brazilian and the French renovation design process. In Proceedings of the Joint CIB W096 Architectural Management and CIB TG49 Architectural Engineering Conference. Rotterdam, Netherlands, p. 10.

OMG, 2011a. Business Process Model and Notation (BPMN) - v2.0, p. 538.

OMG, 2010. OMG Systems Modeling Language (OMG SysMLTM), p. 260.

OMG, 2011b. OMG Unified Modeling Language TM (OMG UML), Superstructure v2.4.1, p. 748.

Op't Land, M. et al., 2009. Enterprise Architecture - Creating Value by Informed Governance, Springer-Verlag Berlin Heidelberg.

Pahl, G. & Beitz, W., 1995. Engineering Design - A Systematic Approach 2nd Editio. K. Wallace, ed., Darmstadt, Germany: Springer Verlag.

Palmer, M.A., 1981. Architect's Guide to Facility Programming, McGraw-Hill Inc.

Paulson, B., 1976. Designing to reduce construction costs. Journal of the Construction Division, pp. 587–592.

Peachavanish, R. et al., 2006. An ontological engineering approach for integrating CAD and GIS in support of infrastructure management. Advanced Engineering Informatics, 20(1), pp. 71–88.

Pena, W.M., 1969. Organizing for Programming. Building Research, (April-June).

Pena, W.M., 1959. People and Program: Background of the Architect. The National Elementary Principal, 33(1).

Pena, W.M. & Caudill, W.W., 1959. Architectural Analysis - Prelude to Good Design. Architectural Record, 125(5), pp. 178–182.

Pena, W.M. & Parshall, S.A., 2001. Problem Seeking - An Architectural Programming Primer 4th editio. S. Fonseca de Nino, ed., John Wiley & Sons.

Pérouse de Montclos, J.-M., 1984. Les Prix de Rome : Concours de l'Académie roayle d'architecture au XVIIIe siècle, Berger-Levrault.

Popper, K., 1936. Conjectures and Refutations 1st Ed., Routledge.

Prasad, B., 1998. Review of QFD and Related Deployment Techniques. Journal of Manufacturing Systems, (3), pp. 221–234.

Preiser, W.F.E., 1972a. Application of Unobtrusive Observation Techniques in Building Performance Appraisal. In Proceedings of the Joint RILEM-ASTM-CIB Symposium, May 2-5 1972. Philadelphia, USA, pp. 101–109.

Preiser, W.F.E., 1977. Programming for Use-Oriented Facilities. In Proceedings of the 7th Environmental Design Research Association Conference (EDRA 7). Los Angeles, USA: Dowden, Hutchinson & Ross.

Preiser, W.F.E., 1985. Programming the Built Environment, Van Nostrand Reinhlod.

Preiser, W.F.E., 1972b. Verbalized User Response and the Building Performance Concept: A Case Study in University Residence Hall Evaluation. In Proceedings of the Joint RILEM-ASTM-CIB Symposium, May 2-5 1972. Philadelphia, USA, pp. 111–119.

Project Management Institute, 1996. A guide to the project management body of knowledge 3rd Ed., Project Management Institute, Inc.

Ramadier, T., 2004. Transdisciplinarity and its challenges: the case of urban studies. Futures, 36(4), pp. 423–439.

RIBA, 2013a. Design Competitions_Guidance for Clients, p. 24.

RIBA, 2013b. RIBA Plan of Work 2013: Consultation document, pp. 1–6.

Rittel, H.W.J., 1973. The State of the Art in Design Methods. Design Research and Methods, 7(2), pp. 143–147.

Rittel, H.W.J. & Webber, M.M., 1973. Dilemmas in a General Theory of Planning. Policy Sciences, 4, pp. 155–169.

Robinson, N.W., 1989. Integrating Multiple Domain Specifications Goals. In Proceedings of the 5th International Workshop on Software Specification and Design (IWSSD'89). ACM, pp. 219–226.

Rodenacker, W.G., 1970. Methodisches Konstruieren (Konstruktiosbucher), Springer Verlag.

Rosenman, M. & Gero, J.S., 1999. Purpose and function in a collaborative CAD environment. Reliability Engineering & System Safety, 64, pp. 167–179.

Rosenman, M. & Gero, J.S., 1998. Purpose and function in design: from the socio-cultural to the techno-physical. Design Studies, (19), pp. 161–186.

Ross, D.T. & Schoman, K.E., 1977. Structured Analysis for Requirements Definition. IEEE Transactions on Software Engineering, 3(1), pp. 6–15.

Ruch, J., 1978. Interactive Space Layout: A Graph Theoretical Approach. In Proceedings of the 15th Design Automation Conference (DAC'78). IEEE Pres, pp. 152–157.

Rykwert, J., 1976. Lodoli on Function and Representation. Architectural Review, CLX(2), pp. 21–26.

Sakao, T. & Lindahl, M., 2012. A value based evaluation method for Product/Service System using design information. CIRP Annals - Manufacturing Technology, 61(1), pp. 51–54.

Scaravetti, D., 2004. Formalisation préalable d'un problème de conception, pour l'aide à la décision en conception préliminaire. Ecole Nationale Supérieure d'Arts et Métiers, CER de Bordeaux.

Séquin, C.H. & Kalay, Y.E., 1998. A suite of prototype CAD tools to support early phases of architectural design. Automation in Construction, (7), pp. 449–464.

Shen, G.Q.P. et al., 2004. A framework for identification and representation of client requirements in the briefing process. Construction Management & Economics, 22(2), pp. 213–221.

Shen, G.Q.P. & Chung, J.K.H., 2006. A critical investigation of the briefing process in Hong Kong's construction industry. Facilities, 24(13/14), pp. 510–522.

Shimomura, Y., Hara, T. & Arai, T., 2009. A unified representation scheme for effective PSS development. CIRP Annals - Manufacturing Technology, 58(1), pp. 379–382.

Simon, H.A., 1969. The Sciences of the Artificial 1st Ed., MIT Press.

Simon, H.A., 1996. The Sciences of the Artificial 3rd Ed., MIT Press.

Simon, H.A., 1973. The structure of ill structured problems. Artificial Intelligence, 4(3-4), pp. 181–201.

Simpson, R.C., 2004. An XML Representation for Crew Procedures, p. 8.

Société APTE, 2007. APTE. Available at: http://www.methode-apte.com/.

Stern, L., 2004. How to Keep Kosher 1st Ed, ed., MorrowCB.

Suh, N.P., 1998. Axiomatic Design Theory for Systems. Research in Engineering Design, 10, pp. 189–209.

Suh, N.P., 2001. Axiomatic design: Advances and applications, Oxford University Press, USA.

Suh, N.P., 1990. The Principle of Design, Oxford University Press.

Sullivan, L.H., 1896. The Tall Office Building Artistically Considered. Lippincott's monthly magazine, pp. 403–409.

Sundin, E., Lindahl, M. & Ijomah, W., 2009. Product design for product/service systems: Design experiences from Swedish industry. Journal of Manufacturing Technology Management, 20(5), pp. 723–753.

SYPAA, 2011. Formations 2011 - Programmation Architecturale et Urbaine, pp. 1–9.

SYPAA, 2005. Syndicat des Programmistes en Architecture et Aménagement. Available at: http://www.sypaa.org/html/orga.html.

Tan, A.R. et al., 2009. Strategies for Designing and Developing Services for Manufacturing Firms. In R. Roy & E. Shehab, eds. Proceedings of the 1st CIRP Industrial Product-Service Systems (IPS2) Conference, 1-2 April. Cranfield University, p. 46.

The Practitioner's Resource, 2014. Programming Basics: An Introduction to an Evidence-Based Design Approach to Architectural Programming. Available at: http://www.practitionersresource.com/CourseDescriptions/11-001.html.

Theyer, R.H. & Dorfman, M., 1990. System and Software Requirements Engineering, IEEE Computer Society Press.

Tjalve, E., 1978. Systematic Design of Industrial Products. Technical University of Denmark, Lyngby.

Tomiyama, T., 1997. A Manufacturing Paradigm Toward the 21st Century. Integrated Computer Aided Engineering, (4), pp. 159–178.

Tomiyama, T. et al., 2009. Design Methodologies: Industrial and Educational Applications. CIRP Annals - Manufacturing Technology, 58(2), pp. 543–565.

Tukker, A., 2004. Eight types of product–service system: eight ways to sustainability? Experiences from SusProNet. Business Strategy and the Environment, (13), pp. 246–260.

Tukker, A. & Tischner, U., 2006. New business for old Europe: product-service development, competitiveness and sustainability, Greenleaf.

Tzortzopoulos, P. et al., 2006. Clients' activities at the design front-end. Design Studies, 27(6), pp. 657–683.

Umeda, Y., Nonomura, A. & Tomiyama, T., 2000. Study on life-cycle design for the post mass production paradigm. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, pp. 149–161.

Umeda, Y. & Tomiyama, T., 1997. Functional reasoning in design. IEEE Expert-Intelligent Systems and their Applications, 12(2), pp. 42–48.

Upitis, A., 2008. Nature Normative - The Design Methods Movement, 1944-1967. Massachussetts Institute of Technology.

Vanneyre, S., 2011. Formation PAMO_Rappel des Interventions et Elements de méthode, p. 66.

Vasantha, G.V.A. et al., 2012. A review of Product–Service Systems Design Methodologies. Journal of Engineering Design, 23(9), pp. 635–659.

Vermaas, P.E., 2011. Accepting ambiguity of engineering functional descriptions. In Proceedings of the 18th International Conference on Engineering Design (ICED11). Copenhagen, Denmark, p. 10.

Vermaas, P.E., 2013. The coexistence of engineering meanings of function: Four responses and their methodological implications. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 27(03), pp. 191–202.

Vermaas, P.E., 2009. The flexible meaning of function in engineering. In Proceedings of the 17th International Conference on Engineering Design (ICED'09). Stanford University, California, United States, pp. 113–124.

Vermaas, P.E. & Dorst, K., 2007. On the conceptual framework of John Gero's FBS-model and the prescriptive aims of design methodology. Design Studies, 28(2), pp. 133–157.

Vernadat, F.B., 1996a. Enterprise Integration: On Business Process and Enterprise Activity Modelling. Concurrent Engineering, 4(3), pp. 219–228.

Vernadat, F.B., 1996b. Enterprise modeling and integration - Principles and application, London, UK: Chapman & Hall.

Vernadat, F.B., 2002. UEML: towards a unified enterprise modelling language. International Journal of Production Research, 40(17), pp. 4309–4321.

Viollet-le-Duc, E.-E., 1863. Entretiens sur l'architecture A. Morel, ed., Paris: Imprimerie de L. Martinet.

Vitruvii, P., 25BC. De architectura libri decem, n.a.

Voordt, D. van der & Wegen, H. van, 2005. Architecture in use: An introduction to the programming, design and evaluation of buildings, Routledge.

VTT, 2008. EcoProP, p. 2.

VTT, 2012. Needscape. Available at: http://cic.vtt.fi/needscape/ [Accessed March 3, 2014].

Wang, L. et al., 2002. Collaborative conceptual design–state of the art and future trends. Computer-Aided Design, 34(13), pp. 981–996.

Warell, A., 1999. Introducing a Use-Perspective in Product Design Theory and Methodology. In Proceedings of the ASME DETC/DTM'99, 12-15 September 1999. Las Vegas, USA.

Weber, C., 2005. CPM/PDD – An Extended Theoretical Approach to Modelling Products and Product development Processes. In H. Bley et al., eds. Proceedings of the 2nd German-Israeli Symposium on Advances in Methods and Systems for Development of Products and Processes. Stuttgart, Germany, pp. 159–179.

Weise, M., Liebich, T. & Wix, J., 2009. Integrating use case definitions for IFC developments. In Zarli & Scherer, eds. eWork and eBusiness in Architecture, Engineering and Construction. Taylor & Francis Group, pp. 637–645.

White, E.T., 1991. Interviews with Architects About Facility Programming, Architectural Media Limited.

White, E.T., 1972. Introduction to Architectural Programming, Tucson: Architectural Media.

White, E.T., 1986. Space adjacency analysis, Architectural Media.

White, S.A., 2004. Business Process Modeling Notation (BPMN), pp. 1–296.

