TOWARDS A DEEPER UNDERSTANDING OF MODELLING IN INFORMATION SYSTEM ANALYSIS

by

Wei Sun
M.Sc., The University of British Columbia, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Business Administration)

THE UNIVERSITY OF BRITISH COLUMBIA
(Vancouver)

August 2015

© Wei Sun, 2015
Abstract

This dissertation defines modelling products in information system (IS) analysis in terms of IS analysis models. Motivated by better understanding these models, it identifies and solves some relevant issues, by method of studying relevant concepts and their connections, in the literature. This method is qualitative and requires our subjective interpretations. However, the utility of the research results have been extensively demonstrated. This dissertation consists of three studies.

The first study focuses on concepts applicable to all types of IS analysis models. Previous views of modelling adopted in IS analysis suffer from being either generic for not tailored to the specific context of IS analysis, or limited in not applicable to the entire scope of IS analysis. To remediate, we propose a new view of modelling in IS analysis that is neither generic nor limited. We call it the generalized view. It can be used to distinguish IS analysis models from other models, to compare different types of IS analysis models, and to explain the representation and interpretation of IS analysis models. We apply it to compare two IS analysis models of an example project.

The second study focuses on the connection among different types of IS analysis models. They were studied in different areas in isolation with inconsistent terminologies, hindering inter-area collaboration. To remediate, we propose another new view of
modelling in IS analysis, to connect different types of IS analysis models in a comprehensive picture. We call it the consolidated view. It can be used to help stakeholders to determine whether two IS analysis modelling methods are substitutes or complements, to compare modelling methods that are substitutes, and to coordinate the use of multiple modelling methods that are complements. We apply it to evaluate and compare Gaia and Tropos, two agent-oriented modelling methods.

The third study suggests how to apply the two views together to analyze modelling methods. It also discusses the differences and relationships between them in such applications. We apply them to analyze the GRAPPLE modelling method. Through the analysis, we test the generalizability of the two views, and identify and solve issues of GRAPPLE.
Preface

This dissertation consists of three studies that are joint work of me and my supervisors. In all three studies, the research programs have been jointly identified and designed by me and my supervisors. Essentially, this dissertation has made a number of arguments to advance the understanding of modelling in information system analysis, and has supported these arguments by analyzing the existing literature. I have conducted most work in making these arguments and supporting them by analyzing the existing literature. These work have been significantly influenced by the feedback from my supervisors.

A preliminary version of some parts of the first and second studies (in Chapters 2 and 3) has been presented in a workshop (Sun, Wand, & Woo, 2011). The first study (in Chapter 2) is currently under final preparation for submission to a research journal for peer review (Sun, Wand, & Woo, 2017). The second study (in Chapter 3) has been presented in a workshop (Sun, Wand, & Woo, 2015), and submitted to a research journal for peer review (Sun, Wand, & Woo, 2016). We plan to present the third study (in Chapter 4) in a research conference in the future.

The research in this dissertation does not involve experimentation with human or animal subjects and does not require any ethics approval.
# Table of Contents

Abstract....................................................................................................................................................... ii

Preface ........................................................................................................................................................ iv

Table of Contents........................................................................................................................................ v

List of Tables .................................................................................................................................................. ix

List of Figures ................................................................................................................................................ x

Acknowledgements ........................................................................................................................................ xi

Dedication ...................................................................................................................................................... xii

1. Introduction............................................................................................................................................... 1
   1.1. BACKGROUND OF DISSERTATION ......................................................................................... 1
   1.2. MOTIVATIONS OF DISSERTATION ...................................................................................... 3
       1.2.1. Motivations of Study 1 (Generalized View) ................................................................. 6
       1.2.2. Motivations of Study 2 (Consolidated View) ............................................................. 7
       1.2.3. Motivations of Study 3 (Joint Application) ................................................................. 8
   1.3. OUTLINE OF DISSERTATION ................................................................................................. 9
   1.4. REMARKS...................................................................................................................................... 9

2. Towards A Generalized View of Modelling in IS Analysis: Explicating the Common Characteristics of All IS Analysis Models .................................................................................. 11
   2.1. INTRODUCTION ...................................................................................................................... 11
       2.1.1. Motivation ..................................................................................................................... 11
       2.1.2. Overview of Method .................................................................................................... 14
       2.1.3. Outline of Chapter ........................................................................................................ 16
   2.2. RELATED WORK ......................................................................................................................... 17
       2.2.1. Related Work on Defining Modelling Products in IS Analysis .................................... 17
       2.2.2. Related Work on Comparing Modelling Products in IS Analysis .............................. 18
   2.3. CONCEPTUAL MODELS IN IS ANALYSIS VERSUS IN GENERIC CONTEXTS ...... 20
       2.3.1. State of the Art Definition of Modelling in IS Analysis ............................................... 21
### Table of Contents

2.3.2. The Generic Issues with the State of the Art ............................................. 21
2.3.3. Proposing A Remedy .................................................................................. 22
2.3.4. Demonstration of Utility: On Clarifying the Scope of Conceptual Modelling in IS Analysis ................................................................. 22
2.3.5. Section Summary and Analysis ................................................................. 24

2.4. **COMPARING DIFFERENT TYPES OF IS ANALYSIS MODELS** .................. 25

2.4.1. Searching for General Characteristics of IS Analysis Models in Their Definitions .................................................................................................................. 26
2.4.2. Demonstration of Utility: On Comparing Different Types of IS Analysis Models For Similarities and Differences ............................................................... 31
2.4.3. Section Summary and Analysis ................................................................. 32

2.5. **REPRESENTATION AND INTERPRETATION OF IS ANALYSIS MODELS** ...... 33

2.5.1. The Distinction Between Model and Script ............................................... 34
2.5.2. The Mappings Between Model and Script ............................................... 40
2.5.3. The Mappings Between Model and Domain ............................................ 42
2.5.4. Semantic Domain and Ontology ............................................................... 44
2.5.5. Section Summary and Analysis ................................................................. 45

2.6. **TOWARDS A GENERALIZED VIEW OF IS ANALYSIS MODELS** ........... 48

2.6.1. The Goals of the Generalized View .......................................................... 49
2.6.2. The Concepts in the Generalized View .................................................... 50
2.6.3. The Relationship Types in the Generalized View ...................................... 51
2.6.4. Visualizing the Generalized View ............................................................. 51
2.6.5. Defining the Elements of the Generalized View ....................................... 52

2.7. **AN EXAMPLE PROJECT** ......................................................................... 58

2.7.1. Background of the Example Project ......................................................... 58
2.7.2. Examples of Concepts in the Generalized View ........................................ 59
2.7.3. Summary of the Example Project .............................................................. 63

2.8. **CONCLUSION OF CHAPTER** ............................................................... 63

2.8.1. Summary ...................................................................................................... 63
2.8.2. Academic Contributions ............................................................................. 65

3. Towards A Consolidated View of Modelling in IS Analysis: Explicating the Connections among Different Types of IS Analysis Models ............................................. 67

3.1. **INTRODUCTION** ...................................................................................... 67

3.1.1. Motivation .................................................................................................... 67
3.1.2. Overview of Method ................................................................................... 68
3.1.3. Outline of Chapter ...................................................................................... 70

3.2. **RELATED WORK** ...................................................................................... 70

3.2.1. General Terminologies of Areas Related to IS Analysis ........................... 70
3.2.2. Evaluation Frameworks of Modelling Grammars and Methods ................ 71
3.2.3. Multi-Product Development Methods ....................................................... 73

3.3. **AREA-SPECIFIC PERSPECTIVES OF MODELLING IN SYSTEM ANALYSIS** .. 74

3.3.1. Method for Summarizing Area-Specific Perspectives ............................... 74
3.3.2. The Requirements Engineering Perspective .............................................. 79
3.3.3. The Conceptual Schema Modelling Perspective ........................................ 83
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.4.</td>
<td>The Domain Understanding Perspective</td>
<td>86</td>
</tr>
<tr>
<td>3.4.</td>
<td><strong>A CONSOLIDATED VIEW OF MODELLING IN IS ANALYSIS</strong></td>
<td>90</td>
</tr>
<tr>
<td>3.4.1.</td>
<td>Method for Building the Consolidated View</td>
<td>90</td>
</tr>
<tr>
<td>3.4.2.</td>
<td>Concepts in the Consolidated View</td>
<td>92</td>
</tr>
<tr>
<td>3.4.3.</td>
<td>Relationship Types in the Consolidated View</td>
<td>94</td>
</tr>
<tr>
<td>3.4.4.</td>
<td>A Visualization of the Consolidated View</td>
<td>95</td>
</tr>
<tr>
<td>3.4.5.</td>
<td>Definitions of the Consolidated View</td>
<td>96</td>
</tr>
<tr>
<td>3.4.6.</td>
<td>Comments about the Consolidated View</td>
<td>100</td>
</tr>
<tr>
<td>3.5.</td>
<td><strong>AN EXAMPLE PROJECT</strong></td>
<td>101</td>
</tr>
<tr>
<td>3.5.1.</td>
<td>Background of the Example Project</td>
<td>101</td>
</tr>
<tr>
<td>3.5.2.</td>
<td>Choosing IS Analysis Modelling Methods for the Example Project</td>
<td>102</td>
</tr>
<tr>
<td>3.5.3.</td>
<td>Modelling Products of the Example Project</td>
<td>105</td>
</tr>
<tr>
<td>3.5.4.</td>
<td>Applying the Consolidated View in Coordinating the Use of Multiple</td>
<td>116</td>
</tr>
<tr>
<td>3.6.</td>
<td><strong>AN EXAMPLE APPLICATION</strong></td>
<td>119</td>
</tr>
<tr>
<td>3.6.1.</td>
<td>Background: Gaia and Tropos</td>
<td>119</td>
</tr>
<tr>
<td>3.6.2.</td>
<td>Evaluation and Comparison</td>
<td>128</td>
</tr>
<tr>
<td>3.6.3.</td>
<td>Summary of the Example Application</td>
<td>129</td>
</tr>
<tr>
<td>3.7.</td>
<td><strong>CONCLUSION OF CHAPTER</strong></td>
<td>130</td>
</tr>
<tr>
<td>3.7.1.</td>
<td>Summary</td>
<td>130</td>
</tr>
<tr>
<td>3.7.2.</td>
<td>Academic Contributions and Applications</td>
<td>131</td>
</tr>
<tr>
<td>4.</td>
<td><strong>Joint Application: Analyzing the GRAPPLE Modelling Method</strong></td>
<td>135</td>
</tr>
<tr>
<td>4.1.</td>
<td><strong>INTRODUCTION</strong></td>
<td>135</td>
</tr>
<tr>
<td>4.1.1.</td>
<td>Background</td>
<td>135</td>
</tr>
<tr>
<td>4.1.2.</td>
<td>Motivation</td>
<td>136</td>
</tr>
<tr>
<td>4.1.3.</td>
<td>Overview of Analysis Method</td>
<td>137</td>
</tr>
<tr>
<td>4.1.4.</td>
<td>Choosing a Modelling Method for the Analysis</td>
<td>138</td>
</tr>
<tr>
<td>4.1.5.</td>
<td>Outline of Chapter</td>
<td>140</td>
</tr>
<tr>
<td>4.2.</td>
<td><strong>THE GRAPPLE MODELLING METHOD: AN OVERVIEW</strong></td>
<td>140</td>
</tr>
<tr>
<td>4.2.1.</td>
<td>GRAPPLE and the Example Project: An Overview</td>
<td>142</td>
</tr>
<tr>
<td>4.2.2.</td>
<td>Example IS Analysis Models of GRAPPLE</td>
<td>145</td>
</tr>
<tr>
<td>4.2.3.</td>
<td>Section Summary</td>
<td>150</td>
</tr>
<tr>
<td>4.3.</td>
<td><strong>ANALYZING GRAPPLE USING THE GENERALIZED VIEW</strong></td>
<td>151</td>
</tr>
<tr>
<td>4.3.1.</td>
<td>Applying Concepts in the “External Product” Category</td>
<td>151</td>
</tr>
<tr>
<td>4.3.2.</td>
<td>Applying Concepts in the “Mental Product” Category</td>
<td>152</td>
</tr>
<tr>
<td>4.3.3.</td>
<td>Applying Concepts in the “Context” Category</td>
<td>152</td>
</tr>
<tr>
<td>4.3.4.</td>
<td>Applying Concepts in the “Theory” Category</td>
<td>154</td>
</tr>
<tr>
<td>4.3.5.</td>
<td>Applying Concepts in the “Language” Category</td>
<td>154</td>
</tr>
<tr>
<td>4.3.6.</td>
<td>Section Summary</td>
<td>156</td>
</tr>
<tr>
<td>4.4.</td>
<td><strong>ANALYZING GRAPPLE USING THE CONSOLIDATED VIEW</strong></td>
<td>156</td>
</tr>
<tr>
<td>4.4.1.</td>
<td>Mapping IS Analysis Modelling Product Types Between GRAPPLE and the Consolidated View</td>
<td>157</td>
</tr>
<tr>
<td>4.4.2.</td>
<td>Evaluating GRAPPLE for Adequacy in Supporting Different Types of IS Analysis Models</td>
<td>160</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Definitions Applicable to Models and Modelling in IS Analysis</td>
</tr>
<tr>
<td>Table 2</td>
<td>A Comparison of Three Types of IS Analysis Models</td>
</tr>
<tr>
<td>Table 3</td>
<td>Applying Modelling Language Concepts: A Example Using the ER Model</td>
</tr>
<tr>
<td>Table 4</td>
<td>Categories of Concepts in the Generalized View</td>
</tr>
<tr>
<td>Table 5</td>
<td>Examples of Concepts in the “External Product” Category</td>
</tr>
<tr>
<td>Table 6</td>
<td>Examples of Concepts in the “Context” Category</td>
</tr>
<tr>
<td>Table 7</td>
<td>A Summary of Improvements on the General Understanding of IS Analysis Models</td>
</tr>
<tr>
<td>Table 8</td>
<td>Categories of Concepts and Relationship Types in the Consolidated View</td>
</tr>
<tr>
<td>Table 9</td>
<td>Concepts in the Consolidated View</td>
</tr>
<tr>
<td>Table 10</td>
<td>Relationship Types in the Consolidated View, Except Those in the “Conceptualization” Category</td>
</tr>
<tr>
<td>Table 12</td>
<td>Example Project – Summary of Modelling Products in IS Analysis</td>
</tr>
<tr>
<td>Table 13</td>
<td>Analysis Results of the IS Analysis Modelling Products in Tropos</td>
</tr>
<tr>
<td>Table 14</td>
<td>Evaluating and Comparing Gaia and Tropos Regarding Modelling Product Support</td>
</tr>
<tr>
<td>Table 15</td>
<td>A Summary of Examples IS Analysis Models of GRAPPLE</td>
</tr>
<tr>
<td>Table 16</td>
<td>Applying the Purpose and Focal Aspects Concepts to Analyze the Four Types of IS Analysis Modelling Products of GRAPPLE</td>
</tr>
<tr>
<td>Table 17</td>
<td>Mapping of IS Analysis Modelling Product Types Between the Consolidated View and GRAPPLE</td>
</tr>
<tr>
<td>Table 18</td>
<td>The Application of the Generalized and Consolidated Views in Analyzing GRAPPLE</td>
</tr>
<tr>
<td>Table 19</td>
<td>Model Types That Correspond to the Conceptual Schema in Three Modelling Methods</td>
</tr>
</tbody>
</table>
# List of Figures

