The following full text is a publisher’s version.

For additional information about this publication click this link.
http://hdl.handle.net/2066/135046

Please be advised that this information was generated on 2021-11-03 and may be subject to change.
Combining System Dynamics with A Domain Modeling Method

Fiona P. Tulinayo
Combining System Dynamics with A Domain Modeling Method

Fiona P. Tulinayo
Promotoren:
Prof. dr. H.A. (Erik) Proper
Prof. dr. ir. Th.P. (Theo) van der Weide

Copromotor:
Dr. P. (Patrick) van Bommel

Manuscriptcommissie:
Prof. dr. E. (Erik) Barendsen
Prof. dr. T.A. (Terry) Halpin (INTI International University, Malaysia)
Prof. dr. E. (Eric) Dubois (University of Luxembourg, CRP Henri Tudor, Facultés Universitaires Notre-Dame de la Paix)
Contents

Dedication 1

1 Introduction 3
   1.1 Identification of the problem area ........................................... 4
   1.2 Main research question ................................................................. 5
   1.3 Justification for this study ............................................................. 6
   1.4 Scope ............................................................................................ 7
   1.5 Research approach ....................................................................... 7
   1.6 Thesis outline ............................................................................... 9
   1.7 Publications ................................................................................ 10

2 Literature 13
   2.1 Ontology modeling .................................................................... 13
      2.1.1 Conceptual modeling .............................................................. 15
   2.2 System dynamics modeling .............................................................. 17
      2.2.1 SD qualitative modeling .......................................................... 19
      2.2.2 SD quantitative modeling ......................................................... 20
   2.3 Conclusion .................................................................................. 21

3 Requirements for this study 23
   3.1 General requirements .................................................................. 23
   3.2 Specific requirements .................................................................. 26
   3.3 Conclusion .................................................................................. 29
7.1.3 Presentation of results ......................................................... 81
7.1.4 Contributions and challenges ................................................ 88
7.2 Analytical evaluation ........................................................................ 90
  7.2.1 Setup of walkthroughs ........................................................... 90
  7.2.2 Analysis of walkthrough feedback ........................................... 91
  7.2.3 Presentation of results ........................................................... 92
7.3 Discussion and reflections on GSD procedure ................................. 105
7.4 Conclusions ..................................................................................... 107

8 Update behaviors and simulations ............................................... 111
  8.1 Object behavior ........................................................................... 112
    8.1.1 Conceptual description for objects ........................................... 112
    8.1.2 States and transitions ........................................................... 112
  8.2 The “Intrapartum process in Ugandan hospitals” case ..................... 118
  8.3 Controlling object flow ............................................................... 122
    8.3.1 A continuous approximation ................................................... 123
    8.3.2 Influencing transitions .......................................................... 124
  8.4 Simulation ..................................................................................... 125
    8.4.1 Model results ........................................................................ 127
  8.5 Conclusion ..................................................................................... 129

9 Suggestions for organizational and tool support ............................ 131
  9.1 Organization involvement in the modeling process ......................... 131
    9.1.1 Alignment of information systems to organizational structure .... 132
    9.1.2 Enterprise architecture modeling ............................................. 133
    9.1.3 Collaborative modeling ......................................................... 134
  9.2 Towards operationalizing the process ............................................ 134
    9.2.1 Methods and procedure used .................................................. 138
    9.2.2 The medium ........................................................................ 141
    9.2.3 Who are the participants and how do they interact ................. 141
    9.2.4 How to bring the model to life ............................................... 143
  9.3 Discussions .................................................................................. 144
# List of Figures

1.1 Research problem and underlying principles .................................................. 5  
1.2 Opted route for this research ............................................................................. 7  
1.3 Cycles in the adopted research methodology were based on [86, 85] . . .  8  
1.4 Research methodology ....................................................................................... 8  
1.5 Thesis layout ........................................................................................................ 9  

4.1 Various paper states ............................................................................................. 33  
4.2 Paper flow concepts in ORM ............................................................................. 33  
4.3 A demonstrating example of a causal loop diagram ..................................... 35  
4.4 Types of flows ........................................................................................................ 40  
4.5 A summary of SFD basic building blocks .......................................................... 41  
4.6 Paper flow .............................................................................................................. 43  
4.7 An integration of ORM with SFD ...................................................................... 44  
4.8 A representation of figure 4.7 with an SD modeling tool .............................. 45  

5.1 Examples of ORM subset constraint and equality constraint ....................... 50  
5.2 Examples of a flow added to stocks ................................................................. 51  
5.3 A supertype-subtype ORM model example plus derived SD model ............ 52  
5.4 An ORM and SD model with connectors introduced ..................................... 53  
5.5 A summary of ORM-SD focus group session activities ................................. 55  
5.6 A summary of how we analyzed focus group data ........................................ 56  
5.7 Possible relations ................................................................................................. 61  

6.1 Overview of the GSD method ............................................................................ 63
8.8 The simplified object life model of a band. Source: [55] .............................. 118
8.9 ORM scheme for the intrapartum process in Ugandan hospitals ................. 119
8.10 ORM sub-scheme for object type patient ......................................................... 119
8.11 ORM sub-scheme for object type patient history ........................................... 119
8.12 Detailed patient Intrapartum model .............................................................. 120
8.13 Baby state model ........................................................................................... 120
8.14 Nurse state model .......................................................................................... 120
8.15 Obstetrician state model ................................................................................ 121
8.16 ORM sub-scheme for object type bed ........................................................... 121
8.17 Life model of a bed ....................................................................................... 122
8.18 Decomposition levels in the ORM scheme shown in figure 8.9 ..................... 122
8.19 A partial stock and flow diagram representation of patient life .................... 125
8.20 Stock and flow diagram representation of patient life .................................... 127

9.1 Decomposition guidelines .............................................................................. 135
9.2 Guidelines for developing a simulation model ................................................ 136
9.3 overview of the way of working (This model was formulated basing on the discussions in chapters 7, 8 and references [107, 7, 8, 10]) ......................... 139
9.4 Approach to GSD simulation .......................................................................... 140
9.5 Overview of the tools used at each modeling phase ..................................... 141
9.6 Application of GSD ....................................................................................... 143

10.1 GSD methodology overview ........................................................................ 149
List of Tables

1.1 Cycle verses Phases ................................................................. 9
1.2 Research questions and thesis outline .............................. 10

2.1 The SD modeling process across classic literature. Source: [126] ................................. 18

3.1 Design science research guidelines. source: [86] ............................ 28
3.2 Summary of thesis requirements and type of questions they answer ......................... 29

4.1 A summary of the causal loop diagram rules .......................... 35
4.2 Forms of stocks, their functions plus their examples ....................... 40

5.1 Results from relations questionnaire ........................................ 57
5.2 Results obtained after focus group discussion ......................... 59
5.3 Revised ORM to SD relations .................................................. 60

6.1 Objects for each CLD variable in Fig. 6.2 ............................... 65
6.2 CLD variables, identified object types, roles and ORM representation .......... 68

7.1 CLD-ORM focus group session execution summary .................. 80
7.2 Objects identified from CLD variable names ............................ 84
7.3 Some of the results of step 3 discussions ................................. 85
7.4 Verbalizations for figure 7.5 .................................................... 87
7.5 Refined ORM to CLD steps, strengths and weaknesses ................. 89
7.6 Walkthrough setup .............................................................. 91
7.7 CLD to ORM transformation steps as provided to ORM experts ............... 93
7.8 CLD to ORM questionnaire feedback ......................................................... 97
7.9 Improved CLD to ORM transformation steps ........................................... 98
7.10 ORM to SFD questionnaire feedback ....................................................... 101

8.1 Variations in stocks EnterHospital, AdmittedPatients and Discharges .... 128
8.2 Variations in stocks Births, MonitoredPatients and Discharges ........... 128
8.3 Variations in flows M, K and N ............................................................... 129
Acknowledgement

T.D. Jakes once said “Anything worth having expect it to hurt”. When I look back, there were times when the world seemed to be closing up on me. Everything seemed to be going wrong and I needed a shoulder to lean on. Surely, those were tough times. Amidst all this, I am so grateful that the Lord my God never left my side. He stood with me in the middle of the storm until it was calm. Thank you Lord.

This journey has not been smooth but no matter how I felt, I got up, dressed up and showed up. With this attitude I was able to get this far, of course there are many other ingredients to this thesis completion. However, the most significant ones which humble me are: the continuous encouragements, support, guidance, knowledge and advice I received from my daily supervisor (Dr. Patrick van Bommel), Dr. Stijn Hoppenbrouwers, Prof. Theo van der Weide, family members especially my sisters (Dr. Juliet Tumwikirize and Hilda Tuhumure) and friends (Dr. Agnes Nakakawa, Dr. Georgita Igna and Dr. Rehema Baguma). I am so grateful and appreciate you.

In cases like these, it is hard to mention each and every person who contributed to this thesis. However I am highly indebted to the coordinators of Nuffic II project for awarding me a scholarship to pursue this PhD. Prof. Ddembe Willeese Williams thank you for your continuous support and advice. Prof. Venansius Baryamureeba thank you for the support you gave me during my research and for being one of the brains behind the NUFFIC II project.

Prof. Erik Proper, thank you for the supervision and contributions. Prof. Theo van der Weide, I am so blessed that you allowed to be part of this research. You came in at a time when I was confused about my thesis progress. Thank you very much for stepping in on such short notice, for believing in me and standing by me during trying times.

Minus my daily supervisor and promoters, many people have provided their expertise and experiences that are added in this thesis. I would therefore like to thank Prof. Terry Halpin and Guido Bakema who advised me on the content in the evaluation and criticisms that tremendously improved the content of this thesis. I would also like to thank the thesis examiners for their valuable comments, the reviewers of conferences, workshops and Journals where this work was submitted. More specifically, the reviewers of the International Journal of Information System Modeling and Design (IJISMD), International Journal of Cooperative Information Systems (IJCIS) and Practice of Enterprise Modeling (POEM). The staff of the following organizations; National Medical Stores, Mulago
Obstetrician and gynecology department, Kawolo gynecology labor ward and Mukono health center labor ward thank you for your corporation. The academicians and practitioners that took part in the evaluation sessions i.e Dr. Ettiëne Rouwette, Dr. Andreas GröBlers, Ger Paulussen, Zwart Jan Peter, Guido Bakema, Elton Manoku, Aminah Zawedde, Saul Kidde, Dr. Agnes Nakakawa, Dr. Mercy Amiyo, Rose Nakibuule, Irene Nakiyimba and Dr. Peter Wakhali. Thank you for your contributions.

The staff of Radboud University Nijmegen, the Radboud International office and the Model Based Systems Development (MBSD) team (2008-2012) thank you for making me feel at home on my arrival and during my stay at Radboud university. My officemates and friends both at Radboud University and at home i.e Dr. Agnes Nakakawa, Dr. Georgita Igna, Dr. Denis Ssebuggwawo, Wenyun Quan, Fides Aarts, Ilona Wilmont, Dr. Farakak Heydarian, Walter Omona, Idda Mosha, Derreck Chitama, Dr. Abiba Longwe, Claire Chagwiza, Brenda N. Kalete, Alexandra Stütz-Karamagi, Jenny Awor, Jackie Onyango Lubega, Winfred Lukowe, Saul Kidde, Dr. Charles Otine, Maureen Nabawanda, Justine N. Karlsen and Benjamin Ajal. Thank you for your friendship, laughter, chats, skating experience and numerous dinners. You surely made may stay at Nijmegen and Radboud University memorable. Ms. Nicole Flipsen, Ms. Irma Haerkens and Ms. Ingrid Berenbroek thank you for your guidance and selfless support.

Special thanks to my parents, brothers, sisters, Dr. Barungi Thaddeus, Mr and Mrs Nzaba’s family, Mr and Mrs Joshua Ssentamu’s family. You have all supported me in many ways I can’t detail here. I am therefore so grateful and blessed to have you in my life. JMK thank you for your Love:))!
Dedication

“There is nothing permanent except change.”
“Where begin by being foolish and we become wise by experience” Maasai Proverb

“To those who thought I could not: they drove me to prove them wrong”
“To my parents who spared a coin and saw me through school”
“To my unborn children, brothers, sisters, nieces and nephews”
“And to the Lord almighty who found me worthy”
Chapter 1

Introduction

It is the desire for many modelers to have a modeling method that exhaustively captures all required information in a systematic yet simple and comprehensible manner. This is because having explicit and comprehensive models may lead to increased use and applicability of the method. Although this is hard to accomplish, we believe that through combining different modeling methods it can be achieved. Combining of methods can be viewed as an improvement of both the methods and process development. This is because concepts from different approaches supplement each other; hence the weaknesses of one approach are overcome by the strength of the other. The combining process requires precise understanding of the relations among or between different constructs, which then lead to reliable scrutiny, transformation and understanding of the underlying principles, hence a more principled description of the method(s) is attained. Through this process, system modeling and underlying principles of complex systems could become easier to conceptualize. In this study therefore we put a domain modeling method Object-Role Modeling (ORM) to work in the context of the creation of System Dynamics (SD) models. The art of SD modeling lies in determining and representing the feedback processes and other elements that determine the dynamics of the system (typically, a process in an organization). However, SD shows a lack of instruments for discovering and expressing precise, language-based concepts in domains. At the same time, the field of conceptual modeling has long since focused on deriving models from natural expressions. We therefore turn to ORM as a prime example of this school of thought to integrate its strong natural language based modeling approach into the creation of SD models. This was inspired by an observed lack of concept-level modeling power in SD. We set out mainly to augment SD modeling by first laying down a sound foundation of domain concepts by means of fact-based ORM modeling. This eventually enables us to link the ORM model to the SD stock-flow diagram.
1.1 Identification of the problem area

Over the years, system dynamics modelers have developed models to increase understanding of the dynamics in a system and for decision makers to have a basis for their decisions on identified complex problems. But these models have been criticized for their lack to capture the structural views which are found at the operational (who and when aspects of the ongoing operations?) and tactical (how will it be done?) levels of the information system pyramid [176, 120]. Their concentration at developing models capturing strategic (what has to be done?) views makes the use and reliability of developed SD models undeniable. If information at all levels of the decision pyramid was captured, better models would be derived although the task entailed is not easy. Secondly, organizational settings are complex in nature which makes them hard for SD modelers to define and conceptualize because the number of variables to consider are many and the question of causation is complicated to untangle [233].

In [13, 117, 140, 148, 164, 176] it is stated that SD as a method has a number of issues among which are ambiguity and lack of detail [13, 140, 176], these make transformation of Causal Loop Diagram (CLD) variables into Stock and Flow Diagram (SFD) hard [140, 164, 176]. Richardson [164] particularly states that; “...of the three tasks-model conceptualization, model formation and model understanding. Unquestionably the two most difficult are the two that are least formal- conceptualization and understanding. The future of the field needs software support for understanding the links between stock and flow/feedback structure and dynamics behavior.” On the other hand Sharif [183] also states that; “...there is a strong case for starting to apply systems dynamics methods more openly in the BPM and MIS research fields, as I feel the tools and techniques available are vastly under-rated in terms of their applicability and capability to provide novel representations of real-world situations...”. These statements are the school of thought for this research.

The identified key issue here is SD model conceptualization as stated in [117, 148, 140, 164]. In order to address this issue, different scholars have put in efforts by advocating for better tools and methods to support SD modeling for example; [140, 164] suggest the mapping of tools to represent SD structure and [175, 193] suggest techniques to analyze reference modes. Here, we join scholars like [49, 105, 118] who have devised efforts to support the SD conceptualization process with tools and techniques used in other scientific disciplines. In our case, we address this issue by using support of a domain modeling method ORM whose focus is on domain conceptualization through data modeling [71]. One of the important features of ORM is its foundation in natural language analysis which makes it resistant to changes that cause attributes to be remodeled as object types or relationships [100]. To position this research, we present a logical framework in figure 1.1 where we show the strength and gaps in both ORM and SD.

In this framework we itemize (allocate an identification character) each method for easy of use in the explanation. We combine method M-1 with method M-3 to arrive at artifact M-2. The gaps and strengths identified in methods M-1 and M-3 are what we use as a basis for our study. We further present the importance or achievements of combining M-1 and M-3 in M-2. As an output of M-1, M-2 and M-3 is the main advantage for
1.2. MAIN RESEARCH QUESTION

From the identified problem, we come up with a main research question.

How can we combine system dynamics with a domain modeling method?

Specific research questions

In order for us to realize the stated main research question, we further come up with smaller manageable sub questions (specific questions).

Concepts: What are the existing SD and ORM constructs?

Relationship: What relationships exist between these two methods? How can we best represent the interactions in these methods?

Procedure: What is the procedure for combining these two methods?

Evaluation: How can we evaluate the procedure(s) for combining system dynamics with ORM and the resulting model?
Guideline: How can the modeling procedure(s) be embedded in a modeling tool(s)?

1.3 Justification for this study

In order to conceptualize a given SD scenario, it is important that the ‘what’, ‘how’ and ‘why’ questions are well answered. In SD however, concentration is drawn to the ‘what’ of the dynamics behavior but not ‘how’ the model elements should be reconfigured to yield a desired behavior [58]. That is why a similar case or model of reality can be depicted (interpreted) in different ways by different SD modelers. The ‘how’ can however be captured in SD models if integrated with domain modeling. This is because domain methods capture a detailed representation of the system in terms of system elements and their interactions to provide a means of understanding a given scenario. Secondly, domain methods have a precise and consistent mechanism for conceptualizing reality, based on which a stock and flow diagram can be realized [58, 156]. Therefore, having domain modeling work in context of SD model building makes SD models more understandable and well focused.

To policy makers: System dynamics modelers build models that are mainly used by policy makers (strategic decision makers) but these models do not capture tactical (responsible for generating procedures) and operational (execute those procedures) views. This affects decisions made by policy makers at the strategic level because SD models derived are loosely coupled. For best results therefore, policy makers should use models that cut across all levels i.e. built with strategic (SD), tactical (domain modeling) and operational (domain modeling) understanding. This can only be achieved if the action distance between ‘knowing’ and ‘doing’ is reduced.

Why the combination: we combine SD with a domain modeling method because domain modeling methods help to identify relationships among entities within the scope of the problem domain and provide a structural view of the domain. This therefore means that the addition of a domain modeling method in the process of developing SD models may improve SD model conceptualization, make SD variable names explicit, and enable transformation and reuse of information. We use ORM in particular as an example of a domain modeling language because of its conceptual focus and roots in verbalization, graphical expressiveness and well-defined semantics. The main advantage of improving SD’s conceptual foundation through ORM is that SD models can be more soundly and readily linked to databases. Secondly, combining SD with ORM provides complementary added value: SD Studies the behavior of a system in terms of discrete quantities of things (stocks and flows) while ORM underpins the models in terms of underlying ontology of the domain. Halpin and Wagner note that “although ORM supports modeling of business terms facts, and many static integrity constraints and derivation rules, it cannot model the reactive behavior of systems which can be described using dynamic integrity constraints”. Therefore, it is the intension of this study that this integration enables stakeholders attain a level of understanding of the dynamics and statics within the SD developed models.
1.4 Scope

We limit this study to combining system dynamics with a domain modeling method ORM. In the process of executing this combination, we opt to use the following route (see figure 1.2). First we transform a Causal Loop Diagram (CLD) model into an ORM model and then from an ORM model into a Stock and Flow Diagram (SFD). In so doing, we are able to improve SD model conceptualization and make the SD variable names explicit. Other alternative routes to combining ORM and SD are highlighted in chapter 7 (figure 7.14).

1.5 Research approach

Before presenting our research approach, we would like to first of all expound on its basing. In this study, we combine two methods. To achieve this combination, we follow the design science (DS) methodology as defined in [86]. But not all is done the design science way, instead we tailor this methodology to fit our way of working (see figure 1.3). Design science is prescriptive and may apply to both nomothetic (attempts to establish general patterns of behavior that can be extended to all elements in a given domain) and idiographic methods (focuses on a unique or individual domain) [125], aims to develop artifacts of heretofore unsolved problem [23, 150, 86]. As an artifact, we define Grounded System Dynamics (GSD) a combination of two existing methods (system dynamics and object role modeling). By combining these two methods we generate synergy effects by using already existing modeling methods and by so doing we overcome some of the weaknesses of SD model building.

To accomplish this, we first of all clarify the goals of the GSD artifact (environment). Thereafter build and carefully evaluate the utility of GSD (design phase), and to a lesser degree, its reliability and validity using analytical and experimental evaluation methods [86]. The design science approach places additional emphasis on the iterative construction and evaluation of artifacts which in our case took place at the evaluation phase where ORM and SD practitioners were asked to experiment and critique both the given steps and the resulting models.

In general, our research methodology was made up of six phases as shown in figure 1.4 and these included: problem identification, requirements specification, conceptualization, design, implementation, and evaluation.
CHAPTER 1. INTRODUCTION

Figure 1.3: Cycles in the adopted research methodology were based on [86, 85]

Figure 1.4: Research methodology

In phase 1, we identify the problem area by reviewing existing literature. In phase 2, we define requirement specifications that we use to arrive at the new artifact. In phase 3, we define constructs in both methods with their underlying principles and thereafter identify relations between constructs. To gain insights into the relations, we review literature and conduct evaluations on the defined relations. In phase 4, we define the new artifact and apply it using examples. In phase 5, we use case studies to experiment and analytically evaluate the GSD artifact. In this same phase, we also present ORM to SD update behavior(s). Note that between phase 4 and phase 5, we have an iteration because after evaluation, the new artifact is refined. In phase 6, we present suggestions for organizational and tool support. To attain comprehensive suggestions, we review related literature inline with our study and finally draw conclusions.

Evaluation methods used in this study are experiments, analysis and a case study. Case study as defined in [44] is a research strategy which focuses on understanding the dynamics present within a single setting. Case study may involve either single or multiple
cases and numerous levels of analysis [230]. It combines data collection methods such as archives, interviews, questionnaires and observations. In this study we used case study methodology to obtain information on a given problem domain. This information is what was used to derive models that were applied in experimental and analytical evaluations. To collect case data, we used observation, archives and interviews. Experimental evaluations were used in a focus group setting. A focus group is defined by [194] as a moderated discussion among six to twelve persons discussing a topic under the direction of a moderator whose role is to promote interaction and keep the discussion on the topic of interest. We used focus group discussions to allow us obtain participants feedback on the relations and GSD procedural transformation steps. For analytical evaluations, we used structured walkthrough sessions and questionnaires which enabled us get an in-depth analysis of the GSD transformation steps and derived models. In table 1.1 we summarize the relationship between figure 1.3 and figure 1.4.

Table 1.1: Cycle verses Phases

<table>
<thead>
<tr>
<th>Cycles in figure 1.3</th>
<th>Fig. 1.4 Research methodology phases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevancy</td>
<td>1</td>
</tr>
<tr>
<td>Design</td>
<td>✓</td>
</tr>
<tr>
<td>Rigor</td>
<td></td>
</tr>
</tbody>
</table>

1.6 Thesis outline

In this section, we present both the thesis layout and thesis outline as shown in figure 1.5 and table 1.2 respectively. In figure 1.5 we depict an overview of the thesis chapters plus what they contain. In chapter 2, we present existing literature relevant to the study. In chapter 3, we present the specified requirements. In chapter 4, we present SD and ORM constructs with their underlying principles. In chapter 5, we present the relations between ORM and SD constructs. In chapter 6, we present the steps for grounding system dynamics with Object-role Modeling. In chapter 7 we present evaluations for Grounded System Dynamics steps. In chapter 8, we present ORM and SD update behaviors. In chapter 9 we present suggestions for organizational and tool support. In chapter 10 we draw conclusions and make recommendation for further research.
Having shown the thesis layout, we now present the thesis outline in table 1.2 with an overview of the research questions, how or what activities were carried out and where they are captured within the thesis.

<table>
<thead>
<tr>
<th>Research Questions</th>
<th>How (Activities)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts</td>
<td>Reviewed literature to identify conceptual link</td>
</tr>
<tr>
<td>Relationship</td>
<td>Reviewed literature, identified and evaluated construct relations</td>
</tr>
<tr>
<td>Procedure</td>
<td>Expert opinion and reviewed literature</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Conducted analytical and experimental evaluations</td>
</tr>
<tr>
<td>Suggestions</td>
<td>Reviewed literature, analytical study</td>
</tr>
</tbody>
</table>

Table 1.2: Research questions and thesis outline

In the process of this study, obtained results were presented as follows; in [207], we used ORM to integrate its strong natural language based modeling approach into the creation of SD models in a two-step schema based approach. By transforming an ORM domain model into a SD stock and flow diagram, we were able to identify the conceptual link between SD and ORM. We further discussed how typical ORM conceptualization can be linked to SD conceptualization and with use of examples showed how such a transformation can be performed. Thereafter, we investigated the basic building blocks of these methods using examples [206]. Investigating the foundation of the two methods enabled us to better understand the underlying concepts and differences in update behavior of the methods that are caused due to state and decision changes. Next we present a procedure which we refer to as GSD (Grounded System Dynamics) a combination of both methods. In this procedure, we outline and define steps on how to underpin an ORM model with a system dynamics model [208].

1.7 Publications

Most of the works presented in this thesis, are extended versions of our publications. These publications were in form of Journals, Conferences and Workshops. Below is a list of the publications.


4. Tulinayo, P. F, (2009) Integrating System Dynamics with Conceptual and Process Modeling. In proceedings of the CAiSE '09 16th Doctoral Consortium held in conjunction with the CAiSE '09 conference, Amsterdam, The Netherlands, June 8th -12th


Chapter 2

Literature

Having provided an introductory chapter to this study, it is important that we establish what we have based our study on, to offer both a coherent and a shared justification. As already stated, we combine two existing methods, these two methods have a lot to be desired but differ in every aspect of modeling. One of the methods has static properties (ORM) while the other has dynamic properties (SD). The static method captures the structure of the system while the dynamic method captures the runtime behavior of a system. In this chapter, we review existing literature to identify the knowledge gap, and acknowledge the work of others in the relevant research area(s). This chapter is made up of two main sections 2.1 and 2.2. In these sections we present existing literature on Ontology Modeling and System Dynamics modeling. This allows us give a clear overview of what is and what was of the methods. Under section 2.1 we define what an ontology is, its use and the ontology domain method we opt to use in this study. In subsection 2.1.1 we present literature on different domain modeling method and also state ORM as our preferred choice for this study. In section 2.2 we present literature on system dynamics, its way of working and areas where it has been applied. Within this section, we also present literature on the two diagrams most commonly used in SD. Finally, we draw conclusions in section 2.3.

2.1 Ontology modeling

The word ‘ontology’ has been defined by different scholars in a number of ways; in [61] Gruber defines an ontology as a formal, explicit specification of a shared conceptualization of a certain domain. On the other hand, Guarino in [63] refers to an ontology as an engineering artifact, comprising of a precise vocabulary used to describe a certain reality (universe of discourse) with a set of explicit assumptions regarding the intended meaning of the vocabulary. Both of these definitions are somewhat similar because the term ‘Conceptualization’ refers to the process of identifying the meaning of particular terminologies (vocabulary as Guarino puts it). The words in the vocabulary appear as
unary, binary or ternary predicates commonly known as relations. In computer science ontology has been redefined as a branch of knowledge engineering, where agreed semantics of a certain domain is represented that enables sharing between information systems [102]. In other words ontology describes a hierarchy of concepts related by subsumption relationships. These relations are added with the aim of expressing other relationships between or among concepts and to limit their interpretations [63, 30, 48]. Guarino further stresses that an ontology is language dependent (application autonomous as Meersman puts it in [134]), and somewhat represent a domain while a conceptualization is language-independent. In [133] Meersman gives clarifications of ontology terminologies detailing its originality, use and how it has been used by other scholars.

Ontologies define a set of constructs used to represent real-world and are used in many ways: 1) to study the type of things that exist or may exist in domains, 2) as general information representation formalisms and overtly during design process to represent knowledge about a particular domain, 3) to describe the semantics in the source of information and to make contents explicit in integration tasks, 4) to facilitate an interaction between domain experts and computer interviewer for knowledge acquisition and 5) for reuse in building a problem solving engine that simulates the problem solving behavior of a domain expert [137, 192, 65, 195]. Their main aim is to capture static domain knowledge in a generic way and provide a commonly agreed upon understanding of that domain, which may be reused and shared across applications. Ample research has been done on ontologies: its philosophy, disciplines where it has been applied [152, 64, 192, 48], issues [153], mapping [43, 104] and integrations [151]. We would like to point out that we do not intend to get into the details of ontology, the references provided in this section exhaustively cover this topic.

Ontologies take on several tasks in three different directions as stated in [219] and different types of ontologies exist for example: 1) Task ontologies; Mizoguchi defines a 'task' as a sequence of problem solving steps and a 'task ontology' as a system of vocabulary for describing problem solving structure of all the existing tasks domain independently [137]. In [98] Ikeda interprets task ontology in two ways; one as an ontology for specifying problem solving processes (this definition is close to the domain ontology definition) and two as a task-subtask decomposition together with task categorization. In task ontology, structures of real world problems are analyzed which helps in overcoming the limitations of generic tasks [137]. 2) Method ontologies on the other hand provide terms specific to particular problem solving methods. Methods and tasks are two distinct terms in knowledge engineering [29]. 3) Application ontologies are specializations of a more general library which includes task and method ontologies as well as domain ontologies [62], etc. In this research however we focus on domain ontologies that specify conceptualizations specific to a domain [195]. Domain ontologies are used to describe concepts or objects in a particular domain by means of their properties as well as their relationships [63]. As a domain ontology modeling method we choose to use ORM essentially as a graphical front-end to Web Ontology Language (OWL) [96, 2, 36, 192] rather than the 'Database design way' [78, 71, 77] as is traditionally done. OWL is a logic-based language that is used to formalize a domain of knowledge [4]. It was developed as a follow on from Resource Description Framework (RDF) and RDFs [20, 111], as well as earlier ontology language projects that included Ontology Interface Language
2.1. ONTOLOGY MODELING

(OWL) [47], DARPA Agent Markup Language (DAML) [83] and DAML+OIL [95, 130].
OWL is intended to be used over the World Wide Web and all its elements are defined as
RDF resources and identified by Uniform Resource Identifier (URIs) [12]. There exists a
number of conceptual/domain modeling methods some of which we discuss below;

2.1.1 Conceptual modeling

Conceptual modeling is a design activity that is influenced by both application and pro­
cess knowledge [182]. It models part of the world in an information base referred to as
the Universe of Discourse (UoD). Understanding the real world and representing it in a
way it can be translated into design is one of the objectives of conceptual modeling [195].
Conceptual modeling is directly related to knowledge representation and validation [186].
In knowledge representation, conceptual models are created to map real-world needs and
in validation, stakeholders verify whether their needs have been correctly specified by
looking at the generated conceptual models. Therefore the quality of a conceptual mod­
eling method has an effect on the quality of the conceptual model [139, 182, 138]. The
outcome of the conceptual modeling process is a conceptual model sometimes referred to
as a domain model. The conceptual model is independent of design or implementation
concerns, aims at expressing the meaning of terms and concepts used by domain ex­
erts while discussing the problem, and to find the correct relationships between different
concepts. There exist various conceptual modeling notations and methods for example;
Unified Modeling Language [17, 174], Entity-relationship diagrams [33], Object-Role
Modeling [77], etc. The goal of these conceptual modeling methods, is to provide a pre­
cise and unambiguous representation of information that performs efficiently and whose
logical structure is easily understood [229]. In more detail:

Entity relationship diagrams

Entity-relationship (ER) diagrams describe the stored data layout of a system and are
made up of entities and relationships [33, 231]. In ER notation, the conceptual model
is described with an ER Diagram in which entities represent concepts and associations
between entities represent relationships. An entity is defined as an object in the real world
with independent existence distinguishable from other objects. A relationship is a tuple
of entities and represents an association between entities [33]. A set of relationships
refers to a set of instances representing an association among different combinations of
instances of entities that participate in that relationship and the arity of a relationship
is the number of ER-roles. To restrict the number of times each instance of an entity
is allowed to participate in the relationship, constraints are attached to ER-roles. These
constraints specify the functionality of relations [25]. Entities are described in a database
by a set of attributes. Such separation of data and operations may be necessitated in cases
where the data and their interrelationships are sufficiently complex [106]. ER models
focus on the structural aspects of database schemas, as opposed to their behavioral aspects
and their purpose is to analyze and describe the database schema of a computer system
[59]. Since they focus on data and their interrelationships, they therefore provide no
constructs for modeling other process elements and provide no information about the functions that make or use this data. They are entirely static representations, providing no time-related information that could drive analysis and measurement [57]. ER diagrams pose limitations similar to Data flow diagrams (DFDs).

The DFD technique was originally described in [37, 231]. It depicts a system network of functional processes interconnected through data flows. They are mainly applied at the analysis phase of the project to describe how the process aspects of information systems work. Their precedence relation is an abstraction of functional semantics. Although DFDs are a useful description for modeling processes, they have a number of limitations; (1) they do not allow sufficient information to describe a problem, (2) they depict the situation from the data point of view and merely show the possible paths taken by an item of data i.e their primary concern is placed on data flow rather than the process [3], (3) they provide no modeling constructs on which to base representation of work flow, people, events, and other business process elements, (4) they provide no information on decisions and event sequences (temporal or precedence relationships), (5) they do not have beginning points, end points and execution paths that is to say, they are static in nature so they do not lend themselves easily to analysis or decision making. To facilitate such analysis, data flow diagramming sometimes is complemented by structured textual descriptions of procedures in which data is to be used; these descriptions are called process specifications [231]. The next conceptual modeling method we discuss is Unified Modeling Language (UML).

Unified modeling language

UML is a graphical modeling language that was adopted by Object Management Group (OMG) for object oriented (OO) analysis and design. It specifies, visualizes, constructs and documents the artifacts of systems [17, 174]. In UML notation, the conceptual model is described with a class diagram in which classes represent concepts, associations represent relationships between concepts and role types of an association represent role types taken by instances of the modeled concepts in various situations. UML class diagrams provide an extended Entity-Relationship (ER) notation that can be annotated with database constructs [73]. They are widely used to describe elements of an information system during the design phase, can be applied to all application domains and used with all major object and component methods. The usability of UML as a language for describing the real world is examined in [46]. The UML notation, models different perspectives of an application using collaboration diagrams, object diagrams, state charts, sequence diagrams, component diagrams, class diagrams, use case diagrams, deployment diagrams, etc. Although UML is popular for designing OO diagrams, it is less suitable for developing and validating a conceptual data model [17]. As already stated conceptual modeling is the activity of formally describing some aspect of a physical and social world around us for purposes of communication and understanding [145]. In order to state or describe what exists in the real world, and to make specific assumptions about how ‘things’ behave, an ontology model is developed. As already stated subsection 2.1, an ontology deals with what exists or assumed to exist [192, 65].
2.2. **SYSTEM DYNAMICS MODELING**

Object-role modeling

As opposed to other data modeling methods, Object-Role Modeling (ORM) evolved from NIAM (Natural-language Information Analysis Method) [226]. It was developed in the early 70's as a fact-oriented approach for modeling information and querying the information content of a business domain at a conceptual level [78]. Originally, ORM was conceived for data modeling purposes and takes a static perspective on the domain in the sense that it aims at capturing the fact types and entity types that play a potentially role in the (dynamic) domain [71]. Fact-orientation means that it includes both types and instances in its models; types are called ‘fact types’, instances ‘facts’. It supports n-ary relations, has an expressive and stable graphical notation. Its diagrams can be automatically verbalized into pseudo natural language sentences which makes it resistant to changes that cause attributes to be remodeled as object types or relationships [100]. Its formal specification and semantics are defined in [69, 211, 222]. ORM is comparable to Entity Relationship (ER) diagrams in use [33]. It is claimed to have a graphical constraint notation that is far more expressive for data modeling purposes than Unified Modeling Language (UML) class diagrams or Entity Relationships diagrams [78]. For a detailed treatment on the comparison between data modeling in UML and ORM see [72, 73, 75]. Although ORM was originally used as a database modeling method, it has been applied to other areas like requirements engineering [45], business rules [70] and also used as a graphical notation in XML-schemes to share information, exchange and process ORM schemas on the internet [14]. In this study however, we use ORM to model a domain ontology as scholars [189, 190, 101, 100] do. Unlike UML, ORM language was built to communicate, clarify, minimize the impact of change and to scope views of a relevant task. Secondly, ORM can be used for abstraction, has a validation mechanism and formal foundations [73]. It is for these decisive factors that we opt to use ORM over other conceptual/ontology modeling methods. More details about ORM can be found in [77].

2.2  System dynamics modeling

Another method we use in this study is system dynamics. System Dynamics (SD) comprises of two concepts: a *system* which is defined as a collection of organized set of interrelated elements and *dynamics* which refer to change over time. System dynamics therefore, seeks to understand how a collection of organized set of interrelated elements change over time. SD as a method dates as far back as 1961, developed by Jay Forrester to handle socio-economic problems with a focus on the structure and behavior of systems composed of interacting feedback loops. A review and history is given in [51]. Although SD has a number of problems [140, 164, 117, 148, 126], it is acknowledged as an excellent medium for exploring and identifying knowledge gaps [181, 34, 167, 227].

In SD it is recognized that, most of the information available to modelers is not numerical in nature, but qualitative [126]. This qualitative data resides in the mental database inform of written text [53, 41]. The underlying assumption is that human mental models are admirable in scrutinizing the basic actions out of which a system is composed, but unable to understand the dynamic implications resulting from these actions. These
mental models are used as a basis and first step in the SD modeling process [132, 41]. They give a description of reality that is usually expressed by a set of sentences in natural language where they portray interactions among elements within the system together with their external influences [41]. In [213] they point out three important attributes of mental models which are: 1) they are not fixed 2) they are not simple and 3) the subject information provided by mental models is usually reliable. To help modelers transfer their mental models into real models, scholars like Richardson and Pugh [167], Randers [159], Roberts et al. [169], Wolstenholme [227] and Sterman [193] came up with steps to guide in SD model building. In table 2.1, Wolstenholme [227] gives three stage; in [167] Richardson gives seven different steps; in [159] Randers categorizes the activities in four steps; in [193] Sterman categorizes the activities into five steps and in [169] Roberts categorizes the SD modeling activities in six steps. Although the processes in these studies are sequentially stated, scholars agree that the SD modeling process is an iterative process that begins and ends with understanding. The greatest strength of this approach is its ability to represent the evolving state of a system through time [126]. Wolstenholme’s reason for sequencing the SD model building process is to capture both quantitative and qualitative analysis because both approaches have important contributions to the ultimate success of any system dynamics study [228].

Numerous attempts have been made to compare and combine SD with other methods. In [24] they briefly discuss how to relate SD with multidimensional data modeling. Their focus however is on the conceptual properties of SD modeling language. They argue that an elaborated meta-model should be related to multidimensional data modeling to enhance the applicability of SD in modern business intelligence. In addition, there has been a lot of interest to compare system dynamics with Discrete Event Simulation (DES) [19, 18, 198, 197]. In all of these studies they aim at knowing which of the two modeling methods should be applied, when and why i.e. for a particular problem [143]. By answering this question, they make sense of the puzzling dynamics. In conclusion to their study, they state that the two methods are complementary but not opposing each other. On the other hand other scholars like [178, 18, 177] embarked on comparing SD with Agent Based Modeling (ABM). In [178] Scholl makes a call for a cross study and joint research on Agent-based and System Dynamics Modeling. In his paper he gives a brief description of each method, areas of applicability, states how the two modeling techniques fit together, their strengths and weaknesses. In [18] however, they compare the three simulation modeling methods (SD, DES and ABM) showing in detail how an ABM can be built.
from an existing SD or DES model. Although their study had not adequately matured, they concluded that by using the ABM approach one is able to capture phenomena than with SD or DES approach.

Other comparative studies have been made with modeling methods that are ostensibly less similar to SD. For example, in [42] Duggan gives a comparison of Petri Net and System Dynamics. He particularly sets out to explore whether Petri Nets could replicate the classic behavior of the standard SD models. In [31] an attempt to integrate SD with UML is done. In this study, scholars note that SD cannot be integrated well with information systems in organizations because it belongs to a specific field of modeling. In [58] SD and enterprise modeling are integrated to address dynamic and structural complexities of a choice situation. In their study, they state that the inability for enterprise managers to come up with good decisions is hampered by their cognitive limits in understanding and addressing the dynamic and structural complexities residing in choice situations. Therefore they came up with a methodology to equip 'them' with the ability to conceptualize and design the architectures at different levels of abstraction. They used SD to create the possibility of problem identification and solution validation through simulation of scenarios. In this study however, we join scholars like [49, 118, 105] who have devised efforts to support the SD conceptualization process with tools and techniques used in other scientific disciplines. In SD two diagrams are most commonly used; Causal Loop Diagrams (CLD) and stock-flow diagrams (SFD). CLDs are qualitative in nature while SFDs are quantitative.

### 2.2.1 SD qualitative modeling

The qualitative phase of system dynamics involves the construction of 'influence diagrams' also known as 'Causal Loop Diagrams'. Causal Loop Diagrams are a visual representation of dynamic influences with inter-relationships amongst a collection of SD variables [188]. They are used to brainstorm on a given problem and to qualitatively capture structures and interactions of feedback loops [221] hence, allowing SD modelers understand how changes manifest in a problem domain. In earlier studies of SD (industrial dynamics) there were no CLDs. CLDs first made an appearance in 1968 in Forrester's book 'Principles of System Dynamics'. In this book he details a description of theoretical definitions and practical applications of the approach as a means of summarizing and explaining the behavior of fully specified simulation models [52].

In [228] Wolstenholme identifies some of the advantages and disadvantages of CLDs which we summarize here starting with advantages: 1) They contain a limited number of symbols with concentration on loop structure. 2) As a conceptualization tool they are easy to understand and use even for non-technical persons. 3) They are very compelling and persuasive. The disadvantages of CLDs include: 1) They do not clearly explain how flows influence stocks. 2) CLDs may lead to mislabeling of loops. 3) When CLDs are used in conceptualization, the resulting diagrams do not provide a sound basis for the rigorous deduction of behavior. Within the SD community there is consensus on the importance of qualitative data during the development of a system dynamics model, but there is not a clear description about how or when to use it. The lack of a defined
systematic procedure on how to obtain and analyze qualitative information creates a gap between the problem modeled and the model of the problem. This causes difficulty in “understanding the links between the observations of reality and the assumptions or formulations in the model especially when the model contains soft variables” [126].

2.2.2 SD quantitative modeling

The quantitative aspect of system dynamics entails the development of a stock and flow diagram where sets of equations are input into the model resulting into simulations. Graphically, the quantitative aspect displays the relationships between stocks and flows that contain underlying information of the model. Secondly, they offer a basis for rigorous deduction of dynamic behavior because their variables and link distinction can be used to explain a wider range of counter intuitive dynamic phenomena than CLDs [116]. SFDs bring together the modeler’s creative thinking ability and their data manipulation ability because they add the dimension of data to mapping of structures which then leads to computer simulation of systems to ascertain the model behavior over time [110, 228]. The derived simulations provide quantitative estimates of system effects and as such, models can be used in a “what if” mode to experiment with alternative configurations, flows and resources [108, 109, 110, 135].

The original concept of system dynamics was purely a computer simulation method centered on bringing the emerging power of computer simulation to analysis of complex socio-economic issues [228]. Computer simulation is a step by step operation of the model structure over compressed time and contains the dynamic behavior of the model [9]. The two main purposes of computer simulation are foresight and insight [177]. To arrive at these simulations, the modeler is expected to have input parameters into the model. These input parameters are inform of equations and must have real-life meaning, be dimensionally consistent, yield valid results, must not unrealistically assume optimality or equilibrium and realism should not be sacrificed for mathematical simplicity [81, 9]. SD simulations also use integral equations and aggregated viewpoints [177, 163, 51]. Two vital questions that may guide a modeler while conducting the simulation experiments are provided by kleijnen in [109] and these include: 1) which combinations of inputs should be simulated? 2) How can the resulting output be analyzed?

After obtaining the simulations, the next step is to conduct sensitivity analysis commonly known as ‘what if’ analysis in SD. Sensitivity analysis is defined as a systematic examination for simulation responses to extreme input model values [110]. Sensitivity analysis is divided into two phases; the pilot stage and regression analysis. The pilot stage searches for the important factors and the second stage approximates the input/output transformation that is implied by the simulation model. Better results are obtained from regression analysis if the simulation experiment is well designed. Sensitivity analysis answers two main questions; 1) what are the effects of changing input values? 2) Are there interactions among inputs? [109].

Tools used in developing SD models include; STELLA, Vensim, Powersim and iThink. These tools provide a practical way to dynamically visualize, communicate how complex systems really work and to derive simulations. They for example allow sim-
ulation 'run' systems over time, sensitivity analysis which reveals key leverage points and optimal conditions, partial model simulations which allows focus analysis on specific sectors or modules of the model. Results from these simulations are presented as graphs, tables and animations. Their files and dynamic data can be imported/exported to Microsoft Excel. The SD tools also contain some aggregation operators like SUM, MIN, MAX etc. to combine values of two or more quantities.