Whitford, F., 2005. Bauhaus World of A., Thames & Hudson Ltd.

Wix, J., 2007. Quick Guide Business Process Modeling Notation (BPMN), p. 14.

Wolstenholme, A., 2009. Never Waste a Good Crisis. Constructing Excellence, p. 32.

Wurzer, G., 2010. Schematic Systems-Constraining Functions Through Processes (and Vice Versa). International Journal of Architectural Computing, 08(02), pp. 197–214.

Wurzer, G., Lorenz, W.E. & Pferzinger, M., 2012. Pre-Tender Hospital Simulation Using Naive Diagrams As Models. In Backfrieder et al., eds. Proceedings of the International Workshop on Innovative Simulation for Health Care. Vienna, Austria, pp. 157–162.

Yahya, I.A. et al., 2007. Value Management in analyzing project brief. In Quantity Surveying International Conference. Kuala Lumpur, Malaysia, p. 12.

Yang, L., Xing, K. & Lee, S.-H., 2010. A new conceptual life cycle model for Result-Oriented Product-Service System development. In Proceedings of the 2010 IEEE International Conference on Service Operations and Logistics, and Informatics. IEEE, pp. 23–28.

Yang, X. et al., 2009. A practical methodology for realizing product service systems for consumer products. Computers & Industrial Engineering, 56(1), pp. 224–235.

Yu, A.T.W., 2007. A value management framework for systematic identification and precise representation of client requirements in the briefing process. The Hong Kong Polytechnic University.

Yu, A.T.W. et al., 2007. An empirical study of the variables affecting construction project briefing/architectural programming. International Journal of Project Management, 25(2), pp. 198–212.

Yu, A.T.W. et al., 2005. Application of value management in project briefing. Facilities, 23(7-8), pp. 330–342 (13).

Yu, A.T.W. et al., 2008. Comparative Study of the Variables in Construction Project Briefing / Architectural Programming. Journal of Construction Engineering and Management, 134(2), pp. 122–138.

Yu, A.T.W. et al., 2006. Investigation of Critical Success Factors in Construction Project Briefing by Way of Content Analysis. Journal of Construction Engineering and Management, (November), pp. 1178–1186.

Yu, E.S.K. et al. eds., 2010. Social Modeling for Requirements Engineering, MIT Press.

Yue, K., 1987. What does it mean to say that a specification is complete? In Proceedings of the 4th International Workshop on Software Specification and Design. Monterrey, USA: IEEE, CS Press.

Zachman, J.A., 1987. A framework for information systems architecture. IBM Systems Journal, 26(3), pp. 276–292.

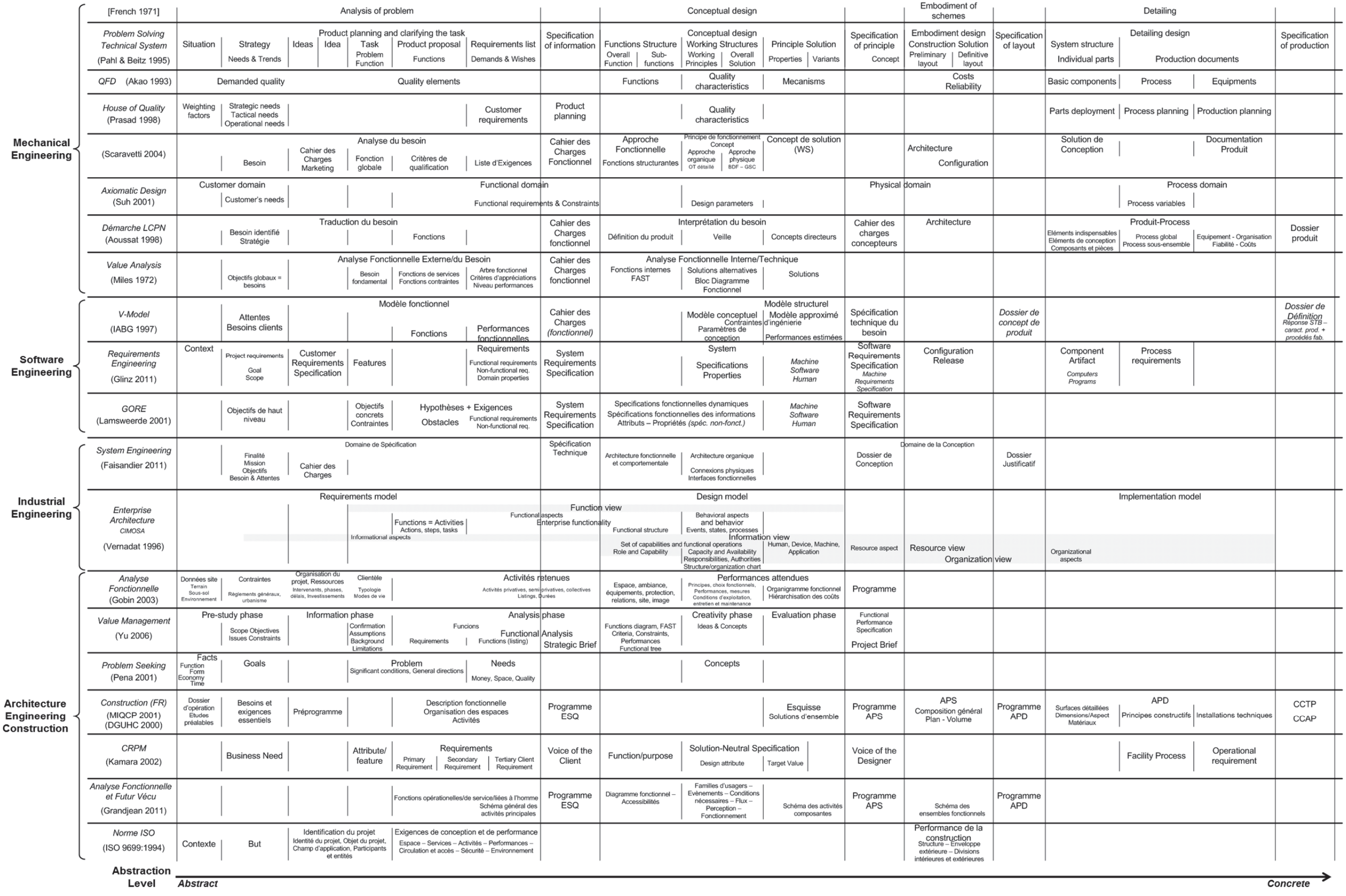
Zaring, O. et al., 2001. Creating eco-efficient producer services, p. 504.

Zave, P. & Jackson, M., 1997. Four dark corners of requirements engineering. ACM Transactions on Software Engineering and Methodology, 6(1), pp. 1–30.

Zetlaoui-Léger, J., 2002. Modalités d'application de démarches programmatiques concertées et participatives pour des projets de proximité, p. 150.

APPENDICES

Appendix A – Concepts used in Various Conceptual Design Methods across Engineering (Figure 13)



Appendix B – Summary of the Proposed Definitions

In this dissertation, a set of definition is proposed across the chapters. This appendix resumes the latest versions to consider.

Definition 1: A **Goal** is a state of affair or condition defined by the clients that the system (i.e. planned facility) must satisfy through cooperation of its components (i.e. sub-systems) and establishes the raison d'être of the system (i.e. answer to the "Why do we need it?" question).

Definition 2: A **Function** is an ability of action, granted by the facility to its clients allowing them to change their environment from a state A to a state B, maintain or avoid a state C, or optimize a state D.

Definition 3: An **Activity** is a unit of work required to perform a function.

Definition 3.1: An **Operation** is an elementary activity.

Definition 3.2: A **Process** is a set of ordered activities.

Definition 4: A **Resource** is any entity (inside or outside the system) required to perform an operation.

Definition 5: A **Space** is the place where activities are performed.

Definition 6: An **Environment** is a specific "as-is" context integrating all the *system*'s surroundings that will impact or be impacted by, directly or not, the *system*'s existence. The surroundings include any environmental factors or external elements (including circumstances, conditions, and knowledge) that exist independently from the system's existence.

Definition 7: An **External Element** is an independent already existing artefact, physical or not, which behaviour or condition will impact or be impacted directly or indirectly by the system.

Definition 8: A **System** is the object of study seen first as a black box representing a collection of structured and organized (inter) dependent components (i.e. sub-systems) either tangible or intangible working together to produce an expected effect on its environment. To be dependent, each component has to be designed, hired, bought, or created in the frame of the system development.

Definition 9: An **Internal Element** is any artefact, physical or not, which existence is totally dependent on the system existence and which contributes to its proper operating.

Definition 10: A **Product** is a tangible or non-tangible resource or space required to perform an activity contributing to the provision of a service.

Definition 11: A **Service** is a human or computed/automated activity, performed by and using a set of tangible as well as non-tangible resources, for other humans or systems providing them value in different ways.

Definition 11.1: A **Core Service** is a customer-oriented service from which the customer will receive the desired value added.

Definition 11.2: A **Support Service** is a product-oriented service or a secondary service required to perform the core service. Its existence is dependent on the existence of the core service or the product.

Definition 12: A **Behaviour** is an activity (action or reaction) or a property of a system (or sub-system) under certain circumstances or triggering events, associated with or derived from its Structure (i.e. its components) explaining the "How and When does/can the system perform a Function?" question.

Definition 13: **Structure** refers to physical and virtual components of the system, their (static) description (i.e. characteristics), and their relationships, describing what it is composed of, more generally answering the "Who (i.e. which resource) has to act in the system to perform a Function when needed, and Where it is performed?" question.

Definition 14: A **Meta-Space** is a virtual ideal space in which all the requirements regarding the proper performance of an elementary activity (i.e. operation) including the associated goals and functions are gathered.

Definition 14.1: An **Operational Meta-Space** is an ideal virtual space associated to an operation (i.e. elementary activity). It is used as an intermediate between the definition of activities and the definition of functional spaces.

Definition 14.2: A **Functional Meta-Space** is an ideal virtual space associated to a functional space and composed of elementary operational meta-spaces. It is used as an intermediate between the textual part of the brief and the graphical functional diagram.

Definition 14.3: A **Design Meta-Space** is an abstract model of a space designed by the architect and represented on a two dimensional drawing.

Appendix C – List of Conceptual Models and their References

Conceptual Model

In Chapter II, a set of conceptual models are used to graphically represent the definitions and taxonomy of each concept. To help the reader, the graphical convention used for the conceptual models is presented in Figure 1.

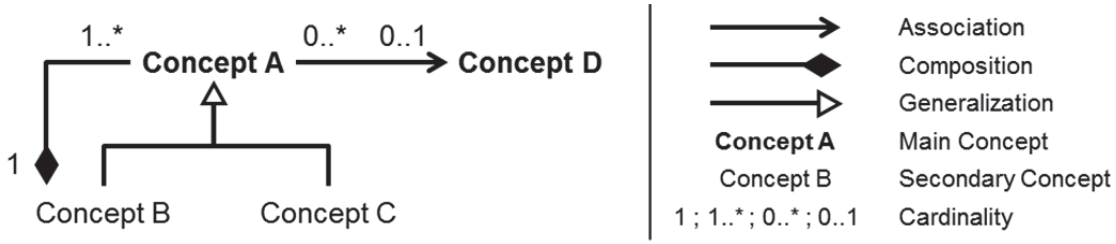


Figure App. C.1 – Graphical Convention used in Chapter II

Figure 1 should be read that way: "*The main concept A is a generalization of concept B and concept C. Respectively, secondary concepts B and C are specializations of the main concept A. Concept B is composed of (at least) one to several concepts A. Therefore, concept B can be composed of concepts B and/or C. The main concept A is optionally linked to maximum one main concept D. The main concept D is optionally linked to several main concepts A*".

Internal References:

- Chapter II p.25
- Chapter III p.55; 59; 62; 67; 68; 86;90; 91-92

Goal Models

Tropos

External References:

- AOSE - The Tropos Methodology.
- Giunchiglia, F., Mylopoulos, J., and Perini, A., 2001. The Tropos Software Development Methodology: Processes, Models and Diagrams.
- Giorgini, P., Kolp, M., Mylopoulos, J., and Pistore, M., 2003. The Tropos Methodology - An Overview, in Methodologies and Software Engineering for Agent Systems, pp. 20.
- Castro, J., 2007. Goal-Oriented Requirements Engineering - The TROPOS Case.
- Giorgini, P., 2008. TROPOS : an Agent-Oriented Methodology, pp. 56.