**Figure 1. Issues with Existing Views of Modelling in IS Analysis** ................................................................. 4
**Figure 2. Concepts and Relationship Types – For Clarifying the Scope of Conceptual Models in IS Analysis** ... 25
**Figure 3. Concepts and Relationship Types – For Explicating the General Characteristics of IS Analysis Models** .................................................................................................................................................. 33
**Figure 4. Concepts and Relationship Types – For Explicating the Representation and Interpretation of IS Analysis Models** .............................................................................................................................................. 46
**Figure 5. The Generalized View of IS Analysis Models** ......................................................................................... 52
**Figure 6. The Requirements Engineering Perspective** .............................................................................................. 80
**Figure 7. The Conceptual Schema Modelling Perspective** ....................................................................................... 83
**Figure 8. The Domain Understanding Perspective** .................................................................................................. 88
**Figure 9. The Consolidated View of Modelling in IS Analysis** .............................................................................. 96
**Figure 10. Example Project – Domain Understanding** ............................................................................................ 106
**Figure 11. Example Project – Entity-Relationship Diagram for the Conceptual Schema** ....................................... 110
**Figure 12. Example Project – Entity-Relationship Diagram for the Functionality Specification** ......................... 113
**Figure 13. An Example Role Model in GAIA** ........................................................................................................ 121
**Figure 14. An Example Interaction Model in GAIA** ............................................................................................... 123
**Figure 15. An Example Strategic Dependency Model in the Early Requirements Phase of Tropos** ..................... 125
**Figure 16. An Example Strategic Dependency Model in the Late Requirements Phase of Tropos** ....................... 126
**Figure 17. An Example Strategic Rationale Model in the Late Requirements Phase of Tropos** ......................... 127
**Figure 18. Example GRAPPLE UML Activity Diagram for Modelling Business Process** ................................. 146
**Figure 19. Example GRAPPLE UML Class Diagram for Modelling Domain Structure** .................................. 147
**Figure 20. Example GRAPPLE UML Package Diagram (of Use Cases) for Listing and Classifying System Functions** ..................................................................................................................................... 148
**Figure 21. Example GRAPPLE UML Use Case Diagram for Describing the Relationship Between Use Cases and Actors** ...................................................................................................................................... 149
**Figure 22. Example GRAPPLE UML Use Case Description** .................................................................................. 149


Acknowledgements

My PhD study has been an exciting intellectual adventure. I am very fortunate to have three top scholars as my mentors. Prof. Yair Wand has shared his great wisdom with me in forming the theoretical ideas in my thesis. Prof. Carson Woo has gave me indispensable guidance in developing the practical applications in my thesis. Prof. Rachel Pottinger has provided me tremendous help in improving the presentation of my thesis.

In addition, I am also thankful for the feedback from my colleagues at UBC, as well as the audiences of my thesis defense, the SIGSAND workshop, and the UBC Sauder School of Business’s weekly workshop. In particular, during the thesis defense, I have received lots of helpful feedback from Prof. Hasan Cavusoglu (university examiner), Prof. Gail Murphy (university examiner), and Prof. Antoni Olivé (external examiner).

I am grateful for the generous financial support from the University of British Columbia and scholarship donators, mainly by means of the Four Year Fellowships for PhD Students, the Dean Earle D MacPhee Memorial Fellowship in Commerce and Business Administration, and the International Tuition Award.

Last but not least, I feel so lucky to have my family and friends for their persistent emotional support during this as well as all the other adventures in my life.
To those who had a winding career path but never gave up trying,

whose journey to excellence inspired many others.
Chapter 1: Introduction

This dissertation consists of three studies. This first chapter serves as an introduction and consists of three sections. First, we introduce the common background of the three studies. Second, we introduce the motivations of the three studies. Third, we present an outline of this dissertation.

1.1. BACKGROUND OF DISSERTATION

The focus of this dissertation is on modelling in the specific yet general context of information system (IS) analysis. In this section, we introduce the background of this dissertation by briefly introducing IS analysis, modelling, the application of modelling in IS analysis, and the diversity of modelling studies in IS analysis.

Information systems (IS’s) are an important type of IT capital that serve the functions of representing, informing about, and changing the state of a firm’s organizational environment (Olivé, 2007). The development of an IS begins with identifying and documenting its requirements to support organizational activities. These early development activities are collectively known as IS analysis (Iivari, Parsons, & Wand, 2006, p. 510). In the IS development process, IS analysis is followed by IS design (i.e., the activities for defining the components of the IS to satisfy the requirements specified in IS
analysis (Iivari et al., 2006, p. 510), and IS implementation (i.e., the activities for constructing the IS using software artifacts).

IS analysis is crucial for the success of IS development projects. Keil, Cule, Lyytinen, and Schmidt (1998) conducted a Delphi study with experienced software project managers from three countries, and found “misunderstanding the requirements” to be among the top three risk factors for project failure.

In general, modelling refers to the technique for creating models and representing them as modelling scripts, which are external and explicit representations of certain parts of reality (Pidd, 2009). As characterized by Stachowiak (1973), a model has three features: mapping (that it represents an original), reduction (that it only represent selected aspects of the original), and pragmatic (that it can be used in place of the original for certain purposes). For example, in IS analysis, the requirements (as a model) represent (or map) the desires of the project stakeholders as certain logical conditions that are not satisfied by the current domain, but are expected to be satisfied by the future domain after the project completes. The requirements are a reduction of the domain, because they do not represent other aspects of the domain that are irrelevant to the purpose of the requirements, such as the names, genders, and mailing addresses of the stakeholders. Once the requirements are created, they can be used for the pragmatic purpose of understanding the desire of the stakeholders, without the need for direct access to the stakeholders.

The study of modelling in IS analysis is important, because modelling plays a central role in facilitating the understanding and communication of requirements by means of formally describing certain aspects of the physical or social world around us as conceptual models (Mylopoulos, 1992).
The study of modelling in IS analysis is also diversified. It is divided into several areas, such as in the context of requirements engineering, conceptual schema modelling, and domain understanding. Such division is probably influenced by the fact that the study of IS analysis is broad in scope while researchers and practitioners tend to specialize in only one or a few areas. In any given area of IS analysis, most studies focus on a cohesive set of subjects. For example, requirements engineering studies mainly focus on “the real-world goals for, functions of, and constraints on software systems” (Zave, 1997). Different areas – having different subjects – use modelling methods to represent different aspects of the reality. For example, unlike requirements engineering studies, domain understanding studies – such as Yu (2010) – focus on modelling the structure and behavior of the domain before envisioning the IS.

In summary, in IS analysis, modelling plays a central role in understanding the requirements. Hence, a good understanding of modelling is important for correctly understanding the requirements, for project success, and for firm productivity. To obtain a good understanding of modelling, one must overcome the challenge of diversity of the IS analysis literature.

1.2. MOTIVATIONS OF DISSERTATION

In this dissertation, we are motivated to improve the understanding of modelling in the particular context of IS analysis. To understand the state of the art, we surveyed views.

---

1 In this dissertation, a view of modelling refers to a set of concepts defined to communicate the understanding of modelling. For example, Zave and Jackson (1997) defined a view of modelling to communicate about modelling in
of modelling proposed or adopted² in the IS analysis literature. By analyzing the existing views, we have found that they all suffered from one issue or another.

The situation with the current literature of IS analysis is illustrated in Figure 1. When we consider IS analysis as a particular context of modelling, the three areas of IS analysis (i.e., requirements engineering, conceptual schema modelling, and domain understanding) are the three sub-contexts of the IS analysis context. Each existing view belongs to a context (or sub-context). For example, as shown in Figure 1, the view of (Zave & Jackson, 1997) belongs to the requirements engineering context. The issues from which the existing views suffer are explained in detail in the rest of this section.