2.3 Conclusion

At the beginning of each section, we have defined the terminologies used. To a small extent, we have also discussed ORM and the two system dynamics representations intentionally leaving out their building blocks for they are presented in chapters 4. In the literature, we have stated the importance of combining methods, the use of ontologies in general, mentioned some of the existing ontologies, briefly discussed ontology integration, and presented literature on Object-Role and System Dynamics modeling.

As stated in the literature, SD offers a systematic approach of qualitative and quantitative analysis which include mapping systems in terms of feedback loops and then translating these maps into quantitative simulation models [122]. Secondly, SD also demonstrates how to identify policies that can improve a given scenario because as a method, it models, simulates and controls complex dynamic systems [141] and is supported by software that allows for process simulation. In the context of enterprise modeling SD is typically used in process analysis design and optimization.
CHAPTER 2. LITERATURE
Chapter 3

Requirements for this study

Requirements can be seen as high-level abstraction of services with precisely stated constraints that are provided by a method under which it functions [68, 218]. Vessey asserts that, since humans have limited information processors, decomposing the problem under study into sub-problems is of great importance [218]. He further says that, the key to successful problem decomposition is to structure the problem into sub-problems that can be integrated to form a complete solution. In this chapter therefore, we break the main problem into smaller manageable problems which we define in form of requirements. To start with, we present general requirements for combining methods in section 3.1 followed by specific requirements for this study in section 3.2. Combining methods can be viewed as a way of improving process development and its outcome. In our study however we combine System Dynamics with Object-Role Modeling. ORM is excellent at conceptualizing and explicitly representing processes in a problem domain while SD is good at analyzing and optimizing processes. Note that the former is in accordance with data modeling and the latter with process modeling. Although these two methods used in this study are applied and used differently, they are very important and play key roles in articulating and analyzing problems.

3.1 General requirements

As we stated in chapter 1, this study aims at combining system dynamics with a domain modeling method. In this section however, we provide some of the general requirements for combining methods. To arrive at these requirements we reviewed numerous scholarly works among which are: [21, 48, 68, 113, 124, 136, 147, 152, 224]. In these studies, scholars were looking at integrating various concepts or methods e.g. method integration, data integration, ontology integration etc. But since they all had a common interest in integration, we were able to use their approaches, insights and findings to arrive at the general requirements stated in this section. From the defined general requirements we derive specific requirements for this study and we present in section 3.2.
CHAPTER 3. REQUIREMENTS FOR THIS STUDY

**GR-1:** The methods and their underlying concepts should be explicitly defined.

A method is defined as an approach for performing system development projects based on a specific way of thinking. It consists of directions, rules, a well defined sequence of elementary operations and when executed, certain results are achieved [21, 136]. It is therefore important that the methods being combined are clearly defined and their constructs are well stated. By explicitly defining these methods, one is able to better understand their way of working, their origin (background), applicability and their constructs. Hence, a good starting point for the combining process since principles pertaining to each particular method are stated.

**GR-2:** The importance and boundary for combining the methods should be clearly stated.

Research on combining methods is conducted to solve existing problem(s). It is therefore important that the problem under study is well defined and backed with existing literature to avoid duplication and misunderstanding of the study. Secondly, the rationale for combining the methods should also be clearly stated and of what use the resulting combination is to the body of knowledge. The identification of the scope and purpose, serves to provide a reasonably well defined target for combining the methods. By scope we mean how far (extensive and detail of study) the researcher intends to go with the combining process. When the importance and boundaries for combining the methods are clearly stated, the researcher is able to 1) plan and schedule his activities 2) exclude unnecessary requirements 3) give the reader a clear guide on what the study is to cover 4) state the value or use of the study.

**GR-3:** A systematic way of working should be defined.

Way of working is discussed by different scholars [180, 170]. Defining the way of working (mode of operation) is important in guiding both the researchers and users (modelers) on how the researchers arrived at the results [170]. This mode of operation should be backed up with an existing methodology. Here we use the term methodology as the study of methods conducted methodically [21, 136]. To systematically combine methods, one is required to not only understand the methods but also have a clear view of the type of methodology to apply. The chosen methodology should be backed with existing literature to give a firm foundation to the study. Such a methodology may contain a number of step or guidelines. These steps should be followed systematically and the techniques applied should be well stated. If at all the researchers make changes in the referred methodology, they should clearly state it and give reasons if necessary. Note that an informative and clear description of the combination leads to better results and easy evaluation.

**GR-4:** Requirements for the study should be specified.

As stated at the beginning of this chapter; requirements can be seen as high-level abstraction of services with precisely stated constraints that are provided by a method under
3.1. GENERAL REQUIREMENTS

which it functions [68, 218]. In GR-4 however we refer to requirements pertaining to the procedure of combining methods. There exist numerous requirements but not all of them are relevant for a particular study therefore, it is important that the researcher specifies the requirements for his/her study. While identifying these requirements, the scope and objectives of the study need to be highly considered that way requirements defined are fit for the aim of the study.

**GR-5:** *The syntax and semantic of the constructs in each method should be clearly stated.*

Syntax have been defined in [187] as the way symbols may be combined to create well-formed sentences in the language, and semantics as correlating sentences and phrases with objects, thoughts and feelings of our experiences. For programming languages, Slonneger states that a computer follows a certain behavior when executing a program therefore the semantics describe this behavior [187]. Describing a step-by-step explanation of how a method will be executed (combined) is of great importance. As opposed to the semantics, the syntax defines the formal relations between the constituents of a method and provide a structural description of the various expressions that make up the method. The combined methods most times differ in some ways. But by defining the syntax and semantics in these methods the researcher is able to 1) understand the underlying principles better 2) identify the relationships between or among the method 3) provide a catalogue of all constructs in the methods.

**GR-6:** *The identified relationships between or among the methods should be explicitly defined.*

Defining the relationships between or among these methods help to bridge the existing gap and finding a common link between or among the methods [68]. Secondly, they help in identifying transitional statement for each construct relationship. It would be good to have all constructs in one method relating to constructs of another method but that would mean having the same method relating to another. In most cases however, some of the constructs have identifiable relations but not very concrete. Therefore being unable to relate all constructs may lead to a number of problems e.g. ambiguity (a construct from one method may be mapped to several other constructs in another method), inconsistency (existence of contradicting relationships), incompleteness (some constructs may not be mapped to any construct in another method) and incorrectness (constructs considered to be appropriate for the combination) [104].

**GR-7:** *A procedure for combining these methods should be defined.*

In [22] it is stated that bringing together a meaningful method requires a procedure to enforce some constraints or rules on the process. Through defining the procedure, a new method is derived. This method may comprise of different phases with numerous steps [86, 85]. A method from the process perspective usually consists of stages, which stages are further divided into individual steps [22]. By stating the existing phases and their steps, the researcher is able to 1) Plan on how to tackle each phase 2) Draw objectives for each phase 3) Decide on the evaluation procedure.
CHAPTER 3. REQUIREMENTS FOR THIS STUDY

GR-8: *The method(s) or techniques applied within the combining process should be explicitly defined.*

In the process of combing methods there are numerous sub-processes among which is application and evaluation. In order to complete each of these sub-processes, certain criteria need to be followed where methods and techniques are applied. The methods we are referring to in this requirement are not similar to the ones mentioned in GR-1. Examples of such methods are used at the application and evaluation phase, and may include: case study, analytical methods, experimental methods etc [201, 136]. Within these methods there are various techniques e.g. observation, structured walkthrough, interview, questionnaire, focus group etc. These techniques should also be clearly defined and backed with literature as to why they are being used and of what importance they are to the study.

GR-9: *For each phase in the combining procedure, guiding steps should be provided and evaluated.*

Evaluation is a more inclusive term often making use of assessment data in addition to other data sources. It is the process of determining the worth of the study [40]. Evaluating the procedure guidelines and the resulting model(s) is the ultimate goal of making the study’s worth and may lead to refinement of both the guidelines and the resulting model(s). The importance of evaluation is to determine or gauge how well the results (new combination) work and requires the development of a metrics. These metrics must be scrutinized by experimental analysis [127]. By refining the resulting combination, the earlier results are modified and more details added. This however does not change the intention of the evaluated results; it instead corrects them, and makes them more accurate. Therefore, refinement can be seen as the process of improving something or the process of removing impurities. Basing on the refined method, guidelines for tool support can be realized.

3.2 Specific requirements

Having identified some of the general requirements for combining methods, we now present requirements specific to the study (combining system dynamics with a domain modeling method). Most of these requirements are derived from the general requirements presented in section 3.1.

R-1: *The constructs and underlying principles of both methods (SD and ORM) should be explicitly defined and the identified relationships should be well documented.*

In our study, we started off by reviewing considerable literature that conforms to the methods we are combining. This enabled us to: understand the concepts in both methods, identify relationships in the methods, have a clear understanding of the methods background, know areas where they are applied and how, importance or use of the methods,
3.2. SPECIFIC REQUIREMENTS

and strengths and weaknesses in these methods. Part of the literature is presented in chapter 2 and the other part is scattered all over the thesis chapters.

In chapter 4 we present ORM and SD constructs. After defining these constructs we identified the relationships and transformation statements that exist in-between these methods presented in chapter 5. By identifying the different constructs we were able to better understand SD and ORM underlying principles.

R-2: Methodically define a procedure for combining the two methods.

Following the design science methodology (DSR) guidelines provided by Hevner in [86] see table 3.1, he identified designing an artifact as the first step. As we stated in chapter 1 section 1.5, we tailor the DSR methodology to suit our way of working without tampering with it. In this study therefore before we define the artifact we had to define the constructs and underlying principles for both methods thereafter identify the existing relations between the method constructs. Having identified the relations we then defined the new artifact which we refer to as Grounded System Dynamics (GSD). GSD is made up of two phases each with guiding steps. System dynamics as mentioned in chapter 2 is made up of two diagrams CLDs and SFDs. To underpin SD with ORM we considered both diagrams leading to two phases in the GSD method. In chapters 5 and 7 we apply various evaluation methods for both the relations and the GSD procedure. Through these evaluations we were able to obtain feedback that was used to refine the relations and the GSD procedure. Guideline 6 states that the search for an effective artifact requires utilizing available means to reach a desired goal. In this study therefore we have used available methods to study the combination of SD with ORM without introducing new techniques or methods. In our evaluations we used practitioners and academicians as our audience for this study.

R-3: The new method must have a theoretical foundation to enable consistent derivation of an evaluation statement.

In this study we follow the design science methodology. A methodology often comprises of various methods and techniques [136]. Here we use the term ‘methodology’ as used in [21, 32, 136] which is: the study of methods with structured sets of guidelines aiming to assist in producing valid and dependable research. In the design science research methodology there are various evaluation methods among which are experimental and analytical evaluation methods [86]. For analytical evaluations we used structured walkthroughs [184] and questionnaires that enabled us have an in-depth analysis of the GSD steps. For experimental evaluations, we used focus group discussions and provided questionnaires to participants to evaluate the GSD transformation steps (for results see chapter 7).

The DSR methodology contributes towards innovation and utility of constructed artifacts [128, 86]. In DSR, artifacts are derived as a final output of the process. They are defined as constructs, models, methods, and instantiations created to enable representation, analysis, understanding, and development of successful information systems within
organizations [128, 127]. To attain these artifacts, March provides six requirements (1) identification and clear description of a relevant organizational IT problem, (2) demonstration that no adequate solutions exist in the extant IT knowledge-base, (3) development and presentation of a novel IT artifact (constructs, models, methods or instantiations) that addresses the problem, (4) rigorous evaluation of the IT artifact enabling the assessment of its utility, (5) articulation of the value added to the IT knowledge-base and to practice, and (6) explanation of the implications for IT management and practice [128]. In addition to the requirements, Hevner defines a list of seven guidelines as presented in table 3.1 and three design science research cycles in [85]. Considering the six requirements, the three DSR cycles and following the design science guidelines; the requirements for a DSR project can be met. Note that in design science the how, where and when to apply these guidelines is left to the researcher(s). Minus the design science methodology, in

<table>
<thead>
<tr>
<th>Guideline 1: Design an artifact</th>
<th>Design Science Research (DSR) Guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSR must produce a viable artifact inform of a construct, a model, a method or an instantiation.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Guideline 2: problem Relevance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The objective of DSR is to develop technology-based solutions to important and relevant business problems.</td>
<td></td>
</tr>
</tbody>
</table>

| Guideline 3: Design Evaluation | The utility, quality and efficacy of a design artifact must be rigorously demonstrated via well executed evaluation methods. |

| Guideline 4: Research Contributions | Effective DSR must provide clear and verifiable contributions in the areas of the design artifact, design foundations and/or design methodologies |

| Guideline 5: Research Rigor | DSR relies upon the application of rigorous methods in both the construction and evaluation of design artifacts. |

| Guideline 6: Design as a search process | The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment. |

| Guideline 7: Communication of research | DSR must be presented effectively both to technology oriented as well as management oriented audiences. |

this study we also put into consideration requirements for combining methods, method integration [147, 113] and system dynamics model building requirements [193].

R-4: The phases in the new method should be explicitly defined, evaluated.

As already stated, GSD method is made up of two phases. In each phase there are guiding steps on how to transform one method component into another method component (see chapter 6). In chapter 7, we present evaluations for the GSD method were a group of practitioners and academicians from both methods are asked to critically review the steps in each phase and support their answers in structured walkthrough and focus group sessions. To apply and evaluate the guiding steps, both examples and cases were used.

R-5: The stock and flow diagrams derived must have valid input parameters and a logical explanation for the resulting simulations should be given.

In system dynamics stock and flow diagrams, a modeler has to input parameters in each variable. These parameters are quantities that are not directly observable and through influences from other variables a behavioral relationship is created. These input parameters
determine the simulation results obtained from the model. The parameters must conform to the variable input requirements which are dependent on a number of factors e.g. influences to a specified variable, role of the variable etc. The behavioral relationships are what we refer to as simulations. These simulations are meant to be more realistic and reliable. For each example and case presented in this study, we present a sample of the resulting simulations and briefly discuss their behavioral relationship.

**R-6: Suggestions for organizational and tool support should be provided.**

In [170] it is stated that method guidance is important and acknowledged. Under this requirement therefore, we discuss organizational involvement in the modeling process, some of the approaches used in organizational modeling and how the derived artifact can be operationalized (see chapter 9). As stated in [35], organization involves arranging concepts into spatial relationships that represent conceptual relationships. Organization activities involve abstracting or creating to identify a set of high-level concepts around which existing concepts can be organized [35]. As a final step to this study therefore, we provide some suggestions for organizational and tool support. This step is important because it facilitates the application of an evaluation approach in practice and provides views on what directions the research should take in future.

### 3.3 Conclusion

In conclusion, since DSR is the key methodology used in this study, we found it fit to give a brief overview of DSR guidelines and requirements. With the provided list of requirements and guidelines, DSR derived artifacts are said to be rarely full-grown information systems that can be used in practice. But they are innovation that define ideas through which implementation and use of information systems can be accomplished [86].

<table>
<thead>
<tr>
<th>Question Type</th>
<th>General Requirements</th>
<th>Specific Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methodological</td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>Conceptual</td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>Related work</td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>Practical Experience</td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓</td>
</tr>
</tbody>
</table>

In table 3.2 we summarize the requirements presented in this chapter. Alongside the requirements, we present the type of question a particular requirement captures. These requirements are mainly derived from existing related works and practical experiences. Basing on the general requirements presented in section 3.1, we have presented and discussed the requirements specific to our solution approach in section 3.2. In chapter 9, we present some of the suggestions for organizational and tool support. Under the same chapter we give insights to future works where we elaborate on what we have been able to achieve and what more should be done.
Chapter 4

Constructs and underlying principles

In this chapter, we present the constructs and underlying principles of Object-Role Modeling and System Dynamics. Each of these modeling methods uses a specific way of thinking, rules and directions on how to model an aspect of a system [21]. These rules and directions are what we refer to as *constructs*. They specify what can be modeled with a given method and define the world view of the method [180, 185]. Here we use the term constructs as concepts, ideas or images specifically conceived for the purpose of organizing and representing knowledge of interest of a given modeling method [86]. As stated in [224], understanding the underlying ideas of different methods helps in defining their transition and relations. If we identify the common core of System Dynamics (SD) and Object-Role Modeling (ORM), then their differences are also identified. In every modeling method there is a set of constructs that define the vocabulary of the method. We therefore believe that before we identify the relations in System Dynamics constructs with Object-Role Modeling constructs, it is important that we identify the constructs used in each method and also explain their underlying principles. This is because they state what can be modeled in a given method and what it entails. To start with, we present constructs that exist in Object-Role Modeling followed by System Dynamics constructs.

4.1 ORM concepts and constructs

The philosophy behind ORM is that it tries to describe a Universe of Discourse (UoD) by describing the communication between its members. An ORM scheme basically is a grammar describing that communication. This grammar is also referred to as information grammar. The general construction of an information grammar is as follows. There is a set of syntactic categories (in ORM terminology: object types) and a set of grammar rules (in ORM terminology: fact types) that describe how these syntactic categories are constructed from other syntactic categories. A grammar rule basically indicates what
object types are involved in a fact type and in what role. The term predicator is used to indicate such a role. Therefore, in ORM a fact type is seen as a set of roles.

The information grammar describes the elementary sentences that are valid in the associated UoD. From these sentences other sentences may be formed. Object Role Calculus (ORC) ([88]) and ORM2 ([76]) are examples of such generic systems for constructing sentences. These sentences will be referred to as information descriptors. The notion of information descriptors was introduced under LISA-D (Language for Information Structure and Access Descriptions) which is based on PSM (Predicator Set Model) in [88]. ORC linked things more explicitly to logic. ORC is the base for formal reasoning about the UoD. A major aspect during the design phase may be a (quantitative) simulation of the model in order to find bottlenecks for the implemented system. Such simulations are quite common in construction technology, for example to prohibit a bridge from destruction by a too heavy load.

ORM's basic building blocks include: entity types, value types and roles [77]. An object type is a collection of objects with similar properties, in the set-theoretical sense. Objects are things of interest, they are either instances of entity or value types. Entity types are designated by solid-line named ellipses in the graphical reproduction of the information grammar. All entity types have a reference scheme, which may be simple (either a reference mode, or an entity to entity relationship) or compound. These reference schemes indicate how a single value relates to that entity type. Value types on the other hand have instances with a universally understood denotation, and hence require no reference scheme. They are identified solely by their values, their state never changes and they are designated by dotted ellipses. The semantic connections between object types are depicted as combinations of boxes and are called fact types. Each box represents a role and is connected to an object type or a value type. The roles denote the way entity types participate in that fact type. To represent some of these definitions let us use an example of the procedures a paper might go through en route from writing to publication. The procedures are stated as:

S1 A person (author) communicates the intent to submit. This can be in the form of an abstract.
S2 Then the content (text of the paper) is submitted, whereby the paper becomes a submitted paper.
S3 Each submitted paper receives a classification.
S4 Each submitted paper is reviewed.
S5 Some submitted papers are accepted, the others are rejected
S6 For each accepted paper a final version is submitted, making the paper a published paper.
S7 Rejected papers are not further processed.

---

1For ORM terminologies in this study, we used Halpin and Morgan [77]; and to model ORM models we used Natural ORM Architect (NORMA).
4.1. ORM CONCEPTS AND CONSTRUCTS

Figure 4.1: Various paper states

Note that the procedure basically describes the various subsequent states recognized for the object type Paper. It is this notion of state that we want to further explore in this thesis, and how states come in naturally during the modeling process. For the above example, the states are displayed in figure 4.1. The conceptual structure of this example is represented on an ORM diagram in figure 4.2.

Figure 4.2: Paper flow concepts in ORM

The numerous ORM symbols and constraints used in figure 4.2 can be verbalized as follows:

C1 (Exclusive): For each Paper, at most one of the following holds: that Paper is rejected or that Paper is accepted.
C2 If some Paper is published then that Paper is accepted.
C3 (mandatory): each paper is submitted by at least one academic.
C4 (mandatory): each paper is written by at least one academic.
C5 If some Academic submits some Paper then that Academic writes that paper.
C6 (Exclusive): For each Academic and Paper, at most one of the following holds:
that Academic writes that Paper; that Academic is assigned to review that Paper.
The number of roles in a fact type are referred to as fact type arity and the semantics of the fact type are put in a fact predicate. A predicate is basically a sentence with object holes in it, one for each role. These predicate names are written beside each role and are read from left to right, or top to bottom. It is through predicates that entity types relate to each other.

Minus the building blocks, ORM also has numerous constraints. Some of these constraints are depicted in figure 4.2 (subset constraint, exclusive constraint, uniqueness constraint and mandatory role constraint). A mandatory role constraint (represented as a dot) means that every instance in the population of the role’s object type must play that role. The uniqueness constraint (internal) means that instances for that role in the relationship type population must be unique. For example, adding a uniqueness constraint over role ‘is classified’ in figure 4.2 means that each paper is classified in at most one class. The subset constraint between roles ‘is accepted’ and ‘is published’ means that if some paper is published then that paper is accepted. Here we do not exhaust all ORM constraints but a detailed explanation of these constraints can be found in [77].

4.2 SD concepts and constructs

As already stated in chapter 2, SD is made up of two notations, Causal Loop Diagrams (CLDs) and Stock and Flow Diagrams (SFDs). Each of these diagrams has a number of constructs with underling principles. In subsections 4.2.1 and 4.2.2 respectively, we present their basic building blocks and underlying principles.

4.2.1 Causal loop diagrams

A Causal Loop Diagram is made up of variables, signs (either a positive or negative) and causal links with arrows representing the causal influence. The arrows are drawn in a circular manner indicating the cause and effect leading to a feedback loop which is a closed sequence of causes and effects sometimes referred to as a closed path of action and information [167].

Constructing a causal loop diagram

To construct a CLD there are different design rules that can be followed. An illustrative summary of these rules is presented in table 4.1. To clarify these rules, let us take figure 4.3 which is an annotated Causal Loop Diagram for a simple process a paper might go through from writing to publication as an illustrative example. This diagram includes elements, arrows (which are called causal links) and signs (either + or -) placed at the arrow tip of the arrows. These signs have the following meanings:

\[\text{For SD terminologies used in this study we use Sterman [193]. All SD stock and flow diagrams are drawn using an SD software called STELLA and all Causal Loop Diagrams were drawn using Vensim. This is because the two platforms are easy to use and offer a practical way to dynamically visualize and communicate how}\]
4.2. SD CONCEPTS AND CONSTRUCTS

Table 4.1: A summary of the causal loop diagram rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>All else equal: A change in A leads to a change in B in the same direction.</td>
</tr>
<tr>
<td>2</td>
<td>All else equal: A change in B leads to a change in A in the opposite direction.</td>
</tr>
<tr>
<td>3</td>
<td>A loop is reinforcing (with an 'R' sign in parentheses) when the number of negative links (polarities) is even.</td>
</tr>
<tr>
<td>4</td>
<td>A loop is balancing (with a 'B' sign in parentheses) when the number of negative links (polarities) is odd.</td>
</tr>
<tr>
<td>5</td>
<td>All else equal: A change in B leads to a change in A in the opposite direction. However, there is a time lapse (lag) before the actual change in A is noticed/evident.</td>
</tr>
</tbody>
</table>

Details:
- Influence arrow
- Reinforcing Loop
- Balancing Loop
- Polarties

Figure 4.3: A demonstrating example of a causal loop diagram

Starting from element Written papers at the top of the diagram, if the number of Written papers increase, the number of Submitted papers also increases thus a positive sign is added to the link i.e. if Written papers increase, Submitted papers will always be higher than they would have been (for further explanation see [193] pg. 141). An increase in Submitted papers leads to an increase in Reviewed papers thus a positive sign is added to the link. If the Reviewed papers increase, both Accepted Papers and Rejected papers also increase. An increase in Accepted Papers leads to an increase in Published papers thus a balancing loop (B) is created. On the other hand if the number of Reviewed papers complex systems and ideas really work.
increase, the number of *Rejected papers* also increases. Thus a positive sign is added to the link between these two elements. However, an increase in the number of *Rejected papers* leads to a decrease in the number of *Published papers* thus a negative sign is added to the link from *Rejected papers* to *Published Papers*. i.e. if *Rejected papers* increase, *Published Papers* will always be lower than they would have been *(for further explanation see [193] pg. 141).* This then is aligned with the example description from section 4.1.

Considering the reinforcing loop in figure 4.3, a typical start for the reinforcement loop is when more papers are being written. But in this example the reinforcing loop starts from *is submitted by* (submitted papers).

Note that there is a delay mark (denoted by two short lines across the causal link) on the arrow from variable *Published papers* and *Submitted papers* indicated with double slashes. This delay mark implies that there is a time lapse (lag) before the actual increase in *Submitted papers* is noticed (becomes evident). That is to say, there is a time delay between the action (increase in published papers) and the observation result (increase in submitted papers) of this action. This delay is a vital determinant of system dynamics behavior and is as a result of various factors in the system [193]. Before we conclude our discussion on CLD constructs, We present a summary of the design rules for a CLD.

**Design Rule 1 (Positive sign):** A causal link from one element ‘A’ to element ‘B’ is positive (+) if either ‘A’ adds to ‘B’ or a change in ‘A’ makes variable ‘B’ change in the same direction. Using figure 4.3 as an example, a causal link from variable *Written papers* to variable *Submitted papers* is positive because an increase in the number of *Written papers* leads to an increase in *Submitted papers* i.e. each time *Written papers* increase, *Submitted papers* also increase so they move in the same direction. Sometimes instead of using a positive sign and a negative sign, either an ‘S’ (Same) or an ‘O’ (Opposite) is used. An ‘S’ means the same as a positive sign and an ‘O’ means the same as a negative sign. Both representations are accepted by the system dynamics community.

**Design Rule 2 (Negative sign):** A causal link from one element ‘A’ to another element ‘B’ is negative (-) if either ‘A’ subtracts from ‘B’ or a change in ‘A’ makes ‘B’ change in the opposite direction. Using the same example in figure 4.3, a causal link from variable *Rejected papers* to variable *Published papers* is negative because *Rejected papers* subtract from the *Published papers*. Therefore a change in *Rejected papers* causes the *Published papers* to change in an opposite direction.

In addition to the signs on each link, is a complete loop sign (either a positive (Reinforcing) or negative (Balancing)). The sign for a particular loop is determined by counting the number of minus (-) signs on all the links that make up that loop.

**Design Rule 3 (Reinforcing Loop):** A feedback loop is called positive or reinforcing, indicated by a plus or ‘R’ sign in parentheses, if it contains an even number of negative causal links. For example in figure 4.3, a reinforcing loop starts from variable *Submitted papers*, to *Reviewed papers*, to *Accepted papers*, to *Published papers* and back to *Submitted papers*.

**Design Rule 4 (Balancing Loop):** A feedback loop is called negative or balancing, indicated by a minus or ‘B’ sign in parentheses, if it contains an odd number of negative causal links. For example in figure 4.3, a balancing loop starts from variable *Submitted papers*.
papers, to _Reviewed papers_, to _Rejected papers_, to _Published papers_ and back to _Submitted papers_.

Thus, the sign of a loop is the algebraic product of the signs of its links. Often a small looping arrow is drawn around the feedback loop sign to more clearly indicate that the sign refers to the loop. Further explanation on how CLD influences operate can be found in [176, 193].

**Design Rule 5 (Delay Mark/Time Delay):** Between variables ‘B’ and ‘A’ in table 4.1 and ‘S6’ and ‘S2’ in figure 4.3, is a delay mark. This delay mark implies that there is a time lapse (lag) between these variables before the actual change is noticed or becomes evident. Delays are of two types: material delays and information delays. Material delays represent a lag in the physical flow while information delays represent gradual adjustment of people’s belief. Identifying delays is an important step in the system dynamics modeling process because they often alter a system’s behavior in significant ways. The longer the delay between cause and effect, the more likely it is that a decision maker will not perceive a connection between the two [193]. A detailed explanation of delays can be found in [193] pg. 409.

**Formal definition of causal loop diagram**

At the beginning of this section, we stated that a Causal Loop Diagram is made up of _variables_ and _causal links_ with arrows representing the causal influence. Causal links have associated a _sign_ (either a positive or negative) and may have an associated _delay_.

A causal link expresses a causal relationship between two factors. If the link has an associated positive sign, then the link expresses a positive influence/relation. We write $F \rightarrow^+ G$ (see design rule 1) to express that a change in variable $F$ causes a similar change in variable $G$. We assume there is a time delay between the cause and its effect; this time delay does not have a lowerbound on its duration (which makes it different from the delay that may be associated with a causal link). When the causal link is effected at time $t$, then it relates the situation of the cause at time $t^-$ with the effect at time $t^+$ (using standard notation for calculus [200]). Note that in this section, we present lemmas and their proofs. These lemmas are only illustrative and are not part of our research results (not linked to any of our research questions). They are however provided as a service to the reader so (s)he can get a better appreciation of SD.

**Lemma 4.2.1** $\rightarrow^\sim$ is a transitive relation.

**Proof:**

Suppose $F \rightarrow^+ G$ and $G \rightarrow^+ H$. Then, because of $F \rightarrow^+ G$, a change of variable $F$ causes a similar change to variable $G$, which in turn leads to a similar change of variable $H$ hence $G \rightarrow^+ H$. Consequently, a change in variable $F$ leads to a similar change of variable $H$, or, $F \rightarrow^+ H$.

A causal link that has associated a minus-sign (see design rule 2) expresses a negative influence relation. So the sign of change in the effect variable is opposite to the change in
the cause variable. We use the notation $F \rightsquigarrow G$ for this case. However, the relation $\rightsquigarrow$ is not a transitive relation:

**Lemma 4.2.2** If $F \rightsquigarrow G$ and $G \rightsquigarrow H$, then $F \rightsquigarrow H$.

**Proof:**

Suppose $F \rightsquigarrow G$ and $G \rightsquigarrow H$. Then, because of $F \rightsquigarrow G$, a change in variable $F$ causes a change in variable $G$ in the opposite direction. This change of variable $G$, because of $G \rightsquigarrow H$, leads to a change in variable $H$ in the direction opposite to the change in variable $G$. Consequently, a change of variable $F$ leads to a similar change of variable $H$, or, $F \rightsquigarrow H$.

Next we consider the combination of positive and negative influence.

**Lemma 4.2.3**

If $F \rightsquigarrow^+ G$ and $G \rightsquigarrow H$, then $F \rightsquigarrow H$.

If $F \rightsquigarrow^+ G$ and $G \rightsquigarrow H$, then $F \rightsquigarrow H$.

**Proof:**

Suppose $F \rightsquigarrow^+ G$ and $G \rightsquigarrow H$. Then, because of $F \rightsquigarrow^+ G$, a change of variable $F$ causes a change of variable $G$ in the same direction. This change of variable $G$, because of $G \rightsquigarrow H$, leads to change of variable $H$ in the direction opposite to the change in variable $G$. Consequently, a change in variable $F$ leads to an opposite change of variable $H$, or, $F \rightsquigarrow H$.

Suppose $F \rightsquigarrow G$ and $G \rightsquigarrow H$. Then, because of $F \rightsquigarrow G$, a change of variable $F$ causes a change in variable $G$ in the opposite direction. This change of variable $G$, because of $G \rightsquigarrow H$, leads to change of variable $H$ in the same direction as the change of variable $G$. Consequently, a change in variable $F$ leads to an opposite change of variable $H$, or, $F \rightsquigarrow H$.

Let $F_1, \ldots, F_n$ be variables, such that $F_i \rightsquigarrow^+ F_{i+1}$ or $F_i \rightsquigarrow^+ F_{i+1}$ for each $1 \leq i < n$, then we call $[F_1, \ldots, F_n]$ a causal path from $F_1$ to $F_n$. This brings us to the following conclusion:

**Lemma 4.2.4** Let $P$ be a causal path from $F$ to $G$, then we have:

$F \rightsquigarrow^+ G$ if the number of negative influences in path $P$ is even

$F \rightsquigarrow^+ G$ if the number of negative influences in path $P$ is odd

**Proof:**

We prove the statement by induction on the length of the causal path $P$.

- For a length 1 path $P = [F, G]$, we have the following cases:
  1. $F \rightsquigarrow^+ G$: then also the number of negative influences on path $P$ is even
2. $F \not\rightarrow G$: then also the number of negative influences on path $P$ is odd

- Suppose the property holds for paths of length $n$, let $P$ be a path of length $n + 1$ from $F$ to $H$. Then we can decompose $P$ as a path of length $n$ from $F$ to some $G$, and path of length 1 from $G$ to $H$. We have the following cases:
  1. $G \not\rightarrow H$: then the number of negative influences on path $[F,G]$ is the same as on the path $[F,H]$. The property now follows from lemma 4.2.1, lemma 4.2.3 and the induction hypothesis.
  2. $G \not\rightarrow H$: this case is similar to the previous case.

A path $P$ from $F$ to $F$ is referred to as a causal loop. The following lemma formalizes design principles 3 and 4:

**Corollary 4.2.1**

If loop $P$ from $F$ to $F$ contains an even number of negative influences, then we can conclude $F \not\rightarrow F$. This means that any change of variable $F$ is reinforced by loop $P$. On the other hand, if path $P$ contains an odd number of negative influences, then any change of variable $F$ is damped by an opposite change by loop $P$.

**Proof:**

Direct consequence of the previous lemma.

Note that these lemmas are no fundamental contribution of the thesis, but are rather intended to aid in explaining the workings of SD.

### 4.2.2 Stock and flow diagram

The Causal Loop Diagram describes variables and how they influence each other. The stock and flow diagram is a materialization of the Causal Loop Diagram, as an easy to use framework for setting up differential equations. The stock and flow diagram is made up of the following building blocks: stocks, flows (inflow and outflows), converters (auxiliary and constant), sources and sinks.

**Constructing a stock and flow diagram**

*Stocks* are depicted as boxes and are defined as containers (reservoirs) containing quantities describing the state of the system. The value of stocks changes overtime through flows (inflows and outflows) [24]. There are different forms stocks can take (reservoir, conveyor, queue, oven) see table 4.2. In each of these forms, quantities held are in a different state as explained in the table.

*Flows* can be imagined as pipelines with a valve that controls the rate of accumulation to and from the stocks. They are represented as double solid lines with a direction arrow. The arrows indicate the direction of a flow into or from a stock. There exists two types of flows; uniflows and bi-flows as represented in figure 4.4. A uniflow means that
### Table 4.2: Forms of stocks, their functions plus their examples

<table>
<thead>
<tr>
<th>Stocks</th>
<th>Functions</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reservoir</td>
<td>A reservoir is a default stock type. A Reservoir is total number of quantities. It passively accumulates its inflows, minus its outflows at each DT.</td>
<td>Population, water in a tank</td>
</tr>
<tr>
<td>Queue</td>
<td>Think of a Queue as a line of items awaiting entry into some process or activity. Queues are FIFO (first in - first out) in their operation. Stuff enters the queue, and remains in line, waiting its turn to exit the Queue.</td>
<td>Grocery store checkout line, airport ticket counter line</td>
</tr>
<tr>
<td>Oven</td>
<td>Think of ovens as batch production system like a bakery process. It opens its doors; fills (either to capacity or until it is time to close the door); bakes its contents for a time (as defined by its outflow logic); then unloads them in an instant.</td>
<td>Elevators depend on door opening or closing to ride and can have a queue waiting to ride in it.</td>
</tr>
<tr>
<td>Conveyor</td>
<td>Think of a Conveyor as a conveyor belt. Material gets on the Conveyor, rides for a period of time (each DT) and then gets off.</td>
<td>Pregnant women, students in school (they ride in a particular class for a period of time and then ride off).</td>
</tr>
</tbody>
</table>

Information in that flow moves (flows) in one direction only and the flow takes on non-negative values only. A bi-flow on the other hand, can take on any value and information flows in two directions. Flows originate from a source and terminate in a sink which are depicted as clouds.

A source represents systems of stocks and rates outside the boundary of the model and a sink is where flows terminate outside the system. A sink is located at the arrow tip of the flow and a source is found at the start of the flow arrow.

Converters either represent fixed quantities (constants) or represent variable quantities (Auxiliaries). Auxiliary variables are informational concepts bearing an independent meaning (add new information). The contained information is in form of equations or values that can be applied to stocks, flows, and other converters in the model [121]. Constants are state variables which do not change [24]. Both auxiliary variables and constants are depicted as small circles on the STELLA SD software. Information from converters and flows is shared through connectors (information links). Two types of connectors exist, the action connectors depicted as solid wires and information connectors depicted as dashed wires [199]. These connectors are immaterial and connect inputs to decision function of a rate. The underpinned meaning to these connectors is that information about the value at the start of the connector influences information at the arrow tip of that connector. Connectors can feed information into or out of flows and converters but only extract information out of stocks [121]. Lastly, we have the concept of sectors.
which are subsystems or subcomponents within a system. They hold/handle all decisions, stocks, information about a particular element or area and contain different information used in an information system. Sectors are not represented in any of the figures but are introduced in one of our later chapters.

Note that among converters we only mention auxiliary and constants but not exogenous variables as building blocks. This is because exogenous variables although they are part of the SFD model, their values are determined by factors outside the model. Secondly, not all SFD models contain exogenous variables, this means that a model can be complete without any external influence(s).

In conclusion, we present a summary of all the discussed stock and flow building blocks except sectors in figure 4.5 followed by some of the SFD design rules.

<table>
<thead>
<tr>
<th>Key:</th>
<th>Stock some scholars refer to it as a Level</th>
<th>Flow (Inflow or Outflow)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sink/Source</td>
<td>Converter/exogenous variable</td>
</tr>
<tr>
<td></td>
<td>Decision process (STELLA) and a Constant (Vensim)</td>
<td>Information connector/ link</td>
</tr>
<tr>
<td></td>
<td>Action</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 4.5: A summary of SFD basic building blocks](image)

**Design Rule 6 (Stocks):** Each stock should have an inflow attached to it. Through information links, a stock can influence all other variables (converters, exogenous variables or other stocks) but can only be influenced through a flow. In other words, there is no direct connection to a stock other than through flows.

**Design Rule 7 (Flows):** Every flow is influenced by another variable (stock or converter) in the model through connectors (information links). This enables the values in either the inflows or outflows to change the contents in the stock. If there is no variable in the model influencing a flow, then it becomes inactive and the rates in the flows cannot be defined. For a rate to be defined, there must be at least one connector influencing that flow. Thus flows can be influenced by stocks, converters and exogenous variables but cannot directly influence converters and exogenous variables or other flows.

**Design Rule 8 (Converters):** As we stated earlier there are two types of converters; a constant and an auxiliary. Converters should be influenced by at least two or more elements in the model. These elements can either be dynamic or static. Converters and exogenous variables can influence flows or other converters and exogenous variables.

**Design Rule 9 (Sink and Source):** A sink and source exist on flows that do not originate
CHAPTER 4. CONSTRUCTS AND UNDERLYING PRINCIPLES

from or terminate into a stock.

**Design Rule 10 (Information links):** Information links can feed information into or out of flows, constants, auxiliary variables and exogenous variables but only extract information out of the stock.

**Formal definition of stock and flow diagram**

A stock is the visualization of the variable of the Causal Loop Diagram. Typically variables represent the relevant states that are distinguished for that type of object. A stock represents a state. A state change corresponds to an object flowing from one stock into another. Consequently, an object can be in at most one stock at each moment.

Causal diagram links correspond to flows in the stock and flow diagram. The size of stock $S$ at time $t$ is denoted as $\text{Stock}_t(S)$ thus definition of stock as a basic concept. We will give a property of stock in terms of differential equations below. Let $\text{Flow}_t(S \rightarrow T)$ be the size of the flow from stock $S$ to stock $T$ via the flow from $S$ to $T$. The total inflow in stock $S$ via any incoming flow at time $t$ is denoted as $\text{Inflow}_t(S)$, the total, outflow is denoted as $\text{Outflow}_t(S)$ (see [193]). So we have:

$$\text{Inflow}_t(S) = \sum_{T \rightarrow S} \text{Flow}_t(T \rightarrow S)$$

$$\text{Outflow}_t(S) = \sum_{S \rightarrow T} \text{Flow}_t(S \rightarrow T)$$

The stock accumulation is expressed as: at any moment $t \geq t_0$. Where $t_0$ is the starting time.

$$\text{Stock}_t(S) = \text{Stock}_{t_0}(S) + \int_{t_0}^{t} [\text{Inflow}_u(S) - \text{Outflow}_u(S)] du$$

As a consequence the change of flow is expressed as:

$$\frac{d(\text{Stock}_t(S))}{dt} = \text{Inflow}_t(S) - \text{Outflow}_t(S)$$

**4.3 Using ORM as a foundation for SD**

In this section an ORM diagram is augmented and replaced by process concepts that lean towards SD like conceptualization.

**4.3.1 An example**

We use figure 4.2 which is an ORM diagram of events (reported as elementary facts) that may be observed in a domain (in this case, the reviewing domain). This approach is in line with the PSM2 [210] approach. Note the constraints requiring that the submitting academic is indeed one of the authors of a paper, and that a reviewer of a paper cannot be
author of that paper. Next, we add temporal dependencies between the roles associated to the paper. This leads to the flow depicted in figure 4.6. The left hand side depicts the full diagram based on the facts types (event types) in the original ORM diagram. The diamond shape is the BPM [223] symbol for a XOR split. Our interest is in the flow statics of submissions, reviews, acceptance, rejection and publication. This leads to the abstracted view depicted on the right hand side.

Figure 4.6: Paper flow
Next we now make explicit the relations between, in particular, the ORM model and stock-flow model. In figure 4.7 the left hand side shows figure 4.6 (right) with some extra information. The extra information pertains to the flow based interpretation. We now see stores of papers that are ready to flow from one state to another. Each time a paper ‘flows’, this is an event (the original events related to ORM diagram, figure 4.2). So:

- A paper is **reviewed**
- A paper is **decided upon**
- A paper is **accepted**
- A paper is **rejected**
- Etc.

Associated to the event-types, we can now also add a rate. Leading to:

- Review rate
- Acceptance rate
- Rejection rate
- Acceptance rate
- Etc.

The right hand side then depicts the SD diagram. Using the SD software, the prelude
4.3. **USING ORM AS A FOUNDATION FOR SD**

to the complete SD Stock and Flow diagram is depicted in figure 4.8. In this figure, when an academic submits a paper, it flows through *submission rate* and is added to stock *submitted papers*, after that it is reviewed and flows through *review rate* to stock *papers to be decided upon*. Stock *submitted papers* accumulates through inflow *submission rate* and is drained through outflow *review rate*. Therefore the value of stock *submitted papers* is determined by these two flows. From stock *papers to be decided upon* the papers move in two different outflows (acceptance rate and rejection rate). The *rejection rate* outflow leads to a sink because after that the papers are terminated out of the system. The *acceptance rate* outflow leads to the stock of *accepted papers* which are later published after corrections from reviewers are effected. We decline to discuss the input variables or parameters of the stock and flow diagram at this stage but we do so in one of our later chapter(s) 6 and 8 respectively.

### 4.3.2 The formal approach

The current state of the Universe of Discourse is recorded by the corresponding ORM scheme as the population of this scheme with all valid (elementary) facts in that particular state at that moment. Consequently, each information descriptor will have a well-defined result. Let \( \text{Pop}_t(D) \) be the result of information descriptor \( D \) at time \( t \) (see also [157]). The goal of applying System Dynamics on an ORM scheme is to obtain quantitative insight.

The quantitative insight may be on the complete ORM scheme, or we may want to focus on a particular part of the scheme. Depending on our needs, we will choose a set \( D_1, \ldots, D_n \) of information descriptors that correspond to relevant aspects. Typically, an information descriptor will refer to instances of an object type in a particular state. We then will be interested in the amount and growth behavior of such objects, expressed as \( P_t(D_i) \) for descriptor \( D_i \) \((1 \leq i \leq n)\), using \( P_t(D_i) \) as a shorthand for the number of...
For an overall description of the quantitative behavior of an ORM scheme \( \Sigma \), the goal is to find a set of factors \( \{D_1, \ldots, D_n\} \) of information descriptors that is complete for \( \Sigma \), meaning:

C-1 The variables are independent: \( \text{Pop}_t(D_i) \cap \text{Pop}_t(D_j) \Rightarrow i = j \)
C-2 From \( \text{Pop}_t(D_1), \ldots, \text{Pop}_t(D_n) \) we can derive \( \text{Pop}_t(X) \) for each object type \( X \).
C-3 We can describe the quantitative behavior of \( D_1, \ldots, D_n \) by a system of differential equations in terms of \( P_t(D_1), \ldots, P_t(D_n) \).