Internal References:

- Chapter III Section 4.1 p.71

*i**

External References:

- Werneck, V.M.B., Oliveira, A. de P.A., and Leite, J.C.S. do P., 2009. Comparing GORE Frameworks: i-star and KAOS, in Ibero-American Workshop of Engineering of Requirements, pp. 12.
- Cares, C., and Franch, X., 2011. A Metamodelling Approach for i* Model Translations, in Mouratidis, H. and Rolland, C. eds., CAiSE 2011, LNCS 6741, Springer Verlag, pp. 337–351.

Internal References:

- Chapter III Section 4.1 p.71

KAOS

External References:

- Matulevičius, R., and Heymans, P., 2005. Analysis of KAOS Meta-model.
- Matulevičius, R., Heymans, P., and Opdahl, A.L., 2006. Ontological Analysis of KAOS Using Separation of Reference, in Proceeedings of the EMMSAD 2006 Conference, Luxembourg, pp. 12.
- Respect-IT, 2007. A KAOS Tutorial, pp. 1–46.

Internal References:

- Chapter III Section 4.1 p.71

Function Models

Interaction Diagram

External References:

- Société APTE, 2007. APTE.
- De la Bretesche, B., 2000. La méthode APTE Analyse de la Valeur Analyse Fonctionnelle: Pétrelle, Paris, France.

Internal References:

- Chapter III Section 4.2 p.72-73
- Chapter IV Section 2.3 p.110-111)

Function-Mean Tree (FMT)

External References:

- Tjalve, E., 1978. Systematic Design of Industrial Products: Technical University of Denmark, Lyngby.
- Buur, J., 1990. A Theoretical Approach to Mechatronics Design: Technical University of Denmark.
- Bracewell, R., and Sharpe, J., 1996. Functional descriptions used in computer support for qualitative scheme generation-'Schemebuilder'. Ai Edam, Vol. 10, No. 04, pp. 333–345.

Internal References:

- Chapter III Section 4.2 p.73

FAST

External References:

- AFNOR, 2000. NF EN 12973-2000: Management par la Valeur, pp. 56.
- Charpentier, F., 2005. Analyse fonctionnelle. Quels outils ? Technologie et formation, Vol. 35, No. 117-118, pp. 10–15 ; 24–35.

Function Block Diagram (FBD):

External References:

- Maussang, N., 2008. Méthodologie de conception des systèmes produits-services: Institut polytechnique de Grenoble.

Behaviour Models

IDEF

External References:

- Ross, D.T., and Schoman, K.E., 1977. Structured Analysis for Requirements Definition. IEEE Transactions on Software Engineering, Vol. 3, No. 1, pp. 6–15.
- National Institute of Standards and Technology, 1993. Integration Definition for Function Modeling (IDEF0).

- Benjamin, P.C., Menzel, C.P., Mayer, R.J., Fillion, F., Futrell, M.T., DeWitte, P.S., and Lingineni, M., 1994. IDEF5 Method Report.
- Mayer, R.J., Menzel, C.P., Painter, M.K., DeWitte, P.S., Blinn, Y., and Perakath, B., 1995. Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report.
- Mayer, R.J., Crump, J.W., Fernandes, R., Keen, A., and Painter, M.K., 1995. Information Integration for Concurrent Engineering (IICE) - Compendium of Methods Report.

Internal References:

- Chapter III Section 4.3.1 p.75
- Chapter III Section 6.2 p.94
- Chapter IV Section 2.5 p.115
- Chapter IV Section 3.2 p.126-128

Business Process Modelling Notation (BPMN)

External References:

- OMG, 2011a. Business Process Model and Notation (BPMN) - v2.0.
- White, S.A., 2004. Business Process Modeling Notation (BPMN).

Internal References:

- Chapter III Section 4.3.2 p.76
- Chapter III Section 6.2 p.95
- Chapter IV Section 1 p.102
- Chapter IV Section 2.6 p.116
- Chapter IV Section 3.1 p.121-125
- Chapter IV Section 4 p. 131-135

Unified Modelling Languages (Behavioural Diagrams)

External References:

- Booch, G., Rumbaugh, J., and Jacobson, I., 1998. The Unified Modeling Language User Guide: Addison Wesley.
- OMG, 2011b. OMG Unified Modeling Language TM (OMG UML), Superstructure v2.4.1.

Internal References:

- Chapter III Section 4.3.3 p.76-77
- Chapter IV Section 2.5 p.113-114
- Chapter IV Section 3.3 p.130

Structure Models

Unified Modelling Languages (Structural Diagrams)

External References:

- Booch, G., Rumbaugh, J., and Jacobson, I., 1998. The Unified Modeling Language User Guide: Addison Wesley.
- OMG, 2011b. OMG Unified Modeling Language TM (OMG UML), Superstructure v2.4.1.

Internal References:

- Chapter III Section 4.4.1 p.78
- Chapter IV Section 2.5 p.114

Entity-Relationship Diagram

External References:

- Chen, P.P.-C., 1976. The Entity-Relationship Unified View of Data Model - Toward a Unified View of Data. ACM Transactions on Database Systems, Vol. 1, No. 1, pp. 9–36.

Internal References:

- Chapter III Section 4.4.2 p.79

Product Breakdown Structure

External References:

- http://en.wikipedia.org/wiki/Product_breakdown_structure

Internal References:

- Chapter III Section 4.4.3 p.80

Functional Diagram

External References:

- Certu, 2006. FICHE 26 : Le schéma fonctionnel.
- Certu, 2010. Pour des bâtiments durables - Guide et outils de programmation: La Documentation Française.
- Vanneyre, S., 2011. Formation PAMO_Rappel des Interventions et Elements de méthode.
- Mauger, C., and Kubicki, S., 2013. A Conceptual Model for Building Requirements Processing, in Proc. of the 11th IPGRC, University of Salford, UK, pp. 318–330.

Internal References:

- Chapter III Section 4.4.5 p.80
- Chapter III Section 6.3 p.96
- Chapter IV Section 2.7 p.118-119
- Chapter IV Section 4.2.1 p.136-137

Adjacency Matrix

Internal References:

- Chapter III Section 4.4.6 p.83

Graph

External References:

- Euler, L., 1736. Solutio problematis ad geometriam situs pertinentis. Commentarii academiae scientiarum Petropolitanae, Vol. 8, pp. 128–140.

Internal References:

- Chapter III Section 4.4.6 p.83

Bubble Diagram

External References:

- Ruch, J., 1978. Interactive Space Layout: A Graph Theoretical Approach, in DAC'78 Proc. of the 15th Design Automation Conference, pp. 152–157.
- Lawson, B., 2005. How designer think: Elsevier.
- Wurzer, G., Lorenz, W.E., and Pferzinger, M., 2012. Pre-Tender Hospital Simulation Using Naive Diagrams As Models, in Backfrieder, Bruzzone, Longo, Novak, and Rosen eds., Proceedings of the International Workshop on Innovative Simulation for Health Care,, pp. 157–162.

Internal References:

- Chapter III Section 4.4.6 p.83-84

Venn Diagram

External References:

- Chilakamarri, K., Hamburger, P., and Pippert, R., 1996. Venn diagrams and planar graphs. Geometriae Dedicata, Vol. 62, No. 1, pp. 73–91.

Internal References:

- Chapter III Section 4.4.6 p.83-84

Schematic Plan

Internal References:

- Chapter III Section 4.4.1 p.78

Zoning Diagram

External References:

- White, E.T., 1986. Space adjacency analysis: Architectural Media.

Internal References:

- Chapter III Section 4.4.6 p.84

Industrial Foundation Classes

External References:

- buildingSMART, 2013. IFC4 Official Release.

Internal References:

- Chapter II Section 3.4.1 p.42
- Chapter II Section 3.5.3 p.46

Meta-Space Diagram

External References:

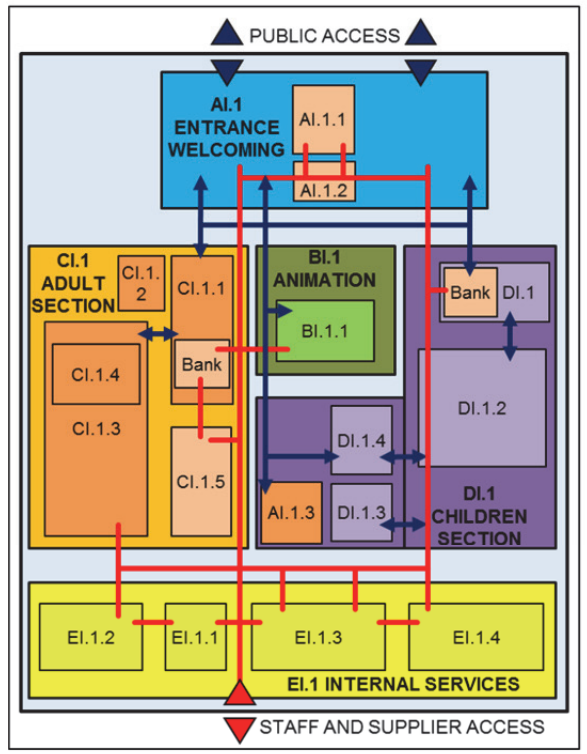
- Mauger, C., 2013. A Design Artifact for Functional Assessment of Construction Projects, in Hadju, M. and Skibniewski, M. eds., Procs Creative Construction Conference 2013, 6-9 July 2013, Budapest, Hungary,, pp. 505–515.
- Mauger, C., and Kubicki, S., 2013. A Conceptual Model for Building Requirements Processing, in Proc. of the 11th IPGRC, University of Salford, UK, pp. 318–330.

Internal References:

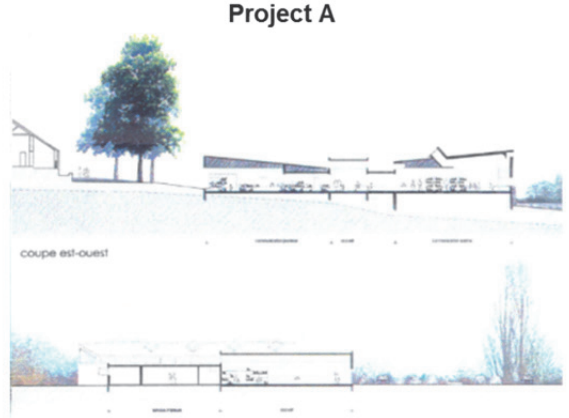
- Chapter III Section 6 p.92-98
- Chapter IV Section 2.6 p.116
- Chapter IV Section 2.7 p.119
- Chapter IV Section 4 p.133-138

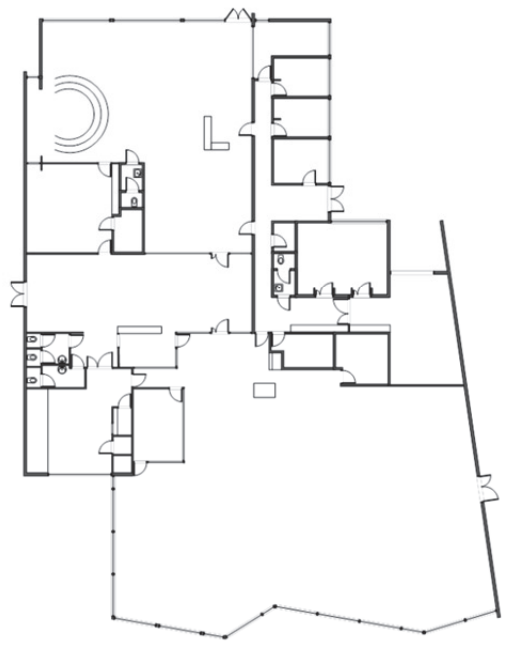
Appendix D – One Functional Diagram, Three Design Proposals (Figure 70)