Figure 1. Issues with Existing Views of Modelling in IS Analysis

---

² An adopted view refers to one that was originally proposed in another literature, but is later used in the IS analysis literature. For example, Almann, Zschaler, and Wagner (2006) defined modelling for IS analysis by adopting Pidd (2009)’s view, which was originally proposed for management science.
First, we found that some existing views of modelling proposed or adopted to accommodate the entire scope of IS analysis suffer from the issue that they apply to modelling in IS analysis as well as in other contexts. Thus these views cannot accurately define modelling in IS analysis because they are too generic. We define this issue as the *generic issue*. For example, Pidd (2009) defined that “A model is a representation of reality intended for some definite purpose.” When his view is adopted in IS analysis, it is generic because it also applies to models beyond the scope of IS analysis. Suppose one wants to study the connections among different types of models in IS analysis (in a study similar to Chapter 3), he or she would not be able to use such a generic definition to precisely define the scope of his or her study. When the paper is sent to peer review, he or she may find it difficult to answer reviewers’ questions such as “why is the technology acceptance model (Davis, Bagozzi, & Warshaw, 1989) not included in the study?”

Second, we found that some other existing views of modelling proposed to accommodate specific areas of IS analysis suffer from the issue that they each apply to a specific area of IS analysis, but not to other areas of IS analysis. We define this issue as the *limited issue*. For example, Zave and Jackson (1997) proposed a view for the requirements engineering area. But their view does not apply to other areas of IS analysis, such as the domain understanding area. (We will further discuss these two areas in §3.3.2 and §3.3.4, respectively.)

Third, we found that the views of modelling in the three specific areas of IS analysis (i.e., the three areas of requirements engineering, conceptual schema modelling, and domain understanding) are disconnected from each other. We define this issue as the *isolated issue*. For example, we have not found any work that explains in details how
modelling in the conceptual schema modelling area is related to that in the domain understanding area.

Fourth, we found that the three specific areas of IS analysis (i.e., the three areas of requirements engineering, conceptual schema modelling, and domain understanding) have inconsistent terminologies. We define this issue as the inconsistent issue. For example, the same concept may have different names in different areas. As a specific example, the “Domain Knowledge” concept in the requirements engineering area and the “Domain Conceptual Schema” concept in the conceptual schema modelling area refer to the same type of modelling products in IS analysis.

Fifth, we found that some specific modelling methods (e.g., i* (Yu, 1997) and e3-value (Gordijn & Akkermans, 2003)) have been proposed for the domain understanding area. However, unlike the requirements engineering and conceptual schema modelling areas, no existing view has generalized about modelling in this area in a way that is independent of any specific modelling method. We define this issue as the missing issue.

Therefore, in this dissertation, we are motivated to improve the understanding of modelling in IS analysis by solving these issues. This dissertation consists of three studies. In the rest of this section, we will discuss the motivations of each of the three studies in further details.

1.2.1. Motivations of Study 1 (Generalized View)

In the first study, we are motivated to solve the generic and limited issues, by proposing a new view of modelling. This new view in study 1 consists of concepts that are
applicable to all types of modelling products in IS analysis, and the relationship types among these concepts. Since it is essentially a generalization about modelling that is tailored to the specific context of IS analysis, we call it the generalized view of modelling in IS analysis.

We want to show that the generalized view is successful in solving the generic and limited issues. To show that it solves the generic issue, we want to show that some of its concepts can be used to distinguish modelling products in IS analysis from other contexts. To show that it solves the limited issue, we want to show that some of its concepts can be used to compare different types of modelling products in IS analysis.

To demonstrate the utility of the generalized view as a whole, we will show that it can be applied to compare different modelling products in IS analysis. In particular, we will use it to compare two modelling products (i.e., requirements and conceptual database schema) of an example project.

1.2.2. Motivations of Study 2 (Consolidated View)

In the second study, we are motivated to solve the isolated and inconsistent issues, by proposing another new view of modelling in IS analysis. This view contains all general types of modelling products from all of the three areas of IS analysis (i.e., the three areas of domain understanding, requirements engineering, and conceptual schema modelling), and explicates the connections among the different types of modelling products. Thus, it solves the isolated issue. In addition, when it is used as a common terminology for all of the specific areas of IS analysis, it solves the inconsistent issue.
Essentially, this new view in study 2 is a consolidation of the existing views of modelling from all of the three specific areas of IS analysis. Thus, we call it the consolidated view of modelling in IS analysis.

To demonstrate the utility of the consolidated view, we will show that it can be used to help stakeholders to determine whether two IS analysis modelling methods are substitutes or complements, to compare and choose from modelling methods that are substitutes, and to coordinate the use of multiple modelling methods that are complements. We will apply it to evaluate and compare modelling methods in IS analysis. In particular, we will use it to evaluate and compare Gaia (Wooldridge, Jennings, & Kinny, 2000) and Tropos (Castro, Kolp, & Mylopoulos, 2002), two agent oriented modelling methods.

In addition, in proposing the consolidated view, we have also proposed our own generalization of modelling products in the domain understanding area, in a way that is independent of any specific modelling methods. This solves the missing issue for the domain understanding area.

1.2.3. Motivations of Study 3 (Joint Application)

In the first and second studies, we have proposed the generalized and consolidated views of modelling in IS analysis, respectively, and applied them individually. In the third study, we want to suggest a way to apply them together to evaluate modelling methods. We will demonstrate their joint application by applying them together to analyze and evaluate GRAPPLE (Schmuller, 2004, §15), a UML based modelling method. Based on the evaluation results, we will suggest improvements to GRAPPLE. In addition, we also
want to clarify the differences and relationships between the roles played by the two views in such applications.

By choosing GRAPPLE, we apply the two views to another modelling method, one that has not been used in the applications of the first two studies. Thus, we put the generalizability of the two views (and external validity of the first two studies) to test.

**1.3. OUTLINE OF DISSERTATION**

The rest of this dissertation is organized as follows. The first study – the generalized view – will be presented in Chapter 2. The second study – the consolidated view – will be presented in Chapter 3. The third study – the joint application of the two views – will be presented in Chapter 4. Finally, a conclusion will be presented in Chapter 5.

**1.4. REMARKS**

In software engineering related fields, the term “meta-model” refers to models of models. In this dissertation, the generalized view (in Chapter 2), the consolidated view (in Chapter 3), and the area-specific perspectives (in Chapter 3) are some examples of meta-models, because they are models representing certain aspects of the models produced in IS analysis. However, in this dissertation, we did not name the views and perspectives as “meta-models”, to reduce the likelihood that readers confuse them with other types of
meta-models that they already know. For example, modelling languages are meta-models
because they are formal generalizations of the models that they are used to represent. (Note:
the modelling language concept is included as a part of the generalized view in Chapter 2.)
An extended discussion of different types of meta-models is beyond the focus of this
dissertation.
Chapter 2: Towards A Generalized View of Modelling in IS Analysis

Explicating the Common Characteristics of All IS Analysis Models

2.1. INTRODUCTION

2.1.1. Motivation

In this chapter, we focus on a particular type of conceptual model: the ones that are produced in IS analysis. As a foundation for our study in this chapter, we surveyed the literature for views of modelling proposed or adopted in IS analysis. In this chapter, a view of modelling refers to a set of defined concepts and relationship types that are used to explicate certain characteristics of conceptual models. A concept is a generalization of certain things. A relationship is an association among certain things. A relationship type is a generalization of relationships. For example, say that we are interested in two concepts: Apple (whose instances are all apples in the world) and Tree (whose instances are all trees in the world). There may be a relationship between a particular apple (an instance of the Apple concept) and a particular tree (an instance of the Tree concept) that the apple grows
on the tree. We may create a relationship type “Grows On” whose instances include all such relationships between an apple and a tree.

When we tried to apply the existing views of modelling to understand the conceptual models produced in IS analysis, however, we found them to be suffering from being either generic or limited.

In the former case, a generic view is formed for conceptual models in general, and thus may not convey a deep understanding of conceptual models produced in IS analysis. This is because a generic view is not tailored to the context of IS analysis. Although a generic view is typically still applicable to the context of IS analysis, it may not be able to convey a deep understanding to effectively guide modelling practices in IS analysis. For example, some IS analysis related publications such as Alßmann et al. (2006, §9.2.1) have adopted Pidd (2009)’s definition that “A model is a representation of reality intended for some definite purpose.” This definition does apply to conceptual models produced in IS analysis, which is a specific type of model after all. However, this generic definition is not tailored to, and thus does not provide sufficient guidance for, producing conceptual models in the specific context of IS analysis. For example, it does not specify which particular purposes are models created and used for in IS analysis.

In the latter case, a limited view is formed within the scope of a specific area of IS analysis, and thus may not be applicable to the entire scope of IS analysis. This is because the view is explicitly or implicitly associated with the area’s limited context, which are often more restricted than the general context of IS analysis. Particularly when the association with the limited context is implicit, the view might be confusing for being misunderstood to be applicable to other similar contexts as well. For example, most
publications in the conceptual schema modelling area (such as van Griethuysen (1982, §1.3), Loucopoulos (1992, p. 4), Rolland and Cauvet (1992, p. 27), and Olivé (2007, p. 1)) only discuss a single type of conceptual modelling product – the conceptual schema. Thus, readers of these publications might mistakenly equate conceptual modelling with conceptual schema modelling, and misunderstand the conceptual schema to be the only type of conceptual model produced in IS analysis. However, in other areas related to IS analysis, the product of conceptual modelling may be a different type of conceptual model. For example, the domain understanding area focuses on the study of producing domain understanding models. Two examples of domain understanding models are the $i^*$ models (Yu, 1997; Yu & Mylopoulos, 1994) and the $e^3$-value models (Akkermans et al., 2004; Gordijn & Akkermans, 2001; Gordijn & Akkermans, 2003; Gordijn, Yu, & van der Raadt, 2006). The view of modelling in the conceptual schema modelling area does not apply to $i^*$ and $e^3$-value models, and hence is limited.

It would be ideal, we think, to strike a new balance with a generalized view of modelling in IS analysis: a view of modelling that is specially tailored to the general context of IS analysis. This generalized view should achieve three goals. First, to avoid being generic, it should explicate the distinct characteristics of modelling products in the specific context of IS analysis versus in a generic context. For example, one should be able to use this definition to determine that conceptual schemas are modelling product in IS analysis, while the behavioral research models (e.g., the Technology Acceptance Model (Davis et al., 1989)) are not.$^3$ Second, to avoid being limited, it should be applicable to all types of

---

$^3$ As we will explicate in §2.3.4, a conceptual schema is produced in the process of developing a specific new IS to analyze the requirements of the IS. Hence, it is a modelling product of IS analysis. In contrast, a behavioral research
modelling products in IS analysis. For example, unlike the limited view of modelling formed in the conceptual schema modelling area discussed in the previous paragraph, this generalized view should apply to not only conceptual schemas but also i* models, e3-value models, as well as all other modelling products in IS analysis. Third, it should explicate the general characteristics with respect to which different types of modelling products in IS analysis can be compared for similarities and differences. For example, say that different types of modelling products in IS analysis can be compared with respect to their purposes, the generalized view should include purpose as such a general characteristic.

2.1.2. Overview of Method

To build such a generalized view and achieve these three goals, we will start by selecting the relevant concepts and relationship types through our analysis of the literature, in the following steps.

First, we will select concepts and relationship types that are relevant to the distinct characteristics of modelling products in the specific context of IS analysis versus in a generic context. This step is to fulfill the first goal, i.e., avoid being generic. It is conducted first because it defines the scope of “modelling products in IS analysis” and thus also the scope of this chapter. We will also demonstrate how these concepts and relationship types can be applied to explain why certain types of conceptual models should not be regarded as modelling products in IS analysis. This selection is based on our own knowledge of model (such as the Technology Acceptance Model) is not associated with any specific IS, but is applicable to all (or certain types of) IS’s in general. Hence it is not a modelling product of IS analysis.
modelling products in IS analysis, rather than based on a specific set of publications. Although we may not have included all distinct characteristics of IS analysis models, we have demonstrated the value of the ones included in our search result. The details and results will be presented in §2.3.