If \( \mathcal{D} \) is a complete set of information descriptors for \( \Sigma \), then \( D \in \mathcal{D} \) is superfluous if \( \mathcal{D} - \{D\} \) also is a complete set for \( \Sigma \). A set \( \mathcal{D} \) of factors that is complete for \( \Sigma \), is called a base for that scheme if this set \( \mathcal{D} \) does not contain superfluous information descriptors. In that case we refer to the factors as dimensions of the scheme. The dimension of an ORM scheme \( \Sigma \) is the minimal number of dimensions required for a base of this scheme. We call \( D_1, \ldots, D_n \) a conceptual base for scheme \( \Sigma \) if it satisfies property C-1. We call a conceptual base a computational base for scheme \( \Sigma \) if it also satisfies property C-3. A complete base for scheme \( \Sigma \) is a computational base that also satisfies property C-2.

Let \( D \) and \( E \) be information descriptors, then there is a flow from \( D \) into \( E \), denoted as \( D \rightarrow^+ E \) if instances from \( D \) may move to \( E \). More precisely:

\[
D \rightarrow^+ E \quad \triangleq \quad \exists_{\text{Pop}_t,x,s,<t} [x \in \text{Pop}_s(D) \land x \in \text{Pop}_t(E)]
\]

meaning that in some population \( \text{Pop}_t \) there is an instance \( x \) from \( D \) that on a later moment is an instance of \( E \). Then we define \( \rightarrow \) as the one-step subrelation of \( \rightarrow^+ \). We will refer to the relation \( \rightarrow \) as the flow relation. A base for a scheme \( \Sigma \) with its induced flow relation \( \rightarrow \) form the base for the SD simulation of \( \Sigma \). Next we motivate that an SD indeed can simulate an ORM scheme.

Let \( \{D_1, \ldots, D_n\} \) be a computational base for scheme \( \Sigma \) and \( \rightarrow \) the induced flow relation. Since the variables \( P_t(D_1), \ldots, P_t(D_n) \) (as functions of \( t \)) take discrete values, it is not obvious that differential equations can be used to describe their behavior. In System Dynamics, rather than determining these differential equations, causal influences between the variables (factors) are detected, leading to a Causal Loop Diagram. That way we can detect basic system properties such as enforcing loops. Another opportunity we have is that we can derive a differential scheme describing the flows between the variables such that we can simulate system behavior in a stepwise manner, leading to the Stock and Flow Diagram. In this thesis we focus on such differential schemes. In this section we describe the formal relation between ORM schemes and differential schemes.

Assume we have successive time steps \( t_0, \ldots, t_n \), with \( t_{i+1} = t_i + \Delta t \). Then the population size of variable \( D \) at time \( t_n \) is obtained as the cumulation of the changes \( dP_t(D) \) during the intervals \( [t_i, t_{i+1}] \):

\[
P_{t_n}(D) = P_{t_0}(D) + \sum_{i=1}^{n} dP_t(D)
\]
During the period \([t_{i-1}, t_i]\) the change may also be described as (using formula 4.2.2):
\[
dP_t(D) = \sum_{E \rightarrow D} \left( \int_{t_{i-1}}^{t_i} \text{Flow}_s(E \rightarrow D) \, ds \right) - \sum_{D \rightarrow E} \left( \int_{t_{i-1}}^{t_i} \text{Flow}_s(D \rightarrow E) \, ds \right)
\]

Flows are approximated as follows:
\[
\int_t^{t+\Delta t} \text{Flow}_s(D_1 \rightarrow D_2) \, ds \approx \text{Rate}_t(D_1 \Rightarrow D_2) \cdot P_t(D_1) \cdot \Delta t
\]

where \(\text{Rate}_t(D_1 \Rightarrow D_2)\) is the fraction of the objects in \(D_1\) that flow from \(D_1\) to \(D_2\) per unit of time, at time \(t\). So we have:
\[
\text{Rate}_t(D_1 \Rightarrow D_2) = \lim_{\Delta t \to 0} \frac{\int_t^{t+\Delta t} \text{Flow}_s(D_1 \rightarrow D_2) \, ds}{\Delta t}
\]

Suppose information descriptors \(D_1\) and \(D_2\) describe two different states of some object type \(X\), such that instances may flow from state \(D_1\) to state \(D_2\). Note that, according to C-2, at any moment \(\text{Pop}_t(D_1)\) and \(\text{Pop}_t(D_2)\) are disjoint. The instances of \(\text{Pop}_{t+\Delta t}(D_2) \cap \text{Pop}_t(D_1)\) may be assumed to have flown from \(D_1\) into \(D_2\) between \(t\) and \(t+\Delta t\), provided \(\Delta t\) is sufficiently small. Consequently, we have proved that the flow may be expressed as a rate of the source stock as follows:

**Theorem 4.3.1**
\[
\text{Rate}_t(D_1 \Rightarrow D_2) = \lim_{\Delta t \to 0} \frac{||\text{Pop}_{t+\Delta t}(D_2) \cap \text{Pop}_t(D_1)||}{||\text{Pop}_t(D_1)||} / P_t(D_1)
\]

In general the rate is not easily measured. However, in the case of a simulation, the error introduced by an incorrect rate estimate, may have a limited effect only. In SD applications, the rate associated with each link is either taken as a constant fraction, or, in case of a converter, as a parameterized fraction. Note that the proof of this theorem is the explanation above. Summarizing, we have the following relation between SD and ORM:

**Corollary 4.3.1**

A SD interpretation of an ORM model consists of a (partial) base of information descriptors for the ORM model and its induced flow relation, the causal dependency links between these base information descriptors, and an estimation of the flow rates.

In the next chapter we will focus on structured translation of an ORM model into an SD model.

### 4.4 Conclusion

In this section we have defined the ORM and SD building blocks, stated some of the SD design rules and identified the extent to which features of ORM static models can be
transformed (with added information) into SD models. The two methods are rather different but when used together for a common goal the results are not only better grounded but also more decisive and reliable. The ORM method equips the modeler with strong conceptualization of the domain. This is key to developing any model.
Chapter 5

Relations between constructs

To reason about the combination of system dynamics with a domain modeling method (ORM), we need to identify relationships between their constructs. Therefore in this chapter we start off by identifying ORM to SD relations and evaluate these relations using questionnaires, a focus group discussion and structured walkthroughs. Basing on the evaluations, we revise ORM to SD relations, present the improved relations in subsection 5.2.3 and finally draw conclusions.

5.1 Defining ORM to SD relations

One aspect of the methods being combined as explained earlier has dynamic properties (SD) while ORM has static properties. Static properties refer to the possible states of the system under study while dynamic properties refer to the possible transitions between the states [84]. Since these two methods are dissimilar, the only way we relate these two methods is by looking at the contents and roles their constructs play in each method.

5.1.1 ORM relations to a stock

As stated in chapter 4, a stock is a container representing an accumulation of either a physical or non-physical quantity and is depicted as a box. To relate an ORM element to a stock, we looked for an ORM element with characteristics similar to a system dynamics stock that is; it holds items, accumulates and can be measured. We relate a stock to a unary fact type (one role) because unary fact types do correspond to properties of entity types (object types) that allow defining of sub-types and they contain properties specific to an object type. To clarify this relation let us take the example of figure 5.1. In this figure, we have object type 'paper' playing a role is published and is accepted. Note that objects in an object type do not make a stock but it’s the objects in a unary fact type that make a stock. This is because the objects in a unary fact type give an independent state to
which some of the objects in an object type take part and in this state they only relate with one object type. This means that all objects in a particular object type are of the same type ‘paper’ but in different state is published and is accepted.

As noted in chapter 4 there are different forms, stocks can take (reservoir, conveyor, queue, oven). See table 4.2. These forms can be defined in cases were an object type has more than one unary fact type. To identify these forms of stock, we use constraints placed between or among these unary fact types. Note that these unary fact types are attached to the same object type which means that they have similar contents (characteristics) but are in different state of affairs (condition). For example, in figure 5.1a, there are two unary fact types (is published and is accepted), objects in unary fact type is published are a subset of objects in unary fact type is accepted. The subset constraint is marked by a dotted arrow running from a subset role to a superset role.

Figure 5.1: Examples of ORM subset constraint and equality constraint

ORM verbalizations for figure 5.1a are as follows:
If some Paper is published then that Paper is accepted. The arrow notion for subset constraints derives from the arrow often used for the logical connective IF.. THEN..

Figure 5.1b Verbalizations
For each Consignment,
that Consignment is in inventory if and only if that Consignment is in good condition.

In figure 5.1b, we have the equality constraint placed between roles is in inventory and is in good condition. This equality constraint indicates that the populations of the role sequences must always be equal. By using the constraints and verbalizations in the ORM figures, we are able to determine the form of stock to use in SD. In figure 5.1a we represent unary fact type ‘is accepted’ as a conveyor stock and ‘is published’ as a reservoir stock because properties of unary fact type ‘is accepted’ are assumed to be in that particular state for a period of time, they then get off and are moved to ‘is published’ which is a reservoir stock well as in figure 5.1b we represent role is in good condition as an oven stock and is in inventory as a reservoir stock. This is because properties of role ‘is in good condition’ are assumed to be in that particular state for a period of time but they all load at the same time and dispatch to the inventory at the same time (this is noted from the equality constraint used ‘if and only if’).

Note that in Figure 5.1b we assume that all consignment that is in good condition is stored in inventory at the same time which is unrealistic because in real life, consignments may be purchased at different time intervals thus a conveyor. But we depict both scenarios to show the difference in use of ORM constraints.
5.1. DEFINING ORM TO SD RELATIONS

For cases where an object type relates to more than one unary fact type, the state change name (unary fact type name) is concatenated with the object type name to give more meaning to the type of quantities flowing into that particular stock (unary fact type). The stock names would then be called ‘accepted papers’, ‘published papers’ ‘consignment in good condition’ and ‘consignment in inventory’.

5.1.2 ORM relations to a flow

Flows as explained in chapter 4 are rates of change of a stock representing the inflow and outflow [24]. They are sometimes referred to as material flows with rates. They are active, change overtime and determine the value of a stock. We relate an object type to a flow. Object types connect different roles to other object types and objects held by these object types play unique roles for each role connection. That is why we see a similarity with SD flows which connect stocks to other stocks. Secondly, objects in each object type are similar and unique this also applies to contents in each flow. To clarify this relationship let us use the example given in figure 5.2. In figure 5.2a) we see that Paper is the object in object type Paper. This therefore means that when creating a flow, it is the content(s) in the object type that flow into stock ‘is published’ and ‘is accepted’ (a state to which objects from object type paper have transformed into). For 5.2b), Consignment is the object in object type Consignment. This means that when creating a flow for the ORM figure in 5.2b), it is the (Consignment) that flow into stock ‘inventory’ and ‘consignment in good condition’.

![Figure 5.2: Examples of a flow added to stocks](image)

Supertype and subtypes: in ORM it is possible to represent unary fact type is published and is accepted as subtypes of supertype paper. This means we would have three object types (one as a supertype and two as subtype). In this case if we refer to both super-types and their subtypes as different flows we will have flows containing similar contents but in different states. Yet we know that a specialization relation between a subtype and a supertype implies that instances of a subtype are also instances of a supertype. E.g. in figure 5.3:

\[ \text{Pop(Male)} \cup \text{Pop(Female)} = \text{Pop(patient)} \]

This means we cannot represent both a supertype and subtype as different flows unless if both the supertype and subtype have different roles they play.

In figure 5.3a) supertype patient has two subtypes (Male and Female). One of the subtypes has unary roles ‘is a non smoker’ and ‘is pregnant’ respectively. The ORM verbalizations for this model are as follows:
Each Male is an instance of Patient.
Each Female is an instance of Patient.
If some Female is pregnant then that Female is a non smoker.

Due to the subset constraint between unary role is a non smoker and is pregnant, we use a Decision Process Diamond (DPD) in figure 5.3b) as a mechanism for managing the diagram complexity associated with the representation of the decision process (If some Female is pregnant then that Female is a non smoker) within this model\(^1\). As a result, the SD model maintains a bi-focal perspective, displaying the female who are not allowed to smoke because they are pregnant and those who smoke\(^2\). In the SD representation of figure 5.3 instead of having a conveyor, we have a uniflow and a DPD. This is because in stock ‘non smoker’ there are other quantities of persons who may exist but are not pregnant and to determine them are few decisions and parameters need to be input into the model. All these decisions (details) can be captured inside the DPD but if we limit the scope of the model to only pregnant patients, then a conveyor would be appropriate for this particular stock.

Another alternative would be to ignore the DPD and instead have the formula for this verbalization (If some Female is pregnant then that Female is a non smoker) input into the flow. However, this would mean that the effect of this decision would only be seen in the simulation results.

In conclusion, the rates that cause change to an SD stock cannot be represented in ORM. But the object type which is identified to be related to a flow can only help in specifying the contents in a particular flow. Therefore we suggest that, when deriving an SD model from an ORM model, the new state objects in a unary fact type take should reflect the outflow of that particular stock. For example in figure 5.3 the outflows should be represented as pregnant female, non smoking female etc that way the modeler is able to relate the type of outflow with the contents (quantities) in the previous stock.

\(^1\)A DPD is a mechanism for managing the diagram complexity associated with the representation of decision processes within a model. Intricacies of decision rules that drive the flows into a “black box” can be “buried” here. On the surface, the modeler and the users of the model can clearly see both the inputs and the outputs associated with a decision process. As a result, the model can maintain a bi-focal perspective, displaying the macro- and micro-structure as needed.

\(^2\)In this study we decline to further discuss or relate DPD to any ORM element due to the complexity attached to it and the fact that some SD software like Vensim use this same representation as a converter.
5.1. DEFINING ORM TO SD RELATIONS

Secondly, there are cases where an object type has no unary fact type but plays a role(s). Here we suggest that, such object types be referred to as converters instead of flows. This is because the objects in this object type are in a constant state but influence fact types (flows and converters in SD).

5.1.3 ORM relations to information links

We do not particularly relate *Information links (Connectors)* to any ORM element but instead use the ORM constraints plus verbalizations to identify the direction of the SD connector. In ORM, constraints are placed in between fact types (binary, ternary, etc), these constraints may play an important role in helping the modeler identify the direction of the connector. Let us use some verbalizations from the ORM model in figure 5.4 to show how ORM verbalizations can help in determining the direction of an information link.

If some Academic submits some Paper then that Academic writes some Paper. *(Connector from paper submissions to paper writing)*

Each Paper is written by some Academic. *(Connector from Academic to paper writing)*

Academic reviews Paper. *(Connector from academic to paper reviewing)*

If some Paper is published then that Paper is accepted. *(Connector from Decision Process Diamond to paper and to published paper)*

However this mechanism may not exhaustively capture all the relevant connectors, we therefore suggest that the modeler puts the convention to a stock and flow model rule listed below into consideration;

**Rule 1:** Information links can feed information into or out of flows and converters (auxiliary variables and constants) but only extract information out of the stock.

**Rule 2:** Stocks are influenced by flows (in and out) and can influence flows or converters but cannot be influenced by other stocks and converters.

**Rule 3:** The flows can be influenced by stocks and converters but cannot influence converters or other flows, and converters can influence flows or other converters.

![Diagram](image-url)
5.1.4 ORM relations to converters and sector

Converters hold values for constants, define external inputs to the model, calculate algebraic relationships, and serve as the repository for graphical functions. In general, they convert inputs into outputs. As stated in chapter 4, converters are of two types; auxiliary and constants. We relate converters to fact types with more than one role (binary, ternary etc). This is because fact types with more than one role combine two or more variables consistently. For fact types with fixed objects, we relate them to constants else they are auxiliaries. Converters vary because they directly or indirectly depend on stocks.

Finally, we relate SD sectors to ORM object types plus their attached roles. This is because ORM conceptual object types act as semantic ‘glue’ [75, 15] and an ORM model is a network of allied object types and relationship types [75]. This means that roles and object types when put together, they make up a complete ORM model. Therefore, object types plus their ‘glued’ roles are similar to SD sectors because when both are ‘glued’ or put together they make up a complete model. A sector is a grouping of elements with related functionally in a model. For example, in a model of a business organization, a sector may be used to represent each of the major processes under consideration e.g. a manufacturing sector, a marketing sector, a transportation sector, a human resources sector, and a financial sector in the model. In model analysis, simulations can be run in a sector-by-sector manner which creates flexibility by enabling sub process analysis. As explained in chapter 4, sectors are subcomponents within a system they handle all information about a particular element. If they are subcomponents, this implies that they contain elements with a common interest and when these elements are put (‘glued’) together they make a complete system. For ORM models with supertypes and subtypes, we relate their SD sector to a super type and not the object types. Therefore a sector in this case would comprise of all elements attached to that supertype.

5.2 Evaluation of ORM to SD relations

Evaluation is an iterative process that took place at every stage of the integration. In this section, we present an evaluation for ORM to SD relations. In this evaluation, we used questionnaires and a focus group session. A focus group as defined in [194, 149] is a moderated discussion among six to twelve persons discussing a topic under the direction of a moderator whose role is to promote interaction and keep the discussion on the topic of interest. We conducted this evaluation to examine the relations inline with the researcher’s pre-knowledge/skills by judging their effectiveness.

In figure 5.5, we present a summary of activities that were used in evaluating ORM to SD relations. Before conducting this focus group discussion, we came up with ORM-SD relations as presented in section 5.1. These relations were used as a basis for the focus group discussion. After defining the initial relations, we prepared documents for evaluating the defined ORM-SD relations. These documents included: defined ORM and

---

3 Note that in this study we only consider binary and ternary fact types to be similar to converters but all fact types with more than one role are referred to as converters.
SD constructs, defined ORM-SD relations and a questionnaire. When the documents were ready, we identified participants and sent them requests asking them to take part in this focus group discussion. Not all participants replied positively but we received a maximum of eight acceptances for the focus group discussion. Those who responded positively, were asked to suggest dates for the focus group discussion. These dates were what we based on to schedule the focus group discussion session.

As earlier stated, prior to the focus group session, participants were given some read text as earlier prepared. This text contained descriptions of both ORM and SD constructs (with examples), initial ORM-SD relations and questionnaires. These contents were later used during the focus group discussion. Normally questionnaires are given at the end of the discussion, in this case however they were given before and after the focus group session. This was to allow participants make individual choices of the relations before conducting the focus group discussion. These choices were given to the moderator before the session started to act as a basis of the discussion(s). As each relation was discussed, each participant’s choice of answer were loudly readout to participants and discussed accordingly. This way of working made the discussions more interactive and focused. For cases where participants had differing opinion, consensus building was required [99]. For consensus build, we borrowed a joint fact finding approach [196]. By borrowing we mean that participants used the idea behind this approach but did not copy or follow this approach step by step. What they did however, was as follows: they gave factual statements that they believed to be relevant to the defined relation, exchanged information, developed common assumptions and together used the gathered information to reach a decision. In cases where given facts were not enough or not available, they negotiated a way to find additional information thus filling the gaps or resolving the disagreement(s).
5.2.1 Analyzing results

We used a focus group discussion to collect data and to successfully analyze the collected data, we used the inductive approach. With this approach, analysis is guided by specific objectives and the procedure is systematic [203]. In this study, the analysis phase comprised of a number of activities e.g. coding, classifying or categorizing of raw data, examining etc. One of the most time consuming activity was: merging of collected data from different sources i.e. audio recorded data (this had to be transcribed first), questionnaire data (from open ended questions), notes taken by the researcher during the sessions and some comments or remarks some of the participants had put down on their note books during the sessions.

![Figure 5.6: A summary of how we analyzed focus group data](image)

As depicted in figure 5.6, we first of all put together all collected data. This enabled us systematically carry on our next step which was categorizing and coding. During coding and categorization, we were able to identify portions of data that discussed a particular relation or construct. The different phrases that where found to be pointing to the same ORM-SD relation were given unique codes. As we did so, we were able to collect and analyze facts. This enabled us find common or different patterns in the text. There were cases when the given text could not fit into the identified relation. In this case, we put our focus on the secluded data to find explanations as to why the given concepts do not fit the patterns. To clearly have a clear portrayal of patterns and relationships, we moved back and forth. That way we were able to find revelations, contradictions, and exceptions in the collected data. After examining all the collected data, we drew conclusions and reported the findings.

5.2.2 Presentation of results

Results from the first questionnaire

Questionnaires were the core of the focus group discussion. In this section therefore, we present a summary of results obtained from the questionnaire followed by focus group remarks.

In table 5.1 we present a summary of the first draft results obtained from the questionnaires. In these results we see that there is no correlation in most of the responses apart from questions 3a) and 5 therefore, no conclusive results could be drawn. To improve these results, we conducted a focus group session with the same participants who took part in answering the questionnaires. However, all participants could not be available for the focus group session therefore original answers for those participants that could not
attend were kept intact. We received nine answered questionnaires but only seven participants took part in the focus group discussion.

Focus group discussion:

In this subsection, we present views or feedback obtained from participants that took part in this study. Before starting the focus group discussion, previously answered questionnaires were given back to participants. This was to enable them have a clear view of what their previous responses were. During the discussions participants were allowed to make changes to their answered questionnaires.

To identify a stock from a flow, a mental exercise of freezing the system is normally done. For example freeze a university, walk around what you still see say people, buildings etc. are the stocks. Anything one sees when time is stopped is a stock and anything that can only be seen over a moment of time is a flow. looking at the examples given (see figure 5.4a), the number of authors has an influence on the number of papers submitted. In other cases there must be one author that submits the paper, therefore it is appropriate to relate a stock to a role because for the contents in a stock to accumulate, there must be a role played (flow/action) and it is the contents in that act/role that make a stock. Secondly, the time constant chosen for a variable also determines its representation e.g. a variable may be referred to as a stock if the time constant chosen is in years and that same variable when its time constant is measured in minutes, it becomes a flow. Considering the definition of object types, they also have some characteristics of a stock. Therefore both a unary fact type and an object type are related to a stock.

Flows are identified to have a relation with object types. But as the definition suggests, flows express a dynamic behavior, whereas object types are static. The static/dynamic (behavior) distinction is not discussed in this chapter because flows as used in earlier examples represent an anti-rigid type or a dynamic subtype in the literature [67, 225]. This idea of anti-rigid types has also been adopted in more recent works on ORM [74]. So, if we take the example of figure 5.4a, “is published” can be represented through a subtype of Paper which is Paper-is-published? The same can be argued for is accepted and is rejected as an anti-rigid (dynamic) subtype of Paper. The obvious advantages of repre-
senting these unary predicates which seem to represent dynamic subtypes is that in this way one could create a taxonomy of those subtypes, ascribe properties that entities have only when instantiating those subtypes (e.g., paper can have properties which are specific for when they are accepted, published or rejected). Note that if a clear real-world semantics for the elements in the language being used is not provided then there is nothing guiding the modeler to choose a modeling primitive over the other (e.g., choose modeling Paper-is-Published as a subtype of Paper in contrast with the unary predicate). One of the problems with this is that different ORM modeling primitives shall be mapped to different SD primitives.

From a full set of variables in an SD model, if a variable is not a stock or a flow then it is a converter. However, it is fine to relate converters to fact types with more than one role but also think of how exogenous variables would be captured in ORM. Maybe they cannot be captured, if not then this should be clarified by ORM experts. Secondly, converters (auxiliaries or constants) should be given different variable names. In case the value of the auxiliary is equal to a stock, the name of the converter should be changed otherwise we could have an extra variable with exactly the same input numerical(s) (parameters) as the ones for the stock. However this occasionally happens, usually when it is represented as a separate auxiliary there is a transformation or operation that is done on the basis of the stock. Note that representation of converters with different SD modeling tool (software) vary since there is no standard representation for them.

To relate information links to any ORM variable is inappropriate. This is because information links define how decisions are made. Representing a decision in ORM may require more than one ORM concept and derivation rules. We therefore suggest information link relates to none of the ORM concepts.

In conclusion, the primary issue here is that we simply cannot completely identify relations between constructs in both methods because each method has something extra which the other method can partly represent. So it is not translating each detail but getting out the essence from one to the other. Therefore we have to accept that we cannot have a complete relation from one into the other, otherwise it would be the same method or it would be such a powerful method such that it has everything in it and ends up being useless because it is so complex.

Results from the second questionnaire

Since the focus group discussion was conducted because of the diverse answers received from the questionnaires, the questions in the first questionnaire were therefore the core of the focus group discussion. The purpose of the focus group discussion was for participants to further explain their independent perception, elaborate their choice of answers and to have a consensus for each relation. At the end of each relation discussions, participants were allowed to change their answers if necessary. In table 5.2 we present a revised version of the results after conducting the focus group discussion. For question one, five out of nine participants stated that object types are related to stocks because object types are seen as a collection of objects and four participants said unary fact types are related to stocks because of the clear ‘object type with one property flavor’. For question two, four
5.2. EVALUATION OF ORM TO SD RELATIONS

Table 5.1: Results from relations questionnaire

<table>
<thead>
<tr>
<th>Mapping Questions</th>
<th>Provided Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Which of the following ORM element is most similar to a system dynamics stock?</td>
<td>0 0 1 3 0 0 1</td>
</tr>
<tr>
<td>2. Which of the following ORM element is most similar to a system dynamics flow?</td>
<td>1 2 0 0 3 3 0 0</td>
</tr>
<tr>
<td>3. There are two SD elements that are classified within a converter; a constant and an auxiliary variable.</td>
<td></td>
</tr>
<tr>
<td>a) Which of the following ORM element is most similar to a system dynamics constant?</td>
<td>7 1 0 0 0 0 1</td>
</tr>
<tr>
<td>b) Which of the following ORM element is most similar to a system dynamics auxiliary variable?</td>
<td>3 0 0 1 0 2 2 1</td>
</tr>
<tr>
<td>4. Which of the following ORM element is most similar to a system dynamics information link?</td>
<td>0 0 0 2 1 3 3 0</td>
</tr>
<tr>
<td>5. Which of the following ORM elements is most similar to a system dynamics sector?</td>
<td>0 0 2 2 0 0 0 5</td>
</tr>
</tbody>
</table>

attend were kept intact. We received nine answered questionnaires but only seven participants took part in the focus group discussion.

Focus group discussion:

In this subsection, we present views or feedback obtained from participants that took part in this study. Before starting the focus group discussion, previously answered questionnaires were given back to participants. This was to enable them have a clear view of what their previous responses were. During the discussions participants were allowed to make changes to their answered questionnaires.

To identify a stock from a flow, a mental exercise of freezing the system is normally done. For example freeze a university, walk around what you still see say people, buildings etc. are the stocks. Anything one sees when time is stopped is a stock and anything that can only be seen over a moment of time is a flow. Looking at the examples given (see figure 5.4a), the number of authors has an influence on the number of papers submitted. In other cases there must be one author that submits the paper, therefore it is appropriate to relate a stock to a role because for the contents in a stock to accumulate, there must be a role played (flow/action) and it is the contents in that act/role that make a stock. Secondly, the time constant chosen for a variable also determines its representation e.g. a variable may be referred to as a stock if the time constant chosen is in years and that same variable when its time constant is measured in minutes, it becomes a flow. Considering the definition of object types, they also have some characteristics of a stock. Therefore both a unary fact type and an object type are related to a stock.

Flows are identified to have a relation with object types. But as the definition suggests, flows express a dynamic behavior whereas object types are static. The static/dynamic (behavior) distinction is not discussed in this chapter because flows as used in earlier examples represent an anti-rigid type or a dynamic subtype in the literature [67, 225]. This idea of anti-rigid types has also been adopted in more recent works on ORM [74]. So, if we take the example of figure 5.4a, “is published” can be represented through a subtype of Paper which is Paper-is-published? The same can be argued for is accepted and is rejected as an anti-rigid (dynamic) subtype of Paper. The obvious advantages of repre-
senting these unary predicates which seem to represent dynamic subtypes is that in this way one could create a taxonomy of those subtypes, ascribe properties that entities have only when instantiating those subtypes (e.g., paper can have properties which are specific for when they are accepted, published or rejected). Note that if a clear real-world semantics for the elements in the language being used is not provided then there is nothing guiding the modeler to choose a modeling primitive over the other (e.g., choose modeling Paper-is-Published as a subtype of Paper in contrast with the unary predicate). One of the problems with this is that different ORM modeling primitives shall be mapped to different SD primitives.

From a full set of variables in an SD model, if a variable is not a stock or a flow then it is a converter. However, it is fine to relate converters to fact types with more than one role but also think of how exogenous variables would be captured in ORM. Maybe they cannot be captured, if not then this should be clarified by ORM experts. Secondly, converters (auxiliaries or constants) should be given different variable names. In case the value of the auxiliary is equal to a stock, the name of the converter should be changed otherwise we could have an extra variable with exactly the same input numerical(s) (parameters) as the ones for the stock. However this occasionally happens, usually when it is represented as a separate auxiliary there is a transformation or operation that is done on the basis of the stock. Note that representation of converters with different SD modeling tool (software) vary since there is no standard representation for them.

To relate information links to any ORM variable is inappropriate. This is because information links define how decisions are made. Representing a decision in ORM may require more than one ORM concept and derivation rules. We therefore suggest information link relates to none of the ORM concepts.

In conclusion, the primary issue here is that we simply cannot completely identify relations between constructs in both methods because each method has something extra which the other method can partly represent. So it is not translating each detail but getting out the essence from one to the other. Therefore we have to accept that we cannot have a complete relation from one into the other, otherwise it would be the same method or it would be such a powerful method such that it has everything in it and ends up being useless because it is so complex.

**Results from the second questionnaire**

Since the focus group discussion was conducted because of the diverse answers received from the questionnaires, the questions in the first questionnaire were therefore the core of the focus group discussion. The purpose of the focus group discussion was for participants to further explain their independent perception, elaborate their choice of answers and to have a consensus for each relation. At the end of each relation discussions, participants were allowed to change their answers if necessary. In table 5.2 we present a revised version of the results after conducting the focus group discussion. For question one, five out of nine participants stated that object types are related to stocks because object types are seen as a collection of objects and four participants said unary fact types are related to stocks because of the clear ‘object type with one property flavor’. For question two, four
5.2. EVALUATION OF ORM TO SD RELATIONS

Table 5.2: Results obtained after focus group discussion

<table>
<thead>
<tr>
<th>Mapping Questions</th>
<th>Provided Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Which of the following ORM element is most similar to a system dynamics stock?</td>
<td>0 0 0 0 4 0 0 0</td>
</tr>
<tr>
<td>2. Which of the following ORM element is most similar to a system dynamics flow?</td>
<td>0 0 0 0 4 0 0 0</td>
</tr>
<tr>
<td>3. There are two SD elements that are classified within a converter; a constant and an auxiliary variable:</td>
<td></td>
</tr>
<tr>
<td>a. Which of the following ORM element is most similar to a system dynamics constant?</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>b. Which of the following ORM element is most similar to a system dynamics auxiliary variable?</td>
<td>0 0 0 0 4 0 0 0</td>
</tr>
<tr>
<td>4. Which of the following ORM element is most similar to a system dynamics information link?</td>
<td>0 0 0 0 1 3 5 5</td>
</tr>
<tr>
<td>5. Which of the following ORM elements is most similar to a system dynamics sector?</td>
<td>0 0 0 0 2 0 0 0 5</td>
</tr>
</tbody>
</table>

out of nine participants said that an object type is related to a flow, two participants said that a flow has a relation with binary and ternary fact types because they contain roles with predicates that describe the relationship between or among objects and three participants said that a flow had no relation to any of the provided options. For question 3 part a) all participants said that a constant has a relation with a value type because the values in a value type do not change. For question 3 part b) two out of nine participants said that an auxiliary relates to a value type, four participants said an auxiliary has a relation with fact types that have more than one role, two participants stated that an auxiliary is related to constraints because they determine objects that can take part in the relationship and can assume any value, and one participant said that an auxiliary has no relation with any of the provided choices. For question four, three participants said constraints were related to information links because constraints hold some ‘decision functions’, one participant said binary and ternary fact types relate to information links and five participant said information links have no relation with any of the provided choices. Finally, for question five, two participants said that a sector has a relation with a supertype and seven participants said a sector has no relation with any of the provided choices. During the focus group session there was a lot of arguing which required consensus building and re-explaining of concepts was often required. However we were able to obtain better or improved results with explanations.

5.2.3 Revised ORM to SD relations

Considering the remarks presented in subsection 5.2.2, we now present the revised ORM to SD relations in table 5.3. In this table we relate a stock to two ORM elements (a unary fact type and an object type). Stocks are related to ORM unary fact types because they both uniquely hold quantities with similar properties, relate to one element (object type for ORM unary fact types and flows for SD stocks). We refer to their elements as containers because of their purpose which is holding items.

For the relation between a flow and an object type we decided to keep it as originally defined since participants neither agreed nor rejected relating flows to object types. Keeping in mind that the operation that change contents in an object type is not repre-
Table 5.3: Revised ORM to SD relations

<table>
<thead>
<tr>
<th>System Dynamics Construct</th>
<th>Identified ORM Construct relations</th>
<th>Transitional Statement</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock</td>
<td>Unary fact type</td>
<td>They both contain &quot;things&quot; or act as containers.</td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td>Object type</td>
<td>They both hold &quot;things&quot;, connect roles (ORM)/flows (SD) to other object type (ORM)/stocks (SD) and contents in both constructs play unique roles for each connection.</td>
<td>Connector</td>
</tr>
<tr>
<td>Quantity</td>
<td>Objects</td>
<td>These can be looked at as the contents within the system.</td>
<td>Content</td>
</tr>
<tr>
<td>Flows (inflow and Outflow)</td>
<td>Object types</td>
<td>They all connect different stocks (SD) and Fact types (ORM).</td>
<td>Homogeneous connectors</td>
</tr>
<tr>
<td>Converter</td>
<td>Fact types with more than one role</td>
<td>Converters are related to fact types with more than one role (binary, ternary and quaternary) because they add new information to the model and combine two or more variables consistently. If fact types have fixed objects, then that converter is a constant else it is an auxiliary. Converters vary because they directly or indirectly depend on stocks.</td>
<td>Heterogeneous connectors</td>
</tr>
<tr>
<td>Information Link (Connector)</td>
<td>None</td>
<td>Information links are not related to any ORM element but ORM constraints and verbalizations can guide in identifying connectors and the direction of the SD connector. In ORM constraints are placed in between fact types (binary, ternary, etc), these constraint may play an important role in helping the modeler identify the direction of the connector.</td>
<td>None</td>
</tr>
</tbody>
</table>

...sent in ORM. Therefore the identified relation between flows (inflows and outflows) and object types is due to the fact that both constructs hold similar contents that change states e.g from a similar object type, objects contained in each unary fact type are in different states and flow quantities change state/differ for each flow-stock connection. Furthermore they all connect different stocks (SD)/unary fact types (ORM) and transfer objects (ORM)/quantities (SD). We refer to their elements as homogeneous connectors because they all connect and transfer quantities with similar properties or concepts. As noted by participants, object types also have some characteristics of a stock. We therefore came up with two choices of option defined below to guide on when to use object types and unary fact types as stocks or flows. The identified two options are progressive refinement and pure mapping.

Progressive refinement is where a first draft model is created and progressively elements of that model are replaced by others. In progressive refinement, both an object type and a role are identified to have relations with a stock. Here each role is independently represented as a stock because a role played by an object type contains similar properties and it relates to one object type. Object types are also related to stocks because they hold objects.

In pure mapping, elements are generated only when one is sure that they are the right ones. Here only unary fact types are identified to have a relation with stocks because they do correspond to properties of the entity type that allow defining of sub-types and they contain properties specific to an object type although these properties are in a different state. Thus objects in a unary fact type are in an independent state to which some of the objects in an object type take part while they are in this state, they relate with only one object type. This means that unary fact types from one object type have similar objects but in different state.

Furthermore, SD quantities are related to ORM counting objects because they are looked at as quantities that flow within the system or process. We use the term ‘quantity’ to represent items or quantifications that flow within an SD system and their elements are referred to as contents.
5.3. **CONCLUSION**

*Converters* are related to fact types with more than one role (binary, ternary etc) because converters add new information to the model and combine two or more variables consistently. For fact types with fixed objects and object types with no attached unary fact type, the converters are constant else they are auxiliaries. Converters vary because they directly or indirectly depend on stocks.

Finally, as participants said, *Information links* define how decisions in the SD model are made therefore we do not relate them with any ORM element. Note that we have not included sectors in table 5.3 because participants said that they are not very important in the SD model.

### 5.3 Conclusion

In this section we have discussed and presented the identified relations to the key constructs in both modeling methods, their transitions and elements. We have also shown the extent to which the features of ORM static models can be transformed (with added information) into SD models.

While identifying relations between ORM constructs to SD constructs we found that it was impossible to exhaustively relate all constructs from one method into the other. The best we could do was to use constructs that were most similar but not identical. In figure 5.7, we illustrate ORM to SD possible relations. In this figure we see that there are cases where one construct from method 1 (*A*) relates to two different constructs in method 2 (*X* and *Y*). This scenario was seen when an SD stock was found to have characteristics similar to both a unary fact type (role) and an object type. The other case is were one construct in method 1 (*B*) relates to only one construct in method 2 (*Z*). The third scenario was where a construct in method 1 (*C*) relates to none of the constructs in method 2. In this study however, we did not have any case where more than one SD construct (Method 1) related to one ORM construct (Method 2) nor where many constructs in method 1 related to many constructs in method 2.
Thus far, we have defined the methods, their constructs, identified the relations between the constructs of the two methods and by doing so we were able to have a clear understanding of the two methods, their constructs and how they relate. In this chapter, however, we present a methodical approach. In this approach we start by defining the initial steps for combining System Dynamics (SD) with Object-role Modeling (ORM). These defined steps are used as a guide to ground system dynamics with a domain method ORM, we therefore refer to this method as Grounded System Dynamics (GSD) [208]. As we stated in chapter 1, combining SD with ORM provides complementary added value: SD studies the behavior of a system in terms of discrete quantities of things (stocks and flows) while ORM underpins the models in terms of underlying ontology of the domain. The GSD method therefore, enables stakeholders (domain experts, decision makers, system analysts etc) attain a level of understanding of the dynamics and statics within the SD developed models.

![Diagram of the GSD method](image)

**Figure 6.1: Overview of the GSD method**

The GSD method is made up of two main phases (Causal Loop Diagram (CLD) to ORM and ORM to Stock and Flow Diagram (SFD)), each with a number of steps. In figure 6.1 we present an overview of the GSD method and indicate each overview
phase with numbers 1-5. In phase 1 we have the informal description where we defined
the constructs with their underlying principles; in phase 2 the initial GSD method steps
comprising of CLD to ORM steps and ORM to SFD steps are presented; in phase 3 we
evaluate the GSD steps using case studies in the analytical and experimental evaluations;
in phase 4 we present the revised GSD method steps; in phase 5 the Stock and Flow
Diagram (SFD) object behavior is presented and conclusions drawn.

Although we partially discuss CLDs in this study, our main focus is on stock and
flow diagrams which are the common notation for system dynamics. CLDs are used
for articulation of a dynamic hypothesis, give insights into the stated problem and show
how positive and negative feedbacks affect each other [159, 165, 193] but they make no
distinction between information links and rate-to-stock links [162].

6.1 CLD to ORM transformation steps

The steps given in this section have some similarities with ORM’s seven Conceptual
Schema Design Procedural (CSDP) steps except here we focus on each CLD variable
and not on each model component (a section of a model) as is the case with ORM’s
CSDP. The CLD to ORM transformation steps look at each variable as a unique entity
[209]. To cross check the quality of the derived ORM model, we recommend that ORM’s
CSDP steps be considered. Secondly, for complex CLD models, we suggest the use of
a summarized top-down approach (three steps) as described in [77]. The CLD to ORM
transformation steps are as follows;

1. Identify object(s) in each CLD variable.
2. Collect and group (Classify) similar objects.
3. Identify and connect roles played by objects to their object types.
4. Add all constraints and mandatory roles.
5. Connect all roles from similar object types.

6.1.1 CLD to ORM transformation definition

Having outlined the CLD to ORM transformation steps, we now explain each step in
detail with examples.

Step 1: Identify object(s) in each CLD variable.
Objects are regarded as “things of interest” that have an object or value types (entity
type or value type in ORM terms). To identify objects in each CLD variable we use
the questions below as a guide:

1. Who are the main role player(s) in a given CLD variable?
2. Who plays a role in this CLD variable?
A role player in this case is the object. By answering one of the questions above, objects taking part in a given CLD variable are identified. Let us use figure 6.2 as an example to make this more explicit. Figure 6.2 is made up of the following variables: Patient arrival, Admitted patients, Patients discharged, Available admission beds and Number of patients in a queue.

For each given CLD variable, we present a question with its anticipated answer (see table 6.1). In question one of table 6.1 the object identified is 'patient' because it is the only role player same applies to question five. In question two, we have objects 'patient' and 'bed'. Object 'patient' is derived directly from the variable while object 'bed' is as a result of the influence or relationship, variable ‘Admitted patients’ has with ‘Available admission beds’ same applies to question three. In question four, the identified object is ‘Bed’ because it is the only role player. Variable ‘Available admission beds’ is influenced by other variables but does not influence any other variable in the loop.

In conclusion, if a variable influences other variables then the objects in the influenced variables are also added to the list of identified objects for that variable. For example; if variable ‘X’ is influenced by variable ‘Y’, then objects identified in ‘Y’ are also identified in variable ‘X’ and if variable ‘X’ influences variable ‘Z’, the objects identified in ‘Z’ are not listed among ‘X’ objects. This is because ‘Z’ does not add to variable ‘X’ but subtracts from it.

Step 2: Collect and group (Classify) similar objects.
Next we collect all similar objects. Each classification of objects makes an object
type. An object type as defined in [77] is a collection of objects with similar properties, it contains objects that play roles and is designated by a solid-line named ellipse with a reference scheme indicated below the object type name. In this step, we collect all objects with similar properties from different CLD variables and group them into one. Using the same example given in step 1 table 6.1, where we have questions and objects identified for each CLD variable, the different types of objects found are two ‘Patient’ and ‘Bed’. These objects are contained in two different object types with similar properties. Out of these classifications are two object types ‘Patient’ and ‘Bed’.

The number of object types in a CLD diagram is equal to the total number of object classifications. Classifications are the different groups of objects with similar properties. Therefore; 

$$\text{Pop}(O_{c}) = \text{Pop}(O_{t})$$

Where;

- $O_{k}$ = Total number of identified object types
- $O_{c}$ = Total number of object classifications

Figure 6.3: A CLD with derived object types

Figure 6.3 shows CLD variables and identified object types from each CLD variable. In the same figure, explanations as to why those object types are derived are included. For example; who plays a role in variable ‘Admitted patients’? Our answers are Patient and Bed. Patient is directly derived from CLD variable ‘Admitted patients’ but ‘Bed’ is as a result of the influence link it has with variable ‘Available admission beds’. Notice that, CLD variables that are influenced by other CLD variables but do not influence any other variable in the CLD model and have no other object types except those directly derived from them. An example of such variable are ‘Available admission Beds’ and ‘Patient arrival’. Other cases are CLD
variables that are influenced by other CLD variables and but have similar object types, for example 'Number of patients in queue'. Since these variables have a similar object type, they therefore maintain object type(s) derived from their particular CLD variable.

Secondly, here we also note that CLD variable names are in plural form but the ORM object type names are represented in singular form. This is because ORM mainly captures information in oneness but that does not mean that multiple objects can not be represented. They can but in this paper we will limit ourselves to the simple concepts of ORM.

**Step 3:** Identify and connect roles played by objects to their object types.

In ORM, roles define the relationships between object types. They are represented by boxes with their predicate names and are connected to object types by solid lines. When these roles are put together they make unary, binary, tertiary, etc, fact types. Using the same example given in step 1, we now show how roles for each object type are identified. To achieve that, we use the following two guiding questions:

- What role do objects in object type (A) play in object type (A)?
- What role do objects in object type (A) play in object type A₁, A₂, A₃, ..., Aₙ?

Where; A₁, A₂, A₃, ..., Aₙ are object types identified in step 2. Note that, for each object type identified in step 2, there is a role attached to it, (see table 6.2). The number of roles played by an object type are determined by the number of times an object types participates (plays) in different roles.

Question in the first bullet relates to one object type and question in the second bullet relates to more than one object type in a problem domain. For every identified role, a connection (link) from that role to the object type it relates with is added. The roles played by objects are explicitly shown as boxes with Predicate names (see ORM representations in table 6.2). These predicate names are written beside each role and are read from left to right, or top to bottom. It is through predicates that object types relate to each other.

**Step 4:** Add all constraints and mandatory roles.

After completing steps 1, 2 and 3, we now add constraints to the model. There are a number of constraints in ORM (see [77]). Constraints make the boundaries of a domain. This step can also be carried out in parallel with step 3 for cases were the CLD model has many variables or is complex. Note that these constraints can not be obtained from the CLD variable(s) directly. Therefore, the modeler validates the ORM model input constraints through verbalizations.