Functional Diagram

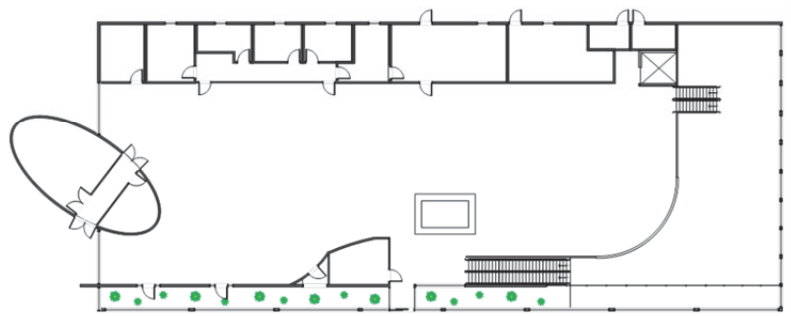


Project A

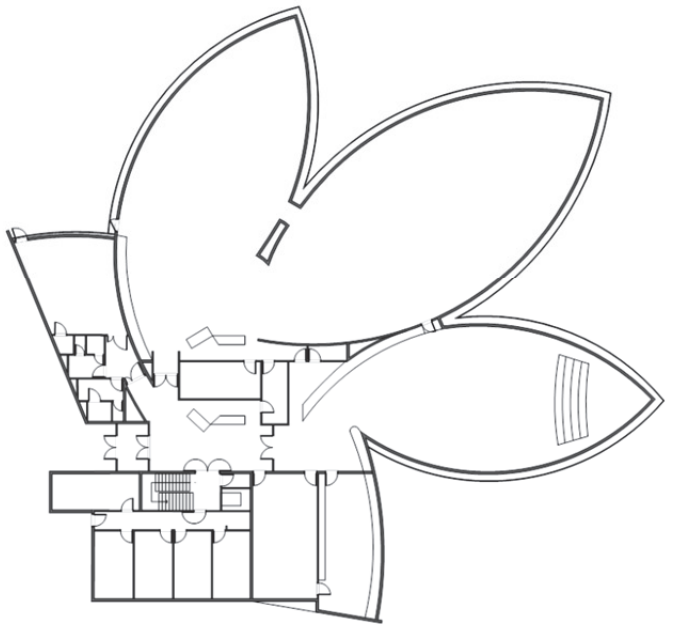




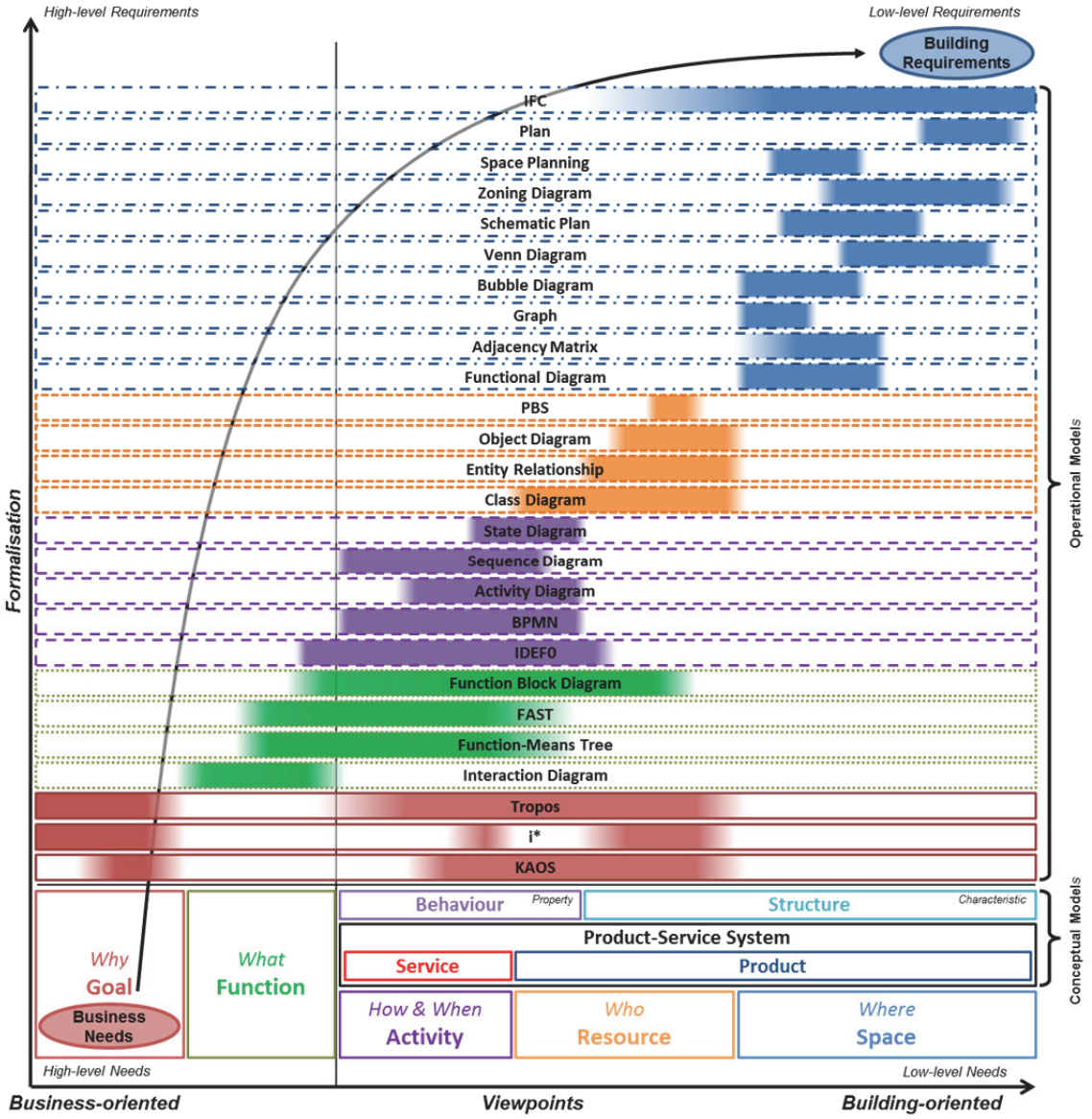
Project B



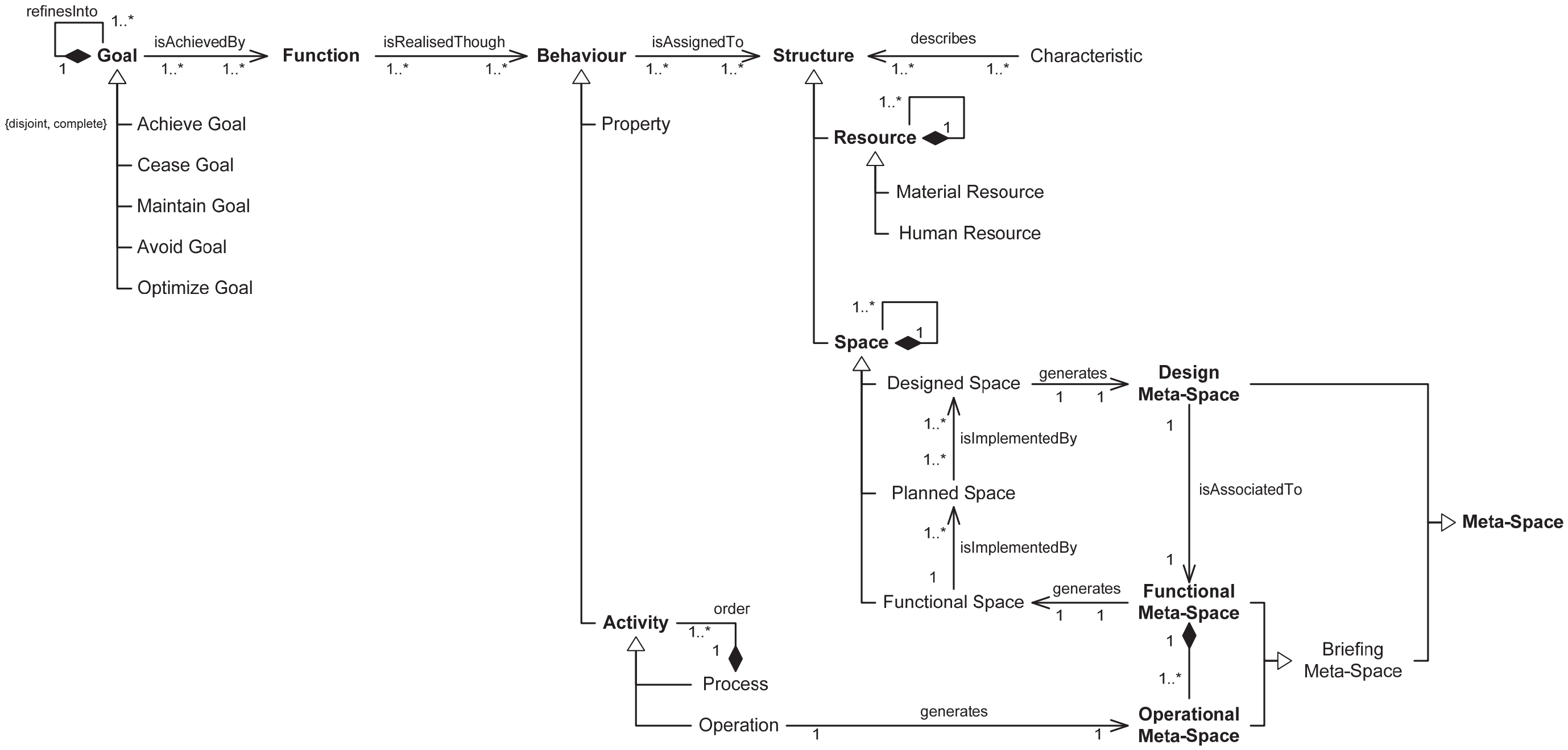
Project C



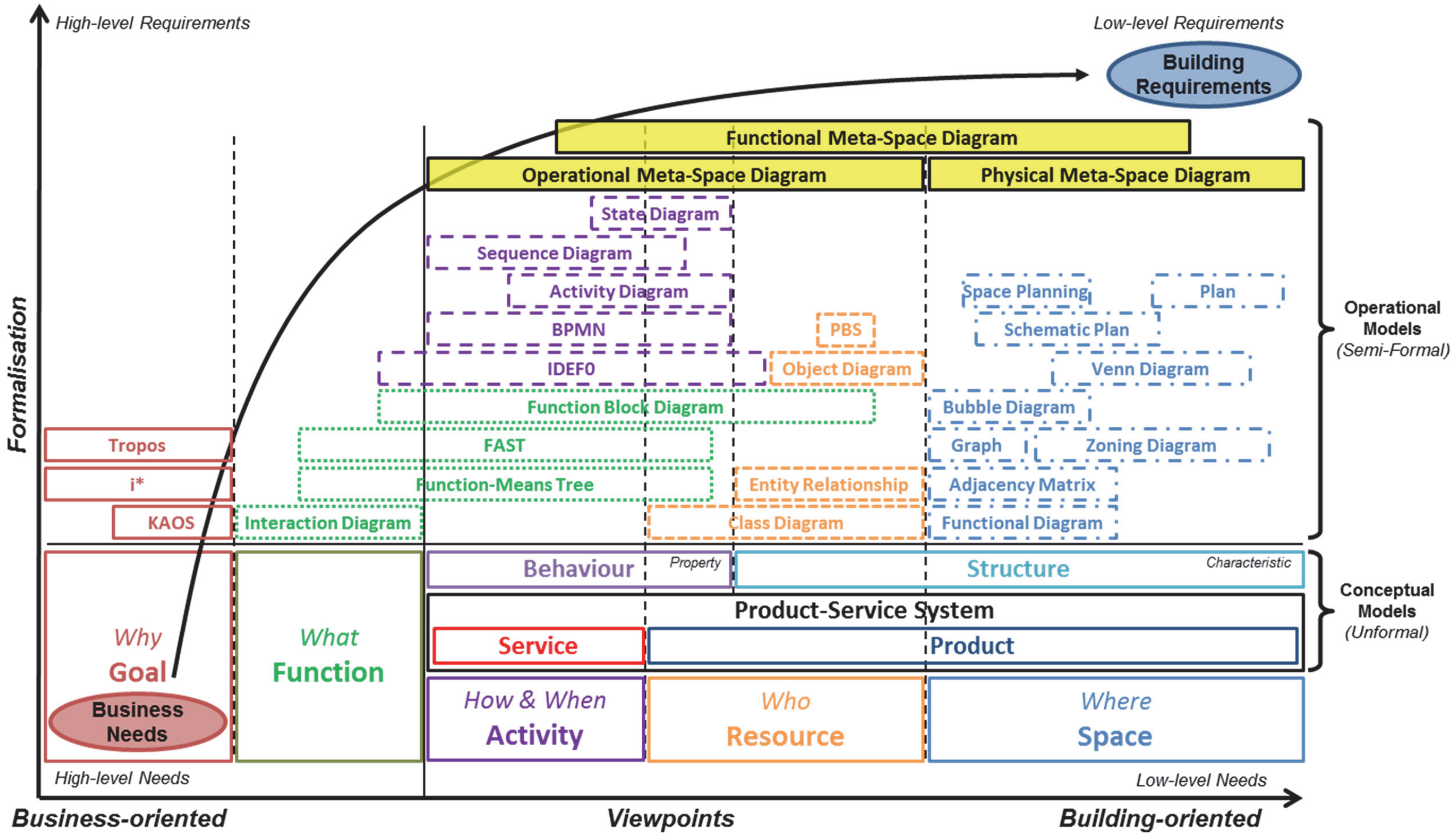
Appendix E – Operational Models Coverage of the Definition Domain (Figure 77)