Second, we will select concepts and relationship types that are relevant to the general characteristics with respect to which different types of modelling products can be compared for similarities and differences, in either generic or IS analysis contexts. For example, it is common for a conceptual model (e.g., a requirements model) to have a “purpose” (e.g., to express how the stakeholders would like the domain to behave at the end of project). Different types of conceptual models may have different, but sometimes similar, purposes. Thus, conceptual models have “purpose” as a common characteristic, with respect to which they can be compared for similarities and differences. This step is to fulfill the third goal, i.e., compare different types of modelling products in IS analysis. This selection is based on an online search using Google Scholar and the relevant keywords. The details and results will be presented in §2.4.1.

Third, we will select concepts and relationship types that are relevant to the representation and interpretation of modelling products in IS analysis. This is an important topic for understanding conceptual models produced in IS analysis because, in both research and practice, these conceptual models often need to be represented as and interpreted from modelling scripts, rather than only exist in stakeholders’ minds. We have selected (Harel & Rumpe, 2004) as a representative publication on the topic, and extracted the relevant concepts and relationship types from it. In addition, we have also discussed a few other concepts (e.g., Ontology) and relationship types (e.g., the one between Model
and Script) based on our own expert knowledge of the topic. The details and results will be presented in §2.5.

In all steps above, if a concept or relationship type does not apply to all types of conceptual models produced in IS analysis, we will exclude it from our study, so as to fulfill the second goal, i.e., avoid being limited.

Finally, once the relevant concepts and relationship types have been selected through the above steps, we will put them together toward the formation of a generalized view of modelling in IS analysis. The presentation of the generalized view will consist of visualization, definitions, and illustrative examples.

2.1.3. Outline of Chapter

In the next section, we will survey the related work. The remaining sections of this chapter are organized as follows. In each of the three subsequent sections, we will select the relevant concepts and relationship types in the first, second, and third steps of our method (as explained in the previous subsection), respectively. Then, we will build the generalized view by putting together these concepts and relationship types. After that, we will present an example project to illustrate the abstract concepts in the generalized view with concrete examples, and to demonstrate applying the generalized view to compare different IS analysis models for similarities and differences. Finally, we will summarize this chapter and discuss its academic contributions.
2.2. RELATED WORK

This section surveys the related work and compares them to the work of this chapter.

2.2.1. Related Work on Defining Modelling Products in IS Analysis

The “generalized view” in this chapter defines the “IS analysis model” concept whose instances are conceptual models produced in IS analysis, along with other concepts that are applicable to IS analysis models. Therefore, it is related to previous work on defining modelling products in IS analysis. We divide them into three different categories.

The first category consists of works that are applicable to all types of modelling products in IS analysis. We will review the definitions from these previous publications in detail in the second next section, and thus will not dive into further details here.

The work in this chapter belongs to the first category, but differs from the previous work in the first category. The previous works in the first category suffer from being generic because they cannot be used to distinguish the conceptual models produced in IS analysis versus in a generic context. As we will demonstrate in the next section, this chapter remedies by proposing a definition of IS analysis models that is not generic.

The second subcategory consists of works that are applicable to certain specific types of modelling products in IS analysis, but are independent of any specific modelling methods. For example, Zave and Jackson (1997) studied three types of modelling products in requirements engineering: requirements, specifications, and domain knowledge. Olivé (2004) studied three types of modelling products in conceptual schema modelling:
conceptual schema, functionality specification, and domain conceptual schema. The works in the second category are studied in details in the next chapter.

Previous works in the second category suffer from being limited because they are not applicable to all types of conceptual models produced in IS analysis. In contrast, the “generalized view” in this chapter differs from previous works in the second category. It is not limited because it is applicable to all types of conceptual models produced in IS analysis.

The third category consists of works that define the modelling products of specific modelling methods or languages. For example, Yu (1997) defined the strategic dependency model and the strategic rationale model, two types of modelling products of the i* modelling language.

Unlike works in the first two categories, works in the third category are associated with specific modelling methods or languages, and thus are not applicable to other modelling methods or languages. The “generalized view” in this chapter differs from works in the third category, because it is applicable independent of any specific modelling methods or languages.

2.2.2. Related Work on Comparing Modelling Products in IS Analysis

Later in this chapter, once we have presented the generalized view, we will demonstrate using it as an analytical framework for comparing modelling products in IS analysis. Some previous work have been done on comparing modelling products, languages, and methods. However, unlike our work, none of them was tailored for the
particular scope of modelling products in IS analysis. In the rest of this subsection, we survey the relevant research topics with a few examples.

The first research topic is on analytically comparing modelling products of a specific type, or modelling languages or methods that produce specific types of modelling products. Some examples of such modelling product types include models of computation (Lee & Sangiovanni-Vincentelli, 1998), agent-oriented models in system analysis and design (Arazy & Woo, 2002; Dam & Winikoff, 2004; Sturm & Shehory, 2004), and business process models (Melão & Pidd, 2000).

Our application of the generalized view belongs to this topic. However, unlike our work, no previous work can be used to compare all types of modelling products in IS analysis. In fact, as we will discuss in the next section, our work is the first one that tries to clearly define the scope of modelling products in IS analysis.

The second research topic is on analytically comparing or evaluating modelling languages. For example, Wand and Weber (1993) proposed a framework to evaluate and compare modelling languages based on their ontological expressiveness.

However, modelling products in IS analysis may differ not only in the modelling languages in which they are represented, but also in other aspects such as the purposes for which they are created and the ontological theories adopted by their creators. Unlike our application of the generalized view, no previous work on the second research topic had investigated comparing IS analysis models with respect to these other aspects.
The third research topic is on empirically comparing or evaluating modelling languages or methods. Some examples are Burton-Jones, Wand, and Weber (2009), Gemino and Wand (2004), and Soffer, Kaner, and Wand (2012).

In the first and second research topics, the comparison or evaluation is conducted by analyzing modelling products, languages, or methods according to certain theoretical frameworks. In contrast, in the third research topic, the comparison or evaluation is conducted by analyzing data collected from the use of the modelling products, languages, or methods in empirical studies. Our work on applying the generalized view to compare modelling products in IS analysis belongs to the first topic and compares modelling products in IS analysis by analyzing them according to the generalized view (which is used as a theoretical framework for such comparison).

2.3. CONCEPTUAL MODELS IN IS ANALYSIS VERSUS IN GENERIC CONTEXTS

In this section, we are motivated to select the distinct characteristics of conceptual models produced in the context of IS analysis that distinguish them from models produced in other contexts. First, we examine the state of the art of the IS analysis literature regarding a general definition of conceptual modelling that is applicable to the entire scope of IS analysis. Second, we point out some issues of the state of the art. Third, we propose a supplement to the state of the art definition as a remedy to solve these issues. Fourth, we demonstrate that the proposed supplement solves these issues. Finally, we summarize and analyze the concepts and relationship types related to the study of this section.
2.3.1. State of the Art Definition of Modelling in IS Analysis

Conceptual modelling is the activity that produces conceptual models. In the current literature of IS analysis, the following renowned statement is widely cited to define conceptual modelling:

“Conceptual modelling is the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication.” (Mylopoulos, 1992).

2.3.2. The Generic Issues with the State of the Art

The statement from Mylopoulos (1992) has no issues per se. It is a very good description for conceptual modelling in general. However, when it is quoted or rephrased by others to define conceptual modelling in IS analysis, it is too generic to exclude some conceptual models that are never studied in IS analysis.

For example, behavioral studies in Management Information Systems (MIS) and other academic disciplines often include a theoretical model that visualize the theorized causal relationships among certain measurable constructs. A famous example is the Technology Acceptance Model (Davis, 1989; Davis et al., 1989) in MIS. Such a model is formal, describes an aspect of the world (that there are certain causal relationships between certain constructs), and may also be termed as a “conceptual model” by some behavioral researchers such as in Xiao and Benbasat (2007). Clearly, the widely cited statement from Mylopoulos (1992) also fits here. However, such models have never been an intended subject of IS analysis studies.
In fact, in the article of Mylopoulos (1992), the definition of conceptual modelling cited above is situated in a context of system analysis. This context is implied by the rest of that article. However, this context was only implicitly attached to the definition, and has not been summarized in a concise statement that can be readily cited by others. Therefore, the definition is always cited alone and lacks explanatory power to exclude conceptual models that are beyond the scope of IS analysis.

2.3.3. **Proposing A Remedy**

For benefit of more precisely describing the research focus of IS analysis, we propose to supplement Mylopoulos (1992)’s renowned definition with an explicit statement of the IS analysis context:

> In IS analysis, conceptual modelling is conducted in the process of developing a specific new IS for certain purposes related to analyzing its requirements to support organizational activities. The modelling product is a specific type of conceptual model, defined as an **IS analysis model**.

2.3.4. **Demonstration of Utility: On Clarifying the Scope of Conceptual Modelling in IS Analysis**

Our supplement proposed above clarifies the scope of conceptual modelling activities in the context of IS analysis by defining its product. In particular, unlike Mylopoulos (1992)’s definition, our supplement definition clearly states that the goal of conceptual modelling in IS analysis is to produce models (of “the physical or social world around us”) for developing specific new IS’s. To elaborate, we will demonstrate how it
adds utility to the state of the art definition by adding the explanatory power to exclude some types of models that are easily confused with IS analysis models.

First, the supplement clarifies that producing behavioral research models such as the Technology Acceptance Model (discussed earlier in this section) are not the goal of conceptual modelling in IS analysis. This is because they apply to all systems in general or all systems of a certain type (such as B2C e-commerce systems) without being associated with any specific IS. Moreover, when such a model is applied to facilitate the development of an IS, the model typically exists before the development process starts. It is not produced in the development process.

Second, similarly, the supplement distinguishes IS analysis models from IS analysis and design (ISA&D) theory models. Some examples of ISA&D theory models are Bunge’s Ontology (Wand & Weber, 1993), Searle’s social ontology (March & Allen, 2014), concept theory (Parsons, 1996), speech act theory (Auramäki, Lehtinen, & Lyytinen, 1988), the representation model (Wand & Weber, 1995), the state tracking model (Wand & Weber, 1995), and the good decomposition model (Wand & Weber, 1995). Although ISA&D theory models are also studied in IS analysis, they fundamentally differ from IS analysis models. While an IS analysis model is produced for developing a specific IS, an ISA&D theory model applies to all IS’s in general and is not associated with any specific IS. When an ISA&D theory model is used in IS development, it is adopted from existing literature and used to provide theoretical guidance for development, rather than produced as a product of development.

Third, similarly, the supplement also distinguishes IS analysis models from modelling languages, which may also be termed “models” in certain literatures related to
IS analysis. While IS analysis models are produced as modelling products in IS development, modelling languages are adopted as tools used in IS development. For example, in the literature of semantic database modelling, the term “Entity-Relationship Model” refers to a modelling language in which a conceptual database schema can be written as an Entity-Relationship diagram. A conceptual database schema (of an IS) – represented as an Entity-Relationship diagram – is an IS analysis model, whereas the Entity-Relationship Model is actually a modelling language.

2.3.5. Section Summary and Analysis

To summarize, in this section, we pointed out that the state of the art general definition of conceptual modelling in IS analysis is not explicitly associated with the IS analysis context. We showed that the definition alone does not distinguish conceptual modelling in the IS analysis context from that in a generic context, with the example of the Technology Acceptance Model. To remediate, we proposed to supplement the state of the art definition with an explicit statement of the IS analysis context. The goal of the supplement is to clarify the scope of conceptual models produced in IS analysis, i.e., IS analysis models.