Modelers use different platforms (tools) to develop ORM models. In this thesis, we used Natural ORM Architect (NORMA). With this tool, when a constraint is added to the role(s), the readings are generated automatically. This therefore makes ORM model validation easier.

**Step 5:** Connect all roles from similar object types.

In our final step, we link all roles connecting to a similar object type into one. Connecting these roles with a common object type makes the ORM model complete.
without repetition of variable names (See figure 6.4). If the ORM model has other constraints e.g irreflexive, ring constraints, occurrence frequency constraints, cardinality joins etc. they can be added to the model at this step. Note that the intent of this example is to capture only a current snapshot without maintaining history.

Figure 6.4: ORM model with all constraints and relationships

The verbalizations for figure 6.4 are as follows:

Patient is admitted to Bed.
Each Patient is admitted to at most one Bed.
Each Bed is for at most one Patient.
Patient is discharged from Bed.
Each Patient is discharged from at most one Bed.
Each Bed was for at most one Patient.
Patient arrives.

Bed is available for admission.

Patient is in queue.

If some Patient is discharged from some Bed then that Patient is admitted to that Bed.

If some Patient is in queue then that Patient arrives.

For each Patient, at most one of the following holds:

that Patient is in queue;

that Patient is admitted to some Bed.

In conclusion, transforming a CLD model into an ORM model is better done incrementally. This prevents the modeler from getting confused in cases when the model boundaries are not explicit or CLD variable names are not clear.

Secondly, we see that in this study we deal with each CLD variable independently which prompts defining of facts pertaining to each particular CLD variable thus clearing the ambiguities in each CLD variable and improving the SD model conceptualization.

6.2 ORM to SFD transformation steps

Having illustrated how a CLD can be transformed into an ORM model, we now present steps on how to transform an ORM model into a stock and flow diagram. To start with we give an outline of the steps and thereafter apply them using a case.

1. Identify all possible stocks.
2. Identify all relevant flows connecting to each stock.
3. Identify all possible converters.
4. Identify all possible connectors (information links).
5. Create sectors.

6.2.1 ORM to SFD transformation

In this section, we use the same example given in chapter 4 of paper flow (figure 6.5) to apply the outlined ORM to SFD transformation steps. To do so, we follow the relations defined in chapter 5. This enables us show the kind of SFD model that would be derived when these steps are followed. As already stated in chapter 5, we have two choices of option that guide on when to use object types and unary fact types as stocks or flows. Here we use the pure mapping option described in chapter 5 subsection 5.2.3. With the pure mapping option, elements are generated only when one is sure that they are the right ones.
**Step 1:** Identify all possible stocks.

With the pure mapping option, only unary fact types are identified to have a relation with stocks. The total number of stocks therefore is equal to the total number of unary fact types. Unary fact types relate to a single object type. There are cases where more than one unary fact type relates to the same object type. In this case, objects in these unary roles have similar quantities (contents) but are in different status (state of affairs). E.g. in figure 6.5 object type ‘paper’ has unary roles is rejected, is accepted and is published. All these unary roles have similar contents/characteristics (Paper) but in different state of affairs/condition (is rejected, is accepted and is published). Each of these unary roles is therefore represented as a unique stock. The condition in which quantities in these stocks are, is determined by the constraints attached to them. In this step therefore, we first of all present all object types with their varying states. Thereafter we separate the unary fact types from binary and ternary fact types. In the example shown in figure 6.5, we only consider two object types: Academic and Paper.

Object type Academic has the following states:

- State \((R_1)\) submits
- State \((R_2)\) writes
- State \((R_3)\) reviews

Object type Paper has the following states:

- State \((R_4)\) is rejected
- State \((R_5)\) is accepted
- State \((R_6)\) is published
- State \((R_7)\) is submitted by
- State \((R_8)\) is written by
- State \((R_9)\) is reviewed by
State $(R_{10})$ is classified in

Among the given states, some of the states make binary relations while other are unary. As earlier stated, in this phase we only consider unary fact types to be stocks. This therefore implies that we have three stocks in the given model and are derived from states

- State $(R_4)$ is rejected
- State $(R_5)$ is accepted
- State $(R_6)$ is published

In figure 6.6, we represent these states as boxes because stocks in SD are depicted as boxes.

![Figure 6.6: Stocks derived from figure 6.5](image)

In 6.6, note that:

1. The unary fact type names are concatenated with the object type names to depict the state and contents in that stock.
2. We represent accepted papers and published papers as separate stocks because not all accepted papers are published. For example, in some conferences if one of the authors of a paper does not show up to present the accepted paper or register for the conference, the accepted paper is not published. Thus a difference in accepted papers and published papers stock quantities.

**Step 2**: Identify all possible flows connecting to each stock.

As stated in chapter 5 object types have a relation with flows because they connect to other object types through fact types. As earlier stated, in figure 6.5 we consider only two object types. This implies that we have two flows (academic and paper). However, object type *academic* has no unary fact type attached to it but contains quantities that influence objects in object type *paper* through roles (submits, writes and reviews). This therefore means that we cannot represent *academic* as a flow, but we instead capture the contents in this object type (academic) in a converter (see step 3). Secondly, since all the identified stocks are attached to the same role, it means they have something in common which are the quantities (objects) flowing in these flows. The flowing common quantity is *paper* but in every stock it takes on a new state therefore the flows into and out of these flows have similar quantities but in different states.

In figure 6.7a, we have a bi-flow in between stocks *accepted papers* and *rejected papers* because the constraint in between these roles is an *or* constraint indicating that the contents within this flow move in either direction. Bi-flows in SD take on
any value and quantities flow in both directions. The existence of a bi-flow indicates that this flow is influenced by a decision of either a positive (accept) or negative (reject) value. To avoid complexities associated with bi-flows one would extend the model to capture submitted papers as represented in figure 6.7b. This however would lead to changes in the ORM model. Here however we limit ourselves to the model depicted in 6.7a.

Step 3: **Identify all possible converters.**

In chapter 5 we stated that fact types with more than one role have a relation with converters. From figure 6.5 we have the following fact types with more than one role:

- Fact type $(F_1)$ submits/is submitted by
- Fact type $(F_2)$ writes/is written by
- Fact type $(F_3)$ reviews/is reviewed by
- Fact type $(F_4)$ is classified in/in which is classified

To derive meaningful converter names, we use both the predicate names plus object type names. That is to say, we concatenate fact type name to object type name. For example, fact type submits/is submitted by we refer to the converter as paper submission, for fact type writes/is written by we refer to the converter as written papers, for fact type reviews/is reviewed by we refer to the converter as paper reviews and for fact type is classified in/in which is classified we refer to the converter as paper classification. To arrive at each converter names we concatenate one word that defines the contents in the fact type with one of the object types relating to that fact type. In figure 6.8, we add the defined converters.

Note that all object types with no unary fact types are also represented as converters. This is because they contain quantities that influence other object types or fact types.

Step 4: **Identify all possible connectors (information links).**

This step does not have a very systematic way because identifying connectors depends both on how the domain expert defines the problem and logic. We therefore suggest that the modeler works in small bits until (s)he reaches a full blown model.
Working in bits helps in understanding the underlying logic and how input parameters in one variable influence other variables they relate with. Connectors as explained in chapter 5 are immaterial and connect inputs to decision function of a rate. One of the ways to add connectors to the model we follow the ORM constraints,

verbalizations plus SD conventions for a stock and flow diagram given in chapter 5 subsection 5.1.3. In figure 6.9 we show the identified connectors. Connectors from a stock to a converter imply that it is the quantities or objects within the stock that influence the variables at the arrow tip of the connectors and not the flow.

**Step 5: Create sectors.**

In chapter 5 we state that each object type plus its roles all make a sector. Therefore, the total number of sectors is equal to the total number of object types. But as participants in chapter 5 said having sectors in the model does not necessarily change the model per se but is good when a modeler intends to carry out sensitive analysis [109]. With sensitive analysis a modeler can lock (switch off) other sectors of the model and only focus on a particular sector.
CHAPTER 6. STEPS FOR GROUNDING SYSTEM DYNAMICS

The sectors presented in figure 6.10 are equal to the number of object types. If the ORM model had supertypes then these supertypes would be the sectors because all elements connecting to that supertype have characteristics in the supertype.

6.3 Conclusion

The GSD artifact is built/defined in the cycle of the adopted research methodology that was based on [86, 85] see figure 1.3 and in the research methodology see phase 4 figure 1.4. This artifact is later assessed using different evaluation methods as indicated in figure 1.3. These evaluations lead to refinement of the defined artifact (see figure 1.3). As an initial design step to the grounded system dynamics method, we have defined procedural steps to show how information in a CLD model can be transferred into an ORM model and from an ORM model to an SFD model using example(s). To arrive at these steps, we put into consideration the relations defined in chapter 5 (from the environment and knowledge base of figure 1.3). The derived SFD model however, lacks a detailed conceptual level description of SD object behaviors and simulations.

In our next chapters, we evaluate the defined GSD method to gain more insights. It is on the basis of these insights that we refine and improve the GSD method. Hence, completing the evaluation phase of the design cycle.
Chapter 7

Evaluation for grounded system dynamics method

In scientific research, one way of improving and widening knowledge is through refinement of methods and construction of new knowledge. However, this does not take place solely [23]. To evaluate the Grounded System Dynamics (GSD) method, we set objective methods and techniques that are founded on the purpose of the resulting method (GSD), rather than basing it on a priori (derived from logic or without observed facts) notion of reality [60].

![Figure 7.1: GSD evaluation overview](image)

In figure 7.1, we present the GSD evaluation overview. As depicted in figure 1.3, the design cycle in design science has two main iterations i.e. assess and refine. In this chapter therefore, we present the evaluation results.

To explain figure 7.1, initially we had two modeling methods i.e. system dynamics and object-role modeling (see figure. After identify relationships between ORM and SD
constructs, a new method was derived (Grounded System Dynamics). This new method was evaluated using three evaluation methods i.e. analytical, experimental and case study. During each of these evaluation methods, a number of techniques were applied. For analytical evaluations, we used walkthroughs and questionnaires; for experimental evaluations, we used focus group and questionnaires. Case study however, was used both as a unit of analysis and a research method during the application of analytical and experimental evaluations; and to derive a case [144]. To arrive at the case study described in subsection 7.1.3, we used the following techniques: interviews, archival reading and observation. As a result of the experimental evaluation, we were able to get the initial evaluations of the GSD transformation steps by applying it to a real case study. The refined steps in the experimental evaluation were further evaluated using analytical evaluations. By conducting these evaluations, we were able to obtain an in-depth expert opinion on the GSD transformation steps. During this phase (evaluation), we started with focus group sessions (experimental) and thereafter structured walkthroughs sessions (analytical) [184]. A detailed explanation is given and a road map presented in figure 7.2. With these evaluations, we were able to assess and refine (design cycle) the GSD method as the design science methodology requires [86].

In figure 7.2, we present a detailed GSD evaluation road map indicating the routes taken while conducting the evaluations for GSD transformation.

**Route A:** As an initial step, we came up with GSD transformation steps. These steps, comprised of CLD to ORM transformation steps and ORM to SFD transformation steps (see chapter 6). To give a firm foundation for these transformations, we first identified the relations between the two methods (SD and ORM), see chapter 5. In figure 7.2, route (A) represents activities carried out in version 1 of the CLD to ORM transformation steps. On completing the initial GSD steps, we prepared documents for evaluating the CLD to ORM transformations steps. Among these documents were identified relations, a case for the CLD, questionnaires, read text comprising of different SD/ORM constructs, etc. After that, we identified participants that would take part in this evaluation. For route A and B, we used modelers with knowledge in either of the methods or other modeling methods. The identified participants were sent requests asking them to take part in the evaluation study. Not all participants accepted but we had a reasonable number for the focus group discussion (which is 6-8 participants per session). Depending on participants' responses, a schedule was drawn. For phase 1 version 1 of the CLD to ORM transformation steps, we used an experimental evaluation method (detailed findings are given in section 7.1).

**Route B:** Here, we used the same participants that took part in the evaluation of phase 1 version 1. We started by preparing documents for the ORM to SFD evaluations since we already had the initial GSD steps. These documents comprised of: ORM to SFD transformation steps with a running example, identified SD to ORM relations and an ORM model that was derived from version 1 or the CLD to ORM steps. During this evaluation, a lot of arguments were involved due to the dissimilarity in the method constructs therefore reaching a consensus took time.

**Route C:** Having received all feedback in phase 1 of the evaluations, we analyzed data
Figure 7.2: GSD evaluation road map

before embarking on the second phase of the evaluation. In this second phase, we used analytical evaluation method (walkthrough). To enable us obtain experts views on the GSD transformation steps, we interacted with a number of practitioners from ORM and SD community (their detailed feedback is presented in subsection 7.2.1). To obtain this data, we followed routes (C and D) represented in figure 7.2. In route (C), we first prepared documents for walkthrough sessions, identified participants
CHAPTER 7. EVALUATION FOR GROUNDED SYSTEM DYNAMICS METHOD

that were to take part in the walkthrough discussions (the identified participants were ORM expert/practitioners because they were in a better position to critique the resulting ORM model) and sent them requests. For those that accepted to take part in the walkthrough session, a schedule was made for each participant. We conducted CLD to ORM sessions first because the outcome of this evaluation was to act as the basis for the ORM to SFD transformation steps (route D).

Route D On completing Route C, we identified participants to take part in ORM to SFD transformation steps. In the interest of time, this step (identifying participants) was done in parallel with CLD to ORM walkthrough version II sessions. Participants that evaluated the ORM to SFD transformation were SD practitioners because they were in a better position to critique the resulting SFD model. After these evaluations (CLD to ORM version II and ORM to SFD version II), the resulting refined steps were merged to make a complete refined GSD steps. From the received feedback, conclusions were drawn and suggestions for improvement were stated. Due to some feedback from SD practitioners, we came up with chapter 8. In chapter 9 we introduce the decomposition mechanism that led to addition of more steps. These steps are given in table 9.1.

The rest of the chapter proceeds as follows: in section 7.1, we present experimental evaluations. Within this section, we present focus group results, contributions and challenges faced during the validation experiments. In section 7.2, we present analytical evaluations. Under this section, feedback from walkthroughs is presented. Finally, we present discussions and reflections on the GSD procedure in section 7.3 and draw conclusions.

7.1 Experimental evaluation

To conduct these experimental evaluations, we followed a process depicted in figure 7.3. First, we came up with initial GSD transformation steps. Then prepared documents in line with the initial GSD transformation steps. These documents included: defined constructs, identified relations, questionnaires and a case to use as an example. After preparing the documents we identified participants that were to take part in this study. These participants were chosen because they had a level of understanding of the methods used and modeling in general. After identifying participants, we sent them requests asking them to take part in evaluating the GSD transformation steps in a focus group setting. Not all participants gave a positive feedback. For those that accepted to take part in the study, we asked them to suggest dates for the focus group session. It was on these provided dates that we based to schedule the focus group sessions. After scheduling the focus group sessions, we sent documents that were to be used in the evaluations to participants. We did so to give them ample time to familiarize with the research problem, case and method constructs. Before starting the discussions, the researcher made sure that all materials that were required for the sessions were available e.g. flip chart, markers, audio recorder etc. At the beginning of the focus group discussions, the moderator started by giving an overview of the study within which a research problem was stated. During the
focus group session, participants used both their knowledge on the methods and the provided documents to evaluate each GSD step. At times, there was need to build consensus. This occurred when they had different views/opinions on either the content or ordering of steps. To manage these differences, the following steps were taken: involved parties were asked to give factual statements that they believed to be relevant to the defined step, develop common assumptions and together use the gathered information to reach a decision. After reaching an agreement, conclusions were drawn and their views incorporated into that evaluated step. In case they agreed that the contents in that step were fine, they proceed to the next step. This was done until all steps were evaluated.

7.1.1 Setup of focus group sessions

As stated earlier, we used experimental evaluations in a focus group setting. A focus group is defined by [194] as a moderated discussion among six to twelve persons discussing a topic under the direction of a moderator whose role is to promote interaction and keep the discussion on the topic of interest. We used focus group discussions to allow us obtain participants feedback on the given steps and examine the applicability (use) of the steps. This feedback was used as the basis for refinement and improvement of CLD to ORM transformation steps. More about focus groups can be found in [114, 16, 194].

Before presenting the feedback, we present a summary of the activities that took place in each focus group session (see table 7.1). In total, we conducted three sessions with 6 to 8 participants. On the first and last days all participants were present (8 participant) but on the second day two participants sent in apologies and we could not get replacements. Prior to conducting these sessions, participants were given reading text ex-
Table 7.1: CLD-ORM focus group session execution summary

<table>
<thead>
<tr>
<th>Experimental evaluations</th>
<th>Number of participants</th>
<th>Aim of the evaluation</th>
<th>Summary of Activities</th>
</tr>
</thead>
</table>
| Session 1                | Eight                  | To evaluate CLD to ORM transformation steps using a given case. | Define research problem | Participants were given 30 minutes to familiarize with the given problem and ask questions where need be.
|                          |                        |                       | Identify CLD variables | During this phase there was need to build consensus on some of the identified variables which led to mismanagement of time.
|                          |                        |                       | Develop a CLD model | The cause and effect relationships among CLD variables were identified.
|                          |                        |                       |                       | Polarities were added to each CLD directional arrow.

| Session 2                | Six                    | - Participants completed the Causal Loop Diagram. |                       |
|                          |                        | - Participants were asked to take a break of 10-15 minutes. |                       |
|                          |                        | - After the break, participants were provided with CLD to ORM transformation steps. |                       |
|                          |                        | - The moderator explained each step as participants applied to the developed CLD. |                       |
|                          |                        | - In this session participants were able to complete steps 1-2 and start on step 3. |                       |

| Session 3                | Eight                  | - Participants were asked to complete step 3 before proceeding to steps 4 and 5. |                       |
|                          |                        | - At each step, participants critiqued the upshot. |                       |
|                          |                        | - On completion, conclusions on the outcome and transformation steps were drawn. |                       |

plaining ORM and SD constructs and how they can be applied. Provision of this text was to enable participants familiarize with the constructs, and to prepare them for the sessions. Materials used in these sessions included; makers, flip charts and an audio recorder. Note that the models provided in this section have been redrawn because flip charts were not easily transferable to the desktop.

7.1.2 Analyzing focus group results

To successfully analyze data collected during the focus group discussions, we used the inductive approach. With this approach, analysis is guided by specific objectives and the procedure is systematic [203]. In this study, the analysis phase comprised of a number of activities e.g. coding, classifying or categorizing of raw data, transcribing of data from audio recorder, looking for emerging patterns etc. One of the most time consuming activity was merging of collected data from different sources i.e. audio recorded data (this had to be transcribed first), questionnaire data (from open ended questions), notes taken by the researcher during the sessions and some comments or remarks some of the participants had put down on their note books during the sessions.

Coding and categorizing of findings

To categorize data, different colored highlighting text makers were used. Each color was used to represent a particular remark/view made by a participant on either the construct/defined ORM-SD relation or GSD step. The essence of categorization was to identify a portion or unit of data as belonging to or representing a common observable fact. Categorization involved giving labels to instances of observable facts found in the collected data. The different phrases, sentences or text that pointed to the same GSD step were given unique codes. Coding is marking pieces of data with symbols, descriptive words or category names. It starts with a summary of the text being examined [203]. The role of coding was to take note of relevant observable facts by collecting examples of these facts and analyzing them in order to find common or different patterns and struc-
7.1. EXPERIMENTAL EVALUATION

7.1.1 TURE S [11]. To code data, first we used symbols to mark views on a particular constructs, then on the defined relation (i.e. between ORM and SD constructs) and finally with the emerging defined GSD step.

Although there are qualitative data analysis packages like NUD. IST that allow codes to be attached to data and sorted in a variety of ways [161], we coded our data manually. This was because we had little knowledge of their (packages) existence worse still how to use them. Through creating categories and coding of data, conceptual schemes that suit the GSD steps were generated. These schemes helped in comparing existing steps with CLD to ORM steps e.g. ORM’s seven Conceptual Schema Design Procedural (CSDP) steps, to change or delete categories and to organize or order steps. Through coding the researcher was able to create order of participant’s remarks.

Finding patterns and relationships

In order to identify patterns, and relationships that emerged from the collected data, we used the detailed coded data. First, we identified similarities in different sets of data followed by differences. There were cases when the given descriptions could not fit into the given steps. In this case, we put our focus on the secluded data to find explanations as to why the given concepts do not fit the patterns. To clearly have a fully blown portrayal of patterns and relationships of the findings, we moved back and forth. Thus we were able to find revelations, contradictions, and exceptions around a particular step.

Examining and summarizing data

After finding the patterns and relationships in the collected data, we examined the data. The process of examining data was iterative with identification of patterns and relationships. This was because in both phases we were required to compare and contrast each given remark, gauge or determine how the given remarks fit together and identify or create flow in the examined data. By doing so we were able to come up with a summary of data presented in subsection 7.1.3.

This summary comprised of what we learnt, a step by step summary of the ORM-SD relations and a summary of GSD steps that emerged in the collected data. We indicated a few key quotations that illustrate the context in which the participant made his/her suggestion. After this we synthesized our findings across multiple data sources.

7.1.3 Presentation of results

To apply the CLD to ORM steps, we used a case “the process a pregnant woman at a maternity ward goes through at Ugandan hospitals (Intrapartum process in Ugandan Hospitals)” in focus group sessions. This case was used both as an experiment for the application and an evaluation mechanism of the CLD-ORM transformation steps [209]. Case study as defined in [44] is a research strategy which focuses on understanding the dynamics present within a single setting. Case study may involve either single or multiple
cases and numerous levels of analysis [230]. It combines data collection methods such as archives, interviews, questionnaires and observations. To collect case data, we used observation, archives and interviews. In this section, we start by giving more background to the case, then present what transpired in the focus group sessions and finally evaluations and challenges faced during the experiment.

Case: Intrapartum process in Ugandan hospitals

Intrapartum is the time from the onset of true labor until the delivery of the infant or placenta. In this case study we sampled out three Ugandan hospitals. Selection of these hospitals depended on a number of factors; 1) Location, 2) Availability of Health services, 3) Human resource (doctors and nurses), 4) Size of the hospital, 4) Level (grade) of the hospital, 5) Patients received per day etc. We visited all hospitals and the health center (Mukono Health Center, Kawolo Hospital and Mulago Hospital) to observe, note down the process, record details on some activities in these hospitals like; doctor monitoring time, patient arrival time, number of patients received per day, activities in the labor ward, archival data, observe patients day to day behaviors. Data was collected from labor suites only. This was done for a period of three month.

The process: A patient comes to the labor ward with her antenatal card from the antenatal clinic. She queues up. Her waiting time depends on a number of factors which are; her arrival time, the number of patients around and number of nurses on duty. When her turn comes, the nurse on duty takes her history and then examines her. This examination takes approximately 30 minutes for Mukono and Kawolo and 15-20 minutes for Mulago. The nurse establishes the patient’s dilation stage. If the patient is 4cm dilated, she is admitted to the general ward. She only returns for examination if there is any complication or after 4 hours. During this time, after every 30 minutes monitoring of the labor progress, status of the mother and cervical dilation is done. When the patient is 8cm dilate, she is taken to the delivery room. While there, the nurse monitors descending of the head 2 hourly and the sticker. When the patient has 10cm dilate, she is ready to give birth. After delivery, she is taken back to the general labor ward. Normal delivery patients stay at the labor ward for a maximum period of 24 hours and patients who have had caesarian birth stay for a period of 4-7 days. On discharge, the baby is taken for immunization.

First session

In the first session we used an open format to allow group members express their views without openly guiding the direction of the discussion [217]. Participants were given approximately 30 minutes to go through the case described in chapter 6 subsection 6.2.1, methods constructs and ask questions where need be. Thereafter, they were asked to brainstorm and come up with CLD variables that were later used to construct a CLD. During the focus group session, questions like; “Why have a CLD to ORM and not a CLD to Stock and flow and then ORM?”, “Of what importance is it to introduce ORM into the traditional System Dynamics CLD to SFD procedure?” were asked and addressed by the moderator. As the participants identified the different CLD variables, the moder-
ator listed them on a flip chart and were as follows: Arrival time, No of patients available, Number of nurses on duty, Doctors on call, Doctor response time, Patient history, Dilation (4cm, 8cm, 10cm), Delay between dilation, Monitoring time, Labor ward(s), Patient examination time, General labor ward, Antenatal card, Waiting time, Delivery room, Normal progress, Number of beds in the delivery room, Abnormal progress, Normal delivery, Baby immunization, Patients discharged, Patient admissions, Babies born, Admission beds. For some of these variables, consensus building was need which wasted a lot of time. On completing the variable identification process, participants embarked on identifying the influences between or among these variables. In figure 7.4, we present both an incomplete and complete CLD model that were derived during the first and second session respectively. In conclusion to this session, participants suggested that some of

Second session

In the second session, participants started off by completing the partial CLD model presented in figure 7.4. Completing this CLD model did not take a lot of time because part of the model had been developed in session one. After completing the CLD, participants took a break of 10-15 minutes and then embarked on what we refer to as the main problem for this evaluation “transforming a CLD into an ORM model”. Each participant was given a sheet of paper containing steps on how to transform a CLD into an ORM model. Participants were asked to follow the given steps, freely critique each step highlighting weaknesses, strength and make suggestions for improvement.

For step 1 of the CLD-ORM transformation process, the moderator provided participants with a guiding question: Who are the main role player(s) in a given CLD variable? and a sheet of paper to develop their ideas. As participants identified objects in each
Table 7.2: Objects identified from CLD variable names

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Identified objects</th>
<th>Variable Name</th>
<th>Identified objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nurses on duty</td>
<td>Nurses</td>
<td>Birth (net or number of births)</td>
<td>Patient and Nurse</td>
</tr>
<tr>
<td>Patient arrival</td>
<td>Patient</td>
<td>Patient discharge</td>
<td>Patient, Nurse and Bed</td>
</tr>
<tr>
<td>Number of patients in queue</td>
<td>Patients</td>
<td>Monitoring time</td>
<td>Patient, Nurse and time</td>
</tr>
<tr>
<td>Available admission beds</td>
<td>Admission bed</td>
<td>Dilation</td>
<td>Patient</td>
</tr>
<tr>
<td>Available delivery beds</td>
<td>Delivery bed</td>
<td>Free ward beds</td>
<td>Ward bed</td>
</tr>
<tr>
<td>Waiting time</td>
<td>Patient, Time and Nurse</td>
<td>Admission rate</td>
<td>Patient, Nurse and Time</td>
</tr>
<tr>
<td>Nurse response time</td>
<td>Nurse and time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient examination time</td>
<td>Patient, Time and Nurse</td>
<td>Admitted Patients</td>
<td>Patient and admission bed</td>
</tr>
<tr>
<td>Patient History available</td>
<td>Patient and Patient history</td>
<td>Delay between dilation</td>
<td>Patient</td>
</tr>
</tbody>
</table>

Having completed step 1, participants proceed with step 2. In this step participants were asked to group and classify similar objects that existed in each CLD variables. By grouping and classifying similar objects in each CLD variable, a list of different object types was made. At this point participants were very involved in the focus discussion and putting all pieces together was not hard thus this step did not take a lot of time. During this session the following classification were identified; Nurse, patient, time, Antenatal card, Dilation stage, delivery room baby ward bed, patient history and bed. Participants suggested that this step be merged with step 1 because they are much alike.

After completing step 2, participants proceeded with step 3. The guiding questions shown in chapter 6 subsection 6.1.1 step 3 (What role do objects in object type (A) play in object type (A)?; What role do objects in object type (A) play in object type A1,A2,A3,...An?) were read to participants by the moderator.

Participants started by identifying all roles played by objects in the same object type then identified roles played by objects in other object types. As a result fact types between or among object types were identified. This step involved a lot of arguments and most times, building consensus was necessary. Due to these arguments, this step took much more time than all the previous steps. We also noted that for each variable, participants referred to the original CLD model in figure 7.4. In table 7.3 we present some of the results including variable names, derived ORM models and verbalizations.

In conclusion, having participants engage in the CLD development process was of great advantage because they get a better understanding of all variables added to the model. Secondly, participants are more active. Thirdly, participants can easily identify and correct issues in the already developed CLD as they proceeded with the transformation steps which leads to a better and improved CLD. Finally, there are cases (especially in step 3) where participants had different views but were quick to consult the moderator and domain expert. This indicates that, in order to apply these steps both the modeler and
Table 7.3: Some of the results of step 3 discussions

<table>
<thead>
<tr>
<th>Variable name</th>
<th>ORM model</th>
<th>Verbalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitoring time</td>
<td><img src="image" alt="Monitoring time ORM diagram" /></td>
<td>Patient is monitored by Nurse for Duration. For each Nurse and Patient, that Patient is monitored by that Nurse for at most one Duration.</td>
</tr>
<tr>
<td>Waiting time</td>
<td><img src="image" alt="Waiting time ORM diagram" /></td>
<td>Patient waits to be admitted for Duration. Each Patient waits to be admitted for at most one Duration. It is possible that the same Duration is for more than one Patient.</td>
</tr>
<tr>
<td>Patient history availability</td>
<td><img src="image" alt="Patient history ORM diagram" /></td>
<td>PatientHistory is available.</td>
</tr>
<tr>
<td>Nurse response time</td>
<td><img src="image" alt="Nurse response time ORM diagram" /></td>
<td>Nurse responds at Time. It is possible that more than one Nurse responds at the same Time and that the same Nurse responds at more than one Time. Each Time, Nurse combination occurs at most once in the population of Nurse responds at Time.</td>
</tr>
<tr>
<td>Available admission beds</td>
<td><img src="image" alt="Available admission beds ORM diagram" /></td>
<td>Each Admission Bed is an instance of Bed. Admission Bed is available.</td>
</tr>
</tbody>
</table>

domain expert need to be present.

**Third session**

Before deciding on whether to merge the models first or constraints, participants had a recap on what transpired in the previous session. Majority of the participants argued to merge the model before adding constraints to avoid repetition of step 4. Their reason was that if they handle step 4 before step 5 they will have to revisit step 4 after the merger.

As a result of the CLD to ORM transformation steps, the ORM model presented in figure 7.5 (with adjustments) is derived. Note that adjustments were made to this model because 1) since a flip chart was used, we had to join different pages to have a complete model thus the model presented in figure 7.5 was reconstructed using NORMA. 2) Some fact types have been changed to give more meaning to the model verbalizations, and 3) after emailed the reconstructed model to participants changes were made and are included in the given model. In table 7.4, we present verbalizations for figure 7.5.

In the interest of time, we did not discuss step by step of the ORM to Stock and Flow Diagram (SFD) transformation steps. However, each participant gave overall remarks on the provided steps. This was possible because we had given them ample time to read and think through the initial steps. The given remarks were noted down by the moderator and for those that had their remarks on paper, they were kind enough to leave them behind. Although participants generally agreed with the ORM to SFD steps, they raised a few concerns. One of the concerns raised was that "at what level/step do we define input
parameters?” The given steps do not state or show at what stage the SFD input parameters are defined. To address this concern, we added new steps to the initially defined steps. This step however cannot stand on its own as defining input parameters is done at every point in the model i.e. every time a new variable is introduced, the parameters have to be defined and these parameters may have an effect on the already defined parameters. Therefore we could not make ‘identification of input parameters’ an independent step instead we state that it is iterative and therefore should be incorporated/integrated within all the defined steps apart from step five.

The second concern was that researchers were limiting converters to fact types with more than one role (binary, ternary fact types etc), this some participants found it to be unrealistic. One of the participants explained as follows: a converter is a decision function. This function receives information, processes it and generates an output inform of action or more information. The converter is usually identified with a specific decision making process and within are quite complex information processing or input parameters. In line with his argument, in [51] Forrester refers to a set of decision makers as ‘players’ whose decisions and actions are coupled. This therefore implies that to a certain extent converts relate to fact types with more than one role because within these fact types are decisions. In [142] Morecroft elucidates Forrester’s statement by using an example of a sales organization. In this example he states that each ‘player’ in that organization is represented by a decision function with information input and output but how these decision functions are coupled is the responsibility of other SFD symbols. Considering Morecroft’s example, a ‘player’ is an object in ORM. Can we assume that each object in an object type has a decision function with information input and output? If so, then there are characteristics of a converter in an object. In conclusion therefore although some participants agreed with step three “Identify all possible converters”, they urged the
7.1. EXPERIMENTAL EVALUATION

Table 7.4: Verbalizations for figure 7.5

<table>
<thead>
<tr>
<th>ORM model Verbalizations</th>
<th>Medical Person assists with births of Baby.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient arrives at LaborWard at Time.</td>
<td>It is possible that more than one Medical Person assists with births of the same Baby.</td>
</tr>
<tr>
<td>For each Time and LaborWard, at most one Patient arrives at that LaborWard at that Time.</td>
<td>and that more than one Baby is birthed by the same Medical Person.</td>
</tr>
<tr>
<td>Patient is in queue.</td>
<td>Each Baby, Medical Person combination occurs at most once in the population of Medical Person assists with births of Baby.</td>
</tr>
<tr>
<td>If some Patient is in queue then that Patient arrives at some LaborWard at some Time.</td>
<td>For each Person, exactly one of the following holds:</td>
</tr>
<tr>
<td>Medical Person admits Patient for Duration.</td>
<td>some Baby is that Person;</td>
</tr>
<tr>
<td>For each Duration and Patient, at most one Medical Person admits Patient for that Duration.</td>
<td>some Medical Person is that Person.</td>
</tr>
<tr>
<td>Patient waits to be admitted for Duration.</td>
<td>Medical Person is on duty.</td>
</tr>
<tr>
<td>Each Patient waits to be admitted for at most one Duration.</td>
<td>For each Medical Person and Patient, that Medical Person examines Patient for Duration.</td>
</tr>
<tr>
<td>It is possible that the same Duration is for more than one Patient.</td>
<td>Nurse monitors Patient for Duration.</td>
</tr>
<tr>
<td>Nurse responds at Time.</td>
<td>Each Patient is monitored by some Nurse for some Duration.</td>
</tr>
<tr>
<td>It is possible that more than one Nurse responds at the same Time and that the same Nurse responds at more than one Time.</td>
<td>for each Patient and Duration, at most one Nurse monitors Patient for that Duration.</td>
</tr>
<tr>
<td>Each Time, Nurse combination occurs at most once in the population of Nurse responds at Time.</td>
<td>Patient is at DilationStage.</td>
</tr>
<tr>
<td>Patient has AntenatalCard.</td>
<td>Each Patient is at at most one DilationStage.</td>
</tr>
<tr>
<td>Each Patient has exactly one AntenatalCard.</td>
<td>Each DilationStage is for some Patient.</td>
</tr>
<tr>
<td>Each AntenatalCard is for exactly one Patient.</td>
<td>It is possible that the same DilationStage is for more than one Patient.</td>
</tr>
<tr>
<td>Patient has PatientHistory.</td>
<td>Nurse establishes DilationStage.</td>
</tr>
<tr>
<td>Each Patient has exactly one PatientHistory.</td>
<td>It is possible that more than one Nurse establishes the same DilationStage.</td>
</tr>
<tr>
<td>Each PatientHistory is for exactly one Patient.</td>
<td>and that more than one DilationStage is established by the same Nurse.</td>
</tr>
<tr>
<td>Nurse records PatientHistory.</td>
<td>Each DilationStage, Nurse combination occurs at most once in the population of Nurse establishes DilationStage.</td>
</tr>
<tr>
<td>Each PatientHistory is recorded by exactly one Nurse.</td>
<td>Each DilationStage is established by some Nurse.</td>
</tr>
<tr>
<td>It is possible that the same Nurse records more than one PatientHistory.</td>
<td>Medical Person discharges Patient.</td>
</tr>
<tr>
<td>For each Medical Person, exactly one of the following holds:</td>
<td>Each Patient is discharged by at most one Medical Person.</td>
</tr>
<tr>
<td>some Nurse is that Medical Person;</td>
<td>It is possible that the same Medical Person discharges more than one Patient.</td>
</tr>
<tr>
<td>some Obstetrician is that Medical Person.</td>
<td>Bed is occupied by Patient for Duration.</td>
</tr>
<tr>
<td>Baby is a newborn</td>
<td>For each Bed and Duration, Bed is occupied by at most one Patient for that Duration.</td>
</tr>
<tr>
<td>Baby is birthed by Patient.</td>
<td>Bed is in DeliveryRoom.</td>
</tr>
<tr>
<td>Each Baby is birthed by exactly one Patient.</td>
<td>Each Bed is in at most one DeliveryRoom.</td>
</tr>
<tr>
<td>It is possible that the same Patient births more than one Baby.</td>
<td>It is possible that more than one Bed is in the same DeliveryRoom.</td>
</tr>
<tr>
<td>For each Bed, exactly one of the following holds:</td>
<td>For each Patient, exactly one of the following holds:</td>
</tr>
<tr>
<td>some Admission Bed is that Bed;</td>
<td>that Patient is admitted;</td>
</tr>
<tr>
<td>some Delivery Bed is that Bed.</td>
<td>that Patient is discharged.</td>
</tr>
<tr>
<td>Admission Bed is occupied.</td>
<td>For each Admission Bed, exactly one of the following holds:</td>
</tr>
<tr>
<td>Delivery Bed is occupied.</td>
<td>that Admission Bed is empty;</td>
</tr>
<tr>
<td>Delivery Bed is empty.</td>
<td>for each Delivery Bed, exactly one of the following holds:</td>
</tr>
<tr>
<td>Bed is in LaborWard.</td>
<td>that Delivery Bed is empty;</td>
</tr>
<tr>
<td>Each Bed is in at most one LaborWard.</td>
<td>that Delivery Bed is occupied.</td>
</tr>
<tr>
<td>It is possible that more than one Bed is in the same LaborWard.</td>
<td>for each Delivery Bed, exactly one of the following holds:</td>
</tr>
<tr>
<td>For each Bed, exactly one of the following holds:</td>
<td>that Admission Bed is occupied;</td>
</tr>
<tr>
<td>that Bed is in some LaborWard;</td>
<td>that Bed is occupied.</td>
</tr>
<tr>
<td>that Bed is in some DeliveryRoom.</td>
<td>for each Delivery Bed, exactly one of the following holds:</td>
</tr>
</tbody>
</table>

researchers not to limit converters to only fact types with more than one role (unary fact types).

Participants suggested that step five "Create sectors" should be made optional and researchers should clearly state that an SD model can be complete without sectors. As this will help the readers not assume that for any SD model to be complete, it should have sectors yet this is not true.

As a general remark, one participant said that, looking at the existing SD modeling steps for constructing system dynamics models e.g. in [159, 167, 169], conceptu-
alization takes the second position after problem definition. Under conceptualization is where causal loop diagrams are constructed and sometimes this phase involves a number of stakeholders e.g. group modeling [216]. At what point do you think conceptualization should be placed in your new method (GSD)? To respond to this question, the researchers stated that, conceptualization (conceiving an idea) does not only stop at the CLD. Although group modeling to a certain extent addresses some of the issues with SD model conceptualization, it does not address ambiguity and lack of detail in these models (CLDs). Yet, it is because of these issues that transformation of causal loop diagram (CLD) variables into Stock and Flow Diagram (SFD) is hard. The introduction of ORM into the process of transforming CLD into SFDs therefore is of great importance as we stated in chapter 1. We introduce a domain modeling method (ORM) in the process of transforming a CLD into SFD because domain modeling methods help to identify relationships among entities within the scope of the problem domain, provide a structural view of the domain, capture a detailed representation of the system in terms of system elements and their interactions to provide a means of understanding a given scenario and they have a precise and consistent mechanism for conceptualizing reality based on which a stock and flow diagram can be realized [156, 58].

7.1.4 Contributions and challenges

Involving clients in SD model formulation and conceptualization is a technique we borrowed from Group Model Building (GMB) [173, 216]. In our case however, we used a small number of modelers with knowledge on ORM, SD and other modeling methods and not clients as is the case in GMB. We also applied a mixture of modeling techniques e.g. brainstorming, case study etc. during the focus group sessions [166, 216]. The case study technique was used as a subset in the focus group sessions to enable us apply the CLD to ORM transformation steps [173].

New insights and refined steps

During the sessions, a number of views were given. The provided views were used not only to refine the CLD to ORM transformation steps but also to streamline the entire Grounded System Dynamics procedure. In table 7.5, we present the refined CLD to ORM transformation steps, purpose for each step, strengths and weaknesses identified during the application of the steps.

During the evaluations, we noted that the CLD to ORM transformation process is better carried out either in parallel or iteratively and with a domain expert present. The role of the domain expert is to help the modeler clarify the ambiguities found within the CLD variables.
Table 7.5: Refined ORM to CLD steps, strengths and weaknesses

<table>
<thead>
<tr>
<th>Steps</th>
<th>Description</th>
<th>purpose</th>
<th>Strengths</th>
<th>weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identify object(s) in each variable</td>
<td>stating objects pertaining to a particular CLD variable</td>
<td>- Gives a clear understanding of objects in each variable.</td>
<td>The steps are exhausting if the SD model has numerous variables or complex.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Easy to complete.</td>
<td>Takes a lot of participant and moderator time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- CLD variable names are made explicit.</td>
<td>Identified object types need to be further streamlined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Easy for the modulator due to limited involvement in succinct formulation of ideas.</td>
<td>Does not differentiate value types from object types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Engaging for participants if they have taken part in the CLD modeling process.</td>
<td>Does not work well if participants do not understand the underlying ORM and SD concepts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Requires a domain expert to be present.</td>
<td>Participants' bias may sift out good ideas.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Produces clear, non-redundant and relevant contributions that lead to refinement of the CLD model.</td>
<td>Consensus building among participants is exhausting for the moderator and takes time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Promotes shared understanding of the SD and ORM constructs among participants.</td>
<td>Hard for participants with limited ORM or SD modeling experience.</td>
</tr>
<tr>
<td>2</td>
<td>Collect and group (classify) similar objects</td>
<td>Specifying object types in a CLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Identify and connect roles played by objects in a particular object type a. To other object types b. To objects in the same object type</td>
<td>specifying roles played by objects in each object type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Merge the models</td>
<td>To view the model as a whole</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Add all constraints and mandatory roles</td>
<td>To define model boundaries</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Challenges faced during the validation experiment

We admit that conducting focus sessions gave us immense benefits but there were numerous challenges too. A concern of one participant was that the two methods (SD and ORM) have different underpinned meaning (concepts) which makes defining their relations more complex. For example, in this experiment, the effects of both CLD polarities (positive and negative) and loops (reinforcing and balancing) cannot be represented in ORM. This therefore means that constructs for both methods cannot be exhaustively captured.

Secondly, considering participant’s views during the focus sessions, some participants were either in agreement or disagreement. This made analysis of audio recorded data difficult because the researcher could not easily tell whether participants’ agreement(s) resulted from coercion or proper explanations from the moderator. Thirdly, getting all participants to fit the experiment sessions into their schedules was not an easy task. Therefore we feel that better results would have been reached if a one to one discussion was conducted. Furthermore, the location (Makerere College of Computing and Information Sciences) where these focus sessions were conducted also left a lot to be desired because of the constant power cuts, strikes etc. This led to postponement and rescheduling of sessions hence delays in the overall experiment.

Finally, design research strives to recruit participants that are familiar and are potential users of the study results [204]. By using participants from different backgrounds (by different we mean they had knowledge of either both or one method(s) under study but specialize in one), we were able to note that some of the participants were rigid especially those that were experts in one of the method. They seemed not to want any changes
although they admitted to the weakness in the methods and saw the importance of the study and the arguments between participants during the sessions consumed a lot of time leading to postponement of some activities for the next session.

7.2 Analytical evaluation

Under analytical evaluations, we used structured walkthroughs to receive more concrete feedback on the GSD transformation steps. A structured walkthrough is a step by step review discussion where practitioner(s) divulge faults in a study that may prevent it from achieving its stated aim.

![Overview of the GSD practitioner feedback procedure](image)

In figure 7.6, we present an overview of the GSD practitioner feedback procedure which is made up of three stages. In stage one, we evaluated CLD to ORM transformation steps [209]. This stage was criticized and evaluated by ORM practitioners. In the second stage, the ORM model was transformed into a system dynamics stock and flow diagram [208]. Stage two was critiqued and evaluated by SD practitioners. In the third and final stage we refine the GSD transformation stages. This last stage depends on the results or views received from the previous two steps. We categorized practitioners into two groups so that the resulting model from each phase was critiqued and evaluated by experts in that field. Hence, a careful evaluation of the utility of the artifact was obtained.

7.2.1 Setup of walkthroughs

To allow us have a detailed evaluation, we used bi-lateral structured walkthroughs. Each walkthrough comprised of two persons i.e. the researcher and the practitioner [184]. The walkthroughs were conducted in two main phases. The first phase involved three ORM academicians and two ORM practitioners. The second phase comprised of three SD practitioners. A summary of the walkthrough setup and activities are depicted in table 7.6.