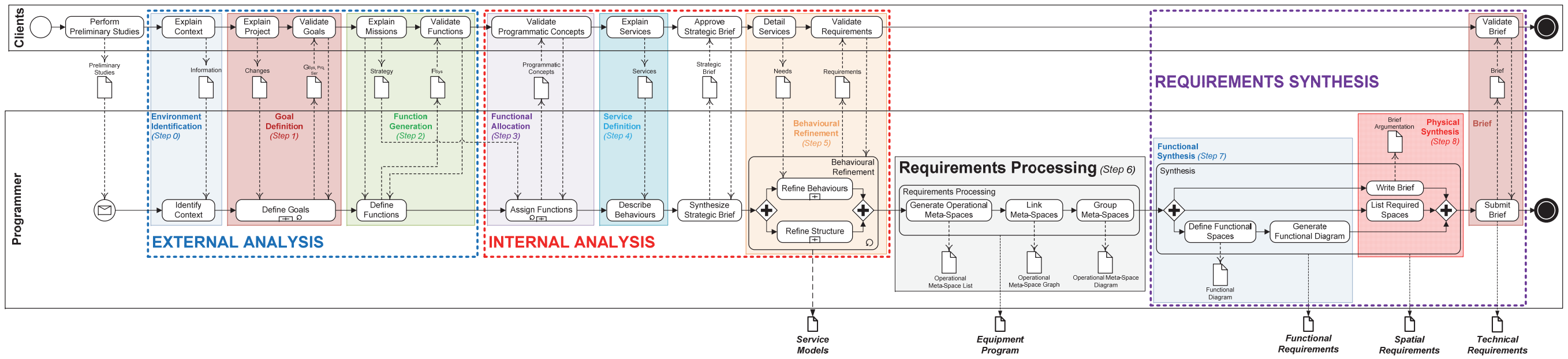
Appendix F – Resulting Conceptual Model of the Building Definition Domain (Restructured Figure 86)



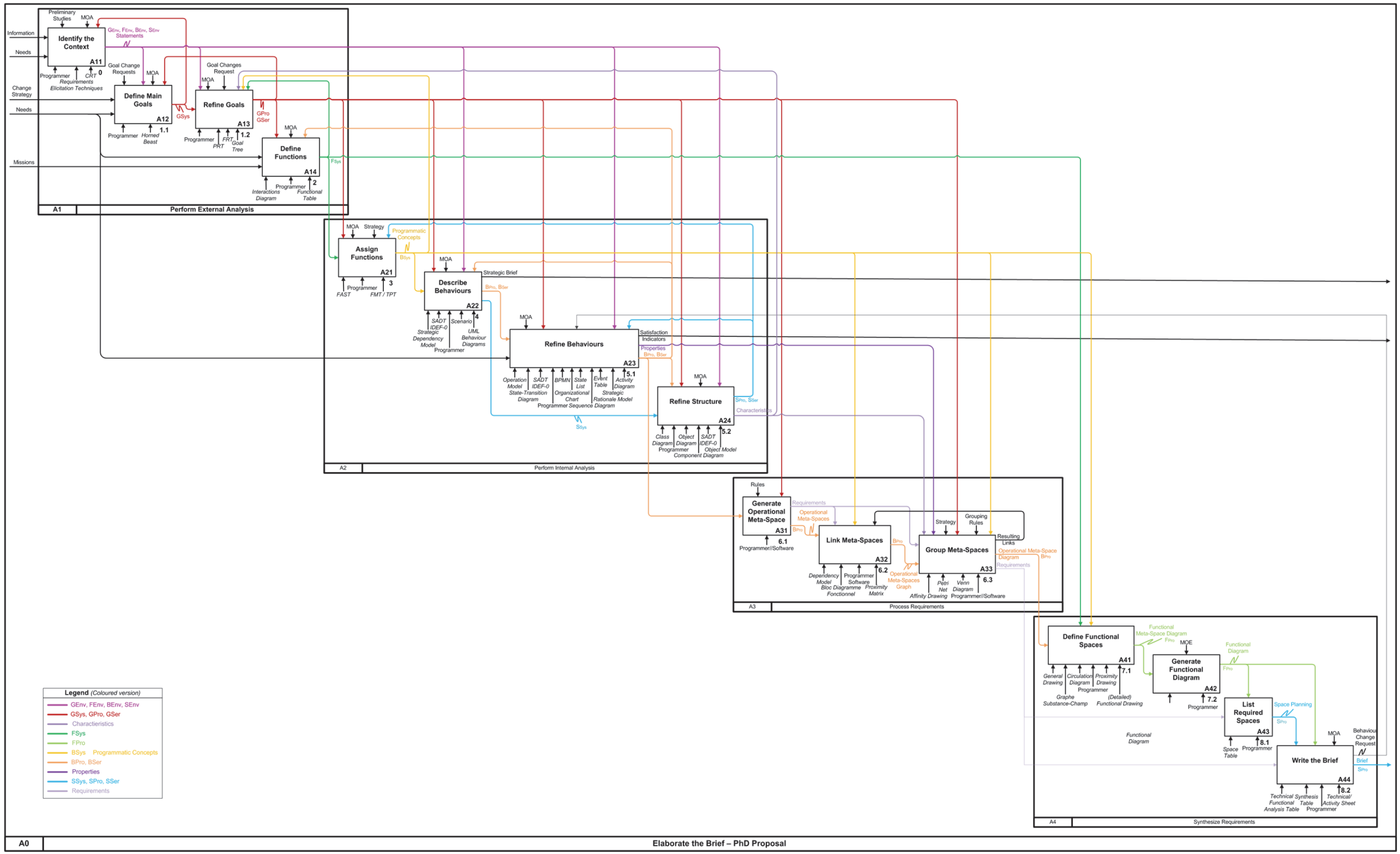
Appendix G – Meta-Space Diagrams Coverage of the Definition Domain (Figure 98)



Appendix H – BPMN process model view of the GFBS briefing (Figure 130)



Appendix I – Detailed IDEF0 Process Model View of the GFBS Framework



Appendix J – University Library of Luxembourg Case Study: Allocation of Book Collections in the Future Building

Introduction

This case study is based on the first results drafted by Miss Hélène LY during her Master thesis performed at Arts et Métiers ParisTech under the supervision of Dr Ali SIADAT and Mr Cyril MAUGER. For the reader information, the context of the case study can be found in Chapter I Section 4.4.4. Information and data about the "Maison du Livre" and the University Library were kindly provided by Mrs Marie-Pierre PAUSCH-ANTOINE and Mrs Julie WILLEMS.

Basic Data and Units

Various data and information are collected, calculated, and deducted in this case study. Table I resumes the conventions taken across the presented tables along this case study presentation.

Table App. J.1 – Data Writing Conventions

Convention
Fixed data
Hypothesis
Formula
Total (sum)
Important if coloured

The measure of space in a library project can be done through several units, depending on the stakeholder involved. Table II sums up the standardized ratio and units of measure based on (Bisbrouck & Renoult 1993). The hypothesis numbers are updated with the requirements specification given by Mme The units used in this document are limited to linear meters (ml), number of documents (Doc), and number of shelves (Shlv).

Table App. J.2 – Ratio, Units, and Basic Data

Free Access Documentation		
Large Free Access	3.5	ml/m²
Number of Documents per linear meter	35	Doc/ml
Surface area for 10,000 documents	107	m²
Number of Documents per square meter	123	Doc/m²
Number of Tablets per Shelve	5	U
Length of shelves	0.9	m

Dimensioning the Collection

The university library opts for using the Dewey Decimal Classification (DDC) as a library classification system in the new university library. The 2013 accessible collection (documents stored in the free access documentation area) counts around 142,000 documents. Among these documents, around 23,000 documents are not classified. It has been decided that these documents will be directly stored in the storehouse.

The first issue to notice concerns the fact that Social Sciences represent 40% of the collection. As a result, this section is refined into its sub-sections to facilitate its repartition in the new building. Table III presents the resulting AS-IS collection used as a basis for the case study.

Table App. J.3 – Repartition of the 2013 Collection based on DDC

		2013 Collection			
Dewey Decimal Classification		**Doc**	**%**	**ml**	**Shlv**
General works	000	3,755	3.2%	107.3	**24**
Philosophy and psychology	100	7,917	6.7%	226.2	**51**
Religion	200	1,045	0.9%	29.9	**7**
Social sciences	300	7,125	6.0%	203.6	**46**
General statistics	310	78	0.1%	2.3	**1**
Political science	320	2,719	2.3%	77.7	**18**
Economics	330	8,438	7.1%	241.1	**54**
Law	340	11,575	9.8%	330.8	**74**
Public administration	350	267	0.2%	7.7	**2**
Social services; association	360	2,235	1.9%	63.9	**15**
Education	370	24,302	20.5%	694.4	**155**
Commerce, communications, transport	380	557	0.5%	16.0	**4**
Customs, etiquette, folklore	390	161	0.1%	4.6	**2**
Language	400	5,578	4.7%	159.4	**36**
Pure Science	500	6,232	5.3%	178.1	**40**
Technology	600	7,074	6.0%	202.2	**45**
Arts & recreation	700	3,189	2.7%	91.2	**21**
Literature	800	19,555	16.5%	558.8	**125**
History & geography	900	6,883	5.8%	196.7	**44**
		118,685	**100%**	**3,391.9**	**911**

This 2013 collection does not include the periodicals. The periodicals will be stored with the other documents of the same section. To consider them, an average of 10% document is added to the current collection.

Regarding its operating, the university library considers a 10% growing rate (%GR) of the collection each year. Considering the new building, the university library wishes to avoid the moving to the storehouse of the collection during the first 5 years of operating.

Table App. J.4 – Collection to divide in the new building

	2013 Collection				%GR	Period.	Total		
DDC	**Doc**	**%**	**ml**	**Shlv**	**%**	**%**	**Doc**	**ml**	**Shlv**
000	3,755	3.2%	107.3	**24**	20%	10%	**4,882**	**140**	**31**
100	7,917	6.7%	226.2	**51**	20%	10%	**10,293**	**294**	**66**
200	1,045	0.9%	29.9	**7**	20%	10%	**1,359**	**39**	**9**
300	7,125	6.0%	203.6	**46**	20%	10%	**9,263**	**265**	**59**
310	78	0.1%	2.3	**1**	20%	10%	**102**	**3**	**1**
320	2,719	2.3%	77.7	**18**	20%	10%	**3,535**	**101**	**23**
330	8,438	7.1%	241.1	**54**	20%	10%	**10,970**	**314**	**70**
340	11,575	9.8%	330.8	**74**	20%	10%	**15,048**	**430**	**96**
350	267	0.2%	7.7	**2**	20%	10%	**348**	**10**	**3**
360	2,235	1.9%	63.9	**15**	20%	10%	**2,906**	**83**	**19**
370	24,302	20.5%	694.4	**155**	20%	10%	**31,594**	**903**	**201**
380	557	0.5%	16.0	**4**	20%	10%	**725**	**21**	**5**
390	161	0.1%	4.6	**2**	20%	10%	**211**	**6**	**2**
400	5,578	4.7%	159.4	**36**	20%	10%	**7,252**	**207**	**47**
500	6,232	5.3%	178.1	**40**	20%	10%	**8,103**	**232**	**52**
600	7,074	6.0%	202.2	**45**	20%	10%	**9,197**	**263**	**59**
700	3,189	2.7%	91.2	**21**	20%	10%	**4,146**	**119**	**27**
800	19,555	16.5%	558.8	**125**	20%	10%	**25,422**	**727**	**162**
900	6,883	5.8%	196.7	**44**	20%	10%	**8,949**	**256**	**57**
Total	**118,685**	**100%**	**3,391.9**	**911**			**154,305**	**4,411.3**	**989**

Based on first iterations of allocation, only a maximum of 20% growing rate (5% per year or 10% over 2 years) can be considered to allow a consistent allocation of the collection in the new building. The main issue concerns the storage of the Education section (370) which is particularly

important in quantity and constraint the most the allocation. Table IV presents the resulting collection to divide based on the required adjustment.