The proposed supplement is based on three concepts and two relationship types. The three concepts are: IS analysis model, purpose, and information system (IS). There is one relevant relationship types: An IS analysis model is created in the process of developing a specific new IS for certain IS analysis relevant purpose. These concepts and relationship types are visualized in Figure 2 below.
The concepts and relationship types in Figure 2 will be incorporated in the generalized view in §2.6. Their definitions will also be presented there.

We have demonstrated the utility of these concepts and relationship types by using them to explain why certain models (e.g., Technology Acceptance Model, Bunge’s Ontology, and Entity-Relationship Model) are not IS analysis models. This shows that our proposed supplement adds utility to the renowned definition of Mylopoulos (1992), when used together to define and understand IS analysis models.

2.4. COMPARING DIFFERENT TYPES OF IS ANALYSIS MODELS

As we have discussed at the beginning of this chapter, existing views of modelling in IS analysis literature suffer from being either generic or limited. In the previous section, we have tackled the generic issue by defining the term “IS analysis model”. In this section, we are motivated to tackle the limited issue by searching for the general characteristics
with respect to which different types of IS analysis models can be compared for similarities and differences. While some existing views of modelling in IS analysis may be limited and incompatible with each other (as discussed at the beginning of this chapter), we hope the general characteristics studied in this section can help researchers to better understand the similarities and differences of the different views.

This section is organized as follows. First, we select these characteristics by surveying the definitions of models and modelling that are proposed or adopted in the IS analysis literature. Second, we demonstrate the utility of these characteristics in comparing different types of IS analysis models. Finally, we summarize and analyze the concepts and relationship types related to the study of this section.

2.4.1. Searching for General Characteristics of IS Analysis Models in Their Definitions

In this subsection, we are motivated to find the general characteristics with respect to which different types of IS analysis models can be compared for similarities and differences. This helps researchers and practitioners to better understand the similarities and differences of different types of IS analysis models.

To figure out these general characteristics, we surveyed the IS analysis literature for definitions of model and modelling with a few selection criteria. The survey was conducted by using a number of relevant keywords (i.e., “model”, “modelling”, “conceptual model”, and “conceptual modelling”) in an online search using Google
First, there are many types of such definitions, some are more applicable to IS analysis models than others. Thus, we focused on those adopted or proposed by publications related to IS analysis. Second, since we are searching for characteristics applicable to all types of IS analysis models, we selected only those that apply to IS analysis models in general. In other words, we excluded definitions for specific types of IS analysis models, such as conceptual schema, domain understanding model, and semantic data model. This exclusion is because, if we extract characteristics from these limited definitions, we cannot be sure that the extracted characteristics are commonly possessed by all types of IS analysis models. Third, some publications may have adopted a generic definition of model or modelling that are not restricted to the scope of IS analysis. Such a generic definitions is still included as long as it is applicable to IS analysis models.

Following these criteria, our literature review results are summarized in Table 1. Each row of the table shows a term related to modelling in IS analysis, the source publication in which it is defined, and its definition from the source.

<table>
<thead>
<tr>
<th>Term and Source</th>
<th>Definition (Quotes from Source)</th>
<th>Extracted Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual Modelling (Mylopoulos, 1992)</td>
<td>“Conceptual modelling is the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication.”</td>
<td>Focal aspects (i.e., some aspects), Purpose, Domain (i.e., world)</td>
</tr>
<tr>
<td>Conceptual Modelling (Embley &amp; Thalheim, 2011)</td>
<td>“Conceptual modelling is about describing the semantics of software applications at a high level of abstraction.”</td>
<td>Domain (i.e., semantics of software application)</td>
</tr>
<tr>
<td>Information Modelling (Siau, 2004)</td>
<td>“Information modelling is the process of formally documenting the problem domain for the purpose of understanding and communication among stakeholders.”</td>
<td>Domain, Purpose, Stakeholder</td>
</tr>
<tr>
<td>Term and Source</td>
<td>Definition (Quotes from Source)</td>
<td>Extracted Characteristics</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Conceptual Model (Wand &amp; Weber, 2002)</td>
<td>“The requirements analysis phase that occurs during information systems development often involves use of models called conceptual models. These models, which are mostly graphic, are used to represent both static phenomena (e.g., things and their properties) and dynamic phenomena (e.g., events and processes) in some domain.”</td>
<td>Domain (i.e., a set of phenomena), Notation (i.e., graphic or otherwise)</td>
</tr>
<tr>
<td>Model (Pidd, 2009)</td>
<td>“A model is a representation of reality intended for some definite purpose. … A model is an external and explicit representation of a part of reality as seen by the people who wish to use that model to understand, change, manage, and control that part of reality.”</td>
<td>Domain (i.e., reality), Purpose, Stakeholder (i.e., people who wish to use that model)</td>
</tr>
<tr>
<td>Model (Bubenko, 1980)</td>
<td>“Whatever it is, we will restrict ourselves to cases that are within reach of our mental and abstraction capacity of create in our minds a ‘model’ of this something. We will perceive, select and classify individual phenomena into classes of sufficiently similar phenomena and generalize relationships between individual phenomena into relations on sets of phenomena. We will also assume that these phenomena obey a set of rules or that their behavior is constrained in some way. By doing this we create a simplified, partial ‘model’ of the environment in our minds.”</td>
<td>Domain (i.e., this something, environment), Focal aspects (i.e., phenomena that are selected to be included in the model)</td>
</tr>
<tr>
<td>Model Originally defined by (Stachowiak, 1973) in German. Summarized by (Kühne, 2005) in English.</td>
<td>“According to Stachowiak a model needs to possess three features: (1) <strong>mapping feature</strong>: A model is based on an original. (2) <strong>reduction feature</strong>: A model only reflects a (relevant) selection of the original’s properties. (3) <strong>pragmatic feature</strong>: A model needs to be usable in place of the original with respect to some purpose.”</td>
<td>Domain (i.e., the mapped original), Focal aspects (i.e., the selection of the original’s properties), Purpose</td>
</tr>
</tbody>
</table>

As shown in the last column of Table 1, we have extracted the common characteristics of models from the definitions. We have conducted the extraction process by finding all the nouns in the definitions, and then selecting – according to our expert judgment – those nouns that describe certain characteristics of models. Because the same characteristics may be described in different terms in different definitions, we have renamed – according to our expert judgment – some characteristics from some definitions, such that the same characteristics have the same names in the extraction results. For
example, in the first row of the table that correspond to the definition of (Mylopoulos, 1992), we have found four nouns: conceptual modelling, activity, some aspects, world, purpose, understanding, and communication. Based on our expert judgment, we have determined that, in this definition, “some aspects”, “world”, and “purpose” are describing certain characteristics of models; while “conceptual modelling” and “activity” are not. We have not included “understanding” and “communication” in the extraction result because they are two specific types of purpose. Based on our expert judgment and the terms used in other definitions, we have renamed “world” into “domain”, and “some aspects” into “focal aspects”.

In summary, from these definitions, we have found five common characteristics of IS analysis models: domain (i.e., the part of reality that is being modeled), purpose (i.e., the intended use of the model), focal aspects (i.e., the types of things in the domain that are selected to be represented in the model), stakeholder (i.e., people who wish to use that model), and notation (i.e., the syntax in which the model is represented).

As introduced at the beginning of this section, the goal of this section is to find the common characteristics with respect to which different types of IS analysis models can be compared for similarities and differences. Not all of the five characteristics serve this goal.

Domain and stakeholder do not serve this goal. Say that multiple types of IS analysis models are used in a single IS development project. These different types of models may share the same domain and stakeholders. Moreover, IS analysis models from different IS development projects are likely to have different domains and stakeholders, even if they are of the same type. Thus, domain and stakeholder cannot be used to compare different types of IS analysis models. Instead, we think it is more appropriate to use them
to compare instances of IS analysis models that are of the same or different types. For example, using these two in such comparisons may yield the result that two conceptual schemas of two different projects have different domains and stakeholders, or that a conceptual schema and a functionality specification of the same project have the same domain and stakeholders.

In addition, notation does not serve this goal for at least three reasons. First, the same type of model may be written in different notations. For example, a conceptual database schema may be written as either a UML class diagram or an ER diagram, among other options. Second, the same notation may sometimes be used to write different types of models. For example, depending on the use and context, a UML class diagram may represent either a system design model or a system analysis model, as shown by Evermann and Wand (2005). Third, as we will explain in detail in the next section, model and script are two distinct concepts. A model is essentially a collection of mental constructs, whereas a script is essentially a collection of syntactic (or notational) expressions recorded in an external medium so as to represent a model. Thus, notation (or syntax) is a characteristic of the script that represents the model, rather than of the model itself.

In contrast, the two remaining characteristics – purpose and focal aspects –indeed serve the goal of comparing different types of IS analysis models for similarities and differences. We will demonstrate such comparison with examples in the next subsection.
2.4.2. Demonstration of Utility: On Comparing Different Types of IS Analysis Models For Similarities and Differences

In the previous subsection, we found five common characteristics of IS analysis models by analyzing the general definitions of modelling adopted or proposed in IS analysis literature. Two of them – purpose and focal aspects – can be used to compare different types of IS analysis models for similarities and differences.

In this subsection, we will use these two characteristics to compare two different types of IS analysis models, conceptual schema and domain understanding. A conceptual schema is a formal, complete, and conceptual description of possible states of a certain domain, including its classifications, rules, laws, etc. (van Griethuysen, 1982). A domain understanding is a conceptual model useful for eliciting the requirements of the project (see §3.3.4). Since different types of domain understanding may serve the purpose of requirements elicitation by modelling different focal aspects of the domain, and no previous publication formally generalized about all types of domain understanding, we will use the i* model (Yu, 2010) and the e3-value model (Gordijn & Akkermans, 2001) as two specific types of domain understanding in this comparison. The i* models focus on modelling the strategic dependencies, while the e3-value models focus on modelling the economic value propositions in the domain.

The comparison results are presented in Table 2. (Readers who are not familiar with the three types of IS analysis models may refer to the corresponding publications cited above for the definitions of the terms and other details. However, these details are not very
important here since the table is only used as an example to demonstrate the use of the two characteristics in comparing different types of IS analysis models.)

**Table 2. A Comparison of Three Types of IS Analysis Models**

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Purpose</th>
<th>Focal aspects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual Schema</td>
<td>To serve as a formal specification of a given set of requirements.</td>
<td>classifications, rules, laws, …</td>
</tr>
<tr>
<td>Domain Understanding</td>
<td>The i* Model To elicit the requirements by understanding strategic dependencies.</td>
<td>actors, strategic dependencies between actors, strategic rationales of actors, …</td>
</tr>
<tr>
<td></td>
<td>The e3 Value Model To elicit the requirements by understanding the economic value propositions.</td>
<td>actors, value objects, value ports, value interfaces, value exchanges, value offerings, market segments, …</td>
</tr>
</tbody>
</table>

**2.4.3. Section Summary and Analysis**

To summarize, in this section, we reviewed the literature for definitions applicable to IS analysis models in general. We analyzed these definitions for concepts and relationship types that explicate the common characteristics of IS analysis models, and found four concepts: domain, purpose, focal aspects, and stakeholder. There are two relationship types among them and the IS analysis model concept. First, the stakeholders create an IS analysis model for certain purpose. Second, the IS analysis model represents the domain with respect to the focal aspects. These concepts and relationship types are visualized in Figure 3.

We have also found the notation (aka. syntax) concept to be relevant. However, as we explained earlier in this section, we consider it as a characteristic not of a model, but of
a corresponding script. Since we will discuss it and other similar concepts in detail in the next section, we have not included it in Figure 3.

**Figure 3. Concepts and Relationship Types – For Explicating the General Characteristics of IS Analysis Models**

The concepts and relationship types in Figure 3 will be incorporated in the generalized view in a subsequent section. Their definitions will also be presented there.