For each given GSD step, the following process was done: Participant evaluated the defined step. If the participant was not content with the defined step, he was asked to give his views before moving to the next step. This feedback was noted down by the
7.2. ANALYTICAL EVALUATION

Table 7.6: Walkthrough setup

<table>
<thead>
<tr>
<th>Walkthrough details</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time requested per session</td>
<td>2 hours</td>
</tr>
<tr>
<td>Documents provided before session</td>
<td>Transformation steps and examples (Models)</td>
</tr>
<tr>
<td>Documents provided after session</td>
<td>Questionnaire</td>
</tr>
<tr>
<td>Number of practitioners per session</td>
<td>One</td>
</tr>
<tr>
<td>Total number of Practitioners that</td>
<td>Three ORM academicians, two ORM practitioners and</td>
</tr>
<tr>
<td>took part in this experiment</td>
<td>three SD practitioners</td>
</tr>
</tbody>
</table>

Activities at the walkthrough session

<table>
<thead>
<tr>
<th>Activities</th>
<th>Participant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduces research problem.</td>
<td>Researcher</td>
</tr>
<tr>
<td>2. Explains transformation steps.</td>
<td></td>
</tr>
<tr>
<td>3. Practitioner asks questions to clarify some details if necessary.</td>
<td>Practitioner</td>
</tr>
<tr>
<td>4. Practitioner and researcher together go through each step using given</td>
<td>Practitioner and</td>
</tr>
<tr>
<td>examples. The practitioner tries them out and makes remarks.</td>
<td>Researcher</td>
</tr>
<tr>
<td>5. Researcher records and takes notes where need be.</td>
<td></td>
</tr>
<tr>
<td>6. On completion of the steps, the practitioner gives his overall remarks</td>
<td>Practitioner</td>
</tr>
<tr>
<td>on the steps and case contents.</td>
<td></td>
</tr>
<tr>
<td>7. Researcher gives the practitioner an evaluation form. This form has</td>
<td>Researcher</td>
</tr>
<tr>
<td>questions which the practitioner answers and hands the evaluation form to</td>
<td></td>
</tr>
<tr>
<td>researcher.</td>
<td></td>
</tr>
<tr>
<td>8. Researcher thanks practitioner and session is concluded</td>
<td></td>
</tr>
<tr>
<td>9. Days after each session, the researcher sends details of the session to</td>
<td>Researcher</td>
</tr>
<tr>
<td>practitioner. Here the practitioner is expected to validate the information</td>
<td></td>
</tr>
<tr>
<td>before the researcher incorporates in the write-up.</td>
<td></td>
</tr>
</tbody>
</table>

researcher. After the walkthrough session, the researcher made an assessment of the given feedback. From which one of the following actions were taken:

1. Revise the evaluated step by incorporating the feedback.
2. Create and add a new step
3. Alternate step(s)
4. Delete step

On completion, participants moved to the next step. In case there were no changes to be made to the step (participants were content with the defined step), they immediately moved to the next step. After evaluating all the steps a final and general conclusion was given by the practitioner. After completing all the steps, we gave a questionnaire comprising of 4-5 open ended questions to the practitioner. The questions in this questionnaire focused on relevance, scope, interpretation and measurement (see Appendices B and C). In summary, figure 7.7 depicts walkthrough activities as were executed step by step.

7.2.2 Analysis of walkthrough feedback

To represent how we analyzed walkthrough feedback, we summarize the activities conducted in figure 7.8. To allow us fill in the gaps between the notes taken during the
walkthrough session, we listened to audio recorded feedback for each participant. As we did so, we transcribed all feedback that was relevant to the study. Thereafter, we examined the feedback and sent it to participants (each participant was sent their own transcribed feedback). After sending the transcribed data, we received three types of reactions:

1. Participants who made changes to their transcribed feedback,
2. Participants who did not respond or react to the sent transcribed feedback,
3. Participants that made no changes to their transcribed feedback (said it was okay).

For participants who made changes, we re-examined transcribed feedback to added new insights. On completion, we categorized and combined feedback according to resemblance. For participants that did not respond or react to the sent transcribed data, we assumed that they had nothing to add. Therefore, we used their original feedback (one we collected from walkthroughs) same applies to participants that made no changes to their transcribed feedback. For easy readability and clarity, we re-arranged feedback with respect to GSD steps. Finally, we reported our findings.

### 7.2.3 Presentation of results

In this section, we start off by presenting feedback from ORM practitioners followed by feedback from SD practitioners. Both parties were requested to experiment with the provided GSD steps and apply them to the provided example(s), critically comment on each step and support their answers. Materials used during these sessions included; Paper, pen and an audio recorder. For ORM practitioner sessions, we also provided CLD to ORM transformation steps shown in table 7.7 and CLD model(s).
### 7.2. ANALYTICAL EVALUATION

#### ORM practitioner feedback

In each session, we first stated the overall aim of the study "combining system dynamics with a domain modeling method (ORM)". Combining SD with a domain method allows us improve SD model conceptualization and reduce on SD model ambiguities [140, 13, 176]. As earlier stated domain modeling methods like ORM help to identify relationships among entities within the scope of the problem domain and provide a structural view of the domain.

![Figure 7.8: A representation of how walkthroughs were analyzed](image)

#### Table 7.7: CLD to ORM transformation steps as provided to ORM experts

<table>
<thead>
<tr>
<th>Steps</th>
<th>Guiding question</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Identify object(s) in each CLD variable.</td>
<td>Who is/are the main role player(s) in a given CLD variable? Who plays a role in this CLD variable?</td>
<td>Role player is a term used in ORM but here we use it as used in natural English.</td>
</tr>
<tr>
<td>2. Collect and group (Classify) similar objects.</td>
<td>Each classification is an object type.</td>
<td></td>
</tr>
<tr>
<td>3. Identify and connect roles played by objects in a particular object type a. To other object types b. To objects in the same object type</td>
<td>What role do objects in object type (A) play in object type (A)? What roles do objects in object type (A) play in object type (A, A, A, ...A)?</td>
<td>Where; [A, A, A, ...A] are object types identified in CLD model.</td>
</tr>
<tr>
<td>4. Add all constraints and mandatory roles.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Connect all roles from similar object types.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In Table 7.7 are the initial CLD to ORM transformation steps as were provided to ORM practitioners. Note that these steps do not cover derived fact types and subtyping. The CLD model in Figure 7.4 worked as an example in the application these steps. The CLD model shown in Figure 7.4 together with the transformation steps in Table 7.7 were given to practitioners a few days to each session date which gave them ample time to familiarize and think through both the models and transformation steps.

In total we conducted eight walkthrough sessions. Five of the walkthrough sessions were with ORM practitioners and three were with SD practitioners. During these sessions, a number of remarks were made and are presented below.

**Feedback from ORM practitioners**

It was interesting to see two out of five practitioners present a global CLD representation shown in Figure 7.9 as their initial interpretation of the CLD to ORM transformation. Their interpretation of the CLD model was a very global representation. They explained this global model as follows; If every variable in a CLD model is depicted as an ORM object type, then it is an object type being influenced by another object type with either a plus (+) or a minus (-) sign. The verbalization for this global representation would then read as follows “every CLD variable influences or is influenced by another CLD variable in a positive or negative way”. Thus for every influence there is an anti symmetric “thing” which prompts the modeler to think of a circular constraint.

![Figure 7.9: A global CLD representation](image)

For step 1 ([Identify object(s) in each CLD variable]), all ORM practitioners agreed that before this step, it is important that the modeler identifies the kind of facts needed. If the modeler starts by identifying objects before having an idea about the kind of facts needed, then (s)he may have all sorts of objects that are not relevant to the problem domain. But if (s)he knows the kind of information required, then it is much easier for him to derive an ORM model.

Secondly, the modeler should have concrete examples of data. For example, in Figure 7.4 variable birth, what is it that the modeler wants to record? Does (s)he want to record that some baby was born at a particular time or the number of births? If a report on the type of data is availed then he would know what kind of data to record. Knowing

---

1 The Intrapartum case was not our first option. But during the first two sessions, ORM practitioners advised us to use a much simpler case because the one we had provided was very complex. With this change, we had to repeat the first two sessions.
that there is an event that occurs which is *birth* of a baby. In ORM, it is important that what the modeler wants to record about a given variable is explicitly stated. More so for variable *birth*, does he want to record when it happened or the name of the baby and the mother who birthed the baby? Just looking at variable *birth* one cannot know the exact data needed. But if the data is shown then the modeler knows what to record and from that, simple English verbalizations are derived. In conclusion, the given *step 1* focuses only on objects yet focus should be on facts first rather than objects. This is so because objects are the ingredients of facts.

Something else we could do to clear the ambiguities in a CLD, is to have a catalogue of terms explaining what the CLD variables mean. For example, looking at variable *free ward beds* does that mean the ward beds that are not occupied? If so then a definition ought to be given so that the modeler knows what ‘free’ really means. What is the difference between *free* and *available*? probably they should be called *available ward beds*, using *free* in a sense of *unoccupied* plus some other condition like the bed being cleaned, broken etc. In ORM we really worry about words say ‘she is free here’ but ‘available there’ since we choose different words may be *free* does not mean the same as *available*. If they do mean the same then we recommend using the same adjective but if they mean something different, it is critical that they are made explicit either in the descriptive paragraphs or in a little terminology column were technical hospital terms used are defined. Another option would be to do this informally in a paragraph.

Figure 7.10: Practitioner identifies objects in a CLD model
In ORM, we decide which facts are simply decided by the case and which facts are derived directly from other facts as part of the procedure. For example from the given CLD, if there are only two types of beds (delivery beds and admission beds) then clearly we have an equation. It would be natural in ORM to specify a derived fact type for the type bed. This would probably be derived from the number of free ward beds because the delivery beds and available admission beds would be fixed. With this remark, practitioners suggested that we include derivation rules. These derivation rules would later guide the modeler in identifying the stock and flow diagram input values. The CLD in figure 7.10 shows the practitioner trying to identify objects in each variable using initials. Thereafter slowly builds the ORM model (see figure 7.11).

Four out of five ORM practitioners found the use of the word ‘role player’ in the guiding question of step 1 confusing because it is a key word in ORM. They said that clarification on the use of this word (role player) may be required for the ORM community. Steps 1, 2 and 3 are quite clear but the order of steps 4 and 5 should be reversed because a lot more work is done if all constraints are added in step 4 before merging the model or the steps should be conducted in parallel. Finally, step 5 should be rephrased from ‘Connect all roles from similar object types’ to ‘Merge similar object types, or Integrate/merge the models or View integration’.

In conclusion, coming up with an ORM model from the given steps may require
7.2. ANALYTICAL EVALUATION

the modeler to keep the stock and flow diagram in mind because as a result of the ORM model, there must be a dynamic model. This means that during the modeling process, the modeler must have a quantitative view in mind. Normally in ORM modeling, this is not needed at all because the quantities are derivable fact types and interest is in individuals. The normal ORM way is an instance view i.e. modeling relationships between instances yet in SD we do not have to model relationships between instances.

Finally, note that information in a single causal loop diagram cannot be captured in a single database. This therefore means that for proper underpinning of SD with ORM, there may be need to integrate or merge different data sources (databases) [124].

ORM questionnaire feedback

At the end of each walkthrough session, a questionnaire with four questions was handed to each practitioner for answering. The questions in this ORM questionnaire focused on CLD to ORM transformation steps’ relevance, coverage/scope, measure and interpretation. In table 7.8, we present a summary of responses received and the questions asked.

<table>
<thead>
<tr>
<th>Questions</th>
<th>Ex</th>
<th>Acc</th>
<th>NI</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. How relevant are the CLD to ORM transformation steps?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Do the given steps provide enough information on how to derive an ORM model from a Causal Loop Diagram?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Overall how would you rank the CLD to ORM transformation steps?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Are the steps given easy to understand or interpret? If NO what do you think should be added to improve the given steps?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the first question, we wanted to know how relevant the CLD to ORM transformation steps were. Three out of five practitioners accepted that the given steps were relevant and the other two practitioners said they needed to be further improved. For the second question, two of the practitioners accepted that the provided steps give enough information on how to derive an ORM model from a causal loop diagram. The other three said that the steps needed to be further improved by either adding the derivation rules or more explanations. For the third question, one practitioner said that overall the CLD to ORM transformation steps were excellent, two practitioners said they were acceptable and the other two said they need to be further improved. For the fourth question, all practitioners found the provided steps easy to understand. Please note that we have not added most of the remarks given for each question. This is because we want to avoid recurrence of remarks since most of the reasons given were already stated during the walkthrough session and appear in subsections 7.2.3 and 7.3 respectively.
Revised CLD to ORM transformation steps

Basing on the given remarks in subsection 7.2.3, we now present the revised CLD to ORM transformation steps in table 7.9. In this table, we see that the number of steps has increased from five to six steps. This is because practitioners suggested that as a first step, facts should be defined before objects since objects are ingredients of facts. Secondly, the order of the last two steps has been reversed because practitioners said that if these steps are to be left as they were presented in table 7.7, the modeler would have to carry out the task of adding constraints twice i.e before the merger and after the merger. Every time a building block is added to another block, fact types and verbalization change this implies that the previously defined facts need to be revised hence, double work for the modeler. Furthermore, step five in table 7.7 has been changed from “Connect all roles from similar object types” to “Merge similar object types”. This is so because practitioners claim that steps make more sense and are more explicit when presented that way.

Table 7.9: Improved CLD to ORM transformation steps

<table>
<thead>
<tr>
<th>Refined CLD to ORM Steps</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. State all facts pertaining to a given problem domain.</strong></td>
<td></td>
</tr>
<tr>
<td>a. Document or make all variable names explicit by using either a catalogue or domain expert.</td>
<td></td>
</tr>
<tr>
<td>b. Since all information in a CLD model cannot be captured in one database consider using multiple databases. At the end, we may be required to merge the data sources in order to have at a complete CLD model.</td>
<td></td>
</tr>
<tr>
<td><strong>2. Identify object(s) for each CLD variable</strong></td>
<td></td>
</tr>
<tr>
<td>a. Make a list of all objects relating to each CLD variable. While listing these objects the modeler should keep in mind that the influence links in the CLD model may help in cases where the variable names are inexplicit.</td>
<td></td>
</tr>
<tr>
<td>b. For each CLD variable, the modeler should be able to answer this question “who are the main role player(s) [actor] in a given CLD variable?” This helps in identifying objects contained in each CLD variable.</td>
<td></td>
</tr>
<tr>
<td><strong>3. Collect and group (Classify) similar objects (each classification is an object type)</strong></td>
<td></td>
</tr>
<tr>
<td>a. All objects identified in the previous step should be grouped to define object types.</td>
<td></td>
</tr>
<tr>
<td>b. Define the type (class) of objects in each category e.g. is it a value type or an object type and state the properties for each classification (object type). Note that all variables identified should be represented irrespective of how often they appear.</td>
<td></td>
</tr>
<tr>
<td><strong>4. Identify and connect roles played by objects in a particular object type</strong></td>
<td></td>
</tr>
<tr>
<td>a. To other object types</td>
<td></td>
</tr>
<tr>
<td>What roles do objects in object type (A), play in object type (A, A_1, A_2, A_3, \ldots, A_n) (A_1 / A_2 / A_3, \ldots, A_n) are object types identified in CLD model</td>
<td></td>
</tr>
<tr>
<td>b. To objects in the same object type</td>
<td></td>
</tr>
<tr>
<td>What role do objects in objects in object type (A) play in object type (A)?</td>
<td></td>
</tr>
<tr>
<td>c. Define all derivable facts and state their formulas.</td>
<td></td>
</tr>
<tr>
<td><strong>5. Merge the models</strong></td>
<td></td>
</tr>
<tr>
<td>Here the modeler is expected to gradually add the different blocks until a full blown model is reached or obtained. Note that step six and five can be done concurrently depending on the size of the model.</td>
<td></td>
</tr>
<tr>
<td><strong>6. Add all constraints and mandatory roles</strong></td>
<td></td>
</tr>
<tr>
<td>As already stated this step can be done in parallel with step five but a thorough check should be done after merging the model to verify its validity</td>
<td></td>
</tr>
</tbody>
</table>

System dynamics practitioner feedback

Prior to system dynamics practitioner sessions, participants were emailed a document containing a running example of the ORM to SFD transformation steps and ORM to SFD
relations. As a running example in these sessions we used a brief paper submission process example in subsection 4.1. ORM and SD element used in this section are defined in chapter 4 but for a detailed treatment of these elements see [77, 193] respectively.

**Feedback from SD practitioners**

In step 1 (Identify all possible stocks), looking at one of the ORM examples provided, 'paper' would be taken as a unit of measurement that flows through some chain and the different states as different stocks in this chain (unary roles). This can be represented as a co-flow. A subset is a parable example of a typical co-flow structure. From the given example academics can do more than one thing at a time (act as both authors and reviewers) in a parallel structure or Sequential structure (see practitioner's representation in figure 7.12). Although Researchers equate a role to a stock, from the given definition of object types they also have some characteristics of a stock. But since the methods are not seemingly similar maybe one should consider using the ORM element that is most similar to stocks. Identifying stocks is a good start because the modeler gets to know the type of quantifications there are in a model before thinking of how they influence each other.

In step 2 (Identify all relevant flows connecting to each stock), every flow has a time coefficient and associated operations that change its population but this is something we do not see in ORM. Therefore, equating an object type to a flow seems unrealistic.
It should instead be equated to an operation that controls a flow but this is not modeled in ORM. We therefore suggest an extension of ORM modeling with a model of dynamic aspects (e.g. similar to methods and state chart diagrams in UML) as an intermediary step. Due to lack of a relation to the flow mechanism that depreciates or increases a stock, we cannot say that a purely static method can capture all data of a dynamic method even when there has been an extension of ORM with dynamic properties ORM \[74, 77, 5, 6\]. However, what the researchers have done is relate object types to flows because of their quantities, their capability to connect different roles plus through them different object types can be connected. Most times steps 1 and 2 are carried out concurrently or in parallel. This is because when a stock is identified, the modeler may find it appropriate to also identify some of the flows relating with that stock.

In step 3 (Identify all possible converters); it is natural that after steps 1 and 2, converters should be identified although in principle we can make a full SD model with only stocks and flows, but this would mean that all the difficult calculations are placed in the flow equations. We do not want to do that here because we want to make the intermediate steps clear.

Something else that should be referred to here are exogenous variables. An exogenous variable comes from outside the model, has an impact on the model but is not impacted by anything inside the model. It is independent and determined by an outside source that has influence to a system or model e.g. the interest rate is determined by the government but has a great influence on the bank services. Exogenous variables are added to an SD model to calculate certain ‘things’ in a model. They can either be constants or time series. However, SD people prefer to have exogenous variables as constants for example, if the interest rate is two percent it should remain that way throughout the model run. But if the modeler wants to make a model that is more exact e.g. applied problems in real organizations (s)he can get the time series of the development in the last 10 years and just paste it in that variable and get something that works. But normally SD people do not like this approach or way of working, they prefer that the system generates the real fluctuations of the feedback. If an auxiliary variable that does exactly ‘this’ is added, then it will result in a lot of random ‘things’ and one may not be able to see the effect of the feedback in principle auxiliaries could be time lines.

Step 4 (Identify all possible connectors or information links) is by far the most difficult step of all because here we define how these decisions are made in the flows. Very often an information link is added from the stock to the related flows because only something which is in there can actually flow out and only something that is in the stock before or flow can actually flow in. Once we have the stock and flow structure, much of the information links and converters come naturally because somehow the flows have to be controlled and this control is done by time coefficient and by the value of the stocks related to the flows. While referring to the different types of connectors (action connectors with a dashed line and information connectors with a solid line), the practitioners said that there is an exception that is allowed in Vensim which is that at time zero, a startup value is needed for a stock. This value can be got by connecting all those things in the model onto a stock but when this is done, the arrow going into the stock becomes a dashed arrow too.
The sectors in SD are an informal notation inform of making things easier to understand but there is really no need to have these sectors. Therefore they are not required in the SD model as such.

In conclusion, there are light spots in both methods and the given examples can partly be translated but we are coming to border that there is something extra in ORM and in SD therefore they are not completely overlapping. The purpose of these steps therefore is to define the visual display of the model. If we were to look at the input parameters into the model then we may need more steps because the five steps lead to a picture and we need the details (input parameter guide) for example, what does variable paper depend on to be rejected or accepted? And that is more complex as it is but may be that is not information captured in this form but information captured from the domain expert. Secondly, in ORM we seem not to have flows explicitly and some of the given roles are mutually excluded thus in a stock and flow diagram, we would have a sequence and that is probably the dynamic aspect (time aspect).

Finally, about the naming of variables; names for some variable say; ‘accepted paper’ would be better referred to as accepting papers; reviewing papers instead of reviewed papers this way of naming variables makes it clear that it is something that happens all the time. Note that accepting papers could also be a name of a stock. Secondly, the converter names (paper writing, paper submission, paper reviewing,....etc) used in the model sound more like flows e.g. ‘paper writing’ may be a flow into ‘written papers’ and ‘paper submission’ may be a flow for ‘submitted papers’ etc. therefore it is not convincing to the SD modeler when represented as converters. It is very important we think of stocks in terms of nouns and flows as verbs.

**System dynamics questionnaire feedback**

At the end of each session, practitioners were given a questionnaire with five questions. In table 7.10, we present a summary of the results obtained from those questionnaires.

<table>
<thead>
<tr>
<th>Questions</th>
<th>Ex</th>
<th>Acc</th>
<th>NI</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. How relevant are the ORM to SFD transformation steps?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Do the given steps provide enough information on how to derive an SFD from an ORM model?</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Overall how would you rank the ORM to SFD transformation steps?</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Are the steps given easy to understand or interpret?</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>If NO what do you think should be added to improve the given steps?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Do you think that object role modeling is a good starting point for the construction of SFDs?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Key: Ex: Excellent; ACC: Acceptable; NI: Needs Improvement

For question 1, out of the three practitioners, one said that the ORM to SFD transformation steps were relevant to the development of the SD model but to solve problems for clients they would depend on the context. Another one practitioner said they were acceptable and one said there was need for improvement because identifying stocks from roles can be understood but there is need for a better explanation.
For question 2, two practitioners accepted that the given steps provide enough information on how to derive an SFD from an ORM model. One practitioner said there was need for improvement. In his reaction to the question he said that in principle he agrees but it must be made clearer which elements in ORM are translated into which SD elements.

For question 3, all practitioners agreed that ORM to SFD transformation steps needed to be further improved. They advised the researchers to be more explicit on what can be transformed and what cannot. For question 4, two practitioners agreed that the steps were easy to understand and interpret but one practitioner disagreed. He said the steps were not ‘yet’ easy to understand and interpret because they depend a lot on domain knowledge. For question 5, all practitioners agreed that ORM is a good starting point for the construction of stock and flow diagrams. They said that it offers some aspects which one needs to consider when constructing an SFD, facilitates communication with IT people but for simulation of SD model (full formalization) more information is needed.

Revised ORM to SFD transformation steps

[Step 1:] Identify all possible stocks and define their input values: As practitioners said there are two ORM elements (a role and an object type) that relate to a stock. To make this explicit we have come up with control parameters to show all possible choices of option for each ORM to SFD step. The control parameters include; RolesObjectStock (ROS) and UnaryStock (US) to be applied in steps 1; SectorObject (SO) and NoSectorObject (NSO) to be applied in step 5. In steps 2, 3 and 4 there are no control parameters need. In summary figure 7.13 represents all control parameters plus identified choices of options for ORM to SFD steps.

These control parameters are explained as follows;

- *RoleObjectStock (ROS):* In this control parameter, all roles and object types are mapped to stocks. Therefore, the total number of stocks in ROS is equal to the total number of roles plus the total number of object types in a given ORM model.
7.2. ANALYTICAL EVALUATION

Therefore: \( \text{Pop}(ROS) = (\text{Pop}(R_i) \cup \text{Pop}(O_k)) \). Where; \( \text{Role} = R_i =: i = 1...n \) and \( \text{Objecttype} = O_k =: k = 1...n \);

- \textit{UnaryStock (US)}: In this control parameter, only unary fact types are mapped to stocks. This is because objects held by a unary fact type relate to one object type. In order for one to derive a stock from an object type, a unary fact type should be connected to it. This unary fact type acts as a container that holds all objects within that object type but in a different state. The total number of stocks in \textit{UnaryStock} is therefore equal to the total number of unary fact types in a given ORM model. That is: \( \text{Pop}(US) = (\text{Pop}(U_f)) \). Where; \( \text{Unaryfacttype} = U_f =: f = 1...n \);

- \textit{SectorObject (SO)}: This control parameter means that an object type plus roles connected to that object type are mapped to a sector. Therefore the total number of sectors is equal to the total number of object types in a given ORM model. That is: \( \text{Pop}(SX) = \text{pop}(O_k) \). Where; \( \text{Stock} = S \) and \( \text{Sector} = S_x =: x = 1...n \);

- \textit{NoSectorObject (NSO)}: This control parameter means that an object type plus roles connected to that object type are not mapped to a sector. Therefore, there are no sectors introduced in the final SD model.

[Step 2:] Identify all relevant flows connecting to each stock and define their input values:

Since participants neither agreed nor rejected the relation of flows to object types, we decided to keep the relation as originally defined but instead use the control parameters given in step 1 and the two choices of option defined below to guide on when to use object types and unary fact types as stocks or flows\(^2\). We identified two options (\textit{progressive refinement} and \textit{pure mapping}) for deriving an SFD model from an ORM model.

As already stated, for \textit{progressive refinement}, a first draft model is created and progressively elements of that model are replaced by others. As depicted in figure 7.13, step 1 of \textit{progressive refinement}, both an object type and a role are mapped to a stock. Here each role is independently represented as a stock because a role played by an object type contains similar properties and it relates to one object type. We also map object types to stocks because they hold objects. Qn: How can we map one ORM element to two SD elements? This issue is cleared if one follows the steps and uses the provided control parameters.

For \textit{pure mapping}, elements are generated only when one is sure that they are the right ones. In figure 7.13 step 1 of \textit{pure mapping}, only unary fact types are mapped to SD stocks. This is because unary fact types do correspond to properties of the entity type that allow defining of sub-types and they contain properties specific to an object type although these properties are in a different state. Due to change(s) in the properties of unary fact types, in \textit{pure mapping}, the objects in an object type do not make a stock but it is the objects in a unary fact type that make a stock. This is because the objects in a unary fact type are in an independent state to which some of the objects in an object type take part and while they are in this state, they relate with only one object type. This means that unary fact types from one object type have similar objects but in different state. Here we

\(^2\)These two options were first introduced in chapter 5
note that step 2 can be done concurrently with step one i.e. the derived model(s) may be sequentially presented or in parallel.

[Step 3:] Identify all possible converters and define their input values: As we already stated there are two types of converters (auxiliary and constant). In this step however we also include exogenous variables as advised by different practitioners. For more explanation see subsection 7.2.3 step three, third paragraph. While conducting this step, it is important that the modeler considers the ORM derivation rules and interacts with the domain expert to guide him as he identifies or defines the input parameters.

[Step 4:] Identify all possible connectors (information links) and connect/link them to required flows, converters and stocks: As practitioners said this step defines how decisions in the model are made. It is therefore very difficult to have guidelines on how to identify information links but it is important that the problem is well stated, proper derivation rules are given and a domain expert is present for any further clarifications. Note that, for every connection made, input values for the variable at the arrow tip change. Therefore they should be redefined.

[Step 5:] Create sectors: Here we propose that this step is made optional as practitioners said, it does not affect the model in anyway.

In conclusion, since practitioners found all the steps simple and relevant except for step five, we have not made a lot of changes but instead added a few pointers to guide the modeler. Secondly, as one practitioner put it, these steps can only lead to a picture of a stock and flow model. In our further works therefore, we are going to add more steps that cover how to define input parameters in a stock and flow model, what constraints or measure(s) should be taken and also show how ORM decomposition helps in defining the stock and flow input parameters.

Limitations

It would be elegant for us to further evaluate the Revised Ground System Dynamics steps however, we did not do so. This was due to a number of reasons:

- Since we had received feedback/views from experts in ORM and SD on changes to incorporate in the GSD transformation steps, it would make no sense for us to reevaluate the new steps without addressing their concerns. We therefore decided to first of all take care of the stated suggestions before reevaluating the GSD method. For example, one of the comments we received from SD practitioners was to show how this new method (GSD) improves conceptualization of the SFD input parameters. “The purpose of these steps therefore is to define the visual display of the model. If we were to look at the input parameters into the model then we may need more steps because the five steps lead to a picture and we need the details (i.e. input parameter guide)”. To address this concern/observation, we came up with a decomposition mechanism presented in chapter 8. In doing so we introduce a method to extend ORM with an integrated mechanism to describe object type behavior. This mechanism allows for decomposition in the description of object type behavior.
7.3. DISCUSSION AND REFLECTIONS ON GSD PROCEDURE

- Secondly, during the evaluations, participants were sent their transcribed data for more scrutiny. Thus allowing them to alter, add and delete whatever they thought was irrelevant or relevant. As represented in figure 7.8 section 7.2.2, we did not receive much response. That is to say, some of the participants did not make any changes or worse still acknowledge receipt of the transcribed data/feedback. This made us conclude that the transcriptions sent to them exhaustively covered what they had to say or they were too busy to respond to our request. To obtain better results, we were of the view that we use the same participants (experts). This is because they are already familiar with our research study and therefore it would be easy for them to see mistakes, improvements and thus give conclusive evaluations or make suggestions for further improvement.

- Further still, to systematically reevaluate the refined GSD steps, would require ample time (i.e. identify participants, send them requests, if response is positive schedule appointments which in most cases are weeks or months apart etc.) and funds. Yet the researcher’s study time had elapsed and the remaining funds were insufficient. Therefore we could not continue with the reevaluation of the refined GSD transformation steps. As one of the future works in chapter 10 section 10.3, we suggest that a field evaluation before implementing this method be done. This kind of evaluation (field) will enable the researcher gauge the applicability, ease of use of the method and have a clear ‘picture’ of what to do next.

7.3 Discussion and reflections on GSD procedure

In subsection 7.2.3 and 7.2.3 respectively, we have presented practitioners’ views on GSD transformation steps. In their feedback they made a few suggestions that we have used to improve the GSD transformation steps. Minus what we have presented in the previous sections, we now present general reflections and discussions on the GSD method inline with the two methods entailed (ORM and SD). For each paragraph, we use a common terminology or phrase to highlight the content in that paragraph.

Use of a domain expert: Both ORM and SD practitioners agreed that having a domain expert during the application of the GSD steps is important because the domain expert helps to clear the ambiguities with the CLD variables (ORM practitioners) and in defining stock and flow input parameters (SD practitioners). However, ORM practitioners further said that, in the absence of a domain expert, the researcher should provide a catalogue of terms explaining each variable name and concrete examples of data. This would help the modeler understand the underlying meaning of the stated CLD variables. For example in step 1, it is difficult to state what an object is for variables like ‘birth’, ‘dilation’ etc. such variables may require a modeler to have more abstract steps.

Populating the ORM model: Putting the system dynamics way of working in perspective requires a mechanism on how to populate ORM object types. This is because there are instances in an object type that may not be playing a role at a point in time. However, ORM being a static method this cannot be achieved. Therefore capturing all information in a CLD into an ORM model may be difficult. On the other hand system dy-
Representation of Soft variables: The issue of how to represent soft variables is not only in ORM but also exists in SD. If a soft variable is key in the model then the modeler has to find a way of representing it. In SD, what is normally done is operationalize the variable by giving it a unit of measure since there is no systematic way of doing it. But that means model reliability is compromised when it comes to simulation of results. Combining soft variables with hard variables in ORM is equally a significant challenge. This therefore raises doubts that all cases (with soft and hard variables) can be successfully combined. Secondly, contiguous entities like masses, amounts of matter etc cannot be represented in an ORM (ER, UML) model unless a specific technical treatment is given to it [66]. This is because ORM is used for structural domain modeling i.e. ORM model types as collections of individuals which can be individuated, are countable and have a definite identity [67]. System Dynamics on the other hand is typically used to model, dynamics of the domain and not necessarily involving the flow of countable discrete entities (that is why converters contain information inform of equations or values that can be applied to stocks, flows, and other converters in the model).

Suggested routes of option: During the structured walkthrough sessions, three of the ORM practitioners suggested a different route of option. This route is represented with letter X in figure 7.14. In the same figure we present option S as the traditional system dynamics route were a domain expert views the real world, interprets it in his mental model, then transforms it into a CLD and finally into a stock and flow diagram. However, route S is not obligatory in system dynamics. Depending on the use of the model, a modeler may opt to have CLDs only, skip the CLD to SFDs or have both under one model. In this study, we have used option G were the researchers came up with examples and a case study to help practitioners apply the GSD procedural steps and to have a clear understanding of how concepts in the two methods relate.

From the given case (option G) a CLD model is derived first and then transformed into an ORM model and finally into an SFD model. During the structured walkthrough sessions, some of the participants said that, although the opted route (option G) represents a good way of underpinning SD with ORM, route X would be optimal and less time consuming. This is because it is much easier to have all facts in a given case stated in an ORM model and variable names made explicit before embarking on the CLD model. Since most of the ambiguities are cleared during the construction of an ORM model, it would be reasonable to move directly to a stock and flow model or to work in parallel. Note that opting for route G the modeler may have to rename some CLD variables as he derives the ORM model hence duplication of tasks (activities). Furthermore, practitioners found route X most convenient, they urge that it would be more appropriate to have an SFD on completing the ORM model instead of the CLD. This is because by using ORM’s derivation rules the modeler would easily identify the SFD input parameters. Finally, since in SD it is important to think of stocks in terms of nouns and flows as verbs [27],

[3]Soft (fuzzy) variables are intangible and can be quantified but not precisely measured. For soft variables numerical metrics and data are not available e.g. expectations, perceptions, satisfaction etc [193].

[4] Hard variables can be precisely measured and, their quantitative metrics and numerical data are available [193].
ORM’s capability to automatically verbalize its diagrams into pseudo natural language sentences would be helpful [100].

7.4 Conclusions

Evaluation is a systematic assessment of the study in order to obtain useful feedback about the study [205]. In this study, we evaluated the GSD method in two phases. First, we evaluated the CLD to ORM transformation steps using experimental evaluation. Thereafter we evaluated the ORM to SFD transformation steps using analytical evaluation methods [86]. The two evaluation methods chosen complement each other because they overlap and crosscheck assessments of the perceptions of participants involved. To conduct GSD evaluations different factors were put into consideration;

1. Participants to take part in the evaluations and what their roles were: This was one of the key points in our evaluations because the subject (methods) we were discussing (combining) required a level of expertise in the methods (ORM ad SD). It was therefore important that we inform our participants why they were chosen and what we expect of them. For the experimental evaluations we used modelers who were using system dynamics and those using ORM in a focus group setting (not necessarily experts) and for analytical evaluation we used only experts and practitioners from both fields. This enabled us to establish the degree to which perceptions were shared, evaluate the relations in both methods and evaluate the GSD methodology.

2. What techniques to use during the evaluations: In our case we used a number
of techniques. In the experimental and analytical evaluations, we used qualitative techniques and for the simulation and update behavior we used a quantitative technique. In the qualitative technique, we aimed at gathering an in-depth understanding of the problem domain and reasons governing participant's choice of action thus investigating the why and how of participant's decision(s). In the quantitative technique, we aimed at developing and employing mathematical formulas, theories and hypotheses pertaining to the problem domain [136, 201]. The process of quantifying the stock and flow diagram is central to quantitative research because it provides the fundamental connection between empirical observation and mathematical expression of quantitative relationships.

3. Communication is another key factor: Communication occurs at different levels and involves a number of factors e.g. how to initiate or make the first contacts with the participants, how to manage disappointments (reject to take part in the experiment), how to handle feedback from participants, how to make sense of the feedback, how to manage participants during evaluation sessions, how to communicate with participants and how to dialogue with them during and after the experiments etc. all these and more are discussed in [92, 93, 91, 94]. In our case communication was very important because participants were from different background e.g. practitioners, academics etc and their way of working was different. It was therefore important that we were explicit in our communications. Secondly, it was important that our research problem was stated early enough and how long our experiments were to take to enable us receive concrete feedback.

4. Number of participants to take part in each evaluation: Knowing the number of participants is very important because it enables the researcher make decisions on who to contact (take part in the experiment), make clear plans on how to manage the evaluations, approximate the time (s)he may require to complete each evaluation, make preparations for analysis of the feedback etc. Hence better management of the entire evaluation process. In our case, since we used different evaluation methods, choosing participants depended on their familiarity with the methods and expertise. For experimental evaluations we had eight participants and for analytical evaluations we also had eight participants.

5. Location were to conduct the evaluations and why: In our case we conducted our evaluations in two different countries (Uganda and The Netherlands). The case study and experimental evaluations were conducted from Uganda because it was easier for the researcher(s) to work in an environment where they felt confirmable (understands the local language and can easily communicate with other people especially with the case study) and analytical evaluations were conducted from the Netherlands because most of the experts were in Europe and internet facilities are good for Skype meetings.

6. How to manage feedback: A lot has been discussed about feedback and analysis of findings from groups of participants [216, 214, 114, 16]. In our case we applied some of these approaches described in literature to manage our findings. Unlike most scholars, our approach involved different techniques because we used different evaluation methods for each phase. As already stated for experimental evalu-
7.4. CONCLUSIONS

In our approach, we used the focus group approach [16, 114, 194]. This approach has some similarities with the Group Model building approach exhibited by Vennix in [216]. In his approach, he involves a number of stakeholders (modelers in our case) in the modeling process showing how he arrives at the final model. For analytical evaluations we used structured walkthroughs because we wanted to get practitioner’s and expert’s opinion on the GSD steps in a one to one discussion. Analyzing data from these evaluations was tedious because audio recorders were used. After each session, we had to listen to the recordings, a number of times before reporting our findings. These recordings however, enabled us tap into different negotiations and participant’s views on different constructs.

Often practitioners from system dynamics have compared their approaches and discussed their methodological assumptions [49, 118, 115, 97], therefore this way of working is not new. In this chapter, we have presented feedback from both SD and ORM practitioners on the GSD transformation steps. The views provided were either insights or suggestions for improvement. One important result of this study however, is to obtain a formal model that can generate empirical behavior. This formal model should allow modelers understand and explain how the behavior is generated from the supposed structure. Therefore the role of this study is not only to underpin SD with ORM but also to a certain extent, explain and understand how this behavior is generated. This directly involves making connection with the way in which ORM represents real systems. In our next chapter therefore, we come up with a mechanism to improve SD model conceptualization through understanding object update behavior. This enables us achieve a complete study and analysis on underpinning SD with a domain modeling method ORM.
Chapter 8

Update behaviors and simulations

This chapter builds on results from chapter 4 and 7. During the evaluation of the GSD method, one of the practitioners said that “If we are to look at the input parameters into the SFD model then we may need more steps because the ORM to SFD provided five steps lead to a picture”. Basing on this remark, we came up with chapter 8. In this chapter, we use examples and the ORM model derived from a case “Intrapartum process in Ugandan Hospitals” already presented in chapter 7 to study the behaviors, and derive more steps for the Grounded System Dynamics (GSD) method by defining stock and flow diagram input parameters, and analyze simulation outputs. One of the goals of this study is to improve System Dynamics (SD) model conceptualization. Understanding the simulations and update behaviors of a system dynamics model at a conceptual level provides a critical means for achieving this goal. To understand what update behavior is all about, one needs to define the two terms involved separately and thereafter merge them to give us a clear definition of update behavior. Update is the act of changing something to bring it up to date and behavior is a value that changes over time. Behavior is reusable when applied to an object i.e. it causes it to respond to user input in meaningful patterns or to operate independently. Update behavior therefore is the act of changing an object with focus on the observable relationship between or among objects with respect to the environment. The main results of this chapter are:

1. A complete theoretical founding of the method of this thesis

2. Allowing quantitative analysis at the level of ORM reasoning

3. The formal transformation rules from ORM into SD
8.1 Object behavior

First, we describe the behavior of single objects. To clarify our descriptions, we use examples showing how states are derived from properties of an object. We also extend ORM to show how various levels of abstraction can be seen using the decomposition mechanism. To demonstrate applicability, we use the “Intrapartum process in Ugandan Hospitals” case to relate our views with an actual scenario. On completion, we also discuss the influencing transitions and simulation outcome.

In this thesis we propose an extension of ORM with the behavioral description of object types, that allows decomposition to master schema complexity. With respect to decomposition, the data modeling technique PSM ([89]) introduces the schema type as a mechanism for decomposition. Besides PSM introduces the grammar type for semi-structured data, allowing the PSM schema to be extended with a grammatical description (which can be compared to a DTD in XML). However, PSM does not cater for the behavioral description of objects. In [26] abstraction layers for data modeling are introduced at a more fundamental level. Several additional methods have been proposed to combine data modeling with behavioral descriptions, such as state charts (see for example [79]). UML (see for example [46]) offers modeling techniques for many aspects of software development, such as a data model and behavior description. PSM\(^2\) ([55]) is an action-based approach to model an application domain, starting from a sample behavioral description (called a logbook, see [210]). PSM\(^2\) does not have a mechanism for decomposition in the behavioral description of object types.

In this thesis we introduce a method to extend ORM with an integrated mechanism to describe object type behavior. This mechanism allows for decomposition in the description of object type behavior. Note that ORM, seen as an extension of PSM, also offers decomposition for grouping object types into meaningful clusters, and the extension with semi-structured data.

8.1.1 Conceptual description for objects

As explained in section 4.1, we use ORM to provide a conceptual description of object types and their relations. In [55] the modeling technique PSM\(^2\) is introduced as an extension of PSM (nowadays more generically referred to as ORM). PSM\(^2\) is based on a description of the application domain in terms of action types rather that fact types as in ORM. By concentrating on the roles attached to each object type, the state transitions for that object type are described in PSM\(^2\). By adding an ordering on those roles, the so-called life model for that object type is obtained.

8.1.2 States and transitions

In this thesis the approach is based on conventional fact-based ORM but extended to cover some of the aspects of PSM\(^2\). We assume that each state of an object is derivable from its (modeled) properties. As a consequence, each state can be described by some information
8.1. OBJECT BEHAVIOR

descriptor (see section 4.1). A base for an object type describes a set of possible states for that object type. Note that an information descriptor \([87, 90]\) can be seen as a path through a conceptual schema, describing a relation between its starting and its ending object type.

In this thesis we restrict ourselves to homogeneous information descriptors (see \([90]\)) meaning that the descriptor has both a unique starting point and a unique ending object type. This is explained as follows. Let \(\text{Start}(D)\) be the set of starting points of information descriptor \(D\) and \(\text{End}(D)\) its set of ending points. Furthermore, we use \(\text{Top}(X)\) to denote the pater familias of the object type hierarchy to which object type \(X\) belongs (see \([87]\)). We call an information descriptor \(D\) homogeneous if (1) all object types in \(\text{Start}(D)\) have the same pater familias (referred to as \(\text{Top}(\text{Start}(D))\)), and (2) all object types in \(\text{End}(D)\) also have the same pater familias (referred to as \(\text{Top}(\text{End}(D))\)). Then the evaluation of \(D\) at point of time \(t\) leads to the pairs of object instances \((x, y)\) such that \(x \in \text{Pop}_t(\text{Top}(\text{Start}(D)))\) and \(y \in \text{Pop}_t(\text{Top}(\text{End}(D)))\) that are related via the \(D\) (see \([157]\)).