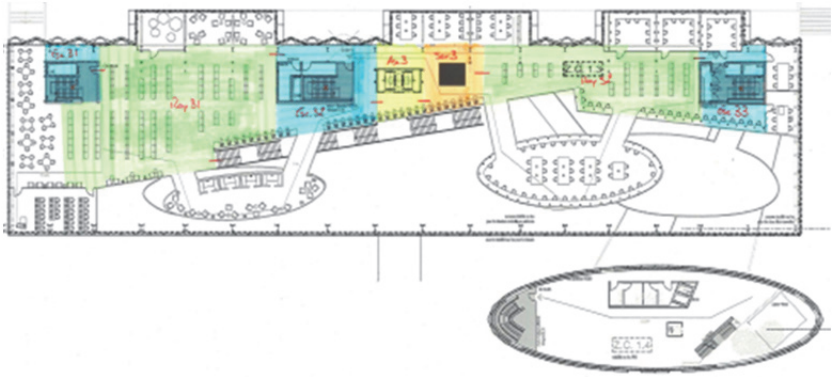


Figure App. J.1 – 3rd Floor Plan of the new building

Based on the architects' design, a consulting company proposes a layout organisation validated by the university library. The proposed layout defines the constraints in terms of available storage in the new building. Figure 1 presents the 3rd floor of the new building including the layout of the furniture. The storage spaces are represented in green. Each floor has storage spaces on the left side and right side of the elevator (in yellow).

Table V presents the number of shelves available per floor with a repartition between the left and right wings of the building. It has to be noticed that the total number of documents to store in the new building represents around 96% of the available storage space. Only 4% are unallocated which confirm the impossibility to reach the 10% growing rate per year over 5 years asked at first.

Table App. J.5 – Number of storage shelves per floor in the new building

	Shlv			Total	
Floor	Left Wing	Right Wing	Total	ml	Doc
3	216	76	292	1,314	45,990
2	174	48	222	999	34,965
1	154	104	258	1,161	40,635
0	0	0	0	0	0
-1	0	0	0	0	0
-2	84	162	246	1,107	38,745
Total	628	390	1,018	4,581	160,335

The direct allocation of the collection from 000 to 900 following the floors and wings is not possible without exploding every section among 2 to 3 different floors. As a result, a grouping of the sections and sub-sections is required (Table IV). The groupings are based on the librarian knowledge about the disciplines tough at the university and relationships across sub-sections of the DDC.

Table App. J.6 – DDC section grouping into thematic hubs

Hubs		DDC	Total Shlv
Sciences, Technology & IT	1	000 + 500 + 600	142
Language & Literature	2	400 + 800	209
Economics, Legal & Social Sciences	3	300 + 320 + 330 + 340 + 350 + 380 + 390	258
Education & Psychology	4	100 + 310 + 360 + 370	287
Arts, History & Geography	5	200 + 700 + 900	93
			989

Based on the thematic hubs, the collection is divided among the various possible storage places in the new building. The objective is to keep each hub on a single floor as much as

possible. Only the sub-section Social Services and Association (360) has to be split between the two wings of the third floor. The resulting allocation is presented in Table VII.

Table App. J.7 – Resulting allocation of the collection

		Number of Shelves per floor (left & right)										
		Floor -2		Floor -1	Floor 0	Floor 1		Floor 2		Floor 3		Total
DDC	Shlv	84	162	0	0	154	104	174	48	216	76	1,018
000	31		100%									100%
100	66										100%	100%
200	9		100%									100%
300	59					100%						100%
310	1									100%		100%
320	23					100%						100%
330	70					100%						100%
340	96						100%					100%
350	3						100%					100%
360	19									50%	50%	100%
370	201									100%		100%
380	5						100%					100%
390	2					100%						100%
400	47								100%			100%
500	52		100%									100%
600	59		100%									100%
700	27	100%										100%
800	162							100%				100%
900	57	100%										100%
	989	0	11	0	0	0	0	12	1	5	1	29.0

All the tables can be found on an Excel sheet that gathers the data in the first worksheet. The second worksheet proposes an abstraction of the building that only represents the shelves and their detailed repartition floor by floor. The allocation of the collection is illustrated on this abstraction to support the representation of circulation issues and practical distribution of collections (Figure 2).

Each shelf is numbered by floor and by wing. White shelves represent unallocated shelves. They can be used to increase the growing rate for specific section (e.g. Literature could dispose of 12 extra shelves on the 2nd floor's left wing which would increase its growing rate from 20% up to 30%).

As a conclusion, we see that the layout and dimensioning of the university library were not performed at the same time or using the right level of detailed information. The Dewey Decimal Classification (DDC) cannot be perfectly followed in the building to structure the collection from bottom-up. As a result, the signage system will be of great importance in the future building to ensure the efficient orientation of the students and public in the building. Indeed, when a student will look for General Statistics (310), he will have to go on the 3rd floor to find it instead of looking between Social Sciences (300) and Political Science (320) both located on the 1st floor.

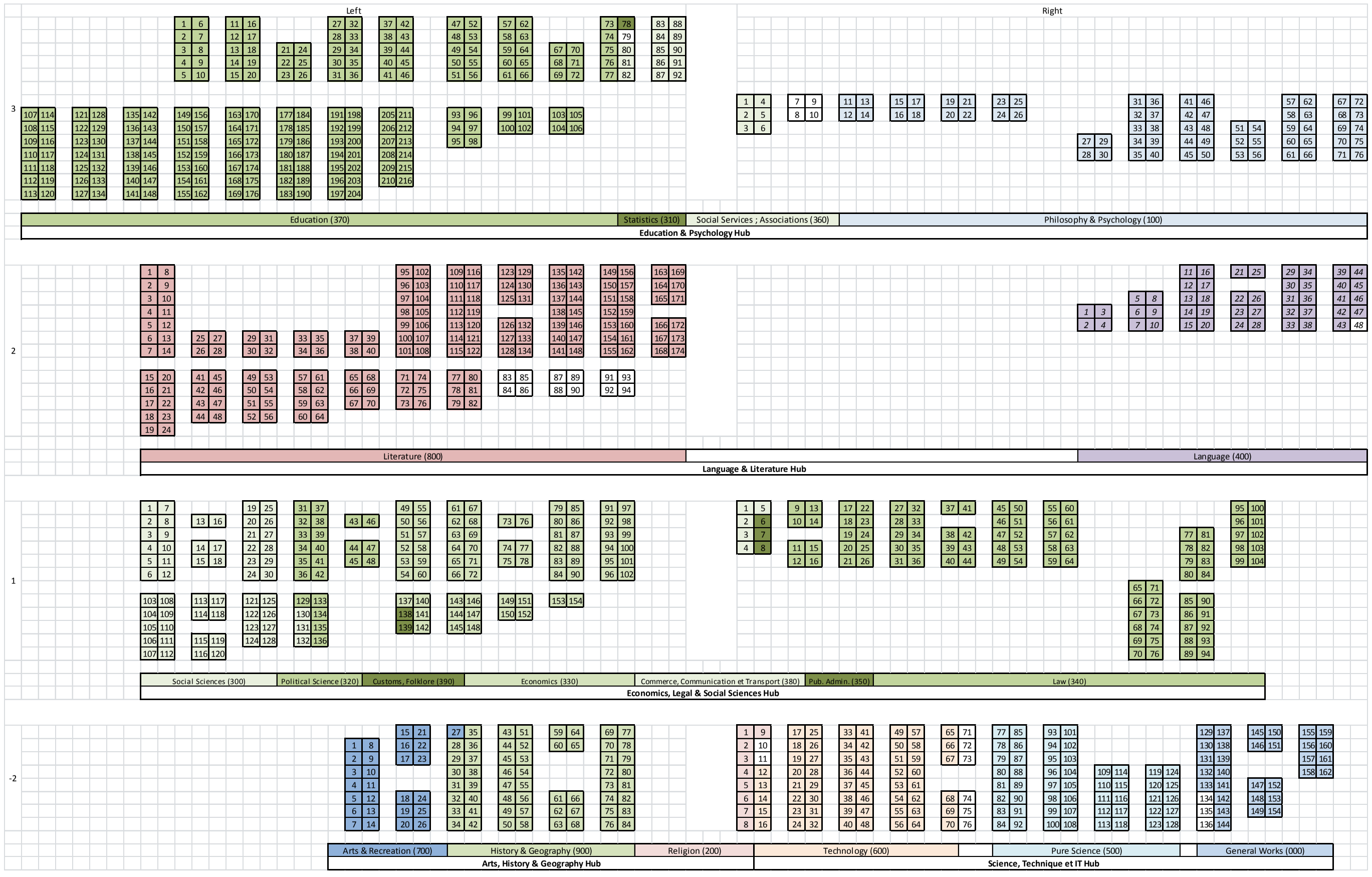


Figure App. J.2 – DDC Allocation represented on an abstraction of the new building

Appendix K – Synagogue Kitchen Case Study

Introduction

This case study is based on an interaction between a Requirements Engineering expert, Pr. Daniel Berry from the University of Waterloo (Canada), and a Canadian Rabbi about the refurbishment of a kitchen in a synagogue. Previous papers were published by Pr. Berry about the relationship between Requirements Engineering and Architecture-Engineering-Construction domains (Berry 2000; Berry 2002; Berry 2003). Pr. Berry kindly shared his thoughts and information about this case as an input for this PhD work. Details about the story behind this case study can be found in the following conference paper: (Mauger & Berry 2014). For the reader information, the context of this case study is already introduced in Chapter I Section 4.4.3.

Method

This appendix aims to present a detailed application of the engineering models on an AEC project (steps 5 of the GFBS briefing framework). The case study is led by the PhD candidate without any direct interaction with the Rabbi and only based on information extracted from the architect's proposal plan for the kitchen, and sketches and use cases provided by Pr. Berry.

The interest of this case study is to show how detailed information can be modelled to provide different viewpoints on the same space. The modelling languages used in this case study are UML and IDEF. Both are well-established modelling languages in the Software Engineering domain. UML counts several diagrams that can be used to model this case study. In this appendix, the comparison between UML and IDEF is performed using use case diagram, class diagram (entity-relationship diagram) and activity diagram from UML on one side, and IDEF0 on the other side.

Based on a set of basic use cases, additional information is generated from the creation of the other diagrams which completes each other. The order of creation is the following: use case diagram, activity diagram, class diagram (entity-relationship), and IDEF0. The creation of the UML and IDEF0 models is performed following the step 5 of the GFBS briefing framework introduced in the Chapter IV Section 2.5.

The assessment of the architect's proposal compared to Pr. Berry's proposal is already presented in a conference poster limited to a flow analysis of the layout (Mauger & Berry 2013) and completed later by a two-stage survey in a conference paper (Mauger & Berry 2014).