Our further analysis revealed that only two of these common characteristics – purpose and focal aspects – are useful for comparing different types of IS analysis models for similarities and differences. We then demonstrated the utility of them in comparing different types of IS analysis models with an example comparison.

### 2.5. REPRESENTATION AND INTERPRETATION OF IS ANALYSIS MODELS

In this section, we study the representation and interpretation of IS analysis models. This is an important topic because, in both research and practice, IS analysis models often need to be represented as and interpreted from modelling scripts, particularly when it is used as a medium to facilitate the communication among multiple stakeholders.
This section is organized as follows. In the first subsection, we make a distinction between models and scripts, and point out why it is important to make such distinction in the IS analysis context. As we then explain, this implies that the representation and interpretation of IS analysis models involves two mappings: the one between model and script, and the one between model and domain. Hence, we study them in the second and third subsections, respectively. In the fourth subsection, we compare semantic domain and ontology – two similar but different concepts relevant to the two mappings. Finally, we summarize the concepts and relationship types related to the study of this section.

2.5.1. The Distinction Between Model and Script

In the previous section, we reviewed the relevant definitions of modelling in IS analysis. We noticed that some of these definitions regard models as mental constructs (i.e., Bubenko (1980)), whereas some other definitions regard them as external representations in certain notations (i.e., Wand and Weber (2002) and Pidd (2009)). The rest of the definitions did not explicitly state whether models are in essence mental constructs or external representations.

Therefore, in this subsection, we are motivated to clarify whether an IS analysis model (i.e., a conceptual model produced in IS analysis) is in essence a mental construct, an external representation, both, or neither.

The rest of this subsection is organized as follows. First, we claim that models are in essence mental constructs that may be externally represented as scripts. Second, we support the claim by examining the importance of regarding model and script as two
distinct concepts in IS analysis. Third, we discuss what the distinction indicates for our study in this section.

2.5.1.1. The Distinction

First, we claim the following. (We will support it in the next sub-subsection.)

In IS analysis, model and script are two distinct concepts. A model consists of mental constructs (e.g., concepts, relationship types, and attributes) that represent certain aspects of a certain domain. In contrast, a script consists of syntactic expressions (e.g., rectangles, diamonds, ovals, and lines) – recorded in an external medium (e.g., paper, whiteboard, and digital file) – that represent the mental constructs of a model. A model is a mental representation of certain aspects of a domain, while a script is an external representation of a model.

Remarks: In certain areas of IS analysis, “models” may almost always be externally represented as “scripts” and hence it may be a convenient shortcut to equate the two concepts within the scope of those areas. This reflects a limited view on the distinction between model and script in these areas, and may be the reason why some definitions (surveyed in the previous section) regard “models” as external representations in certain notations. However, in the broader scope of IS analysis, we claim that it is important to make a distinction between the two, as we will elaborate next.

2.5.1.2. The Importance of the Distinction in IS Analysis

We claimed that it is important to distinguish between model and script in the broader scope of IS analysis. To support our claim, it suffices to show the existence of some IS analysis research problems for which the distinction matters. Hence, we provide
some examples of research problems in IS analysis and explain why it is important to make
this distinction in addressing these research problems.

First, some IS analysis research is conducted to empirically compare different ways
to model the same domain as different scripts. To better conduct this kind of research, it is
helpful to explicate the subtle difference between model and script. In particular, when the
research goal is to compare two modelling grammars, in order to control for other
differences between the treatment and control groups, researchers need to bear a clear mind
whether the grammatical difference results in difference merely in the script, or in both the
script and the model represented by the script. In the next two paragraphs, we will give an
example of each case, respectively. The two examples are very similar to Ramsey, Atwood,
and Doren (1983) and Brosey and Shneiderman (1978), respectively, except that the two
cited publications studied modelling in system design, rather than in system analysis.

To give an example of the first case, say that a study – very similar to Ramsey et al. (1983) – compares modelling the same business process model (i.e., the mental model)
as a script in a flowchart style notation versus as an equivalent script in a “if-then-else”
style textual notation. In designing such a study, researchers need to make sure that
differences exist in only the two scripts, but not the model represented by the two scripts.
In other words, they need to make sure that the flowcharts and the “if-then-else” textual
scripts in the study represent the same business process model. If the two scripts represent
two different business process models, the research design would be flawed.

To give an example of the second case, say that another study – very similar to
Brosey and Shneiderman (1978) – compares semantic database modelling as a script using
the relational data model versus as another script using the hierarchical data model. Because of the difference between relational and hierarchical data models, the mental models represented by the two scripts are also different. When interpreting the results of such a study, researchers need to realize that the differences exist in not only the two scripts, but also the models represented by them.

Second, since the difference between model and script is relevant to empirical comparisons between conceptual modelling techniques (as we have just elaborated above), it is also relevant to frameworks proposed to classify and guide such empirical comparisons. We surveyed such frameworks proposed in the IS analysis literature and have not found any framework that explicates the difference between model and script. The survey was conducted by using the relevant keyword combinations (i.e., “framework modelling language”, “framework modelling grammar”, “framework modelling method”, and “framework modelling technique”) in an online search using Google Scholar. We believe that the published frameworks can be improved if they would explicated the difference between model and script. For example, in the previous paragraph, we pointed out a difference between two example studies. In the first example study, the treatment and control groups use two different scripts that represent the same model. In the second example study, the treatment and control groups use two different scripts that represent two different models. This difference between the two example studies is also present between Ramsey et al. (1983) and Brosey and Shneiderman (1978), which are very similar to the first and second example studies, respectively. However, the framework proposed by

---

4 These two data models are designed for logical database modelling. They are not ideal tools for semantic database modelling. Nonetheless, they could be used for semantic database modelling as well. We are just using them as examples here.
Gemino and Wand (2004, p. 254) have compared them but have not identified this particular difference.

Third, to study “implementation bias”, an important phenomena in requirements engineering (which is an important area related to IS analysis), it is necessary to make a distinction between model and script. An implementation bias is an arbitrary constraint imposed on the system to be developed that may lead to a sub-optimal solution to satisfy the stakeholders’ requirements (Straub & Zelkowitz, 1992). According to Straub and Zelkowitz (1992), implementation bias are introduced in two cases.

The first case happens when stakeholders incorrectly interprets a requirements modelling script into a requirements model with more constraints than actually represented by the script, and as a result introduce these additional fictitious constraints in modelling scripts in future stages of requirements modelling activities. For example, a requirements modelling script may state that the new IS to be developed needs to be integrated with existing ones implemented using the Java programming language. A group of stakeholders might somehow misread this statement as a fictitious constraint that the new IS to be developed needs to be implemented using Java as well. This is a misinterpretation because machines implemented using other programming languages (e.g., C#, Python, and Ruby) can usually also be integrated with those using Java. If they introduce this fictitious constraint into scripts in future stages of modelling activities, they would introduce an implementation bias.

The second case happens when the stakeholders are forced to represent the requirements model as a script with more constraints than the requirements model, due to limitations of the chosen modelling language. As an example adapted from Straub and
Zelkowitz (1992, p. 20), the requirement model may state that the IS can provide the set of current employees. However, the stakeholders may have been using a formal modelling language that does not have the “set” construct. Hence they used “array”, the closest approximation available in that language. While a set can be implemented as an array, it can also be alternatively implemented as heap, tree, hash table, etc. Thus, due to the lack of the “set” construct in the formal modelling language, the script introduces an extra constraint (which does not exist in the model it represents) that the “set” is implemented as an “array”.

In both cases, “implementation bias” is defined based on the difference between a model and a relevant script. Hence we conclude that, to define and study “implementation bias”, an important phenomena in both requirements engineering and IS analysis, it is necessary to make a distinction between model and script.

Fourth, in some IS analysis research, the same script may be interpreted into different models. The work by Straub and Zelkowitz (1992) – that we have just discussed above – provides an example where implementation bias may be introduced because a script representing a certain requirements model is misinterpreted into another model with extra constraints. As another example, some researchers – such as Evermann and Wand (2005) – proposed techniques to adapt IS design modelling languages (e.g., UML) for the purpose of IS analysis. Using such techniques, the same script (e.g., a UML class diagram) may represent either a design model of software artifacts or an analysis model of real-world phenomena, and thus interpretation from scripts into models is dependent on the context (i.e., design or analysis) in which they are used.
In summary, in this subsection, we have discussed four types of IS analysis research work. They all indicate that the distinction between model and script is important in the broader scope of IS analysis.

2.5.1.3. What the Distinction Indicates

In summary, in this subsection, we made a distinction between model as a mental representation of a domain and script as an external representation of a model, and claimed that it is important to make this distinction in the general scope of IS analysis research. We supported this claim with examples of IS analysis research problems for which this distinction matters.

Since models are often externally represented by scripts in IS analysis rather than only mentally represented, the distinction indicates that, to study the representation and interpretation of IS analysis models, we need to study the representation and interpretation mappings both between model and script and between model and domain. Hence, we study them in the next two subsections, respectively.

2.5.2. The Mappings Between Model and Script

In the previous subsection, we established that a model consists of mental constructs, while a script consists of syntactic expressions. In this subsection, we want to study the mapping between the two. This can be done by studying modelling languages, which are mechanisms through which models are represented by scripts, and through which scripts are interpreted into models.
Many studies have been done on modelling languages. Although the terminologies used in these studies may differ, their conceptualization of elements of modelling languages are more or less the same. In this subsection, we will refer to the view\(^5\) of Harel and Rumpe (2004) as a representative, because it focuses on explaining semantics, which is particularly relevant to the mapping between model and script. In their view, a modelling language consists of syntax and semantics. The *syntax* provides a mechanism through which *syntactic expressions* of the script are written. More specifically, the syntax consists of syntactic constructs and rules. A *syntactic construct* is an elementary notation used in forming syntactic expressions in the scripts. The *syntactic rules* are the rules according to which syntactic constructs are combined to form syntactic expressions. The *semantics* consists of the semantic domain and the semantic mapping. The *semantic domain* defines phenomena that can be represented by the language. The *semantic mapping* defines the mapping between syntactic expressions and the semantic domain.

To explicate the aforementioned concepts from Harel and Rumpe (2004) with an example, we apply them to the classic Entity-Relationship (ER) model (Chen, 1976) in Table 3. Interested readers may refer to Harel and Rumpe (2004) for further details of these concepts.

---

\(^5\) This view is applicable to modelling languages in general, including but not limited to IS analysis and design.
Table 3. Applying Modelling Language Concepts: A Example Using the ER model

<table>
<thead>
<tr>
<th>Concepts</th>
<th>Examples in ER Modelling Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Syntactic Construct: Graphical notations such as Rectangle, Diamond, Oval, Line, Label, …</td>
</tr>
<tr>
<td></td>
<td>Syntactic Rule: An Oval must be connected to one and only one Rectangle or Diamond by a Line; A Diamond must be connected to one or more Rectangles with Lines; Each Rectangle, Diamond, and Oval must have a Label; …</td>
</tr>
<tr>
<td></td>
<td>Syntactic Expression: A Rectangle with a label “Person”; A Diamond with a label “Is Mother Of”; An Oval with a label “age”; …</td>
</tr>
<tr>
<td>Semantics</td>
<td>Semantic Domain: A semantic domain that consists of elements such as Entity Type, Relationship Type, Attribute, …</td>
</tr>
<tr>
<td></td>
<td>(Readers who are not familiar with the ER model may refer to (Chen, 1976) for definitions and other details.)</td>
</tr>
<tr>
<td></td>
<td>Semantic Mapping: A Rectangle with a label “Person” is mapped to the “Person” Entity Type; A Diamond with a label “Is Mother Of” is mapped to the “Is Mother Of” Relationship Type; An Oval with a label “age” is mapped to the “age” Attribute; …</td>
</tr>
</tbody>
</table>

The mapping between model and script can thus be explained by elements of the modelling language in the view of (Harel & Rumpe, 2004). On one hand, the mental constructs of the model are created as instances of elements of the semantic domain. On the other hand, the syntactic expressions of the script are composed of the syntactic constructs using the syntactic rules. The mappings between the mental constructs of the model and syntactic expressions of the script are defined by the semantic mapping.