In section 4.3.2 we introduced the notion of conceptual base. A conceptual base is a set of information descriptors, where each information descriptor of a conceptual base describes a typical state of its starting object type. We will call \(\mathcal{D}\) a base for object type \(X\) if all descriptors in \(\mathcal{D}\) start from object type \(X\):

\[
\forall D \in \mathcal{D} \ [\text{Top}(\text{Start}(D)) = X]
\]

Base \(\mathcal{D}\) is called complete for object type \(X\) if at each point of time \(t\):

\[
\bigcup_{D \in \mathcal{D}} \text{Pop}_t(D) = \text{Pop}_t(X)
\]

In that case we will call the descriptors in \(\mathcal{D}\) elementary states for object type \(X\). So for a complete base \(\mathcal{D}\) at each moment each instance of \(X\) is in precisely one of the elementary states in \(\mathcal{D}\). Besides elementary states, we also have compound states. Basically, a compound state is a grouping of (elementary or compound) states. This leads to a hierarchy of states. Let \(\mathcal{S}\) be the set of all (elementary and compound) states. The nature of \(\mathcal{S}\) is described by the following rules. Since \(\mathcal{D}\) is complete, we have:

**S-1** \(\mathcal{D} \subseteq \mathcal{S}\).

The hierarchy of states is represented by the relation \(\text{SubState}\), where \(\text{SubState}(x, y)\) is representing the fact that \(x\) is a substate of \(y\). The hierarchical structure is enforced by the following rules:

**S-2** \(\neg \text{SubState}(x, x)\)

**S-3** \(\text{SubState}(x, y) \land \text{SubState}(y, z) \Rightarrow \text{SubState}(x, z)\)

States without decomposition are elementary. We will use \(\text{Elem}(x)\) to denote state \(x\) being elementary, abbreviating \(\neg \exists y [\text{SubState}(y, x)]\). Based on this abbreviation we have:

**S-4** \(\text{Elem}(x) \iff x \in \mathcal{D}\)

To express that \(x\) is a direct substate of \(y\) we use \(\text{SubState}_1(x, y)\) which is an abbreviation for \(\text{SubState}(x, y) \land \neg \exists z [\text{SubState}(x, z) \land \text{SubState}(z, y)]\). States can be in at most one direct decomposition:
S-5 SubState\(_1\)(x, y) \land SubState\(_1\)(x, z) \Rightarrow y = z

Using the states \(S\), we further extend the ORM scheme by adding state transitions as a binary relation \(--\rightarrow\) on the states \(S\). The expression \(S_1 \rightarrow S_2\) is interpreted as: an object in state \(S_1\) may move to state \(S_2\). Furthermore we assume a set \(D_{in}\) of initial (elementary) states of object type \(X\), and a set \(D_{out}\) of final (elementary) states. The following rules describe the transition relation and the distributive rules for substate and state transition. Compound states have (at least) one initial and (at least) one final state:

\[\begin{align*}
T-1 & & \neg \text{Elem}(x) \Rightarrow \exists y [\text{SubState}\(_1\)(y, x) \land y \in S_{in}] \\
T-2 & & \neg \text{Elem}(x) \Rightarrow \exists y [\text{SubState}\(_1\)(y, x) \land y \in S_{out}] \\
\end{align*}\]

Each object type will have an initial state. The instances of an object type may be persistent, in which case the object type may not have a final state. Within a compound state, an initial state has no predecessor, and a final state has no successor:

\[\begin{align*}
T-3 & & \text{SubState}\(_1\)(x, y) \land z \rightarrow x \land \neg \text{SubState}\(_1\)(z, y) \Rightarrow x \in S_{in} \\
T-4 & & \text{SubState}\(_1\)(x, y) \land x \rightarrow z \land \neg \text{SubState}\(_1\)(z, y) \Rightarrow x \in S_{out} \\
\end{align*}\]

If a compound state has a successor then this is effectuated by any of its final states. The same holds for the case of a predecessor. This is formalized by the following rules:

\[\begin{align*}
T-5 & & u \rightarrow v \land \neg \text{Elem}(u) \land \neg \text{Elem}(v) \Rightarrow \exists x, y [\text{SubState}\(_1\)(x, u) \land \text{SubState}\(_1\)(y, v) \land x \rightarrow y] \\
T-6 & & u \rightarrow v \land \neg \text{Elem}(u) \land \text{Elem}(v) \Rightarrow \exists x [\text{SubState}\(_1\)(x, u) \land x \rightarrow v] \\
T-7 & & u \rightarrow v \land \text{Elem}(u) \land \neg \text{Elem}(v) \Rightarrow \exists y [\text{SubState}\(_1\)(y, v) \land u \rightarrow y] \\
\end{align*}\]

Finally, transitions are inherited upward in the hierarchy:

\[\begin{align*}
T-8 & & \text{SubState}\(_1\)(x, u) \land \text{SubState}\(_1\)(y, v) \land u \neq v \land x \rightarrow y \Rightarrow u \rightarrow v \\
\end{align*}\]

We call a (compound) state \(x\) a start state of compound state \(y\) if \(x\) is an initial state and also a direct substate of \(y\):

\[
\text{StartState}(x, y) \triangleq x \in D_{in} \land \text{SubState}\(_1\)(x, y)
\]

We call \(x\) a birth state if it is an initial state but not the start state of another state:

\[
\text{BirthState}(x) \triangleq x \in D_{in} \land \neg \exists y [\text{SubState}\(_1\)(x, y)]
\]

If \(x\) is a birth state, then we also write:

\[
\omega \rightarrow x
\]

where \(\omega\) is virtual (source) state. If \(x\) is a death state, then we also write:

\[
x \rightarrow \Omega
\]

where \(\Omega\) also is a virtual (sink) state. Analogously we introduce final state and death state:

\[
\text{FinalState}(x, y) \triangleq x \in D_{out} \land \text{SubState}\(_1\)(x, y)
\]

\[
\text{DeathState}(x) \triangleq x \in D_{out} \land \neg \exists y [\text{SubState}\(_1\)(x, y)]
\]

We call the structure \(\mathcal{B}(X) = \langle S, \text{SubState}\(_1\), D_{in}, \rightarrow, D_{out}\rangle\) a behavioral description of object type \(X\). Note that an object type may have more than one behavioral description, but this will be generally not the case in practice.
Example 8.1.1

Consider the scheme in Figure 8.1. It shows object type A having a single unary fact type consisting of role p, i.e. objects (population instances in an object type) in A play role p. In this case we thus have:

- $D = \{ \text{D} \}$ with $D = \text{ObjNm}(A) \text{RoleNm}(p)$ where $D$ is the concatenation of the name $\text{ObjNm}(A)$ of object type A, and the name $\text{RoleNm}(p)$ of role p
- $S = \emptyset$
- $\text{SubState}_1 = \emptyset$
- $\rightarrow = \emptyset$
- $D_{\text{in}} = \{ D \}$
- $D_{\text{out}} = \emptyset$

In simple terms, let object type (A) be person and role (P) be born in EU. In this example, we have a single state change. This is because it requires a single decision and occurs once an event is triggered and never again. E.g. A person can only be born once, therefore a single decision is required. In ORM this would be represented as: Object type Person plays role born-in-EU. If the person is not born in EU then he does not play that role. In system dynamics, we would have a converter feeding new information into inflow (EU Birth) see figure 8.1. This inflow is activated every time a person plays that role born-in-EU. The quantities in inflow EU births are stored in stock born in EU (see figure 8.2). Note that stock born in EU does not have an outflow. This is because the quantities in stock born in EU do not change. The resulting extended ORM scheme is displayed in Figure 8.3.

Example 8.1.2

Consider again the scheme in Figure 8.1. Now we assume that instances of A do not necessarily play role p, but when they do, they will always keep playing that role. Then we have two states for instances of object type A:

1. $D_1 = \text{ObjNm}(A) \text{BUT NOT RoleNm}(p)$
2. \( D_2 = \text{ObjNm}(A) \text{RoleNm}(p) \)

So we have:

- \( D = \{D_1, D_2\} \)
- \( S = D \)
- \( \text{SubState}_1 = \emptyset \)
- \( \rightarrow = \{D_1 \rightarrow D_2\} \)
- \( D_{\text{in}} = \{D_1\} \)
- \( D_{\text{out}} = \emptyset \)

In this example we represent a single state change with multiple decisions. In this case there is an occurrence of the event that is likely to appear again in the system. For example; person has visited EU. Once a person starts to play the role has-visited-EU he never stops. This person may decide to play this role again and when he does play this role again (after a period of time), multiple decisions are made. This new visit leads to updates in existing information hence, the latest visits and frequency of the visits are captured. In the single state change with multiple decision, the ORM model will not differ from the one in figure 8.1 although information has changed; yet for the SD model there is a significant difference because of the multiple decisions made as reflected by the converters and connectors (information links) in figure 8.4. The converter (new visit and old visit) represent the new information that is to be fed into inflow (visiting rate) that causes a change to stock has visited EU.
8.1. OBJECT BEHAVIOR

Figure 8.5: Extended ORM scheme

- \( \mathcal{D} = \{D_1, D_2\} \)
- \( \mathcal{S} = \{D_1, D_2, S\} \)
- \( \text{SubState}_1 = \{(D_1, S), (D_2, S)\} \)
- \( \rightarrow = \{(D_1, D_2)\} \)
- \( \mathcal{D}_{\text{in}} = \{D_1\} \)
- \( \mathcal{D}_{\text{out}} = \emptyset \)

In Figure 8.6 we also see a shorthand notation for the state that is decomposed into a linear sequence of substates. In SD terminology, such a state is referred to as a conveyor state.

Example 8.1.3

As a more complex example, we take an example from [55]. In this example the life of a pop band is described. After setting up the band \((D_1)\) the band gets a name \((D_2)\). Then the band repeatedly performs actions like writing a song \((D_3)\) followed by recording that song \((D_4)\), or producing a recording \((D_5)\). Then the band may stop \((D_6)\). The resulting extended ORM scheme is shown in Figure 8.7, focusing at the life model only. We see two compound states, \(S_1\) that comprises the band foundation process, and \(S_2\) covering the active life of the band. Formally, we have:

- \( \mathcal{D} = \{D_1, \ldots, D_6\} \)
- \( \mathcal{S} = \{D_1, \ldots, D_2, S_1, S_2\} \)
- \( \text{SubState}_1 = \{(D_1, S_1), (D_2, S_1), (D_3, S_2), (D_4, S_2), (D_5, S_2)\} \)
- \( \rightarrow = \{(D_1, D_2), (D_2, D_3), (D_2, D_5), (S_1, S_2), (D_3, D_4), (D_4, D_3), (D_4, D_5), (D_5, D_3), (D_5, D_5), (S_2, S_2), (D_4, D_6), (D_5, D_6), (S_2, D_6)\} \)
- \( \mathcal{D}_{\text{in}} = \{D_1, D_3, D_5\} \)
Using the decomposition mechanism, it is possible to see the life of a band at various levels of abstraction. For example, in Figure 8.8 we have a high level view on band life.

In the next subsection we will demonstrate these concepts in our case “Intrapartum process in Ugandan Hospitals”.

8.2 The “Intrapartum process in Ugandan hospitals” case

The ORM scheme for the “Intrapartum process in Ugandan Hospitals” case is displayed in figure 8.9. Note that ORM does not provide a mechanism for decomposition. As a consequence, there is no effective means to master scheme complexity such as this is possible using a decomposition mechanism. A consequence ORM schemes tend to be very large and overly detailed. In this example we show how stepwise refinement can be used as an effective mechanism to build and to understand conceptual schemes. Using the Intrapartum process in Ugandan Hospitals ORM scheme depicted in figure 8.9, we illustrate how the decomposition mechanism is used at various levels of abstraction. First, we break the ORM scheme into manageable sub-schemes.

To do so, we take object types with unique properties, these object types are Bed and Person, and are depicted in figures 8.10 and 8.16. Through these sub-schemes, we show how the decomposition mechanism works on a real-life example.
In the life of a person depicted in figure 8.10 we have three elementary states; *Baby*, *Patient* and *Medical person*. In state *patient*, when the patient arrives at the hospital, she queues up ($P_1$) then she is registered ($P_2$) by the nurse, after registration patient history is updated ($P_3$). Patient history has two states is initiated and record. Once patient history is initiated, it can only be updated every time a patient visits the hospital (see figure 8.11).
turn comes she is examined ($P_3$), depending on the findings, she is either admitted ($P_6$) or discharged ($P_9$). If she is admitted ($P_6$), she is monitored ($P_7$) every 30 minutes until she gives birth ($P_8$). After giving birth she is discharged ($P_9$). In figure 8.12, we represent the patient intrapartum process. At a higher level of abstraction the patient intrapartum starts with *Enter hospital*. Then the patient is examined, leading to either *Treatment* or (if no treatment is required) *Leave hospital*. After treatment, the patient leaves the hospital. In Figure 8.12 we can see this main structure, but the figure also displays the further details of the compound states.

Although the delivery of babies is a main activity of our hospital department, there is little administration about the babies. In fact, only their birth is recorded in this department. Babies are represented by the state *Baby*, and consequently there is only one base state is born ($C_1$). This, and the baby life is depicted in figure 8.13.

Next we focus on the *Medical persons* of this department, consisting of *nurses* and *obstetricians*. This is represented by state *Medical person*, which has two base states *nurses* and *obstetricians*. The life model of a *nurse* has the following states; administrates ($A_1$), monitors ($A_2$), examines ($A_3$), admits ($A_4$), records history ($A_5$), establishes dilation stage ($A_6$), births baby ($A_7$) and discharges ($A_8$). These states are contained in four different complementary states Preparation, examination, treatment and discharge. Some of these states are similar with the obstetrician’s state. This is because in a hospital, if a case is very sensitive for example an operation, it is handled by an expert who in this case is the obstetrician.

The life model of an obstetrician has four states; examines ($A_3$), admits ($A_4$), birth...
8.2. THE “INTRAPARTUM PROCESS IN UGANDAN HOSPITALS” CASE

baby ($A_7$) and discharges ($A_8$). These states are contained in three different complementary states Examination, Treatment and Discharge. In figures 8.14 and figures 8.15 we present the life model of a nurse and an obstetrician respectively.

In figure 8.16 we present an ORM scheme for object type Bed. In the life of a bed we have two elementary states; Admission bed and Delivery bed.

In figure 8.17, we depict the life of states Admission bed and Delivery bed. In state Admission bed, a bed is either empty ($B_1$) or occupied ($B_2$) also in state Delivery bed, a bed is either empty ($B_3$) or occupied ($B_4$).

Before any admissions, the bed is empty ($B_1$), when a patient is admitted, she occupies an empty bed thus ($B_2$). The admission bed may be emptied on discharge or for a while when the patient is delivering. During delivery time the patient occupies the delivery bed ($B_4$) (in this case one patient occupies two beds the admission bed and delivery bed). The delivery bed is also initially empty but when a patient is ready for delivery she occupies this delivery bed for a period of time before and after she is taken back to her admission bed ($B_2$) thus the arrows between ($B_1$), ($B_2$), and ($B_3$), ($B_4$). Note that in this case study, the hospital(s) have limited resources e.g. beds, medical personnel etc. Therefore a patient does not have the luxury of not occupying a bed when allocated one. I.e. once a patient is allocated a bed, she occupies it else someone else will. Due to constrained resources, in figure 8.17, we do not distinguish between allocation of a patient to a bed and actual occupancy of a bed by a patient.

In figure 8.18 we represent all the decomposition levels contained in the ORM...
scheme shown in figure 8.9. As earlier stated, in figure 8.9 there are two object types (supertypes) with unique properties (person and bed). In these object types, are objects in different states that relate to each other causing changes. For example admitted patient influences states admission bed and delivery bed respectively. Understanding these state changes help the modeler define SD influencing transitions and input parameters. These influencing transitions and input parameters are discussed in subsections 8.3.2 and 8.4 respectively.

8.3 Controlling object flow

At this point we assume an ORM scheme extended with the behavioral description $B(X) = \langle S, \text{SubState}_1, \mathcal{D}_1, \cdots \cdots, \mathcal{D}_{\text{in}}, \mathcal{D}_{\text{out}} \rangle$ for each object type $X$. In this section we discuss how object flows can be described quantitatively at the conceptual ORM level, and how these object flows depend on each other.
8.3. CONTROLLING OBJECT FLOW

8.3.1 A continuous approximation

Sofar we have extended ORM with the concepts of state and state transition. However, extending ORM with state transitions is not new. In [5, 6] they explore the extension of ORM to support declarative specification of dynamic rules restricted to single-step transitions. The extension of ORM in this study allows us to analyze ORM model behavior at a conceptual level. We have showed how decomposition helps system analysts to think and analyze at the most convenient level of abstraction (granularity). The analysis is by reasoning, or qualitative. Population quantities can be involved in this reasoning. Population sizes are discrete numbers (even natural numbers).

For a deeper understanding of the dynamics of the conceptual system described by the ORM model we focus on groups of objects rather than individual object instances and their transitions. Typically, we focus on (compound) states and their size, and how these sizes vary over time.

Our intention is to analyze the dynamics by continuous simulation. In [1], Albrecht defines continuous simulation as a computer model of a physical system that continuously tracks response of a system over time according to a set of defined equations typically involving differential equations. More concretely, Lee [123] states that a computer simulation or computer model, attempts to create and analyze an abstract model or program that simulate the behaviors of real-world systems. Our intention is to introduce a computational model at the conceptual level of ORM.

To facilitate the analysis, and to find the differential equations involved, we apply a continuous approximation of the discrete world. We assume time to be continuous, thus taking values from the real numbers ($\mathbb{R}$). According to [112], the closed-world transformation is the most popular continuous-to-discrete transformations in digital system design and also used in digital simulation. Typically for a System Dynamics is to use a fixed sample interval. At this point we have two options:

1. Assuming that population sizes also take values from the continuous domain $\mathbb{R}$. Then differential equations can be used to describe the system behavior. Differential equations are a powerful mechanism to derive properties of a system. From a differential equation a differential scheme is easily derived.

2. Setting up a system of differential equations directly.

We feel that a system of differential equations more adequately can describe system behavior. In this subsection we discuss the basis and motivation for this approximation in a formal way. In the next subsection we introduce simulation as an effective realization for this formal approach. Basically we show how a System Dynamics interpretation can be done at conceptual level. In the next chapter we discuss how the conceptual analysis is transformed (compiled) into a System Dynamics realization.
8.3.2 Influencing transitions

As in system dynamics (see section 4.2.1), causal relations are introduced as $D_1 \succ^{op} D_2$ or $D_1 \prec D_2$ between (compound) states of objects, expressing a positive or negative effect of a change in the population of $D_1$ at time $t$ on the population of $D_2$ at time $t$. We assume there is a time delay between the cause and its effects; this time delay does not have a lower bound on its duration, and may be seen as infinitely small. When the causal link is effected at time $t$, then it relates the situation of the cause at time $t$ with the effect at time $t^+$ (using standard notation for calculus [200]). The causal relations are defined as follows:

$$D_1 \succ^{op} D_2 \triangleq \text{Sign} \left( \frac{d\text{Pop}(D_1)}{dt} \right)(t^-) = \text{op} \ \text{Sign} \left( \frac{d\text{Pop}(D_2)}{dt} \right)(t^+) \text{ at each moment } t$$

for $\text{op} \in \{-, +\}$. As an example, considering 8.18 we notice that an increase of incoming patients will lead to an increase in babies born. This is expressed as:

$$\text{Patient} \succ^{+} \text{Baby}$$

Another rule may be that a decline in the number of obstetricians is to be followed by an increase of the number of nurses:

$$\text{Obstetrician} \prec \text{Nurse}$$

When more beds are being used for delivery, then there are less admission beds, and vice versa:

$$\text{Admission Bed} \prec \text{Delivery Bed}$$

$$\text{Delivery Bed} \prec \text{Admission Bed}$$

Besides rules set up by the system analyst, the relations $\rightarrow$ and $\succ^{op}$ also are related in the following way:

1. if $D_1 \rightarrow D_2$ then also $D_1 \succ^{+} D_2$.

2. if both $D_1 \rightarrow D_2$ and $D_1 \rightarrow D_2$ then also $D_2 \prec D_3$ and $D_3 \succ D_2$.

Causal influences are special kinds of growth relations between states of object types. We call the causal relation $D_1 \succ^{op} D_2$ homogeneous when also $D_1 \rightarrow D_2$. In the other case, the causal relation is called a converter. For example, an increase in the number of patients will lead to an increase of the number of beds:

$$\text{Patient} \succ^{+} \text{Bed}$$

This is an example of a converter. The statement expresses the fact that there will be more new beds as a result of more patients in the hospital. So the number of patients positively influences the transition $\omega \rightarrow \text{Bed}$.

$$\text{Depends} (\text{Patient}, \omega \rightarrow \text{Bed})$$
8.4 Simulation

Simulation is defined as the artificial imitation of adequate components of a real-world situation to achieve certain goals. It enforces the internal reliability of the theory and ensures that the behavior it claims to explain can be generated by its underlying assumptions [160]. In this section we discuss how an extended ORM scheme is converted into a stock and flow diagram. The ORM extensions we introduced, add essential System Dynamics concepts to the conceptual level description of ORM. As a result, the conversion is rather straightforward. The conversion focuses only on the states of object types and their transitions. The decomposition structure is ignored, since System Dynamics has no decomposition mechanism.

As an example, we use figure 8.18 focusing on patients only. Under patient there are nine states \((P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9)\) placed in four main categories (Enter hospital, Examination, Treatment and Leave Hospital). However, to arrive at the Stock and Flow Diagram (SFD) we take states \(P_1, P_2, P_3\) and \(P_4\) as one stock Enter Hospital because they constitute of activities patients take on entering the hospital. Other states are represented independently as stocks and they include; Examination \((P_5)\), Admitted patients \((P_6)\), Monitored patients \((P_7)\), Births\((P_8)\) and Discharges \((\text{Leave Hospital})\) \((P_9)\). In each of these stocks are different quantities at time \(t\). The quantities in these stocks are determined by inflows and outflows connected to each stock.

![Figure 8.19: A partial stock and flow diagram representation of patient life](image)

In figure 8.19 we depict a stock and flow diagram that defines the flow of patient into different stocks (states). After a small interval of time \(dt\), the quantities in these stocks equal to the number of patients at the start of the time interval plus or minus the net flow of patients into these stocks.

**Defining input parameters**

Patient(s) enter hospital through inflow \((A)\), after a period of time they are examined. To take care of the change in stock Enter hospital, we add an outflow \((X)\). The quantities in stock Enter hospital are defined as;

\[
\text{EnterHospital} (t) = \text{EnterHospital} (t - dt) + (A - X) \times dt
\]

Where \(A\) is the rate at which patients are coming into hospital, \(X\) is the rate at which...
patients are being examined and $dt$ is the time interval. Inflow $A$ parameters are defined as:

$$A = \text{if } ( \text{mod (time, 2)} = 1 ) \text{ then } 2 \text{ else } 0$$

and outflow $X$ is defined as

$$X = \frac{\text{ExaminationCapacity}}{\text{Medical Persons on duty} \times \text{ExaminationDuration}}$$

Where

$\text{ExaminationCapacity} = \max(3 \text{ patients})$

$\text{Medical Persons on duty} = \min(2 \text{ persons})$

$\text{ExaminationDuration} = \max(30 \text{ minutes})$

Note that in this example, converter $\text{Medical Persons on duty}$ acts as an exogenous variable because in this example we focus on patients only. Quantities in stock examination are defined as:

$$\text{Examination} (t) = \text{Examination} (t - dt) + (X - Z - Y) \times dt$$

Where

$$Z = \text{Treatment duration} + \text{Examination duration}$$

$$Y = X / \text{Treatment duration}$$

Quantities in stock admitted patients are defined as:

$$\text{AdmittedPatients} (t) = \text{AdmittedPatients} (t - dt) + (Z - K) \times dt$$

Where

$$K = \text{if } ( \text{Dilation} \geq 4 ) \text{ then } \text{MonitoredPatient} \text{ else } 0$$

$$\text{Dilation} = \max ( \text{round} (\text{normal}(10,8,4)), 0)$$

Quantities in stock Monitored Patients are defined as:

$$\text{MonitoredPatient} (t) = \text{MonitoredPatient} (t - dt) + (K - M) \times dt$$

Where

$$M = \text{if } ( \text{Dilation} \geq 8 ) \text{ then } \text{Births} \text{ else } \text{MonitoredPatient}$$

Quantities in stock Births are defined as:
Births \((t) = \text{Births} \ (t - dt) + (M - N) \cdot dt\)

Where

\[ N = \text{delay (Births, 0) / TreatmentDuration} \]

And quantities in stock Discharges are defined as:

\[ \text{Discharges} \ (t) = \text{Discharges} \ (t - dt) + (N + Y) \cdot dt \]

Figure 8.20: Stock and flow diagram representation of patient life

In figure 8.20, we add a few converters pertaining to the problem domain that cause change in the behavior of the quantities in stocks through flows. Each stock is given an initial value and the equations defining the stock quantities are placed in the converters and flows. The added converters include: Medical persons on duty, Examination capacity, Examination duration, Discharge time and Treatment duration.

8.4.1 Model results

Having defined the different stock and flow diagram input parameters, we now present the simulation results for the model in figure 8.20.

SD tools are continuous simulation tools. However, some have discrete capabilities like conveyors, large array handling and discrete flows. In table 8.1, we show the variations in stocks Discharges, Admitted patients and Enter hospital. In these variations we see that Discharges (reservoir stock) have an exponential growth well as Enter hospital and admitted patients (conveyor stocks) have an exponential decay. The exponential growth in stock Discharges occurs due to the positive influences other variables have on this stock. For example, an increase in the number of births leads to an increase in discharges. Secondly the discharge stock is a reservoir stock meaning the simulation results of stock Discharges is an accumulation of patients that leave the hospital.
In the simulation results of *Enter hospital* and *Admitted patients*, there is a close gap. This is because most of the pregnant women who come into hospital are due for labor therefore the duration of admission varies depending on the complexity of the pregnancy. Since patients do not come in at the same time but in varying time intervals, we define the rate at which patients come into hospital (Inflow A) as IF(MOD(time, 2) = 1)then2else0. This input parameter creates oscillations that affect stock *Enter hospital*. From the simulation results, we see that *Enter hospital* and *Admitted patients* have an exponential decay. First these stocks are given varying transit time and once a patient gives birth she is discharged. These discharges reduce the number of admitted patients and those that enter hospital. Note that, for stock *Enter Hospital* we only consider patients not attendants and workers.

Table 8.2: Variations in stocks Births, MonitoredPatients and Discharges
8.5. Conclusion

In table 8.2, we show the simulation output for stocks *Births*, *MonitoredPatients* and *Discharges*. In these variations we see that all the three stocks have an exponential growth. However, *MonitoredPatients* and *Births* are less stable due to the reflected fluctuations over a period of time. These fluctuations are due to the influences both variables have on each other as defined in the simulation input parameters.

Table 8.3: Variations in flows M, K and N

In table 8.3, we show variations in flows *K*, *M* and *N*. Flow *K* reflects the rate at which patients flow from stock *Admitted patients* to stock *Monitored patients*. The simulations for the flow rate *K* depict random variations because patients flow at different time intervals. Flow *M* reflects the rate at which patients flow from stock *Monitored patients* to stock *Births*. The simulations results for flow rate *M* also depicts random variations because patients flow at different time intervals. Flows *N* reflects the rate at which patients flow from stock *Births* to stock *Discharges*. The simulation results for flow rate *N* depict an exponential growth because of the continuous accumulation and positive growth of stock *Births*.

8.5 Conclusion

In this chapter, we have discussed behavior of single objects applying it to various examples and a case "Intrapartum process in Ugandan Hospitals". Secondly, we have extended ORM to show how various levels of abstraction can be seen using the decomposition mechanism. This has led to more steps to the GSD method and these steps are presented in chapter 9. Finally, we have discussed how an extended ORM scheme is converted into a Stock and Flow Diagram.
Chapter 9

Suggestions for organizational and tool support

In this study we have particularly used Object-Role Modeling (ORM) to support System Dynamics (SD). We started off by defining the concepts in each method (see chapter 4), thereafter identified relations in these two methods (see chapter 5), evaluated the ORM to SD relations, defined the Grounded System Dynamics (GSD) method, evaluated the steps in the GSD method and discussed the update behavior(s). The GSD method comprises of numerous steps, these steps require a clear understanding of the domain. Understanding of the domain enables the modeler to easily define the derivation rules, stock and flow input parameters and, better understand object behavior. In this chapter, we start by discussing organizational involvement in the modeling process, as a subsection we discuss two of the approaches used in organizational modeling (collaboration and enterprise modeling) emphasizing the use of system dynamics. In section 9.2 we show how the GSD method can be operationalized and thereafter draw conclusions.

9.1 Organization involvement in the modeling process

In organizations, change is the only constant variable. This is because problems encountered are not only structured but dynamic and interconnected, which means that decisions made should less depend on intuitive and gut feelings. One of the ways organizations can deal with change is by understanding organizational complexity and business operations. To achieve this, decision makers should be equipped with the right tools and information systems should be aligned to organizations and the work practices [212]. Thus, decision makers get to use informative and well grounded models. To understand organization operations, modeling is fundamental. It is therefore important that the modeling processes, activities, conceptualization are well understood, as this will not only improve the outcome but also the decisions being made. To achieve this, it is essential that organizations are involved in the modeling process, that way modelers are able to obtain first hand infor-
CHAPTER 9. SUGGESTIONS FOR ORGANIZATIONAL AND TOOL SUPPORT

Information from stakeholders hence comprehensive and valid models. Having valid models leads to more credibility and applicability of the model thus reducing the gap between what is done in academics and in practice.

9.1.1 Alignment of information systems to organizational structure

Building organizational structures and sets of business processes that reflect the interdependence of enterprise strategy and information technology capabilities requires linking of information systems to organizational needs. This however cannot be achieved in isolation. It is therefore vital that stakeholders work together with modelers to arrive at substantive results. Secondly, Information System (IS) strategies should be aligned to the organizational structure. Alignment can modify the basic stakeholders’ view on the nature of an organization and decisions thus effective and efficient utilization of organization’s information system. Alignment however, is not a precise activity, it involves lateral thinking and opportunism. Two mechanisms of alignment are identified in [28] that is strategic alignment and structural alignment. In [82], they are of the view that strategy concerns both formulation and implementation and strategic alignment is not an event but a process of continuous adaptation and change. Alignment by definition is a state of agreement or collaboration with a common cause or perspective. Strategic alignment therefore is an agreed upon collaboration process to continuously adapt, formulate and implement IS to the organizational strategy. On the other hand structural alignment examines the degree of structural fit between IS and the business, specifically in the areas of IS decision-making [28]. The two types of alignment include horizontal and vertical alignment [82]. Horizontal alignment describes the relation between internal processes and external customers while vertical alignment describes the relation between the top strategy and the lower levels of the information system (tactical and operational) [82, 120]. In this study we used vertical alignment because we combine system dynamics (used at the strategic level) with ORM (used at the operational level). By so doing we capture decision maker’s views at the strategic level through CLDs and use them in context with operational concepts (ORM and SFD) to arrive at a grounded method. Thus all stakeholders especially decision makers are able to view the impact of their decisions based on what is inexistence (actual data) not what is assumed to exist (prediction). Nevertheless predictions are also made possible through tuning of SFD converter and flow input parameters.

Arriving at the model alone does not guarantee the effectiveness of the model (this is inline with system dynamics Causal Loop Diagram). Normally, stakeholders identify variables and state their causal relations according to their interpretation and perception of the problem. What they do not do is get into details of understanding facts pertaining to each variable, the relations or better still clearly give meaningful variable names (names that are précis and easy to understand). In the Grounded System Dynamics method we urge that models derived from the modeling sessions with stakeholders should go through other stages like identification of facts for each variable, stating of derivation rules attached to each variable, describing of object behaviors etc. This will improve model conceptualization and rationalize ambiguities. Finally, system dynamics being a problem structuring method, it is commonly used at the strategic level whereas ORM is a domain modeling method (data modeling method) meaning it is used at the lower levels of mod-
9.1. ORGANIZATION INVOLVEMENT IN THE MODELING PROCESS

By combining these two methods, we are able to reduce the gap between structural methods (commonly used by decision makers) and domain methods.

9.1.2 Enterprise architecture modeling

One of the modeling methods used to understand the complexities involved in today's organizations is enterprise architectural modeling. Enterprise architectural modeling is made up of three main words; Enterprise, architecture and modeling. An enterprise is a collection of organizations with a common set of goals. Architecture personifies components of organization systems, their relationships to each other, their environment, and the principle guiding its design and progress [103, 119, 232]. Enterprise Architecture (EA) therefore is the process of transforming the organization's visualized goal and IT infrastructure to mirror the company's operating model by creating, communicating and improving the key requirements, principles and models that describe the enterprise's future. It consists of principles, methods and models that are used in the design and realization of the enterprise's organizational structure, business processes, information systems and infrastructure [119]. The process of constructing structured descriptions relevant to the activity of an enterprise is referred to as enterprise modeling. The enterprise modeling process involves a number of stakeholders whose aim is to develop an enterprise model for the organization following modeling set rules. The outcome of the enterprise modeling process is an enterprise model. An enterprise model represents the structure, activities, processes, resources, behavior, goals and constraints of an enterprise thus the language to define an enterprise should be provided [54]. Enterprise modeling can be enriched by adopting collaborative problem solving and decision making approaches. Such approaches can be generally referred to as Group Support Systems (GSSs) [50]. GSSs is broadly categorize into two, i.e. Problem Structuring Methods (PSMs) and Electronic Meeting Systems (EMSs) [172, 217]. PSMs support creative and rational thinking among groups and individuals about an organization’s baseline and target situations, while EMSs enable computer based collaborative work practices. PSMs include systems dynamics, Group model building, Soft Systems Methodology (SSM), Strategic choice approach, journey making etc. Research efforts towards enriching enterprise modeling with PSMs and EMSs are underway. For example, the Collaborative Evaluation of (Enterprise) Architecture Design Alternatives (CEADA) project [146] explores how Collaboration engineering can be used to enable the supplementary use of EMSs, SSM, and SD Causal Loop Diagrams along with architecture modeling approaches (i.e. languages like BPML, UML, DEMO, ORM; tools, techniques, frameworks) during enterprise architecture creation. The resultant of the CEADA project is the CEADA method that supports collaborative sessions in which enterprise architects involve stakeholders in the architecture modeling process. The Grounded System Dynamics (GSD) method can be one of the methods used in such modeling sessions to offer dynamic perspectives and shared understanding on various organizational processes and data that supports those processes. The effectiveness and efficiency of such collaborative modeling sessions can be evaluated using an evaluation framework that involve expert (domain and modelers) and stakeholders [191].
9.1.3 Collaborative modeling

Collaborative modeling is one of the approaches used in EA. Collaborative modeling is a process that engages stakeholders (people who are affected by, can influence, or are interested in the decision at hand) in executing a modeling task. The outcome of this modeling task is a model that supports the decision processes [168]. The collaboration modeling group may comprise of stakeholders, decision makers and domain experts. In [93, 92] they discuss and study the collaborative modeling process in more detail. In their study they identify some of the key activities involved in the collaborative modeling process among which are; dialogue structure, negotiations, rules, interactions and Models. During the collaborative modeling process, modeling languages like UML [17, 174], DEMO [38, 39], BPMN [223] are used and models developed. These models help to better understand the existing conditions and future prospects of the organization, for management and decision support, and the modeling express information in a structure that is defined by a consistent set of rules. For example, in chapter 8 we define a set of rules for the update behavior, these rules interpret the meaning of components in the structure.

The approach of involving organization stakeholders in the modeling process is not new, examples of such approaches include; participatory modeling [131, 179], Group Model Building (GMB) [216] etc. These fields may have different specifications but their underlying idea is more or less the same because they all involve the support of some form of dialogue between stakeholders in the development of a rational model. In [171, 215] some of the advantages for involving stakeholders in the modeling process are stated and these include;

- Lead to a shared understanding of the problem among participants.
- Participants get to express their views freely.
- There is consensus.

More in particular, in [215] Vennix gives reasons as to why using Group Model Building (GMB) is important.

9.2 Towards operationalizing the process

GSD provides steps that are more formally structured to successfully improve quantification of simulations. The reliability of GSD depends critically on the reliability of the defined rules, conceptual object descriptions, relations and transformations. To facilitate operationalization of the GSD method, we provide guidelines. These guidelines provide additional support for users of the GSD method.

GL-1: Develop a CLD model.

The first step when developing a system dynamics model is normally constructing a CLD. This CLD model represents articulated mental models. The CLD modeling process however, does not impose many restrictions and does not separate structure from behavior.
9.2. TOWARDS OPERATIONALIZING THE PROCESS

But helps to express, organize knowledge and assess learning about complex situations [176]. Secondly, CLDs are important at articulating and understanding how variables influence each other. In this methodology therefore, we use a CLD model as a basis for the GSD model.

GL-2: Transform the CLD model into an ORM model.

As stated in chapter 1, we transform a CLD model into an ORM model to clear ambiguities in the CLD model and to improve SD model conceptualization (refinement and specification of abstract concepts). To clear ambiguities, we apply the steps in table 7.9 chapter 7 and to improve conceptualization, we study the update behaviors.

GL-3: Decompose the ORM model.

Decomposition is the disintegration or breaking down of a given ORM model into small manageable fragments. As presented in chapter 8, decomposing of an ORM model follows a number of steps. A summary of these decomposition guiding steps is presented in table 9.1.

<table>
<thead>
<tr>
<th>Decomposition guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a)</strong> Separate object types with their roles into unique ORM schemas.</td>
</tr>
<tr>
<td><strong>b)</strong> Handle each object type independently.</td>
</tr>
<tr>
<td>Give each role a unique identification.</td>
</tr>
<tr>
<td>Categorize states into elementary and compound states.</td>
</tr>
<tr>
<td>Represent states in order of activity occurrence.</td>
</tr>
<tr>
<td>Add directional paths.</td>
</tr>
<tr>
<td><strong>c)</strong> Merge different decomposition levels and represent them as a whole.</td>
</tr>
</tbody>
</table>

Figure 9.1: Decomposition guidelines

a) Separate object types with their roles into unique ORM schemas.

In most case, object types in an ORM model are more than one and contain different objects. To apply the decomposition mechanism to an ORM model, object types should be separated into different ORM schemes. However, in some cases the ORM model has super types and for each supertype there is a hierarchy of substates. It is therefore important to separate these object types so that the modeler is able to represent states for each ORM schema with a unique identifier. Although the contents in each state have similar properties.

b) Handle each object type independently.

In order to improve conceptualization of the SFD model, handle each object type independently. Doing so enables a modeler understand how objects in each state relate to one another. For each object type;

1. Give each role a unique identification.
2. Categorize states into elementary and compound states.
3. Represent states in order of activity occurrence.
4. Add directional paths.

Once the modeler understands the states and state transitions in each object type or hierarchy, (s)he is able to analyze the model behavior at a conceptual level.

c) Merge different decomposition levels and represent them as a whole.

Having handled each object type or hierarchy independently it is important that the different ORM schemas are merged into one complete model. In this merged model, all the decomposition levels can be viewed. After this step, SFD input parameters or values are easy to define because all existing states and transitions are already identified. During the development of an SFD model, the guidelines presented in table 9.2 are used to show how the modeler can arrive at simulation results.

GL-4: Construct a stock and flow diagram and simulate the model.

To transform a decomposed ORM model into an SFD, we use the identified relations between ORM elements and SFD elements (See chapter 5). Thereafter follow the defined ORM to SFD transformation steps (see chapter 7 subsection 7.2.3). These steps however do not give a complete SFD model with behavioral descriptions, yet one of the objectives of GSD is to improve model conceptualization. To achieve this, we came up with a decomposition mechanism discussed in chapter 8 and a summary of the decomposition guidelines is presented in table 9.1.

Another important aspect in SFD models is simulation. Simulation allows continuous testing of assumptions and sensitivity analysis of parameters [142]. A simulation model distinguishes between stocks and flows. As stated in chapter 4, stocks change over-time through flows to produce the dynamic behavioral patterns of the SFD model. In table 9.2, we present guidelines on how to arrive at an SFD simulation model. These guidelines were derived as we constructed the stock and flow diagram in chapter 8 section 8.4.

<table>
<thead>
<tr>
<th>Simulation Guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Represent each state category as a stock and use meaningful names.</td>
</tr>
<tr>
<td>b) For each stock, define initial values or quantities.</td>
</tr>
<tr>
<td>c) Identify flows (inflow or outflow) connecting to each stock.</td>
</tr>
<tr>
<td>d) For each added information link or converter, define input values or parameters.</td>
</tr>
<tr>
<td>e) Ensure that all input equations are logical and units are consistent.</td>
</tr>
<tr>
<td>f) When evaluating the model, focus should be on interactions rather than individual elements.</td>
</tr>
<tr>
<td>g) Conduct tests e.g. mass balance, logical and extreme-value tests etc.</td>
</tr>
</tbody>
</table>

Figure 9.2: Guidelines for developing a simulation model

a) Represent each state category as a stock and use meaningful names.

After applying the decomposition mechanism, we can now see various levels of model abstraction. In each level there are elementary and compound states as explained in chapter 8. Each of these states (compound or elementary) is depicted
as a stock in SFD because they contain elements (objects) with similar properties. Note that compound states are made up of more than elementary states. Secondly, to improve understanding of the evaluation models, use simple and precise variable names [27].

b) *For each stock, define initial values or quantities.*

For simulation computations, it is very important that defined initial values are appropriate. An initial value is a point at which quantities in a stock start to grow. For example if a stock has an initial value of 50, then in the simulation graph quantities in that stock start at 50 and if the initial value is zero, then the simulation of the stock also starts at zero.

c) *Identify flows (inflow or outflow) connecting to each stock.*

Flows occur over a period of time. In business settings, there are several interactions and there exists many possible flow equations that are consistent with the stock and flow diagram. But each problem domain has different variables and causal influences, the equations of the flows therefore must be entered or defined by the modeler. To successfully construct a stock and flow diagram, it is necessary to understand the difference between stocks and flows [107]. Flows have rates at which quantities flow into or out of the stocks.

d) *For each added information link or converter, define input values or parameters.*

Information links and converters are a central component in SFD models and play an important role. Information links can be difficult to model because of their abstract nature. Through information links, information from one converter/flow/stock can simultaneously flow to other SFD elements rapidly and with substantial twist. The addition of an information link can lead to profound impact on the model performance and simulation results [107]. It is therefore important that the modeler has a logical explanation for each information link added.

e) *Ensure that all input equations are logical and units are consistent.*

SFD input equations or parameters should have a logical explanation. Each variable at the start of the connector should be included into the equations of the variable at the arrow tip of the connector. Secondly, the units on the right hand side of each equation should have the same unit measure as the ones on the left.

Ensuring that all input equations are logical and units are consistent is a way of validating the internal structure of the SD model. Validity of the model in system dynamics means that the internal structure of the model is valid not its output behavior [10]. A valid SD model structure (the totality of the relationships between or among variables) can be used to test the effect of changes on the defined problem. A model that generates a behavior with little or no relationship with that of the system is most times unreliable and thus invalid. But if the model behavior replicates the behavior of the real system then it is valid.

f) *When evaluating the model, focus should be on interactions rather than individual elements.*
When evaluating the SFD model, focus should be on interactions between or among variables rather than individual elements. This is because SFD elements influence each other through causal links and flows. If SFD elements are dealt with in isolation there would be a possibility of not arriving at a valid model because all elements in the model contribute to SFD model validity.

Secondly, interactions or relations should be logical and before using the model to examine policy decisions, it should be validated against observed or likely trends. If the defined model reproduces or represents the real system (in the reference graph), then it is assumed to contain critical elements generating the problem but if it does not reproduce the reference graph then the model structure and causal influences should be revised.

g) Conduct tests.

In system dynamics modelers and users gain confidence in the model through testing. In [7, 8] three classes of tests are suggested: structure tests, behavior tests and policy implication tests. **Structure tests** determine how well the structure of the model matches the structure of reality, **Behavior pattern tests** determine how consistently model outputs match real world behavior and **Policy implication tests** determine whether the observed system responses to policy changes replicate model predictions. In the process of conducting these tests, simulations depicting the behaviors of input variables are derived. To conduct these simulations:

- Choose appropriate simulation details when conducting test runs.
- Use behavior over time graphs to understand the correlations between model variables.
- Change input values to analyze the effect to change.

### 9.2.1 Methods and procedure used

As already stated in chapter 6, to come up with the GSD method, we combined two methods (ORM and SD) following the design science methodology. Here we are not discussing the methods per se because we did that in chapters 2 and 4 respectively but we suggest how these methods are to be used and at what level they are applied in an organization. Earlier on we mentioned that system dynamics models are used at the strategic level and help decision makers articulate and study the causal influences on which they base their decisions. On the other hand ORM as a domain conceptualization method identifies facts pertaining to each variable as shown in chapters 6, 7 and 8. The combination of these two methods implies that a treatment from both methods is contained in the same method (GSD). Each of these combined methods has guiding steps, for system dynamics see [167, 159, 169, 227, 193] and for ORM, seven conceptual scheme design procedure steps are also given in [77]. Here we do not further outline or show ORM and SD guiding steps but draw our focus on the GSD steps.

The GSD method uses a step by step approach to underpin a system dynamics model with a domain method. An overview of these phases are presented in figure 9.3.
9.2. TOWARDS OPERATIONALIZING THE PROCESS

Identify and connect roles played by objects in a particular object type
   a) To other object types
   b) To objects in the same object type

Identify all possible connectors (information links) and connect/link them to required flows, converters and stocks.

Iterative and in parallel with phase 2

In phase one we have six steps, these steps show how information in a CLD can be transformed into an ORM model. In phase two we have five steps and a sub-phase with four steps. The steps in phase two show how an ORM model can be transformed into a stock and flow diagram. These phases were first introduced in chapter 6 and evaluated in chapter 7. As a result of the feedback we received from participants in chapter 7, we came up with a sub-phase for phase two. Without the sub-phase, what we had at the end of the first two phases was a “picture” of an SFD. The sub-phase therefore gives life to the SFD model by showing how ORM derivation rules, decomposition and facts can help in determining the input parameters and understanding of the SD model behavior. In chapter 8 section 8.4, we explicitly explain how decomposition (an extension of ORM) can be used to capture details and complexities associated with ORM and SD models. To clearly represent this, we used the case “Intrapartum process in Ugandan Hospitals”.