Usage Description of a Kitchen

In this section, we provide examples of text that could be found in an architectural brief that describe a kitchen usage. These examples are however more detailed to the benefit of the illustration. The provided extra details are mostly implicit knowledge about kitchens. The identification of these extra details is left to the own appreciation of the reader (as it depends on his/her experience with kitchens). The other sections are based on an extensive description of kitchen usages. These usages were defined during the case study presented in (Mauger & Berry 2013) and completed by a short literature review (Frederick 1923):

"The kitchen is used to feed between 15 and 20 people for dinner every day and lunch on week-ends. The cooker does the groceries twice a week. He usually buys various kinds of things such as food supplies, drinks, and from time to time utensils. Fresh food is stored in a fridge. Frozen food (or to be frozen) is stored in a freezer. The rest of the food is stored in a pantry. Utensils are stored in cupboards. To ease the groceries storing and not to put groceries on the floor, the cooker would like to have countertops close to storage equipment. Moreover, all utensils should be grouped at the place where they are used."

This first example contains generic information about kitchens. It can be applied on any kind of kitchen with slight adjustments on quantities and habits. The second example provides information specific to synagogue kitchens:

"The synagogue kitchen provides a support for Jewish believers to keep kosher, more particularly during Shabbat (i.e. from Friday 18 minutes before sunset till Saturday 40 minutes after sunset). Keeping kosher involves specific requirements. Firstly, only kosher food can be brought in the kitchen (i.e. fresh fruits and vegetables and certified kosher meats, fish, and packaged food). Secondly, dairy and meat products are never mixed during cooking, eating, and storing leftovers in containers. This strict separation is extended to any utensil, instrument, or dish which came in contact with the food. To be able to use a spoon with both, it is necessary to boil it (i.e. sterilise it from the previous food). Moreover, wood utensils and countertops are excluded (as we cannot be sure of their cleaning)."

For a Rabbi, all of these requirements can be considered as obvious. They may not be naturally provided to the architect at first, assuming that somehow the architect is familiar with keeping kosher. Unfortunately, the architect is often unaware of such things and these requirements stay at an implicit state (Mauger & Berry 2014).

The amount of text required to describe the kitchen usage can be quite extensive to cover all the details and to make explicit all the requirements. In Frederick's book, more than forty pages are proposed to describe a generic kitchen usage back in the 1920s (including utensils and surfaces dimensioning). With today's new equipment, we can barely imagine how many pages it would require to do the same for the 2010s kitchen. Regarding the more specific case of synagogue kitchens, Stern proposes a 300 pages book on "how to keep kosher" (Stern 2004). As a result, the architect ends on drowning in a huge quantity of more or less structured information (and we are only talking about a kitchen). In this appendix, we spare the readers' time by only providing the models that synthesise a fraction of this information through different viewpoints.

Models

The first part focuses on models applied during the 5th step of the GFBS Briefing Process: the behavioural refinement (Figure App. K.1). This step is introduced in Chapter IV Section 2.5 of the dissertation. Two models are used in this appendix: UML and IDEF0. Both models provide different viewpoints and cover different kinds of information. Compared to the dissertation, this appendix presents a consequent amount of use cases used to describe kitchen usages. The list of use cases is far from being exhaustive but represents a decent amount of information to illustrate the value added of engineering models and meta-space artefacts.

Figure App. K.1 – Step 5: Behavioural Refinement

The first one presented is the UML limited to use case diagram, activity diagram, and class diagram (entity-relationship diagram in reality). The second one is the IDEF0. The models were developed in that order based on the same information.

Use Case Diagram – The use case diagram gathers 23 use cases performed by the cooker in the kitchen (Figure App. K.2). Two levels of details can be identified among the 23 use cases. The first level concerns 9 generic use cases. The second level expands them into 21 specific instances. The specific instances represent variants of the generic use cases. Among the instances, only the difference between meat and dairy meals is presented regarding keeping kosher and the synagogue kitchen.

The reading of each use case is the following: "*A – The cooker put away fresh food in a refrigerator*". "*Cooker*" is a position (S_{Ser}) assigned to a human actor (S_{Pro}). "*Put away*" is an activity (B_{Ser}) performed by the "*Cooker*". "*Fresh*" is a state or property (B_{Pro}) of the food. "*Food*" is a kind of goods and "*Refrigerator*" a kind of equipment (S_{Pro}).

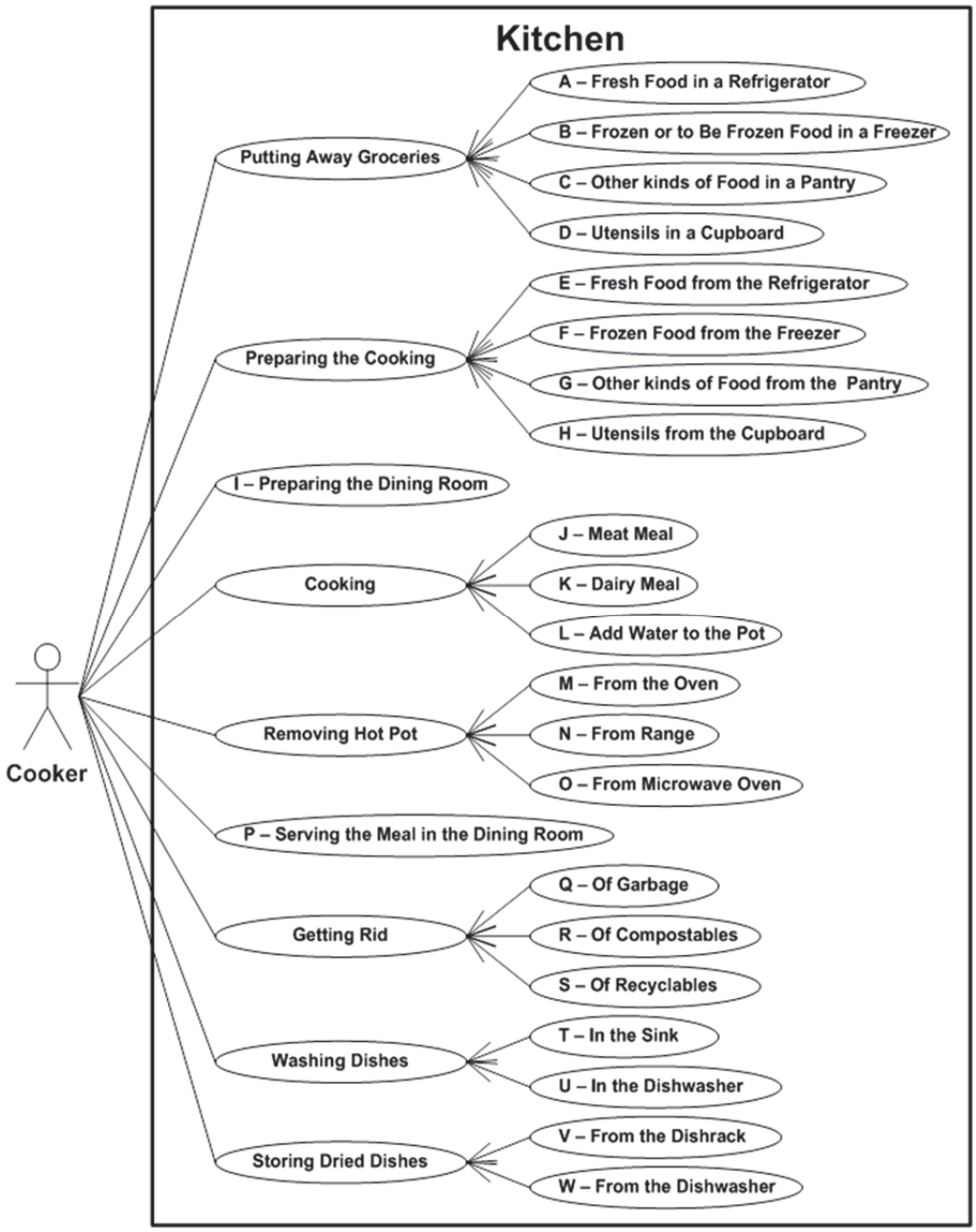


Figure App. K.2 – Use Case Diagram of the Kitchen Usage

Activity Diagram – Based on the set of use case contained in the use case diagram, we build an activity diagram (Figure App. K.3). The activity diagram describes in more details the steps followed to perform each use cases and their sequential dependencies from the actor (i.e. the cooker) viewpoint.

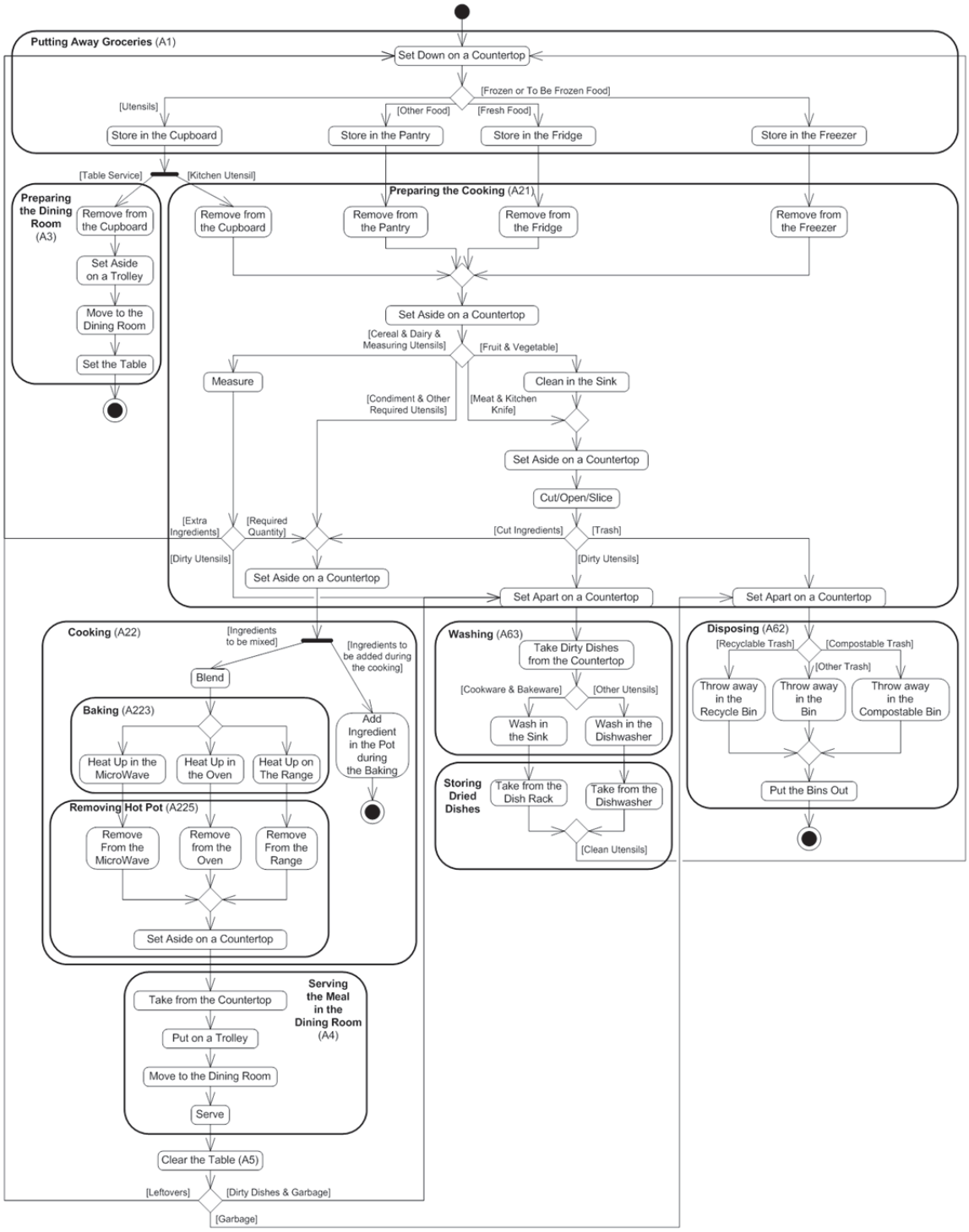


Figure App. K.3 – Activity Diagram of the Synagogue Kitchen Usage

Class Diagram (Entity-Relationship Diagram) - Figure App. K.4 represents a possible formalisation of business knowledge related to kitchens. This example is quite generic and does not integrate specific rules regarding keeping kashrut (e.g. meat and dairy food should not be mixed). Such entity relationship diagram could be used or applied on any kitchen. Programmers could use it to capitalise generic knowledge on typical case studies reducing the requirement elicitation process to a validation of the formalised knowledge by the paying, user, or customer clients. To take full benefits of such models, its automation and implementation in a database are essential (also due to the huge amount of information generated by the briefing process).