2.5.3. The Mappings Between Model and Domain

Earlier in this section, we established that a model consists of mental constructs that represent certain aspects of the domain. Hence, there is a mapping between model and domain. In this subsection, we study the factors that influence this mapping.
According to the definition of model by Stachowiak (1973) and others that we surveyed in the previous section, a model has the “reduction” feature and represents only selected focal aspects of the domain. Thus, the focal aspects determines which types of phenomena of the domain are represented in the model, and hence is an influencing factor. Moreover, a model has the “pragmatic” feature that it is created for a certain purpose. Since the selected content of the model needs to support such purpose, the focal aspects is necessarily influenced by the purpose. Therefore, purpose is another influencing factor. In Table 2 from the previous section, we discussed the purposes and focal aspects of three different types of models (i.e., conceptual schema, i* model, and e³-value model). We will continue with this example here. Suppose three instances of the three types of models are developed for the same IS. Then, the three models share the same domain, but their contents are mapped to different things in the domain, because of their differences in purpose and focal aspects.

Moreover, in order to represent selected phenomena of a domain as a model, the phenomena need to be perceived by human first. Thus, factors that influence perception of the domain also influence the mapping between model and domain. Thus, IS researchers have realized for a long time that IS analysis is related to Ontology, “the branch of philosophy that deals with theories about the structure and behavior of the worlds that human perceive” (Wand & Weber, 2004). An important research topic in IS analysis is to articulate the ontological constructs that we need to describe perceptions of the domain, at various levels of abstraction (Wand & Weber, 2004). In IS analysis, an ontology is a collection of ontological constructs. When an individual modeler or a group of stakeholders

---

6 The i* and e³-value models are two specific types of domain understanding models.
adopt an ontology in the creation of a model, it become a factor that influences the perception of the domain and hence also the mapping between model and domain. Interested readers may refer to the Bunge’s ontology (Wand & Weber, 1993, Table 1) for an example of ontology and ontological constructs.

2.5.4. Semantic Domain and Ontology

In the two preceding subsections, we discussed semantic domain and ontology. In this subsection, we will discuss their similarities, differences, and connections.

In the literature, elements of a semantic domain are called semantic constructs, whereas elements of an ontology are called ontological constructs. The two types of constructs are similarly defined by different researchers. A semantic construct defines a concept for describing a certain type of things in a domain, for user of the corresponding language. Similarly, an ontological construct also defines a concept for describing a certain type of things in a domain, but for user of the corresponding ontology.

In fact, in the IS analysis literature, a semantic domain is commonly considered as a type of ontology. For example, when Gordijn and Akkermans (2001, p. 13) defined the e³-value modelling language, they named the semantic domain as “the e³-value ontology”. However, the reverse is not true: not every ontology is a semantic domain, because a semantic domain by its definition is part of a language, whereas an ontology can be used for purposes other than forming part of a language. For example, the Bunge’s ontology (Wand & Weber, 1993, Table 1) is proposed to describe all kinds of phenomena in the
world at a very general level, but has not been used as the semantic domain of any well-known modelling language yet.

Therefore, a semantic domain is a type of ontology that is used to define things that a language is designed to represent. What distinguishes semantic domain from other types of ontologies is not the nature of their elementary constructs, but the purpose for which the elementary constructs are put together for. Thus, for benefit of simplicity of the generalized view that we seek to build in the next section, we regard ontological constructs as the elements of both semantic domain and ontology, and regard the semantic domain as a type of ontology.

We comment that some research opportunities exist in studying the relationship between ontology (in general) and semantic domain. For example, as shown by the work of Wand and Weber (1993), one can evaluate a modelling language by mappings its semantic domain to a benchmark ontology (e.g., the Bunge’s ontology). As another example, a modelling language designer can explore the possibility of creating the semantic domain of a new modelling language based on an existing general ontology.

Finally, readers who are interested about the general types of ontologies can refer to Wand and Weber (2004) and Guarino (1998, §2.4), among other relevant publications.

2.5.5. Section Summary and Analysis

To summarize, in this section, we studied the representation and interpretation of IS analysis models. This is important because, in both research and practice, IS analysis
models often need to be represented as and interpreted from modelling scripts. In our study, we found many related concepts and relationship types, as illustrated in Figure 4.

**Figure 4. Concepts and Relationship Types – For Explicating the Representation and Interpretation of IS Analysis Models**

These concepts and relationship types have been studied in four parts in this section.

In the first part, we studied the distinction between model and script. This part is mainly related to four concepts and two relationship types. The concepts include IS analysis model, script, mental construct, and syntactic expression. There are two relevant relationship types. First, an IS analysis model consists of mental constructs. Second, a script consists of syntactic expressions. We used these concepts and relationship types to distinguish between model and script, and demonstrated that the distinction matters in a

---

7 In constructing the generalized view, we focus only on IS analysis models. Therefore, we used IS analysis model rather than model in this summary.
number of IS analysis research problems. This distinction indicates that, to study the representation and interpretation of IS analysis models, we need to study the mappings both between model and script and between model and domain.

In the second part, we studied the mapping between model and script, by studying the basic concepts of modelling languages. The concepts studied in this part include: modelling language, syntax, syntactic construct, syntactic rule, semantics, semantic domain, and semantic mapping. There are several relevant relationship types. First, a modelling language has two parts, syntax and semantics. Second, the syntax consists of a number of syntactic constructs and syntactic rules. Third, the semantics consists of a semantic domain and a number of semantic mappings. Fourth, a syntactic expression is composed of syntactic constructs according to certain syntactic rules. Fifth, an IS analysis model can be mapped to or from a script according to the semantic mappings.

In the third part, we studied the mapping between model and domain. The concepts studied in this part include: domain, purpose, focal aspects, ontology, and ontological construct. There are several relevant relationship types. First, an IS analysis model represents perceptions of a domain with respect to certain focal aspects. Second, the perception of the domain is influenced by the adopted ontology. Third, the focal aspects are determined by the purpose of the IS analysis model. Fourth, an ontology consists of ontological constructs. Fifth, a mental construct is an instance of an ontological construct.

In the fourth part, we studied the similarity, difference, and connections between the semantic domain concept studied in the second part and the ontology concept studied in the third part. Our study revealed a relationship type between the two, that a semantic domain is a type of ontology (which is used to define part of a modelling language).
These concepts and relationship types will be incorporated into the generalized view in the next section. Their definitions will also be presented there.

2.6. TOWARDS A GENERALIZED VIEW OF IS ANALYSIS MODELS

In the three preceding sections, we have pointed out several issues with the existing views of IS analysis models (i.e., conceptual models produced in the context of IS analysis), that they are either generic (i.e., not specific enough for guiding IS analysis) or limited (i.e., too specific to be applicable to all types of IS analysis models) in some way. We proposed remedies by selecting and studying some concepts that generalize about the common characteristics of all IS analysis models. We used examples to demonstrate the utilities of these concepts for understanding IS analysis models.

In this section, we will put all these concepts and the relevant relationship types together in a whole picture toward the construction of a generalized view of IS analysis models, in the following steps. First, we introduce the goals of the generalized view. Second, we determine the set of concepts in the generalized view, and organize them into categories. Third, we determine the set of relationship types in the generalized view. Fourth, we present a visualization of the generalized view. Finally, we define the concepts and relationship types in the generalized view.

Note: A preliminary version of the generalized view in this section had been presented in one of our workshop presentations (Sun et al., 2011) as “the unified view of conceptual modelling”.
2.6.1. The Goals of the Generalized View

In this subsection, we introduce the goals of the generalized view.

In each of the three preceding sections, we proposed solutions to certain issues regarding the understanding of IS analysis models being either generic or limited, and summarized our solutions with a set of concepts and relationship types (among these concepts) in the section summary. At the end of each section, we summarized the solutions proposed in the corresponding section. The generalized view will summarize all the solutions proposed in this chapter, by combining and organizing all the concepts and relationship types in all the section summaries.

Therefore, the goal of the generalized view is to serve as a whole picture for summarizing this chapter’s efforts made in improving the understanding of IS analysis models, by putting together the concepts and relationship types that are relevant to this chapter’s solutions to a number of important issues in IS analysis.

However, while we believe that the generalized view made good progress in shaping a general – rather than generic or limited – understanding of IS analysis models, its goal is not set unrealistically to solve every single issue regarding the general understanding of IS analysis models. Hence, we do not aim at including every single concept relevant to IS analysis models. For example, an IS analysis model may be created through a modelling method. Therefore, “modelling method” is a concept relevant to IS analysis models. However, this concept is not included in the generalized view, because we have decided to focus on the “product” aspect of conceptual modelling in this study, while “modelling method” is more relevant to the “process” aspect of conceptual modelling.
Nonetheless, we hope that future work by us and our colleagues will continue to study other concepts and relationship types that are helpful for shaping a better general understanding of IS analysis model, and use them to extend the generalized view.

2.6.2. The Concepts in the Generalized View

In this subsection, we will determine the set of concepts in the generalized view, and organize them into a few categories.

According to the goal of the generalized view introduced in the previous subsection, the concepts in the generalized view will include all concepts from the previous sections. To present them in an organized way, we classify the concepts into five categories. The definitions and contents of the categories are presented in Table 4.

Table 4. Categories of Concepts in the Generalized View

<table>
<thead>
<tr>
<th>Category</th>
<th>Definition of Category</th>
<th>Included Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>Concepts in this category describe characteristics of the specific context (e.g., a specific project) of the relevant IS analysis work.</td>
<td>Information System (IS), Stakeholder, Purpose, Focal Aspects, Domain</td>
</tr>
<tr>
<td>Theory</td>
<td>Concepts in this category describe characteristics of the theory (e.g., Bunge’s ontology) used to guide the relevant IS analysis work.</td>
<td>Ontological Construct, Ontology</td>
</tr>
<tr>
<td>Language</td>
<td>Concepts in this category describe characteristics of the modelling language (e.g., UML, ER model, ( \mathcal{F} ), and ( e^3 )-value.) that is used to represent the relevant IS analysis model.</td>
<td>Syntactic Rule, Syntactic Construct, Semantic Domain, Semantic Mapping, Syntax, Semantics, Modelling Language</td>
</tr>
<tr>
<td>Mental Product</td>
<td>Concepts in this category describe characteristics of the modelling products of the relevant IS analysis work, that exist internal to the mind.</td>
<td>Mental Construct, IS Analysis Model</td>
</tr>
<tr>
<td>External Product</td>
<td>Concepts in this category describe characteristics of the modelling products of the relevant IS analysis work, that are recorded as scripts in certain mediums external to the mind (e.g., recorded in paper, whiteboard, and digital file).</td>
<td>Syntactic Expression, Script</td>
</tr>
</tbody>
</table>
2.6.3. **The Relationship Types in the Generalized View**

In this subsection, we determine the set of relationship types in the generalized view. According to the goal of the generalized view introduced at the beginning of this section, the relationship types in the generalized view will include all relationship types from the previous sections.

We find that two relationship types are very similar in semantic meaning:

- An IS analysis model is created in the process of developing a specific new IS for some certain IS analysis relevant purpose. (From Figure 2.)
- Stakeholders create an IS analysis model for certain purpose. (From Figure 3.)

We think that they complement each other by describing different aspects of the creation of IS analysis models. Hence, in the generalized view, we combine them into a single relationship type that captures semantic meaning of both of them:

- Stakeholders create an IS analysis model for certain IS analysis relevant purpose in the process of developing a specific new IS. (To be included in the generalized view.)