In step (a) and (b), we first of all focus on each object type, define roles per object type and then make a grouping of roles. In doing so, compound states are differentiated from single state. For a clear explanation (see subsection 8.2). After differentiating the

Figure 9.3: overview of the way of working (This model was formulated basing on the discussions in chapters 7, 8 and references [107, 7, 8, 10])
compound states from single states, we describe input parameters (step (c)). This is done in an incremental manner to ensure consistency and to avoid complexities associated with big detailed models. On completing step (c) we gauge the effect of variable uncertainties using the defined functions. These uncertainties are due to a combination of variables in the function, measurement limitations and inaccuracy of input values. The process of gauging the effect of variables is what we refer to as simulation and the analysis of the simulation output is error propagation. Error propagation is the effect on a function by a variable’s uncertainty. Through error propagation a modeler is able to provide an accurate measurement of uncertainty. Error propagation is a calculus derived statistical calculation designed to combine uncertainties from multiple variables. To determine the parameter of a function requires several measured variables which are subject to uncertainty in their values\(^1\). The relationships of these variables to the parameter can be expressed as known mathematical functions valid over a certain range of variables. All this is done in step (d). Conclusions are drawn based on the results obtained from those studies. The bidirectional arrow from phase two to the sub-phase in figure 9.3 indicates that some of the steps in the sub-phase can be carried out concurrently with steps in phase two. For example steps (a) and (b) are better carried out concurrently with steps 7 and 8; steps 9 and 10 can be carried out concurrently with step (c).

\[\text{ORM model} \quad \begin{array}{c} \text{Domain Knowledge} \quad \text{Defined ORM derivation rules, facts and decomposition mechanism} \\ \text{and assumptions} \quad \text{Mathematical} \\ \quad \text{definitions} \quad \text{Physical Properties} \quad \text{Differential equations, values} \\ \quad \quad \text{and converter equations} \\ \text{Simulations} \end{array} \]

Figure 9.4: Approach to GSD simulation

In figure 9.4 we present a summary of our approach to the GSD simulation depicted in the sub-phase. In GSD it is important to have all information about the problem domain captured in the ORM model that way the modeler gets a grip of the problem domain facts before applying any assumptions. During the ORM model construction, derivation rules and facts are stated. In the process of constructing the ORM model, derivation rules and facts are stated. After validating the ORM model, the ORM to SFD steps are applied. Included in these steps is the decomposition mechanism. This mechanism helps the modeler conceptualize and abstract details of the ORM model. Applying the decomposition mechanism enables us to better understand the underlying concepts of the SFD model and update behaviors of ORM objects. Using the abstraction view of the decomposition levels, we derive an SFD model. In this model, equations inform of differentials, mathematical formulas and values are input into flows and converters, initial values for stocks are also defined. Defining of input values or equations requires physical properties (see figure 9.4). These properties should be measurable because they define the

\(^1\)wikipedia.org/wiki/Propagation_of_uncertainty
physical state of the system. The changes in these properties describe the behavior of the
defined variables over a period of time (simulations).

9.2.2 The medium

The GSD method mainly requires two platforms, system dynamics tools(s) and ORM
tool(s). To represent information in the mental models and to understand the feedback
relationships, a CLD model is constructed using Vensim PLE or any other convenient SD
tool. To create an unambiguous definition of a logical universe of discourse, an ORM
support tool is used (NORMA). To systematically represent the behaviors of different
objects (decomposition mechanism), the modeler or system analyst may use any platform
that is convenient for him. In our case we used Microsoft Word and Microsoft PowerPoint.
To develop and execute simulation models, system dynamics modeling tools like Vensim
PLE, Stella/iThink, Powersim [154] etc can be used. In figure 9.5, we present an overview
of the tools used at each modeling phase.

![Figure 9.5: Overview of the tools used at each modeling phase](image)

In activity 1 and 4, SD tools are used. In activity 2, ORM tools are used and in
activity 3 the modeler(s) can use any graphical tool or platform of their choice. ORM
tool’s limitations for abstraction and capturing decomposition mechanism hinder its app-
licability for the GSD method. We therefore recommend an extension of both SD and
ORM tools to capture the static (SD) and decomposition mechanism (ORM). Capturing
information in this manner may lead to something like a ‘dashboard’ where changes to a
model are reflected instantly hence a dynamic way of working and better decision making.

9.2.3 Who are the participants and how do they interact

In the GSD method, the key players are domain experts (stakeholders, clients, and de-
cision makers), modelers and system analysts. These key persons should work together
to arrive at the desired goal. Since domain experts know what data to use and which
business rules are relevant to individual data items or groups of data items, they therefore play an active part in the GSD process. The key role for the domain expert(s) is to verify, clarify the defined variables and to make sure that the problem domain is properly represented. In the first phase of the GSD method, the domain expert(s) should work together with the modeler to make sense of the details of the problem domain. That is to say domain experts should be involved in the process of defining facts pertaining to the identified variable, defining the causal influences and identifying relationships between or among different objects. In the second phase, the domain expert(s) and modeler(s) work with the system analyst to plan on how to implement the outcome and how to design the components that are provided to the developer.

The GSD modeling process comprises of numerous tasks. Getting the best out of this modeling process requires a modeler to understand the methods being used, the stakeholder’s interests, how the organization operates, and the processes involved. Knowing all this puts the researcher or modeler in a better position to communicate with the stakeholders. The communications aspect is well discussed in chapter 7 of [94]. Note that our focus is on how to come up with the method and not what is required to accomplish the task [212]. The GSD method requires that the persons taking part in the modeling task have knowledge in either of the methods being combined hence, clear interpretation and better understanding of the underlying principles. Understanding of these principles plays an important role especially when it comes to identifying the simulation input parameters, model behavior and applying the decomposition mechanism. Since it is hard to come across modelers with knowledge in both methods we suggest that at each phase, the resulting model should be evaluated by expert(s) using that method for example; phase one (from a CLD to ORM) model should be evaluated by ORM experts and phase two (from an ORM model to a stock and flow diagram) should be evaluated by SD expert(s). When working on a real-life case, in GSD it is advisable to first go through the CLD with SD experts before applying the steps in the first phase. This helps the modeler(s) achieve a proper interpretation of the case. If there are numerous participants in the modeling process, it is important that for each step, all participants have a shared understanding of the resulting model(s). For consistency and transparency, the GSD steps should be followed systematically with less alternations if need be.

During the GSD evaluations a number of activities like communication, dialogue etc. were involved. In [191, 92] three items (rules, interactions and models) involved in the communication process are presented. In the communication process, different participants with either a common goal (cooperative) or conflicting (adversarial) are involved [202]. Here participants interact and share their views with other participants. Another activity involved is dialogue, some scholars refer to dialogue as a game [92, 93]. Dialogue games occur during argumentation to regulate disagreements between two parties on the tenability of one or more claims. During focus group sessions, we noted that sometimes participants executed persuasion dialogues. Persuasion dialogue is where participants persuade other participant to adopt their point of view [220]. During dialogue games, conditions under which statements are appropriate are defined. The principles governing the meaning and use of statements are defined at the stage of the dialogue in which the statement is made [155].
9.2.4 How to bring the model to life

In order to bring the model to life, there are three main phases (conceptual model, specification of sets of variables and simulation of behavior). First the modeler comes up with a conceptual model. This is done by hand and the modeler’s concern is to represent the domain expert opinions on what is true. By so doing, different concepts and relations in the methods are defined thus interpretations of these concepts by different parties (modelers, system analyst, stakeholders) are streamlined. This implies that stakeholders, especially those responsible for designing and implementing the solutions, get to better understand the problem as it evolves. The second phase is to select specific sets of variables. These variables are based on the system dynamics model because it allows visualization of behavior through simulations. In the final phase, the behavior of the model is simulated. Tuning of the flows is manually done as the modeler defines the input parameters and quantities directly. In figure 9.6 we present an illustration of how the GSD method is applied.

![Figure 9.6: Application of GSD](image)

In the process of defining facts pertaining to each CLD variable, it is good to state all derivation rules and also have a catalogue of terms because they make defining SFD input parameters easier. As facts are defined, some of the CLD variables are rephrased hence ambiguities are cleared and naming of CLD variables is more explicit. Construction of the CLD model requires input from decision makers (see GMB [214]), same applies to the GSD method however, in GSD once the CLD model is complete, decision makers may not be required unless if they are the same as domain experts. Domain experts are contacted from time to time for clarifications.

To have a complete GSD model, merging or integrating of more than one databases may be required. For example if we want to know the ‘total number of people living in EU’, we need to know the number of births, immigrants, death and those emigrating into other countries. As all this information is likely to be stored in various databases, having casual influences alone without actual data may lead to inaccurate simulations.

Finally, to execute this method, the act of modeling should be well understood. Understanding the act of modeling requires understanding of the underlying principles (con-
CHAPTER 9. SUGGESTIONS FOR ORGANIZATIONAL AND TOOL SUPPORT

cepts), analysis and evaluations of these concepts, defining of relations between method constructs and having a systematic way of working. All these have been discussed in our earlier chapters.

9.3 Discussions

Complex system dynamics models are hard to conceptualize because modelers or stakeholders cannot understand how various parts of the system interact and add up to the whole [158]. In this thesis however, we have demonstrated how support from ORM can help SD modeler(s) better understand the behaviors in a real system using the decomposition mechanism. Understanding the relationships between objects and their (object) behavior(s) leads to better SD models. In GSD, conceptualization is an iterative process where the modeler does not only define the variables and causal links but also has to clearly understand the objects (actors) in each variables, their roles (how they relate with other variables) and of what use they are to the model as a whole [129].

System dynamics as a method is known to handle all sorts of problems (quantifiable and unquantifiable), however as we stated in chapter 7 the GSD method does not work well with unquantifiable (uncountable) objects. This is because the GSD method requires that the variables used are explicit with derivable facts. Having explicit variables allows the modeler understand the objects (key players/actors) in each variable and their relationships with other variables. Further still, it is recognized in SD that, most of the information available to modelers resides in the mental database [132, 41]. These mental databases give a description of reality that is usually expressed by a set of sentences in natural language [41]. Since GSD combines both SD with ORM that has its conceptual focus and roots in verbalization, the variable names used in a GSD model are therefore made explicit.

Finally, in GSD a CLD is the basis from which other models are derived, it is therefore important to have a CLD before applying the GSD method. In our case we used a CLD constructed with a group of people (in a typical organization, these would be stakeholders who are the decision makers) and facilitated by one experienced system dynamics person. A similar approach can be found in [215]. The CLD model derived was dependant on participant’s interpretation of the problem however our concentration here is not on the CLD modeling process but the GSD method. The first phase of GSD typically involves conceptualization of the problem and streamlining of the defined CLD variable names. The second phase is about showing how ORM can be used to understand the object behavior(s) and simulation output. In section 9.2 we provided all the GSD steps. However, these steps have not been applied to an actual experiment for validation. Validity of the GSD method therefore, depends on understanding the underlying method concepts, the relations between the two methods(ORM and SD), having a shared conceptualization of the problem domain and knowing which set of variables to use in the model. The advantage of this method is that validation occurs at every phase. In the first phase, validation is done through ORM verbalizations. If the modeling in the first phase is well executed and validated, then fewer or no errors are expected in the second phase. The few
remaining errors can however be cleared through conceptual description of objects and experimental analysis of the simulation output (error propagation analysis).

9.4 Conclusion

In this chapter, we have discussed the importance of organizational involvement in the modeling process and tool support. We particularly state that, to improve the quality of the models being developed and to have a shared understanding of the model, stakeholders and modelers should be involved in the modeling process. We have also stated that there are research efforts towards enriching the enterprise modeling with PSMs and EMs and how efficiency and effectiveness of such collaboration modeling sessions can be evaluated. In subsection 9.2, we have discussed how the GSD process can be operationalized, stated the methods and procedures used, participants involved and how to bring the GSD model to life. In GSD, the normal system dynamics procedure (from a Causal Loop Diagram to a Stock and Flow Diagram (SFD)) is broken down into smaller manageable steps. These steps help the modeler gain insight into the compositional subsystems of the model. An overview of the problem domain is formulated through CLDs by specifying the causal influences, their polarities and constituting variables. This however does not include detailing of any information flow subsystems.
CHAPTER 9. SUGGESTIONS FOR ORGANIZATIONAL AND TOOL SUPPORT
Chapter 10

Conclusions

In this thesis, we have studied how to combine system dynamics with a domain modeling method (ORM). In the process of combining these two methods, we have highlighted some of their strengths and weaknesses. In this chapter however, we present a summary of the thesis contributions and direction for further research.

10.1 Contributions

To combine system dynamics with a domain modeling method, we followed a design science research methodology. Design science creates and evaluates IT artifacts intended to solve identified organizational problems [86]. In [86] it is stated that, development of the artifact should be drawn from existing theories and knowledge through a search process to come up with a solution for the defined problem. In this section, we demonstrate how research questions presented in chapter 1 have been answered and in what section or chapter of the thesis. First, we present the main research question.

**Main RQ: How can we combine system dynamics with a domain modeling method?**

In general, combining methods is the act of unifying method elements together methodically. The process of combining methods requires defining relationships between the methods, precise understanding of the relations between the methods and understanding of the underlying ideas of the methods, which then lead to a more principled exposition of the methods [224]. Through combining methods, system modeling and operating of complex systems could become easier to computerize [56]. This research question is too broad and requires that researchers carry out numerous tasks e.g identifying existing method elements, defining relationship between the methods etc. To answer this research question, we divided it into constituent specific research questions. As Vessey says, the key to a successful problem solution is to structure the problem into sub-problems that can be integrated to form a complete solution for a specific problem [218]. Through an-
swearing the given specific questions, we were able to answer the main research question in a methodical manner.

**RQ-1: What are the existing SD and ORM constructs?**

This research question is answered in chapter 4. To arrive at comprehensive definitions, we reviewed literature pertaining to SD and ORM. We started off by presenting ORM constructs (see section 4.1) followed by SD constructs (see section 4.2). In our presentations, we used examples to explain the underlying principles. Under system dynamics, there are two notations (Causal Loop Diagrams (CLD) and Stock and Flow Diagrams (SFD)). First, we presented CLD constructs followed by SFD constructs. Under CLDs we showed how a CLD is constructed, stated the existing constructs and defined the entailed design rules. In more detail, we showed how variables, sign and causal links interact to create loops. Having presented CLD constructs, we embarked on SFD constructs. Under SFDs, we defined each building block, design rules and also gave SFD formal definitions. In the last part of chapter 4, we discuss ORM as a Foundation for SD.

**RQ-2: What relationships exist between these two methods? How can we best represent the interactions in these methods?**

This research question was answered in chapter 5. To arrive at this answer, we reviewed literature and also carried out an evaluation using a focus group session. Conducting this session enabled us get participants views on the already defined relations and methods in general. To make these relationships explicit, we used numerous examples. In particular, we focused on key constructs, which are central to construction of stock and flow diagrams but seemingly difficult to use in a comprehensible and unambiguous manner. On answering research questions RQ-1 and RQ-2, so was requirement R-1 (*The constructs and underlying principles of both methods (System Dynamics and Object Role Modeling) should be explicitly defined and the identified relationships should be well documented.*)

**RQ-3: What is the procedure for combining these two methods?**

This research question is similar to the second requirement (R-2) defined in chapter 3. To answer this question, a method Grounded System Dynamics (GSD) was defined in chapter 6. GSD is a step by step method that shows how information from CLD can be transformed into an ORM schema and from an ORM schema to an SFD, and how ORM helps in understanding object behavior. In figure 10.1, we present an overview of the GSD method. In this figure, we have an initial transformation stage where information in a CLD is used to derive an ORM schema. In the process of transforming a CLD into an ORM schema, we are able to rename some of the CLD variables hence clearing ambiguities and better understanding of how variables in a CLD relate to each other. To achieve concrete results however, a domain expert and modeler should work together.

After transforming CLD variables into an ORM schema, we embarked on transforming an ORM schema into an SFD. Initially, this phase had five steps but after evaluation, we added a sub-phase as shown in chapter 9 figure 9.3. In this phase, transformation
10.1. CONTRIBUTIONS

decisions are taken on the basis of the ORM schema. These transformations depend on identified ORM to SD relations. This phase is carried out by a modeler and system analyst.

In conclusion the procedure of combining system dynamics with a domain modeling method is a step by step procedure that comprises of two main phase (CLD to ORM and from ORM to SFD). Within these steps are relations and rules on how to execute a given step. RQ-3 is answered in chapters 6 and 9.

RQ-4: How can we evaluate the procedure(s) for combining system dynamics with ORM and the resulting model?

To answer this question, we used examples and a case study in experimental and analytical evaluations (see chapter 7). These evaluations enabled us determine the worth and significance of the GSD transformation steps. Secondly, we were able to receive participant’s views and gain insights into prior GSD transformation steps. With the provided views and insights, we were able to refine and improve the GSD method.

In prior ORM to SFD transformation steps, we had five steps. These steps led to a ‘picture’ of an SFD model as some of the participants put it. This was because the derived SFD model model lacked the ability to show the behavior(s) between or among constructs. Due to this insight, we came up with chapter 8. In this chapter, we used the decomposition
mechanism to better conceptualize and understand the SD model behavior, and show how using ORM helps in to deal with complex SD models. When one understands how different elements in a model interact, determining SFD input parameters is made easier. The formulas and input values are what we refer to as SFD input parameters from which simulation are derived. These simulations are used for sensitive analysis. During sensitive analysis, simulation responses are examined with various input values to study the effect of changing these input values and interactions among inputs [110]. This last phase is carried out by a system analyst, modeler and domain expert.

**RQ-5: How can the modeling procedure(s) be embedded in a modeling tool(s)?**

This research question has been answered in chapter 9. In this chapter, we discussed organization involvement in the modeling process, how the GSD method is operationalized and provided guidelines for tool support. Note that this research question is similar to requirement R-6 (Suggestions for organizational and tool support should be provided.)

### 10.2 Discussion

Understanding how different variables in a system dynamics model interact [158] is hard. This is due to the ambiguities and complexity of the SD model. As demonstrated in this thesis (specifically chapters 5, 6 and 8), when SD is combined with a domain modeling method (ORM) this issue is minimized. This is because ORM has its focus on domain conceptualization. In the defined artifact (GSD), conceptualization is iterative i.e. a modeler defines the variables and causal links, identifies objects (actors) in each variables, their roles (how they relate with other variables) and of what use they are to the model as a whole. By so doing, a modeler is able to better understand the interactions in the model and clear ambiguities in the model thus transformation of a Causal Loop Diagram into a Stock and Flow Diagram is much easier. In this study, our main research question was ("How can we combine system dynamics with a domain modeling method?"). Answering this question required a systematic procedure thus dividing it into sub-questions (specific research questions).

**Defining constructs:** To begin with, we defined each method construct and underlying principles. Constructs are sometimes referred to as components, fragments or semantic primitives [80, 185]. A constructs as defined in [86] are concepts, ideas or images specifically conceived for the purpose of organizing and representing knowledge of interest of a given modeling method. Since we were combining different methods, defining ORM and SD constructs seemed the most logical thing to do. In the process of defining these constructs the researchers gained knowledge on the method underlying concepts (see chapter 4). Constructs helped researchers specify what can be modeled with a given method [180, 185]. On completing this definition phase, the researchers were able to identify some of the relations between ORM and SD constructs (see chapter 5).

**Identifying relations:** Before defining a new artifact, we identified the relations between different method constructs. This enabled us transfer a set of rules and techniques
10.3. FUTURE WORK

used in one method into another method (even model transformation). Relation identifi-
cation is a re-engineering approach and in our thesis, it was done manually (see chapter 5). While identifying relations between the two methods (ORM and SD), we discovered that it was impossible to exhaustively relate all constructs from one method into the other. What we did was to use constructs that were most similar but not identical.

Defining an artifact: Artifacts are models, constructs, methods and instantiations [85]. In our case the artifact is a method (GSD). In [86], designing an artifact is defined as the first guideline. However in some research projects it is done after problem relevance (Guideline 2). This does not affect the outcome in any way as long as the researcher meets the requirements of a design sciences approach. In our study, we defined the artifact after identifying existing method constructs and relations. This was because we needed to reason about the combination of ORM and SD before defining the artifact.

Evaluating the defined artifact: In our study, evaluation occurred at each phase of building the artifact. In our case, evaluation first occurred at the relation identification stage. The outcome of these evaluations were used as a basing for defining the artifact thus evaluation and refinement go hand in hand. In our study, evaluation and refinement continued to occur at different stages of the study. However, we did not reevaluate the last output due to the limitations stated in chapter 7 section 7.2.3. Evaluation is important in most studies because it draws insights into the study and it gives a basis for improvement/refinement for the study.

10.3 Future work

In this thesis, we have come up with a new method Grounded System Dynamics (GSD) to improve SD model conceptualization. In chapter 6, we defined the GSD steps. These steps were evaluated in chapter 7 using experimental (where the researcher guided participants in using the method) and analytical evaluation methods (structured walkthroughs involving ORM and SD practitioners). Basing on the feedback we received from these evaluations, we came up with chapter 8. In this chapter we introduced a decomposition mechanism to help system analysts think and analyze at the most convenient level of abstraction. By introducing this mechanism more steps were added to the GSD method. These steps are described under GSD guidelines in chapter 9. The limitation to this study however, is that we did not apply the refined GSD steps (GSD guidelines) to a case study where the researcher is not the facilitator of the modeling session(s) or a field study. For further work therefore we recommend that the GSD method as a whole be applied to a case study or field study. In so doing, the applicability of the GSD method, described in chapter 9 subsection 9.2.1 will be evaluated and validated.

Secondly, an extension of the ORM tool to capture the decomposition mechanism is worth implementing. This implementation will allow modelers represent object behaviors, view and understand the conceptual representation of a scenario before transforming it into an SFD model. Furthermore, in chapter 7 we noted that information in a single Causal Loop Diagram cannot be captured in a single database and in chapter 9 we presented guidelines for tool support. As a direction for future work, we recommend that the
developed tool connects to different data sources (databases), various levels of abstraction can be seen (decomposition mechanism), and the ORM-SFD model can be viewed on a common platform. In real life, this tool may look like a ‘dash board’ were decision makers are able to view the effect of change as per available data sources hence improved validity of the simulation results.

Finally, in chapter 8, we limited our study to objects (tangible) but we know that in a system dynamics model there exists variables that are not tangible e.g. exogenous variables and soft variables. These variables influence other variables in an SD model, for further work, the researcher(s) may look at the behavior of such variables and how best they can be represented.
Appendices
Appendix A. Relations Questionnaire

A questionnaire for identifying the relations between ORM and SD method elements

Use the given study instrument as a guide to answering the given questions. Please tick and give reason(s) explaining why you think the chosen element is most appropriate to the given question.

1. Which of the following Object-Role Model element(s) is most similar to a System Dynamics stock?

   □ Value type  □ Object type  □ Object  □ Subtype  □ Supertype

   Fact types
   a) unary (one role)  b) binary (two roles)  c) Ternary fact type (three roles)

   • Give reason(s) why you think the selected ORM element is most similar to an SD stock?

2. Which of the following Object-Role Model element is most similar to a System Dynamics flow?

   □ Value type  □ Object type  □ Object  □ Subtype  □ Supertype

   Fact types
   a) unary (one role)  b) binary (two roles)  c) Ternary fact type (three roles)

   • Give reason(s) why you think the selected ORM element is most similar to an SD flow?

3. There are two SD elements that are classified within a converter (parameter). These include; a constant and an auxiliary variable.

   a. Which of the following Object-Role Model element is most similar to a System Dynamics constant?

      □ Value type  □ Object type  □ Object  □ Subtype  □ Supertype

      Fact types
      a) unary (one role)  b) binary (two roles)  c) Ternary fact type (three roles)

      • Give reason(s) why you think the selected ORM element is most similar to an SD constant?
b. Which of the following Object-Role Model element is most similar to a System Dynamics auxiliary variable?

☐ Value type  ☐ Object type  ☐ Object  ☐ Subtype  ☐ Super type

*Fact types*
   a) unary (one role)  b) binary (two roles)  c) Ternary fact type (three roles)

*Constraints*
   a) Mandatory role  b) Uniqueness constraints  c) subset  d) Equality

• Give reason(s) why you think the selected ORM element is most similar to an SD auxiliary variable?

---------------------------------------------------------------

4. Which of the following Object-Role Model element is most similar to a System Dynamics information link (connector)?

☐ Value type  ☐ Object type  ☐ Object  ☐ role connector

*Fact types*
   a) unary (one role)  b) binary (two roles)  c) Ternary fact type (three roles)

*Constraints*
   a) Mandatory role  b) Uniqueness constraints  c) subset  d) Equality

• Give reason(s) why you think the selected ORM element is most similar to an SD information link (connector)?

---------------------------------------------------------------

5. Which of the following Object-Role Model element is most similar to a System Dynamics sector?

☐ Value type  ☐ Object type  ☐ Object  ☐ Subtype  ☐ Super type

*Fact types*
   a) unary (one role)  b) binary (two roles)  c) Ternary fact type (three roles)

• Give reason(s) why you think the selected ORM element is most similar to an SD sector?

---------------------------------------------------------------

Thank you very much for taking part in this exercise. I pray that you once more participate in my upcoming GSD (transforming and ORM model into and SD model) procedure evaluation study.

************************END**************************
Appendix B: Walkthrough Instrument for ORM practitioner

Grounded System Dynamics Validation Experiment (For ORM Practitioners)

The Main aim of this validation experiment is to allow us assess the effectiveness of the GSD procedure. Participants are expected to critically comment on each step and be ready to backup (support) their answers.

GSD is a combination of two methods, System Dynamics (SD) and Object-Role Modeling (ORM). This procedure has two sections and each of these sections has five short steps. These steps guide a modeler on how to move from an SD causal loop diagram model to an ORM model and from an ORM model to an SD stock and flow diagram model. Participants are requested to take part only in one section of the GSD procedure. ORM experts take part in the transformation of the Causal Loop Diagram (CLD) to an ORM model, whereas System Dynamics experts take part in the transformation of the ORM model into an SD stock and flow diagram section.

CLDs are visual representation of dynamic influences with inter-relationships amongst a collection of SD variables. They are used in system dynamics modeling to qualitatively capture structures and interactions of feedback loops and to brainstorm on a given problem which enables them understand how changes manifest in a problem domain.

Materials to be used during the walkthroughs

1. Paper
2. Pen
3. Audio Recorder
4. ORM to CLD transformation steps
5. Running example.

Overall Aim of the study

The overall aim of this study is to underpin SD with a domain modeling method ORM. Domain modeling methods help identify relationships among entities within the scope of the problem domain and provide a structural view of the domain (we use ORM as an example of the domain modeling method). Underpinning of SD with a domain method enables us improve SD model conceptualization, reduce on SD model ambiguity, and enable transformation and reuse of information. By conducting these walkthroughs we are evaluating and validating the GSD steps.
Walkthrough Instrument for ORM practitioner

Process
Participants are provided with a running example (Case) and CLD-ORM transformation steps before the walkthrough sessions. We use the running example to evaluate and critic the steps there after the practitioner is provided an evaluation form comprising of five questions which the researchers use in their final evaluations.

From a causal loop diagram to an ORM model we have the following steps:
1. Identify object(s) for each CLD variable.
2. Collect and group similar objects into one object type.
3. Identify and connect roles played by objects in a particular object type
   a. With other object types
   b. With the same object type
4. Merge similar object types.
5. Add all constraints and mandatory roles.

<table>
<thead>
<tr>
<th>Num</th>
<th>Steps</th>
<th>Guiding question</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identify object(s) for each CLD variable.</td>
<td>Who are the main role player(s) in a given CLD variable? Who plays a role in this CLD variable?</td>
<td>Role player is a terminology used by ORM modelers but we use it here in its normal English way because this study is for both ORM modelers and non ORM modelers.</td>
</tr>
<tr>
<td>2</td>
<td>Collect and group (Classify) similar objects.</td>
<td>Here each classification makes an object type</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Identify and connect roles played by objects in a particular object type a. To other object types b. To objects in the same object type</td>
<td>What role do objects in object type (A) play in object type (A)? What roles do objects in object type (A) play in object type {A, A₁, A₂, A₃...Aₙ}?</td>
<td>Where: {A,A₁,A₂,A₃...Aₙ} are object types identified in CLD model. Here we are trying to identify roles played by objects in the same object type and in other object types</td>
</tr>
<tr>
<td>4</td>
<td>Merge similar object types.</td>
<td>Here we are putting all models to get a complete model</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Add all constraints and mandatory roles.</td>
<td>As a last step, all constraints are added and verbalizations derived.</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Transformation of a CLD into an ORM model guiding steps

In this validation experiment, we use a Causal Loop Diagram derived from the Intrapartum process in Ugandan hospitals to work as an example in the application phase. This case is given to practitioners together with the steps prior to the session date to allow them have a view of what we will be working with in the walkthrough sessions.
Causal Loop Diagram for Intrapartum in Ugandan Hospitals
Evaluation of Causal Loop Diagram (CLD) to Object-Role Modeling (ORM) Transformation Steps

1. How relevant are the Causal Loop Diagram transformation steps to ORM?
   □ Excellent □ Acceptable □ Needs Improvement
   Comments:____________________________________________________________________________

2. Do the given steps give enough information (guide) on how to derive an ORM model from a CLD?
   □ Excellent □ Acceptable □ Needs Improvement
   Comments:____________________________________________________________________________

3. Do the steps provide any higher levels of thinking?
   □ Excellent □ Acceptable □ Needs Improvement
   Comments:____________________________________________________________________________

4. Overall how would you rank this CLD to ORM transformation steps?
   □ Excellent □ Acceptable □ Needs Improvement
   Comments:____________________________________________________________________________

5. Are the steps given easy to understand or interpret?
   □ Yes □ No
   If NO what do you think should be added to improve the given steps?
   Comments:____________________________________________________________________________
Appendix C: Walkthrough Instrument for SD practitioner

**Grounded System Dynamics Validation Experiment (For SD Practitioners)**

The main aim of this validation experiment is for us to evaluate the GSD procedural steps. Participants are requested to critically comment on each step and be ready to back up (support) their answers.

GSD is a procedure that combines two methods, System Dynamics (SD) and Object-Role Modeling (ORM). This procedure comprises of two parts and each of these parts has five short steps. These steps guide a modeler on how to move from an SD causal loop diagram model to an ORM model and from an ORM model to an SD stock and flow diagram (SFD). Participants are requested to take part only in one part of the GSD procedure. SD experts are requested to take part in the transformation of the ORM model into a stock and flow diagram, whereas ORM experts take part in the transformation of a Causal Loop Diagram (CLD) into an ORM model.

ORM is a fact-oriented approach for modeling information and querying the information content of a business domain at a conceptual level. Fact-orientation means that it includes both types and instances in its models; types are called "fact types", instances "facts". Including the instance level is crucial in linking concepts with advanced SD modeling.

**Materials to be used during the walkthroughs**

1. Paper
2. Pen
3. Audio Recorder
4. ORM to SFD transformation steps
5. Running example.

**Overall Aim of the study**

The overall aim of this study is to underpin SD with a domain modeling method ORM. Domain modeling methods help identify relationships among entities within the scope of the problem domain and provide a structural view of the domain (we use ORM as an example of the domain modeling method). Underpinning of SD with a domain method enables us improve SD model conceptualization, reduce on SD model ambiguity, and enable transformation and reuse of information. By conducting these walkthroughs we are evaluating and validating the GSD steps.
Participants are provided with a running example, a summary table of the mappings and CLD-ORM transformation steps before the walkthrough sessions. We use the running example to evaluate and critic the steps there after the practitioner is provided an evaluation form comprising of five questions which the researchers use for their final evaluations.

**Steps from an ORM Model to an SD model**

1. Identify all possible stocks.
2. Identify all relevant flows connecting to each stock.
3. Identify all possible converters.
4. Identify all possible connectors (information links) and link them to converters, flows and stocks they relate with.
5. Identify and create sectors.

<table>
<thead>
<tr>
<th>#</th>
<th>Step</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identify all possible stocks.</td>
<td>We map a stock to a unary fact type. There exist four types of stocks (conveyor, reservoir, oven and queue). These are determined though constraints attached to the unary fact type.</td>
</tr>
<tr>
<td>2</td>
<td>Identify all relevant flows connecting to each stock</td>
<td>We map a flow to an object type. But the direction of flow and mechanism of populating an object type is not stated in ORM because ORM is a static method.</td>
</tr>
<tr>
<td>3</td>
<td>Identify all possible converters</td>
<td>We map converters to fact types with more than one role. Therefore all fact types with more than one role are turned into converters and there predicate names are concatenated with the object type name to make them more explicit.</td>
</tr>
<tr>
<td>4</td>
<td>Identify all possible connectors (information links) and link them to converters, flows and stocks they relate with.</td>
<td>There is no clear way of adding information links but we use the verbalizations.</td>
</tr>
<tr>
<td>5</td>
<td>Identify and create sectors</td>
<td>We map sectors to Object types with all roles connecting to that object type</td>
</tr>
</tbody>
</table>

Table 1: Transformation steps from an ORM model into an SFD model

In this validation experiment we will use an ORM example model that we used in one of our publications. This model looks at the "paper submission process". This model is given to practitioners together with the steps prior to the session date to allow them have a view of what we will be working with in the walkthrough sessions.

**Running example**

- A person (author) writes intent of submission. This can be in the form of an abstract.
- Then the content (text of the paper) is submitted, whereby the paper becomes a submitted paper.
- Each submitted paper receives a classification.
- Each submitted paper is reviewed.
- Some submitted papers are accepted and some are rejected.
- For each accepted paper new content is submitted, which makes the paper a published paper that is added to the publications.
These statements are represented in the ORM diagram below.

The symbol and Constraint in this figure can be verbalized as follows:

C1 (Exclusive): For each Paper, at most one of the following holds: that Paper is rejected or that Paper is accepted.
C2: If some Paper is published then that Paper is accepted.
C3 (mandatory): each paper is submitted by at least one academic.
C4 (mandatory): each paper is written by at least one academic.
C5: If some Academic submits some Paper then that Academic writes that paper.
C6 (Exclusive): For each Academic and Paper, at most one of the following holds: that Academic writes that Paper; that Academic is assigned to review that Paper.

Step 1: Identify all possible stocks.
Stocks are containers (reservoirs) containing quantities describing the state of the system. Stocks are mapped to unary fact types (one role) because they both contain "things" or act as containers.

A unary fact type is a single role. A role denote the way entity types participate in a fact type.

Step 2: Identify all relevant flows connecting to each stock.
Flows can be imagined as pipelines with a valve that controls the rate of accumulation to and from the stocks. Flows are mapped to an object type because they both connect different stocks (SD) and Fact types (ORM).
An object type is a collection of objects with similar properties and through an object type connections to other object types are achieved.

**Step 3: Identify all possible converters.**
Converters either represent fixed quantities (constants) or represent variable quantities (Auxiliaries). Auxiliary variables are informational concepts bearing an independent meaning (add new information) which contains information of equations or values that can be applied to stocks, flows, and other converters in the model. Converters are mapped to fact types with more than one role (binary, ternary and quaternary) because they add new information to the model and combine two or more variables consistently. If fact types have fixed objects, then that converter is a constant else it is an auxiliary. Converters vary because they directly or indirectly depend on stocks.

**Step 4: Identify all possible connectors (information links).**
Information from converters and flows is shared through connectors (information links). Two types of connectors exist, the action connectors depicted as solid wires and information connectors depicted as dashed wires. These connectors are immaterial and connect inputs to decision function of a rate. The underpinned meaning to these connectors is that information about the value at the start of the connector influences information at the arrow tip of that connector. Connectors can feed information into or out of flows and converters but only extract information out of stocks.
Information links are not mapped to any ORM element but we use ORM constraints plus verbalizations to identify connectors and the direction of the SD connector. In ORM constraints are placed in between fact types (binary, ternary, etc), these constraint play an important role in helping the modeler identify the direction of the connector.

Step 5: Create sectors.
A sector is a grouping of elements with related functionally in a model. Sectors are subsystems or subcomponents within a system. They hold/handle all decisions, stocks, information about a particular element or area and contain different information used in an information system.

We map SD sectors to ORM object types plus their attached roles. This is because ORM conceptual object types act as semantic "glue" and an ORM model is a network of allied object types and relationship types. This means that roles and object types when put together, they make up a complete ORM model. Therefore, object types plus their 'glued' roles are similar to SD sectors because when both are 'glued' or put together they make up a complete model.
Evaluation form for ORM to Stock and Flow Transformation Steps

1. How relevant are the ORM to SFD transformation steps?
   □ Excellent □ Acceptable □ Needs Improvement
   Comments:_________________________________________________________________________

2. Do the given steps give enough information (guide) on how to derive an SFD model from an ORM model?
   □ Excellent □ Acceptable □ Needs Improvement
   Comments:_________________________________________________________________________

3. Overall how would you rank these ORM to SFD transformation steps?
   □ Excellent □ Acceptable □ Needs Improvement
   Comments:_________________________________________________________________________

4. Are the steps given easy to understand or interpret?
   □ Yes □ No
   If NO what do you think should be added to improve the given steps?
   Comments:_________________________________________________________________________

5. Do you think that Object role modeling is a good starting point for the construction of SFD?
   _________________________________________________________________________________
   _________________________________________________________________________________
   ________________________________________________________________________________
   ________________________________________________________________________________
Appendix D: SFD Equations

\[ \text{Admitted\_patients}(t) = \text{Admitted\_patients}(t - \text{dt}) + (Z - K) \times \text{dt} \]
\[ \text{INIT Admitted\_patients} = 5, 2, 4, 1, 8 \]
\[ \text{TRANSIT TIME} = \text{varies} \]
\[ \text{INFLOW LIMIT} = \text{INF} \]
\[ \text{CAPACITY} = \text{INF} \]
\[ \text{INFLOWS}: \]
\[ \bowtie \quad Z = \text{CONVEYOR OUTFLOW} \]
\[ \text{TRANSIT TIME} = \text{Treatment\_duration} + \text{Examination\_duration} \]
\[ \text{OUTFLOWS}: \]
\[ \bowtie \quad K = \text{CONVEYOR OUTFLOW} \]
\[ \text{TRANSIT TIME} = \text{IF}(\text{Dilation} \leq 4) \text{THEN}(\text{Monitored\_patient}); \text{ELSE}(0) \]

\[ \text{Births}(t) = \text{Births}(t - \text{dt}) + (M - N) \times \text{dt} \]
\[ \text{INIT Births} = 0 \]
\[ \text{INFLOWS}: \]
\[ \bowtie \quad M = \text{CONVEYOR OUTFLOW} \]
\[ \text{TRANSIT TIME} = \text{IF}(\text{Dilation} \leq 8) \text{THEN}(\text{Births}); \text{ELSE}(0) \]
\[ \text{OUTFLOWS}: \]
\[ \bowtie \quad N = \text{DELAY}(\text{Births}, 0) / \text{Treatment\_duration} \]

\[ \text{Discharges}(t) = \text{Discharges}(t - \text{dt}) + (N + Y) \times \text{dt} \]
\[ \text{INIT Discharges} = 0 \]
\[ \text{INFLOWS}: \]
\[ \bowtie \quad N = \text{DELAY}(\text{Births}, 0) / \text{Treatment\_duration} \]
\[ \bowtie \quad Y = \text{LEAKAGE OUTFLOW} \]
\[ \text{LEAKAGE FRACTION} = X / \text{Treatment\_duration} \]
\[ \text{NO-LEAK ZONE} = 100\% \]

\[ \text{Enter\_hospital}(t) = \text{Enter\_hospital}(t - \text{dt}) + (A - X) \times \text{dt} \]
\[ \text{INIT Enter\_hospital} = 2, 1, 0, 2, 3 \]
\[ \text{TRANSIT TIME} = \text{varies} \]
\[ \text{INFLOW LIMIT} = \text{INF} \]
\[ \text{CAPACITY} = \text{INF} \]
\[ \text{INFLOWS}: \]
\[ \bowtie \quad A = \text{IF}(\text{MOD}(\text{time}, 2) = 1) \text{then} 2 \text{else} 0 \]
\[ \text{OUTFLOWS}: \]
\[ \bowtie \quad X = \text{CONVEYOR OUTFLOW} \]
\[ \text{TRANSIT TIME} = \]
Examination(t) = Examination(t - dt) + (X - Z - Y) * dt
INIT Examination = 3,1,3,1,3
TRANSIT TIME = varies
INFLOW LIMIT = INF
CAPACITY = INF
INFLOWS:
  - X = CONVEYOR OUTFLOW
    TRANSIT TIME =
OUTFLOWS:
  - Z = CONVEYOR OUTFLOW
    TRANSIT TIME = Treatment_duration + Examination__duration
  - Y = LEAKAGE OUTFLOW
    LEAKAGE FRACTION = X/Treatment_duration
    NO-LEAK ZONE = 100%

Monitored_patient(t) = Monitored_patient(t - dt) + (K - M) * dt
INIT Monitored_patient = 1,2,3,1,2
TRANSIT TIME = varies
INFLOW LIMIT = INF
CAPACITY = INF
INFLOWS:
  - K = CONVEYOR OUTFLOW
    TRANSIT TIME = IF(Dilation <= 4) THEN(Monitored_patient) ELSE(0)
OUTFLOWS:
  - M = CONVEYOR OUTFLOW
    TRANSIT TIME = IF(Dilation <= 8) THEN(Births) ELSE(0)

- Dilation = MAX(ROUND(NORMAL(10,8,4)),0)
- Examination__capacity = 3
- Examination__duration = MAX(30)
- Medical_persons__on_duty = MIN(2)
- Treatment_duration = MIN(24)
Bibliography


Combining methods can be viewed as a way of improving model development processes and their outcome. In this thesis, we have studied how to combine system dynamics with a domain modeling method (Object-Role Modeling (ORM)). As a domain modeling method, we have used ORM. ORM is excellent at conceptualizing and explicitly representing processes in a problem domain while System Dynamics (SD) is good at analyzing and optimizing processes. Although these two methods are applied and used differently, they are both very important and play key roles in articulating and analyzing problems. SD offers a systematic approach of qualitative and quantitative analysis but it shows a lack of instruments for discovering and expressing precise, language-based concepts in domains. At the same time, the field of domain modeling has long since focused on deriving models from natural expressions.

In this thesis, we started by presenting an introductory chapter, in this chapter we defined a research problem, objectives, justification for the study and research approach. After presenting the introductory chapter, we reviewed existing literature to identify the knowledge gap, and acknowledge the work of others. Next, we used various scholarly works to come up with requirements. These requirements are of two types; general requirements and specific requirements. The requirements defined under general requirements are broad for combining methods in general and those under specific requirements pertain to our research study in particular. Through identifying these requirements, we were able to break the main research problem into smaller manageable problems.

After defining the requirements, we presented constructs and underlying principles for both ORM and SD. Through identifying these constructs, we were able to denote what can be modeled by each method (ORM and SD). By using the defined constructs and underlying principles, we were able to identify relationships between ORM and SD constructs. These defined relationships were evaluated using a focus group discussion. Basing on the identified relations, we came up with steps for the Grounding System Dynamics (GSD) method. These steps are in two phases. In the first phase, we defined steps for transforming a Causal Loop Diagram (CLD) into an ORM model, variable names were made more explicit hence, the system under study is better understood. In the second phase, we defined steps for transforming an ORM model into a Stock and Flow Diagram (SFD). In so doing we were able to improve the stock and flow diagram conceptualization.

Having defined the steps for grounded system dynamics, we evaluated the defined steps using analytical and experimental evaluation methods. Under experimental evalu-
tions, we used focus group discussions and a case study. Under analytical evaluations, we used walkthroughs. In these walkthroughs we used an instrument (concept definitions, examples and questions) as guide in our study and were conducted in pair (the researcher and participant).

During the evaluations, we received a lot of insightful feedback among which was to transform a SFD from a ‘picture’ into a working model. To take care of this remark, we came up with chapter 8. In this chapter, we study the object behavior(s) and extend ORM to show how various levels of abstraction can be seen using the decomposition mechanism. In so doing, we derived more GSD steps which we presented in chapter 9. To attain concrete results, we used examples and an ORM model that was derived from a case “Intrapartum process in Ugandan Hospitals”. On completion, we made suggestions for organizational and tool support. To sum it all up, we drew conclusions for the study showing contributions for the study and future works.
Samenvatting

Door bestaande methoden te combineren kan het proces van modelontwikkeling, en de resultaten daarvan, verbeterd worden. In dit proefschrift hebben we onderzocht hoe System Dynamics kan worden gecombineerd met Object-Role Modeling (ORM), een methode voor domein-modellering. ORM is sterk op het gebied van conceptualiseren en het expliciet representeren van processen in een probleem domein, terwijl de kracht van System Dynamics (SD) juist ligt in het analyseren en optimaliseren van processen. Ook al werden deze twee methoden op verschillende wijze toegepast, ze zijn beide erg belangrijk en spelen een sleutelrol bij het verwoorden en analyseren van problemen. SD biedt een systematische aanpak voor kwalitatieve en kwantitatieve analyse, maar het toont een gebrek aan instrumenten voor het ontdekken en uitdrukken van nauwkeurige, op taal gebaseerde concepten in applicatiedomeinen. Daarentegen heeft het gebied van domein modelleren een lange traditie in het afleiden van modellen uit expressies in natuurlijke taal.