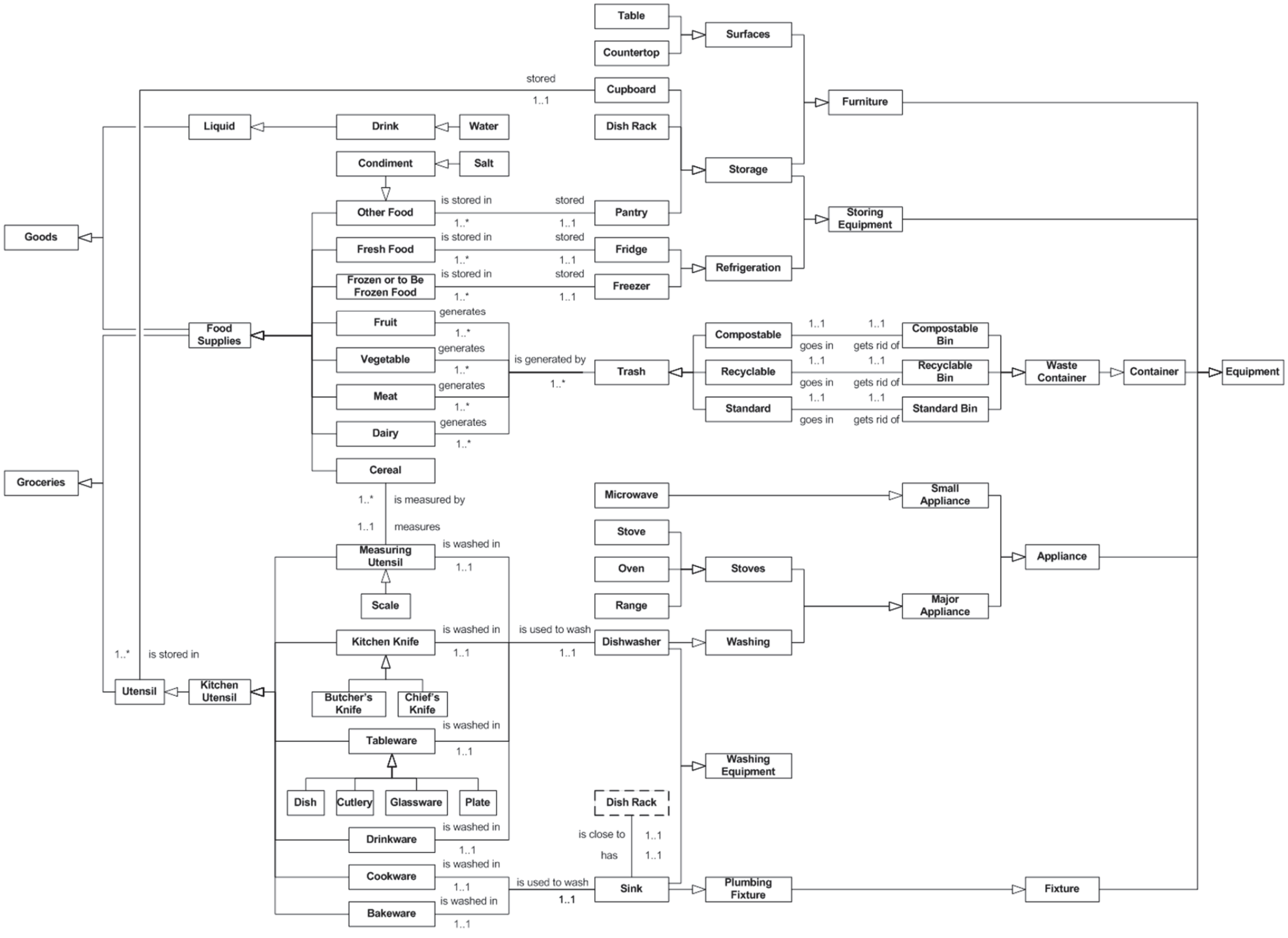


Figure App. K.4 – Entity Relationship Diagram of the Synagogue Kitchen Usage

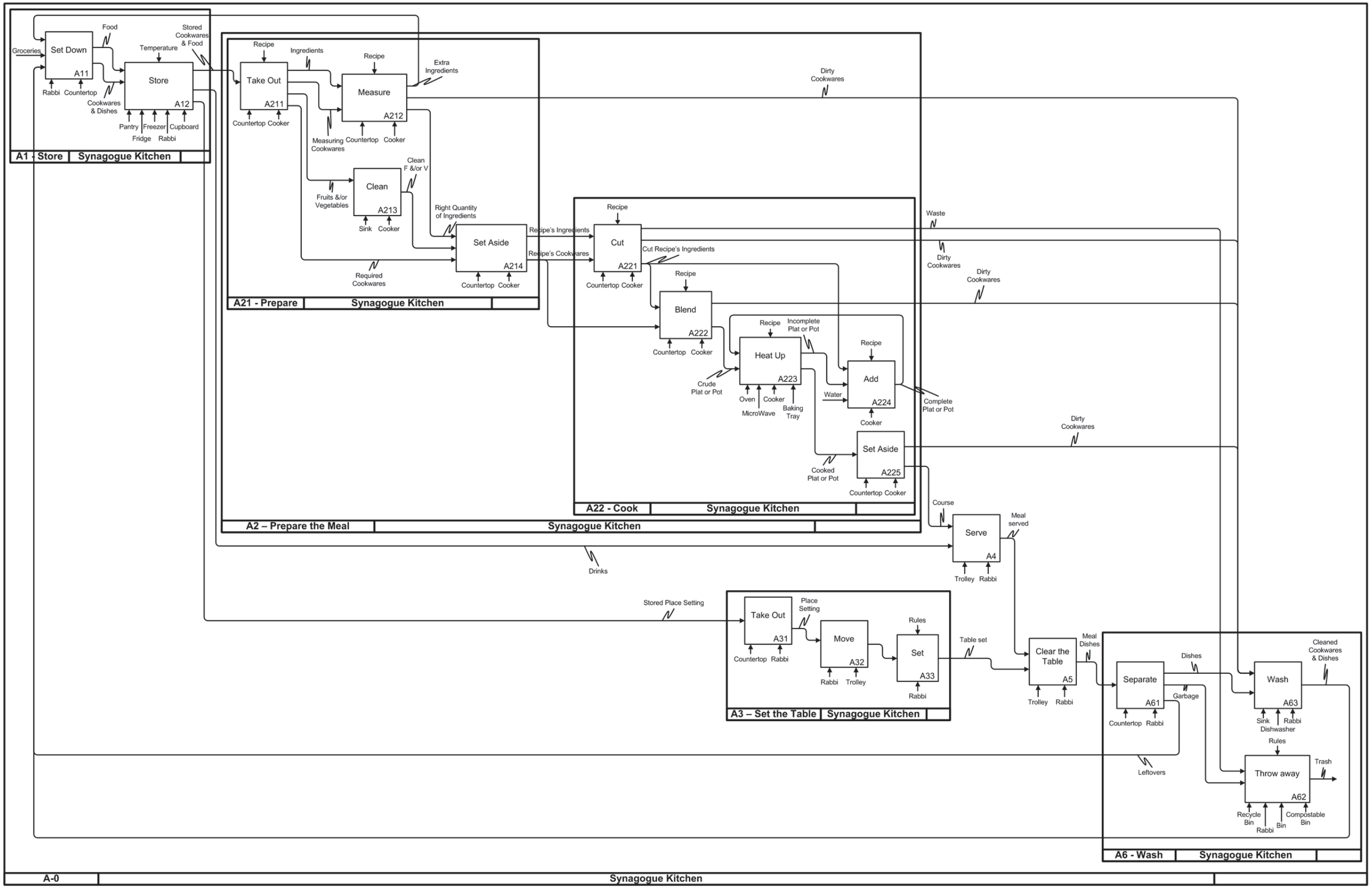


Figure App. K.5 – IDEF0 of the Synagogue Kitchen Usage

IDEF0 – Based on the same information used for the UML models, an IDEF0 model is proposed (Figure App. K.5). The IDEF0 model provides a different viewpoint compared to UML. UML focuses on the actors and the structure elements. IDEF0 represents the viewpoint of the goods transformed during each activity. As a result, the number of activities is less important as there is no distinction between "mechanisms", i.e. the required resources. On the other hand, the number of distinct flows is more important due to the separation of goods. The state changes of the goods are also represented in this model.

Synthesis

The use of the two different models (i.e. UML and IDEF0) implies a certain effort in terms of consistency. Information modelled in the first set of models can be partially represented in the second one. As the viewpoint is different, the completion of the second model can also lead to the identification of missing information in the first ones. As a result, both models support each other to refine and complete the information on the kitchen use. The programmer can benefit from their synergies to complete his understanding of the clients' behaviours.

METHODE DE DEFINITION DES EXIGENCES D'UN PRODUIT INTEGRANT SES SERVICES APPLIQUEE AUX BATIMENTS PUBLICS

RÉSUMÉ : La phase conceptuelle, basée sur des informations imprécises et incomplètes, est à la base de la satisfaction des clients par le futur système. La programmation architecturale se concentre sur une des premières étapes de cette phase dans le domaine de la construction : la définition des besoins d'un bâtiment. Les pratiques et recherches actuelles en Architecture-Ingénierie-Construction ont tendance à être basées sur l'expérience des programmistes et architectes. Elles sont rapidement orientées vers la production de solutions architecturales et avec une formalisation tardive des besoins métiers en exigences bâtiment. La définition des exigences est bien plus développée dans les autres domaines d'ingénierie (i.e. génie mécanique, industriel et logiciel) au sein de théories, modèles, techniques et outils. Ces travaux de recherche visent à proposer un raisonnement de programmation architecturale basé sur ces connaissances en matière d'ingénierie des exigences dans ces autres domaines. La méthodologie suivie pour développer ce raisonnement s'appuie sur une démarche transdisciplinaire basée sur une combinaison de méthodologies issues des Sciences de la Conception. Outre l'identification de langages de modélisation couvrant différents points de vues associés à la programmation architecturale, un manque est identifié en ce qui concerne les concepts définissant l'objet d'étude : la transition entre les concepts d'activité et d'espace. Le concept de ''méta-espace'' est proposé pour combler ce manque ainsi qu'un langage de modélisation permettant l'exploitation des relations entre ces concept. Une formalisation très en amont de la transition des besoins métiers des clients vers les exigences bâtiment est proposée via une démarche conceptuelle résultant de l'ajout du concept de ''méta-espace''. Cette démarche est présentée étape par étape en s'appuyant sur les divers cas d'étude traités au cours de la thèse.

Mots clés : programmation architecturale, architecture-ingénierie-construction, ingénierie des exigences, phase conceptuelle, système produit-service, conception de service

FRAMEWORK FOR INTEGRATION OF SERVICES IN PRODUCT REQUIREMENTS DEFINITION APPLIED TO PUBLIC BUILDINGS

ABSTRACT: The conceptual phase is based on very fuzzy and incomplete information about clients' needs. It constitutes the core of clients' satisfaction about the future system. Architectural programming, or briefing process, focuses on the early stages of the conceptual phase of construction projects, i.e. the requirements definition. Current practices and research in Architecture-Engineering-Construction tend to be experience-based, solution-oriented, and with a late formalisation of business needs into building requirements. The requirements definition is far more supported by theories, models, tools, and techniques in other engineering disciplines (i.e. Mechanical, Industrial, and Software Engineering). This research aims to propose an architectural programming reasoning based on existing requirements engineering knowledge from these other engineering disciplines. The methodology followed to develop this reasoning is a transdisciplinary approach based on a combination of Design Research Methodology, and Design Science. Besides the identification of modelling languages covering different viewpoints associated with the briefing process, a gap is identified regarding modelling constructs associated with a "meta-space" design artefact. This design artefact and its associated modelling language (i.e. the meta-space diagram) support the formalisation of the transition from activities to space. The resulting conceptual framework proposes a step by step early formalisation of the transition from clients' business needs to building requirements. All along the thesis, different case studies are used to illustrate the proposals.

Keywords : briefing process, architecture-engineering-construction, requirements engineering, conceptual design, product-service system, service design