We beginnen dit proefschrift met een inleidend hoofdstuk, waarin we probleemstelling, doelstellingen en rechtvaardiging van het onderzoek formuleren en de onderzoeksaanpak beschrijven. Na dit inleidende hoofdstuk, geven we een overzicht van de bestaande literatuur om de kennisloof vast te stellen, en het werk van anderen te erkennen. Vervolgens maken we gebruik van verscheidene wetenschappelijke methoden om te komen tot de vereisten. We onderscheiden twee typen vereisten: algemene en specifieke vereisten. De algemene vereisten geven brede eisen om methoden in het algemeen te combineren, terwijl de specifieke eisen betrekking hebben op ons onderzoek in het bijzonder. Door deze vereisten te identificeren, waren we in staat het onderzoeksthema op te splitsen in kleinere, hanteerbare problemen.

Na het vaststellen van de vereisten, presenteren we de constructen en de onderliggende principes van zowel ORM als SD. Door deze constructen te identificeren, kunnen we aangeven wat door de beide methodes (ORM en SD) gemodelleerd kan worden. Daardoor we gebruik maken van gedefinieerde constructen en onderliggende principes kunnen we de relatie tussen ORM en SD constructen identificeren. Deze gedefinieerde relaties werden gevalueerd door middel van een focusgroep-discussie. Vanuit de gevonden relaties kwamen we tot de stappen voor de Grounded System Dynamic (GSD) methode. Deze stappen zijn verdeeld in twee fasen. In de eerste fase definieren we de stappen om een Causal Loop Diagram (CLD) om te zetten in een ORM-model. Daarbij worden de namen van variabelen nader gexpliciteerd zodat het beschouwde systeem beter wordt begrepen. In de tweede fase definieren we de stappen voor de transformatie van een ORM model naar
een Stock and Flow Diagram (SFD). Daarbij konden we de conceptualisatie van het Stock and Flow Diagram verbeteren.

Nadat de stappen voor Grounded System Dynamics zijn vastgesteld, evalueren we deze stappen met analytische en experimentele evaluatiemethoden. Bij de experimentele evaluaties gebruikten we focusgroep-discussies en een case study. Bij de analytische evaluaties hebben we gebruik gemaakt van walkthroughs. Bij walkthroughs gebruikten we een instrument (concept definities, voorbeelden en vragen) als leidraad en deze werden uitgevoerd in paren (de onderzoeker en deelnemer).

Tijdens de evaluaties kregen we veel inzichtelijke feedback waaronder een voorstel om een SFD vanuit een 'plaatje' te transformeren naar een werkend model. Op deze wens komen we terug in hoofdstuk 8. In dat hoofdstuk bestuderen we het objectgedrag(ingen), en breiden we ORM zodanig uit dat verschillende niveaus van abstractie zichtbaar zijn met behulp van het decompositie mechanisme. Daarbij kwamen we tot meer GSD stappen, die we in hoofdstuk 9 presenteren. Als demonstratie hiervan geven we voor de casus “Het intrapartum proces in de Oegandese ziekenhuizen” het uitgebreide ORM model. Tenslotte geven we suggesties voor organisatorische en tool-matige ondersteuning. We geven een overzicht van de bijdragen van deze studie en geven richtlijnen voor vervolgonderzoek.
Curriculum Vitae

Tulinayo Fiona was born in Kampala, Uganda. She graduated with a bachelors in industrial and fine arts at Makerere University. Thereafter, she worked as a graphics designer for Monitor Publications Uganda. In 2005, she started pursuing her masters degree and on 28th January, 2008 she graduated with a master in computer science. In March 2008 she started pursuing her PhD.
SIKS Dissertatiereeks

1998
1998-1 Johan van den Akker (CWI) DEGAS - An Active, Temporal Database of Autonomous Objects
1998-2 Floris Wiesman (UM) Information Retrieval by Graphically Browsing Meta-Information
1998-3 Ans Steuten (TUD) A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective
1998-4 Dennis Breuker (UM) Memory versus Search in Games
1998-5 E.W.Oskamp (RUL) Computerondersteuning bij Straftoemeting

1999
1999-1 Mark Sloof (VU) Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products
1999-2 Rob Potharst (EUR) Classification using decision trees and neural nets
1999-3 Don Beal (UM) The Nature of Minimax Search
1999-4 Jacques Penders (UM) The practical Art of Moving Physical Objects
1999-5 Aldo de Moor (KUB) Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems
1999-6 Niels J.E. Wijngaards (VU) Re-design of compositional systems
1999-7 David Spelt (UT) Verification support for object database design
1999-8 Jacques H.J. Lening (UM) Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation

2000
2000-1 Frank Niessink (VU) Perspectives on Improving Software Maintenance
2000-2 Koen Holtman (TUE) Prototyping of CMS Storage Management
2000-3 Carolien M.T. Metselaar (UvA) Sociaal-organisatorische gevolgen van kennistechnologie; een proceeshenadering en actorperspectief
2000-4 Geert de Haan (VU) ETAG, A Formal Model of Competence Knowledge for User Interface Design
2000-5 Ruud van der Pol (UM) Knowledge-based Query Formulation in Information Retrieval
2000-6 Rogier van Eijk (UU) Programming Languages for Agent Communication
2000-7 Niels Peak (UU) Decision-theoretic Planning of Clinical Patient Management
2000-8 Veerle Coupé (EUR) Sensitivity Analysis of Decision-Theoretic Networks
2000-9 Florian Waas (CWI) Principles of Probabilistic Query Optimization
2000-11 Jonas Karlsson (CWI) Scalable Distributed Data Structures for Database Management

2001
2001-1 Silja Renooij (UU) Qualitative Approaches to Quantifying Probabilistic Networks
2001-2 Koen Hindriks (UU) Agent Programming Languages: Programming with Mental Models
2001-3 Maarten van Someren (UvA) Learning as problem solving
2001-4 Evgenii Smirnov (UM) Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets
2001-5 Jacco van Ossebracht (VU) Processing Structured Hypermedia: A Matter of Style
2001-6 Martijn van Welie (VU) Task-based User Interface Design
2001-7 Bastiaan Schonhage (VU) Diva: Architectural Perspectives on Information Visualization
2001-8 Pascal van Eck (VU) Compositional Semantic Structure for Multi-Agent Systems Dynamics

2001-9 Pieter Jan’t Hoorn (RUL) Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes

2001-10 Maarten Siekhuys (UvA) Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VUA) Knowledge Management: The Role of Mental Models in Business Systems Design

2002

2002-01 Nico Lassing (VU) Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwol (UT) Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT) Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castelo Valdueza (UU) The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU) The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL) Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Broncz (CWI) Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

2002-08 Jaap Gordijn (VU) Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-09 Wim Jan van den Heuvel (KUB) Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM) Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU) Agent Based Modelling of Dynamics:

2002-12 Albrecht Schmidt (UvA) Processing XML in Database Systems

2002-13 Hongling Wu (TUE) A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wiebe de Vries (UvT) Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems

2002-15 Rik Eshuis (UT) Semantics and Verification of UML Activity Diagrams for Workflow Modelling


2002-17 Stefan Manegold (UVA) Understanding, Modeling, and Improving Main-Memory Database Performance

2003

2003-01 Heiner Stuckenschmidt (VU) Ontology-Based Information Sharing in Weakly Structured Environments

2003-02 Jan Broersen (VU) Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuermie (TUD) man-Computer Interaction and Presence in Virtual Reality Exposure Therapy

2003-04 Milan Petkovic (UT) Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UVA) Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT) Development and specification of virtual environments

2003-07 Machtel Jansen (UVA) Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM) Repair Based Scheduling

2003-09 Rens Kortmann (UM) The resolution of visually guided behaviour

2003-10 Andreas Lincke (UvT) Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT) Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT) Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM) Nonce Hostem - Searching with Opponent Models

2003-14 Sijin Hoppenbrouwers (KUN) Freezing Language: Conceptualisation Processes across ICT-Supported Organisations

2003-15 Mathijs de Weerdt (TUD) Plan Merging in Multi-Agent Systems

2003-16 Menno Windhouwer (CWI) Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses


2003-18 Levente Kocsis (UM) Learning Search Decisions

2004

2004-01 Virginia Dignum (UU) A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UT) Monitoring Multi-party Contracts for E-business
<table>
<thead>
<tr>
<th>Year</th>
<th>Title</th>
<th>Author(s)</th>
<th>Institutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004-03</td>
<td>A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving</td>
<td>Perry Groot (VU)</td>
<td></td>
</tr>
<tr>
<td>2004-04</td>
<td>Organizational Principles for Multi-Agent Architectures</td>
<td>Chris van Aart (UVA)</td>
<td></td>
</tr>
<tr>
<td>2004-05</td>
<td>Knowledge discovery and monotonicity</td>
<td>Viara Popova (EUR)</td>
<td></td>
</tr>
<tr>
<td>2004-06</td>
<td>The Evaluation of Business Process Modeling Techniques</td>
<td>Bart-Jan Hommes (TUD)</td>
<td></td>
</tr>
<tr>
<td>2004-07</td>
<td>Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes</td>
<td>Elise Boltjes (UM)</td>
<td></td>
</tr>
<tr>
<td>2004-08</td>
<td>Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise</td>
<td>Joop Verbeek (UM)</td>
<td></td>
</tr>
<tr>
<td>2004-09</td>
<td>For the Sake of the Argument; explorations into argument-based reasoning</td>
<td>Martin Caminada (VU)</td>
<td></td>
</tr>
<tr>
<td>2004-10</td>
<td>Knowledge-rich indexing of learning-objects</td>
<td>Suzanne Kabel (UVA)</td>
<td></td>
</tr>
<tr>
<td>2004-12</td>
<td>Creating emotions and facial expressions for embodied agents</td>
<td>The Duy Bui (UM)</td>
<td></td>
</tr>
<tr>
<td>2004-13</td>
<td>Using Multiple Models of Reality: On Agents who Know how to Play</td>
<td>Wojciech Janmog (UT)</td>
<td></td>
</tr>
<tr>
<td>2004-14</td>
<td>Logic in Conflict. Logical Explorations in Strategic Equilibrium</td>
<td>Paul Harrenstein (UU)</td>
<td></td>
</tr>
<tr>
<td>2004-15</td>
<td>Multi-Relational Data Mining</td>
<td>Arno Knobbe (UU)</td>
<td></td>
</tr>
<tr>
<td>2004-16</td>
<td>Using Generative Probabilistic Models for multimedia retrieval</td>
<td>Federico Divina (VU)</td>
<td></td>
</tr>
<tr>
<td>2004-17</td>
<td>Informed Search in Complex Games</td>
<td>Mark Winands (UM)</td>
<td></td>
</tr>
<tr>
<td>2004-18</td>
<td>Supporting the Construction of Qualitative Knowledge Models</td>
<td>Vania Bessa Machado (UvA)</td>
<td></td>
</tr>
<tr>
<td>2004-19</td>
<td>Using generative probabilistic models for multimedia retrieval</td>
<td>Thijs Westerveld (UT)</td>
<td></td>
</tr>
<tr>
<td>2004-20</td>
<td>Learning from Design: facilitating multidisciplinary design teams</td>
<td>Madelon Evers (Nyenrode)</td>
<td></td>
</tr>
<tr>
<td>2005-01</td>
<td>Methodological Aspects of Designing Induction-Based Applications</td>
<td>Floor Verdenius (UVA)</td>
<td></td>
</tr>
<tr>
<td>2005-02</td>
<td>AI techniques for the game of Go</td>
<td>Erik van der Werf (UM)</td>
<td></td>
</tr>
<tr>
<td>2005-03</td>
<td>A Pragmatic Approach to the Conceptualisation of Language</td>
<td>Franc Grooten (RUN)</td>
<td></td>
</tr>
<tr>
<td>2005-04</td>
<td>Towards Database Support for Moving Object data</td>
<td>Nirvana Meratnia (UT)</td>
<td></td>
</tr>
<tr>
<td>2005-05</td>
<td>Two-Level Probabilistic Grammars for Natural Language Parsing</td>
<td>Gabriel Infante-Lopez (UVA)</td>
<td></td>
</tr>
<tr>
<td>2005-06</td>
<td>Adaptive Game AI</td>
<td>Pieter Sprock (UM)</td>
<td></td>
</tr>
<tr>
<td>2005-07</td>
<td>Hypermedia Presentation Generation for Semantic Web Information Systems</td>
<td>Flavius Frasineac (TUE)</td>
<td></td>
</tr>
<tr>
<td>2005-08</td>
<td>A Model-driven Approach for Building Distributed Ontology-based Web Applications</td>
<td>Richard Vdovjak (TUE)</td>
<td></td>
</tr>
<tr>
<td>2005-09</td>
<td>Storage, Querying and Inference for Semantic Web Languages</td>
<td>Jeen Broekstra (VU)</td>
<td></td>
</tr>
<tr>
<td>2005-10</td>
<td>Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments</td>
<td>Anders Bouwer (UVA)</td>
<td></td>
</tr>
<tr>
<td>2005-11</td>
<td>Agent Based Matchmaking and Clustering - A Decentralized Approach to Search</td>
<td>Elib Ogston (VU)</td>
<td></td>
</tr>
<tr>
<td>2005-12</td>
<td>Distributed Simulation in Industry</td>
<td>Csaba Boer (EUR)</td>
<td></td>
</tr>
<tr>
<td>2005-13</td>
<td>Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen</td>
<td>Fred Hamburg (UL)</td>
<td></td>
</tr>
<tr>
<td>2005-14</td>
<td>Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics</td>
<td>Borys Omelayenko (VU)</td>
<td></td>
</tr>
<tr>
<td>2005-15</td>
<td>Analysis of the Dynamics of Cognitive Processes</td>
<td>Tibor Bosse (VU)</td>
<td></td>
</tr>
<tr>
<td>2005-16</td>
<td>Usability of XMI. Query Languages</td>
<td>Joris Graaumanns (UU)</td>
<td></td>
</tr>
<tr>
<td>2005-17</td>
<td>Software Specification Based on Re-usable Business Components</td>
<td>Boris Shishkov (TUD)</td>
<td></td>
</tr>
<tr>
<td>2005-18</td>
<td>Test-selection strategies for probabilistic networks</td>
<td>Danielle Sent (UU)</td>
<td></td>
</tr>
<tr>
<td>2005-19</td>
<td>Situated Representation</td>
<td>Michel van Dartel (UM)</td>
<td></td>
</tr>
<tr>
<td>2005-20</td>
<td>Cyber Consumer Law, State of the Art and Perspectives</td>
<td>Cristina Costeau (UL)</td>
<td></td>
</tr>
<tr>
<td>2005-21</td>
<td>Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics</td>
<td>Wijnand Derks (UT)</td>
<td></td>
</tr>
<tr>
<td>2006-01</td>
<td>Foundations of B2B Electronic Contracting</td>
<td>Samuil Angelov (TUE)</td>
<td></td>
</tr>
<tr>
<td>2006-02</td>
<td>Contextual issues in the design and use of information technology in organizations</td>
<td>Cristina Chialiti (VIU)</td>
<td></td>
</tr>
<tr>
<td>2006-03</td>
<td>The role of metacognitive skills in learning to solve problems</td>
<td>Noor Christoph (UVA)</td>
<td></td>
</tr>
<tr>
<td>2006-04</td>
<td>Building Web Service Ontologies</td>
<td>Marta Sabou (UVA)</td>
<td></td>
</tr>
<tr>
<td>2006-05</td>
<td>Validation Techniques for Object-Oriented Proof Outlines</td>
<td>Cecs Plerik (UU)</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>Name (Institute)</td>
<td>Title</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Marko Smiljanic (UT)</td>
<td>XML schema matching - balancing efficiency and effectiveness by means of clustering</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Eelco Herder (UT)</td>
<td>Forward, Back and Home Again - Analyzing User Behavior on the</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Mohamed Wahdan (UM)</td>
<td>Automatic Formulation of the Auditor’s Opinion</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Ronny Siebes (VU)</td>
<td>Semantic Routing in Peer-to-Peer Systems</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Joeri van Ruth (UT)</td>
<td>Flattening Queries over Nested Data Types</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Bert Bongers (VU)</td>
<td>Interactivation - Towards an ecology of people, our technological environment, and the arts</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Henk-Jan Lebbink (UU)</td>
<td>Dialogue and Decision Games for Information Exchanging Agents</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Johan Hoorn (VU)</td>
<td>Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Rainer Malik (UU)</td>
<td>CONAN: Text Mining in the Biomedical Domain</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Carsten Riggelsen (UU)</td>
<td>Approximation Methods for Efficient Learning of Bayesian Networks</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Stacey Nagata (UU)</td>
<td>User Assistance for Multitasking with Interruptions on a Mobile Device</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Valentin Zhizhkon (UVA)</td>
<td>Graph transformation for Natural Language Processing</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Birna van Riemsdijk (UU)</td>
<td>Cognitive Agent Programming: A Semantic Approach</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Marina Velikova (UvT)</td>
<td>Monotone models for prediction in data mining</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Bas van Gils (RUN)</td>
<td>Aptness on the Web</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Paul de Vrieze (RUN)</td>
<td>Fundaments of Adaptive Personalisation</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Laura Hollink (VU)</td>
<td>Development of Cognitive Model for Navigating on the Web</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Laura Hollink (VU)</td>
<td>Semantic Annotation for Retrieval of Visual Resources</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Madalina Drugan (UU)</td>
<td>Conditional log-likelihood MML and Evolutionary MCMC</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Vojkan Mihajlovic (UT)</td>
<td>Score Region Algebra: A Flexible Framework for Structured Information Retrieval</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Stefano Bocconi (CW1)</td>
<td>Vox Populi: generating video documentaries from semantically annotated media repositories</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Borkur Sigurbjornsson (UVA)</td>
<td>Focused Information Access using XML Element Retrieval</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Kees Leune (UvT)</td>
<td>Access Control and Service-Oriented Architectures</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Wouter Teepe (RUG)</td>
<td>Reconciling Information Exchange and Confidentiality: A Formal Approach</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Peter Mika (VU)</td>
<td>Social Networks and the Semantic Web</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Gilad Mishne (UVA)</td>
<td>Applied Text Analytics for Blogs</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Natasa Jovanovic (UT)</td>
<td>To Whom It May Concern – Addressee Identification in Face-to-Face Meetings</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Mark Hoogendoorn (VU)</td>
<td>Modeling of Change in Multi-Agent Organizations</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>David Mobach (VU)</td>
<td>Agent-Based Mediated Service Negotiation</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Natalia Stash (TUE)</td>
<td>Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Marcel van Gerven (RUN)</td>
<td>Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Rutger Rienks (UT)</td>
<td>Meetings in Smart Environments; Implications of Progressing Technology</td>
<td></td>
</tr>
</tbody>
</table>
| 2007 | Niels Bergboer (UM) | Context-Based Image Analysis  
2007.15 Joyca Lacroix (UM) NIM: a Situated Computational Memory Model |
| 2007 | Davide Grossi (UU) | Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi agent Systems |
| 2007 | Theodore Charitos (UU) | Reasoning with Dynamic Networks in Practice |
| 2007 | Bart Orriens (UvT) | On the development an management of adaptive business collaborations |
| 2007 | David Levy (UM) | Intimate relationships with artificial partners |
| 2007 | Slinger Jansen (UU) | Customer Configuration Updating in a Software Supply Network |
2007-21 Karianne Vermaas (UU) Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005

2007-22 Zlatko Zlatev (UT) Goal-oriented design of value and process models from patterns

2007-23 Peter Barna (TUE) Specification of Application Logic in Web Information Systems

2007-24 Georgina Ramirez Camps (CWI) Structural Features in XML Retrieval


2007-22 Zlatko Zlatev (UT) Goal-oriented design of value and process models from patterns

2007-23 Peter Barna (TUE) Specification of Application Logic in Web Information Systems

2007-24 Georgina Ramirez Camps (CWI) Structural Features in XML Retrieval


2008-01 Katalin Boer-Sorban (EUR) Agent-Based Simulation of Financial Markets: A modular, continuous-time approach


2008-03 Vera Hollink (UVA) Optimizing hierarchical menus: a usage-based approach

2008-04 Ander de Keijzer (UT) Management of Uncertain Data - towards unattended integration

2008-05 Bela Mutschler (UT) Modeling and simulating causal dependencies on process-aware information systems from a cost perspective

2008-06 Arjen Hommersom (RUN) On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective

2008-07 Peter van Rosmalen (OU) Supporting the tutor in the design and support of adaptive e-learning

2008-08 Janneke Bolt (UU) Bayesian Networks: Aspects of Approximate Inference

2008-09 Christof van Nimwegen (UU) The paradox of the guided user: assistance can be counter-effective

2008-10 Wouter Bosma (UT) Discourse oriented summarization

2008-11 Vera Kartseva (VU) Designing Controls for Network Organizations: A Value-Based Approach

2008-12 Jooszef Farkas (RUN) A Semiotically Oriented Cognitive Model of Knowledge Representation

2008-13 Caerina Carraciolo (UVA) Topic Driven Access to Scientific Handbooks

2008-14 Arthur van Bunningen (UT) Context-Aware Querying; Better Answers with Less Effort


2008-16 Henriette van Vugt (VU) Embodied agents from a user’s perspective

2008-17 Martin Op’t Land (TUD) Applying Architecture and Ontology to the Splitting and Allying of Enterprises

2008-18 Guido de Croon (UM) Adaptive Active Vision

2008-19 Henning Rode (UT) From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search

2008-20 Rex Arendsen (UVA) Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven

2008-21 Kristian Balog (UVA) jPeople Search in the Enterprise

2008-22 Henk Koning (UU) Communication of IT-Architecture

2008-23 Stefan Visscher (UU) Bayesian network models for the management of ventilator-associated pneumonia

2008-24 Zharke Aleksovyvl (VU) Using background knowledge in ontology matching


2008-26 Marijn Huijbregts (UT) Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled

2008-27 Hubert Vogten (OU) Design and Implementation Strategies for IMS Learning Design

2008-28 Ildiko Flesch (RUN) On the Use of Independence Relations in Bayesian Networks

2008-29 Dennis Reidman (UT) Annotations and Subjective Machines – Of Annotators, Embodied Agents, Users, and Other Humans

2008-30 Wouter van Atteveldt (VU) Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content

2008-31 Loes Braun (UM) Pro-Active Medical Information Retrieval


2008-33 Frank Terpstra (UVA) Scientific Workflow Design; theoretical and practical issues

2008-34 Jeroen de Knijf (UU) Studies in Frequent Tree Mining

2008-35 Ben Torben Nielsen (UvT) Dendritic morphologies: function shapes structure

2009-01 Rasa Jurgelenaité (RUN) Symmetric Causal Independence Models
2009-02 Willem Robert van Hage (VU) Evaluating Ontology-Alignment Techniques

2009-03 Hans Stol (UvT) A Framework for Evidence-based Policy Making Using IT

2009-04 Josephine Nabukenya (RUN) Improving the Quality of Organisational Policy Making using Collaboration Engineering

2009-05 Sietse Overbeek (RUN) Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality

2009-06 Muhammad Subianto (UU) Understanding Classification

2009-07 Ronald Poppe (UT) Discriminative Vision-Based Recovery and Recognition of Human Motion

2009-08 Volker Nannen (VU) Evolutionary Agent-Based Policy Analysis in Dynamic Environments

2009-09 Benjamin Kangwa (RUN) Design, Discovery and Construction of Service-oriented Systems

2009-10 Jan Wielemaker (UvT) Logic programming for knowledge-intensive interactive applications

2009-11 Alexander Boer (UvA) Legal Theory, Sources of Law & the Semantic Web

2009-12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin) Operating Guidelines for Services

2009-13 Steven de Jong (UM) Fairness in Multi-Agent Systems

2009-14 Maksym Korotkiy (VU) From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)

2009-15 Rinke Hoekstra (UvA) Ontology Representation - Design Patterns and Ontologies that Make Sense

2009-16 Fritz Reul (UvT) New Architectures in Computer Chess

2009-17 Laurens van der Maaten (UvT) Feature Extraction from Visual Data

2009-18 Fabian Groffen (CWI) Armada, An Evolving Database System

2009-19 Valentijn Robu (CWI) Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets

2009-20 Bob van der Yecht (UU) Adjustable Autonomy: Controlling Influences on Decision Making

2009-21 Stijn Vanderloooy (UM) Ranking and Reliable Classification

2009-22 Pavel Serdyukov (UT) Search For Expertise: Going beyond direct evidence

2009-23 Peter Hofgesang (VU) Modelling Web Usage in a Changing Environment

2009-24 Annerieke Heuvelink (VU) Cognitive Models for Training Simulations

2009-25 Alex van Ballegooij (CWI) RAM: Array Database Management through Relational Mapping

2009-26 Fernando Koch (UU) An Agent-Based Model for the Development of Intelligent Mobile Services

2009-27 Christian Glahn (OU) Contextual Support of social Engagement and Reflection on the Web

2009-28 Sander Evers (UT) Sensor Data Management with Probabilistic Models

2009-29 Stanislav Pokraev (UT) Model-Driven Semantic Integration of Service-Oriented Applications

2009-30 Marcin Zukowski (CWI) Balancing vectorized query execution with bandwidth-optimized storage

2009-31 Sofiya Katrenko (UvA) A Closer Look at Learning Relations from Text

2009-32 Rik Farenhorst (VU) and Remco de Boer (VU) Architectural Knowledge Management: Supporting Architects and Auditors

2009-33 Khiet Truong (UT) How Does Real Affect Affect Affect Recognition In Speech

2009-34 Inge van de Weerd (UU) Advancing in Software Product Management: An Incremental Method Engineering Approach

2009-35 Wouter Koelewijn (UL) Privacy en Politiegegevens: Over geautomatiseerde normatieve informatieuitwisseling

2009-36 Marco Kalz (OU) Placement Support for Learners in Learning Networks

2009-37 Hendrik Drachsler (OU) Navigation Support for Learners in Informal Learning Networks

2009-38 Riina Vuorikari (OU) Tags and self-organisation: a metadata ecology for learning resources in a multilingual context


2009-40 Stephan Raaijmakers (UvT) Multinomial Language Learning: Investigations into the Geometry of Language

2009-41 Igor Berezhnyy (UvT) Digital Analysis of Paintings

2009-42 Toine Bogers (UvT) Recommender Systems for Social Bookmarking

2009-43 Virginia Nunes Leal Franquera (UT) Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients

2009-44 Roberto Santana Tapia (UT) Assessing Business-IT Alignment in Networked Organizations

2009-45 Jilles Vreeken (UU) Making Pattern Mining Useful
SIKS Dissertatiereeks

2009-46 Loredana Afanasiev (UvA) Querying XML: Benchmarks and Recursion

2010

2010-01 Matthijs van Leeuwen (UU) Patterns that Matter
2010-02 Ingo Wassink (UT) Work flows in Life Science
2010-04 Olga Kulyk (UT) Do You Know What I Know? Situational Awareness of Co-located Teams in Multi-display Environments
2010-05 Claudia Hauff (UT) Predicting the Effectiveness of Queries and Retrieval Systems
2010-06 Sander Bakkes (UvT) Rapid Adaptation of Video Game AI
2010-07 Wim Fikkert (UT) Gesture interaction at a Distance
2010-08 Krzysztof Siewicz (UL) Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments
2010-09 Hugo Kielman (UL) Politiële gegevensverwerking en Privacy, Naar een effectieve waarborging
2010-10 Rebecca Ong (UL) Mobile Communication and Protection of Children
2010-11 Adriaan Ter Mors (TUD) The world according to MARP: Multi-Agent Route Planning
2010-12 Susan van den Braak (UU) Sensemaking software for crime analysis
2010-13 Gianluigi Folino (RUN) High Performance Data Mining using Bio-inspired techniques
2010-14 Sander van Splunter (VU) Automated Web Service Reconfiguration
2010-15 Lianne Bodenstaff (UT) Managing Dependency Relations in Inter-Organizational Models
2010-16 Sicco Verwer (TUD) Efficient Identification of Timed Automata, theory and practice
2010-17 Spyros Kotoulas (VU) Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications
2010-18 Charlotte Gerritsen (VU) Caught in the Act: Investigating Crime by Agent-Based Simulation
2010-19 Henriette Cramer (UvA) People’s Responses to Autonomous and Adaptive Systems
2010-20 Ivo Swartjes (UT) Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative

2010-21 Harold van Heerde (UT) Privacy-aware data management by means of data degradation
2010-22 Michiel Hildebrand (CWI) End-user Support for Access to Heterogeneous Linked Data
2010-23 Bas Steunebrink (UU) The Logical Structure of Emotions
2010-24 Dmytro Tykhonov (TUD) Designing Generic and Efficient Negotiation Strategies
2010-25 Zulfugar Ali Memon (VU) Modelling Human-Awareness for Ambient Agents: A Human Min-dreading Perspective
2010-26 Ying Zhang (CWI) XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines
2010-27 Marten Voulon (UL) Automatisch contracteren
2010-28 Arne Koopman (UU) Characteristic Relational Patterns
2010-29 Stratos Idreos (CWI) Database Cracking: Towards Auto-tuning Database Kernels
2010-30 Marike van Erp (UvT) Accessing Natural History Discoveries in data cleaning, structuring, and retrieval
2010-31 Victor de Boer (UVA) Ontology Enrichment from Heterogeneous Sources on the Web
2010-32 Marcel Hiel (UvT) An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems
2010-33 Robin Aly (UT) Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval
2010-34 Teduh Dirghaayu (UT) Interaction Design in Service Compositions
2010-35 Dolf Trieschnigg (UT) Proof of Concept: Concept-based Biomedical Information Retrieval
2010-36 Jose Janssen (OU) Paving the Way for Life-long Learning; Facilitating competence development through a learning path specification
2010-37 Niels Lohmann (TUE) Correctness of services and their composition
2010-38 Dirk Fahland (TUE) From Scenarios to components
2010-39 Ghaeanfar Farooq Siddiqui (VU) Integrative modeling of emotions in virtual agents
2010-40 Mark van Assen (VU) Converting and Integrating Vocabularies for the Semantic Web
2010-41 Guillaume Chaslot (UM) Monte-Carlo Tree Search
2010-42 Sytren de Kinderen (VU) Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach
2010-43 Peter van Kranenburg (UU) A Computational Approach to Content-Based Retrieval of Folk Song Melodies
2010-44 Pieter Bellekens (TUE) An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain

2010-45 Vasilios Andriopoulos (UvT) A theory and model for the evolution of software services

2010-46 Vincent Pijpers (VU) e3alignment: Exploring Inter-Organizational Business-ICT Alignment

2010-47 Chen Li (UT) Mining Process Model Variants: Challenges, Techniques, Examples

2010-48 Withdrawn

2010-49 Jahn-Takeshi Saito (UM) Solving difficult game positions

2010-50 Bouke Huurnink (UVA) Search in Audiovisual Broadcast Archives

2010-51 Alia Khairia Amin (CWI) Understanding and supporting information seeking tasks in multiple sources

2010-52 Peter-Paul van Maanen (VU) Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention

2010-53 Edgar Meij (UVA) Combining Concepts and Language Models for Information Access

2011-01 Botond Cseke (RUN) Variational Algorithms for Bayesian Inference in Latent Gaussian Models

2011-02 Nick Tinnemeier (UU) Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language

2011-03 Jan Martijn van der Werf (TUE) Compositional Design and Verification of Component-Based Information Systems

2011-04 Hado Philip van Hasselt (UU) Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms

2011-05 Bas van de Ruudt (VU) Enterprise Architecture Coming of Age – Increasing the Performance of an Emerging Discipline

2011-06 Yiwen Wang (TUE) Semantically-Enhanced Recommendations in Cultural Heritage

2011-07 Yujia Cao (UT) Multimodal Information Presentation for High Load Human Computer Interaction

2011-08 Nieske Vergunst (UU) BDI-based Generation of Robust Task-Oriented Dialogues

2011-09 Tim de Jong (OU) Contextualised Mobile Media for Learning

2011-10 Bart Bogaert (UvT) Cloud Content Contention

2011-11 Dhaval Vyas (UT) Designing for Awareness: An Experience-focused HCI Perspective

2011-12 Carmen Bratosin (TUE) Grid Architecture for Distributed Process Mining

2011-13 Xiaoou Mao (UvT) Airport under Control; Multiagent Scheduling for Airport Ground Handling

2011-14 Milan Lovric (EUR) Behavioral Finance and Agent-Based Artificial Markets


2011-16 Maarten Schadd (UM) Selective Search in Games of Different Complexity

2011-17 Jiyin He (UVA) Exploring Topic Structure: Coherence, Diversity and Relatedness

2011-18 Mark Ponsen (UM) Strategic Decision-Making in complex games

2011-19 Ellen Rusman (OU) The Mind’s Eye on Personal Profiles

2011-20 Qing Gu (VU) Guiding service-oriented software engineering – A view-based approach

2011-21 Linda Terlouw (TUD) Modularization and Specification of Service-Oriented Systems

2011-22 Junte Zhang (UVA) System Evaluation of Archival Description and Access

2011-23 Wouter Weerkamp (UVA) Finding People and their Utterances in Social Media

2011-24 Herwin van Welbergen (UT) Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior

2011-25 Syed Waqar ul Quamin Jaffry (VU) Analysis and Validation of Models for Trust Dynamics

2011-26 Matthijs Aart Pontier (VU) Virtual Agents for Human Communication – Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots

2011-27 Aniel Bhulai (VU) Dynamic website optimization through autonomous management of design patterns


2011-29 Faisal Kamiran (TUE) Discrimination-aware Classification

2011-30 Egon van den Brook (UT) Affective Signal Processing (ASP): Unraveling the mystery of emotions

2011-31 Ludo Waltman (EUR) Computational and Game-Theoretic Approaches for Modeling Bounded Rationality

2011-32 Nees-Jan van Eck (EUR) Methodological Advances in Bibliometric Mapping of Science

2011-33 Tom van der Weide (UU) Arguing to Motivate Decisions
2011-34 Paolo Turrini (UU) Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations

2011-35 Maaike Harbers (UU) Explaining Agent Behavior in Virtual Training

2011-36 Erik van der Spek (UU) Experiments in serious game design: a cognitive approach

2011-37 Adriana Burliau (RUN) Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference

2011-38 Nyree Lammens (UM) Bee-inspired Distributed Optimization

2011-39 Joost Westra (UU) Organizing Adaptation using Agents in Serious Games

2011-40 Viktor Clerc (VU) Architectural Knowledge Management in Global Software Development

2011-41 Luan Ibraimi (UT) Cryptographically Enforced Distributed Data Access Control

2011-42 Michal Sindlar (UU) Explaining Behavior through Mental State Attribution

2011-43 Henk van der Schuur (UU) Process Improvement through Software Operation Knowledge

2011-44 Boris Reuderink (UT) Robust Brain-Computer Interfaces

2011-45 Herman Stehouwer (UvT) Statistical Language Models for Alternative Sequence Selection

2012-01 Terry Kakeeto (UvT) Relationship Marketing for SMEs in Uganda

2012-02 Muhammad Umair(VU) Adaptivity, emotion, and Rationality in Human and Ambient Agent Models

2012-03 Adam Vanya (VU) Supporting Architecture Evolution by Mining Software Repositories

2012-04 Jurriaan Sauer (UU) Development of Content Management System-based Web Applications

2012-05 Marijn Plomp (UU) Maturing Interorganisational Information Systems

2012-06 Wolfgang Reinhardt (OU) Awareness Support for Knowledge Workers in Research Networks

2012-07 Rianne van Lambalgen (VU) When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions

2012-08 Gerben de Vries (UVA) Kernel Methods for Vessel Trajectories

2012-09 Ricardo Neisse (UT) Trust and Privacy Management Support for Context-Aware Service Platforms

2012-10 David Smits (TUE) Towards a Generic Distributed Adaptive Hypermedia Environment

2012-11 J.C.B. Rantham Prabhakara (TUE) Process Mining in the Large: Preprocessing, Discovery, and Diagnostics

2012-12 Kees van der Sluijs (TUE) Model Driven Design and Data Integration in Semantic Web Information Systems

2012-13 Suleman Shahid (UvT) Fun and Face: Exploring non-verbal expressions of emotion during playful interactions

2012-14 Evgeny Knutov(TUE) Generic Adaptation Framework for Unifying Adaptive Web-based Systems

2012-15 Natalie van der Wald (VU) Social Agents: Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.

2012-16 Fienke Both (VU) Helping people by understanding them - Ambient Agents supporting task execution and depression treatment

2012-17 Amal Elgammal (UvT) Towards a Comprehensive Framework for Business Process Compliance

2012-18 Eltjo Poort (VU) Improving Solution Architecting Practices


2012-20 Ali Bahramisharif (RUN) Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing

2012-21 Roberto Cornacchia (TUD) Querying Sparse Matrices for Information Retrieval

2012-22 Thijs Vis (UvT) Intelligence, politie en veiligheidsdienst: verenigbare grootheden?

2012-23 Christian Muehl (UT) Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction

2012-24 Laurens van der Werff (UT) Evaluation of Noisy Transcripts for Spoken Document Retrieval

2012-25 Silja Eckartz (UT) Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application

2012-26 Emile de Maat (UVA) Making Sense of Legal Text
2012-27 Hayrettin Gürkük (UT) Mind the Sheep! User Experience Evaluation and Brain-Computer Interface Games

2012-28 Nancy Pascall (UvT) Engendering Technology Empowering Women

2012-29 Almer Tigelaar (UT) Peer-to-Peer Information Retrieval

2012-30 Alina Pommeranz (TUD) Designing Human-Centered Systems for Reflective Decision Making

2012-31 Emily Bagarukayo (RUN) A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure

2012-32 Wieteke Visser (TUD) Qualitative multi-criteria preference representation and reasoning

2012-33 Rory Sie (OUN) Coalitions in Cooperation Networks (COCOON)

2012-34 Pavlo Jan-cur (RUN) Evolutionary analysis in PPI networks and applications

2012-35 Evert Haasdijk (VU) Never Too Old To Learn - Online Evolution of Controllers in Swarm- and Modular Robotics

2012-36 Denis Ssebugwawo (RUN) Analysis and Evaluation of Collaborative Modeling Processes


2012-38 Evert Haasdijk (VU) Never Too Old To Learn - Online Evolution of Controllers in Swarm- and Modular Robotics

2012-39 Hassan Fatemi (UT) Risk-aware design of value and coordination networks

2012-40 Agus Gunawan (UvT) Information Access for SMEs in Indonesia

2012-41 Sebastian Kelle (OU) Game Design Patterns for Learning

2012-42 Dominique Verpoorten (OU) Reflection Amplifiers in self-regulated Learning

2012-44 Anna Tordai (VU) On Combining Alignment Techniques

2012-45 Benedikt Kratz (UvT) A Model and Language for Business-aware Transactions

2012-46 Simon Carter (UVA) Exploration and Exploitation of Multilingual Data for Statistical Machine Translation

2012-47 Manos Tsagkias (UVA) Mining Social Media: Tracking Content and Predicting Behavior

2012-48 Jorn Bakker (TUE) Handling Abrupt Changes in Evolving Time-series Data

2012-49 Michael Kaeser (UM) Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions

2012-50 Steven van Kervel (TUD) Ontology driven Enterprise Information Systems Engineering

2012-51 Jeroen de Jong (TUD) Heuristics in Dynamic Scheduling; a practical framework with a case study in elevator dispatching

2013-01 Viorel Milea (EUR) News Analytics for Financial Decision Support

2013-02 Erietta Liarou (CWI) MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing

2013-03 Szymon Klarmann (VU) Reasoning with Contexts in Description Logics

2013-04 Chetan Yadati (TUD) Coordinating autonomous planning and scheduling

2013-05 Duke Pumarcja (UT) Groupware Requirements Evolutions Patterns

2013-06 Romulo Goncalves (CWI) The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience

2013-07 Giel van Lankveld (UvT) Quantifying Individual Player Differences

2013-08 Robbert-Jan Merk (VU) Making enemies: cognitive modeling for opponent agents in fighter pilot simulators

2013-09 Fabio Gori (RUN) Metagenomic Data Analysis: Computational Methods and Applications

2013-10 Jeewanie Jayasinghe Arachchige (UvT) A Unified Modeling Framework for Service Design

2013-11 Evangelos Pournaras (TUD) Multilevel Reconfigurable Self-organization in Overlay Services

2013-12 Marian Razavian (VU) Knowledge-driven Migration to Services

2013-13 Mohammad Safiri (UT) Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly

2013-14 Jafar Tanha (UVA) Ensemble Approaches to Semi-Supervised Learning Learning

2013-15 Daniel Hennes (UM) Multiagent Learning - Dynamic Games and Applications

2013-16 Ivin Kok (UU) Exploring the practical benefits of argumentation in multi-agent deliberation

2013-17 Koern Kok (VU) The PowerMatcher: Smart Coordination for the Smart Electricity Grid

2013-18 Jeroen Janssens (UvT) “Outlier Selection and One-Class Classification”

2013-19 Renze Steenhuizen (TUD) Coordinated Multi-Agent Planning and Scheduling
2013-20 Katja Hofmann (UvA) Fast and Reliable Online Learning to Rank for Information Retrieval
2013-21 Sander Wubben (UvT) Text-to-text generation by monolingual machine translation
2013-22 Tom Claassen (RUN) Causal Discovery and Logic
2013-23 Patricio de Alencar Silva (UvT) Value Activity Monitoring
2013-24 Haitham Bou Ammar (UM) Automated Transfer in Reinforcement Learning
2013-26 Alireza Zarghami (UT) Architectural Support for Dynamic Homecare Service Provisioning
2013-27 Mohammad Huq (UT) Inference-based Framework Managing Data Provenance
2013-28 Frans van der Sluis (UT) When Complexity becomes Interesting: An Inquiry into the Information eXperience
2013-29 Iwan de Kok (UT) Listening Heads
2013-31 Dinh Khoa Nguyen (UvT) Blueprint Model and Language for Engineering Cloud Applications
2013-32 Kamakshi Rajagopal (OUN) Networking For Learning; The role of Networking in a Lifelong Learner’s Professional Development
2013-33 Qi Gao (TUD) User Modeling and Personalization in the Microblogging Sphere
2013-34 Kien Tjin-Kam-Jet (UT) Distributed Deep Web Search
2013-36 Than Lam Hoang (TUE) Pattern Mining in Data Streams
2013-37 Dirk Börner (OUN) Ambient Learning Displays
2013-38 Eelco den Heijer (VU) Autonomous Evolutionary Art
2013-40 Pim Nijssen (UM) Monte-Carlo Tree Search for Multi-Player Games
2013-41 Jochem Liem (UvA) Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning
2013-43 Mare Bron (UVA) Exploration and Contextualization through Interaction and Concepts
2014
2014-01 Nicola Barile (UU) Studies in Learning Monotone Models from Data
2014-02 Fiona P. Tulinayo (RUN) Combining System Dynamics with A Domain Modeling Method
In this study we put a domain modeling method Object-Role Modeling (ORM) to work in the context of the creation of System Dynamics (SD) models. The art of SD modeling lies in determining and representing the feedback processes and other elements that determine the dynamics of the system (typically, a process in an organization). SD however, shows a lack of instruments for discovering and expressing precise, language-based concepts in domains. At the same time, the field of conceptual modeling has long since focused on deriving models from natural expressions. We therefore turn to ORM as a prime example of this school of thought to integrate its strong natural language based modeling approach into the creation of SD models. This was inspired by an observed lack of concept-level modeling power in SD. We set out mainly to augment SD modeling by first laying down a sound foundation of domain concepts by means of fact-based ORM modeling. This eventually enables us to link the ORM model to the SD stock-flow diagram.
In this study we put a domain modeling method Object-Role Modeling (ORM) to work in the context of the creation of System Dynamics (SD) models. The art of SD modeling lies in determining and representing the feedback processes and other elements that determine the dynamics of the system (typically, a process in an organization). SD however, shows a lack of instruments for discovering and expressing precise, language-based concepts in domains. At the same time, the field of conceptual modeling has long since focused on deriving models from natural expressions. We therefore turn to ORM as a prime example of this school of thought to integrate its strong natural language based modeling approach into the creation of SD models. This was inspired by an observed lack of concept-level modeling power in SD. We set out mainly to augment SD modeling by first laying down a sound foundation of domain concepts by means of fact-based ORM modeling. This eventually enables us to link the ORM model to the SD stock-flow diagram.