We find that the other relationship types from the section summaries can be directly copied into the generalized view. To save space, we will not list every one of them here since we will visualize and define them in subsequent subsections.

2.6.4. **Visualizing the Generalized View**

In the two preceding subsections, we have determined the set of concepts of the generalized view, organized the concepts into five categories, and determined the set of
relationship types of the generalized view. In this subsection, we present a visualization of the generalized view in Figure 5. The legends in the diagram explains the mapping between mental constructs of the generalized view (i.e., concepts and relationship types) and syntactic expressions of the diagram (e.g., boxes, diamonds, and lines).

Figure 5. The Generalized View of IS Analysis Models

2.6.5. Defining the Elements of the Generalized View

In this subsection, we define the elements of the generalized view. The five concept categories have been defined earlier in this section. Hence, in this subsection, we only need to define concepts and relationship types of the generalized view, based on the literature as well as our discussions in the previous sections of this chapter. Because of the scope of this chapter, when defining the concepts and relationship types, we focus on the case that they
are used to describe something that is related to an IS analysis model, although some of them may be applicable to other types of models as well.

For the benefit of the readers, we present the definition of each concept as a bullet-point, and the definition of each relationship type as a sub-bullet-point attached to one of the associated concepts. This is because, the definitions are usually quite long and of varying lengths, thus making it awkward to present all of them in tables or other formats. Moreover, it is helpful for understanding to examine the definitions of associated concepts and relationship types together, rather than separately.

The definitions are presented below, organized according to the five concept categories as defined in Table 4 earlier in this section. To help the readers to better understand the generalized view, an example project will be presented in the next section to provide concrete example for each abstract concept.

Concepts in the “Context” category:

- **Domain**: “The collection of all objects (entities) that ever have been, are, or ever will be in a selected portion of a real world or postulated world of interest that is being described.” (van Griethuysen, 1982, §1.3).

- **Information System (IS)**: “A designed system that collects, stores, processes, and distributes information about the state of a domain.” (Olivé, 2007, p. 3).

- **Stakeholder**: In the scope of an IS development project, a stakeholder is a person, group, or organization that will affect or will be affected by the outcome of the project.

- **Focal Aspects**: The types of things in the domain that are selected to be represented in a certain IS analysis model.
- The Focal Aspects are determined by the Purpose: The focal aspects of an IS analysis model is determined by its purpose, because the types of things that are represented in the model need to support the purpose for which the model is created. For example, the purpose of a conceptual schema is to define the structure and behavior of the domain at the end of the project. Thus, the focal aspects of conceptual schemas should include both static aspects (such as entity types, relationship types, and attributes), and dynamic aspects (such as events) of the domain.

- **Purpose**: The intended use of an IS analysis model. For example, the purpose of a conceptual schema is to define the structure and behavior of the domain at the end of the project.

Concepts in the “Theory” category:

- **Ontological Construct**: An ontological construct is an element of an ontology, and is used to articulate a certain type of phenomena in the world regarding the nature and structure of the world.

- **Ontology**: An ontology is a model that articulates the nature and structure of the world (Wand & Weber, 2004). An ontology, for people who adopt it, provides a theory about what types of phenomena exist in a domain, as well as a vocabulary for describing these types of phenomena.

  - An Ontology consists of Ontological Constructs: This is implied by the definition of the two concepts.

  - The perceptions of Domain are influenced by an Ontology: When people adopt an ontology in perceiving a domain, their perception of the domain are
influenced by the ontology, because the ontology provides a theory about what types of phenomena exist in the domain.

Concepts in the “Language” category:

- **Syntactic Construct**: An elementary symbol of a modelling language that can be combined with other syntactic constructs to form syntactic expressions, according to certain syntactic rules.

- **Syntactic Rule**: A rule according to which syntactic constructs are combined together to form syntactic expressions.

- **Syntax**: The syntax of a modelling language provides a mechanism to form syntactic expressions. It consists of all of its syntactic constructs and syntactic rules.
  
  - The Syntactic Constructs are parts of the Syntax; the Syntactic Rules are parts of the Syntax: the syntax of a modelling language consists of a collection of syntactic constructs and a collection of syntactic rules.

- **Semantic Domain**: A definition of the types of things in the world that the modelling language is designed to represent.
  
  - A Semantic Domain is a type of Ontology: A semantic domain is a type of ontology that is used to define the types of things in the world that a certain language is designed to represent.

- **Semantic Mapping**: A mapping between syntactic expressions and mental constructs. For example, under the semantic mapping of the entity-relationship model, a box with a label “Person” (a syntactic expression) is mapped to an entity type named as “Person” (a mental construct).
• **Semantics**: The semantics of a modelling language consists of its semantic domain and all of its semantic mappings.
  
  o The Semantic Mappings *are parts of* the Semantics; the Semantic Domain *is a part of* the Semantics: The semantics of a modelling language consists of a semantic domain and a number of semantic mappings.

• **Modelling Language**: The language in which models are represented as scripts.
  
  o A Syntax *is a part of* a Modelling Language; A Semantics *is a part of* a Modelling Language: A modelling language consists of two parts: syntax and semantics.

Concepts in the “**Mental Product**” category:

• **Mental Construct**: A construct that people form in their mind to describe their perceptions of certain phenomena of a certain domain.
  
  o A Mental Construct is an *instance of* an Ontological Construct: When people adopt an ontology in creating a model, their perception of the corresponding domain is influenced by the ontology, and a mental construct of the created model will be an instance of an ontological construct of the adopted ontology.

  For example, when a person adopt the Bunge’s ontology to perceive the world, he may create in his mind a “person” class as a mental construct to describe his perception of persons in the domain. This “person” class is an instance of the “class” ontological construct in the Bunge’s ontology.

• **IS Analysis Model**: A type of conceptual model that is produced in the process of developing a specific new IS by representing selected focal aspects of the perceptions of a certain domain, for certain purpose related to analyzing the requirements of the IS.
The Stakeholders *create* an IS Analysis Model *in the process of developing a specific new IS for* certain IS analysis relevant purpose: This is implied by the definition of IS analysis model.

- An IS Analysis Model *consists of* Mental Constructs: This describes the nature of an IS analysis model as a collection of mental constructs.
- An IS Analysis Model *represents* perceptions of a certain Domain *with respect* to selected Focal Aspects: This is implied by the definition of IS analysis model and Focal Aspects.

Concepts in the “External Product” category:

- **Syntactic Expression**: An expression written in a modelling language, that is formed by combining a number of syntactic constructs according to the syntactic rules of that language.
  - A Syntactic Expression *is composed of* Syntactic Constructs *according to* Syntactic Rules: This is implied by the definition of syntactic expression.
- **Script**: A script consists of a number of syntactic expressions recorded in a certain external (i.e., outside of the mind) medium (e.g., paper, whiteboard, and digital file) so as to represent a model.
  - A Script *consists of* Syntactic Expressions: This describes the nature of a script as a collection of syntactic constructs recorded in an external medium.
  - A Script *is mapped to or from* an IS analysis Model *according to* the Semantic Mapping: In IS analysis, a script is an external representation of an IS analysis model. It is written in a modelling language whose semantic mapping defines the mapping between the elements of the script and the elements of the
represented model. For example, the mapping between an Entity-Relationship diagram (i.e., a script) and the corresponding IS analysis model is defined by the semantic mapping component of the Entity-Relationship Model (i.e., a modelling language).

2.7. AN EXAMPLE PROJECT

In this subsection, we present an example project of a family tree IS for two purposes. The first purpose is to provide concrete examples to illustrate the abstract concepts in the generalized view. The second purpose is to demonstrate applying the generalized view to compare different IS analysis models for similarities and differences.

To help readers focus on understanding the generalized view rather than this example project itself, we make this example project as simple as possible, and fictitious rather than realistic. This example project is similar to a case study, except that case studies are based on observations of real-world phenomena in their natural settings (Benbasat, Goldstein, & Mead, 1987, p. 371).

2.7.1. Background of the Example Project

The background of this (fictitious) example project is as follows. A big family has decided to develop a family tree IS to keep track of the family members, and certain relationships among them. They have decided to use plain English (a natural language) to
informally represent the requirements (a type of IS analysis model), and use the Entity-Relationship Model to formally represent the conceptual database schema (another type of IS analysis model) of the IS. Because of their education background, the family members adopt the Bunge’s ontology (Wand & Weber, 1990a, 1993) in perceiving the world. (They have also created other IS analysis models for the project, such as a domain understanding model represented as an i* (Yu, 2010) diagram. However, for sake of simplicity, we will not include them in this section.)

Note: While English is not originally designed as a modelling language, it can be and is sometimes used as one. In this section we are using English as a modelling language to informally represent the requirements.

2.7.2. Examples of Concepts in the Generalized View

Based on this example project, we can provide examples for the concepts in the generalized view. Moreover, we can use concepts in generalized view to compare the two IS analysis models (i.e., the requirements and the conceptual database schema) of this project for similarities and differences. In the rest of this subsection, we present the discussion for each category of concepts in the generalized view.

2.7.2.1. Examples of Concepts in the “External Product” Category

We start the discussion by presenting in Table 5 the modelling scripts of the two models, and give an example syntactic expression of each script.
Table 5. Examples of Concepts in the “External Product” Category

<table>
<thead>
<tr>
<th>Script</th>
<th>Requirements</th>
<th>Conceptual Database Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1.</td>
<td>The user can query the IS about personal information of a given family member.</td>
<td></td>
</tr>
<tr>
<td>R2.</td>
<td>The user can query the IS about which other family members are related to a given family member in a marriage or parental relationship.</td>
<td></td>
</tr>
<tr>
<td>Syntactic Expression</td>
<td>R1. The user can query the IS about personal information of a given family member.</td>
<td></td>
</tr>
</tbody>
</table>

As shown in Table 5, the two IS analysis models are represented by two different modelling scripts. The requirements is represented in a textual script that consists of syntactic expressions written as English sentences, whereas the conceptual database schema is represented in a diagrammatic script that consists of syntactic expressions written as geometric symbols (e.g., rectangular, diamond, and oval).

2.7.2.2. Examples of Concepts in the “Language” Category

In this example project, the conceptual database schema is written in the Entity-Relationship Model, while the requirements is written in plain English. Here, the Entity-Relationship Model and English are two instances of modelling languages that are used to externally represent IS analysis models as scripts. In Table 3, we have already used the Entity-Relationship Model and its components to provide examples for concepts in the language category of the generalized view.
The concepts in the “language” category of the generalized view can be used to compare different modelling languages. However, since the differences between the Entity-Relationship Model and the English language are so obvious, a comparison between them would be trivial. Thus, we omit their comparison here.

Instead, we briefly demonstrate using concepts in the “language” category to analyze the work of Evermann and Wand (2005). They proposed a variation of the UML language for modelling organizational environments. Essentially, they have created a modelling language for modelling the organizational environment, that shares the same syntax (including syntactic constructs and syntactic rules) with UML, but with different semantics (including semantic domain and semantic mapping). Their semantic domain consists of real-world things in an organizational environment, whereas UML’s original semantic domain consists of software artifacts. Under their semantic mapping, syntactic expressions written in UML are mapped to real-world things in an organizational environment, rather than software artifacts under the original UML semantic mapping.

2.7.2.3. Examples of Concepts in the “Theory” Category

In this project, in creating both the requirements and the conceptual database schema, the stakeholders have used the same Bunge’s ontology in perceiving the world. The ontological constructs of the Bunge’s ontology had been presented in Wand and Weber (1993, Table 1), and provide examples for concepts in the theory category.
2.7.2.4. Examples of Concepts in the “Mental Product” Category

The requirements and the conceptual database schema are two examples of the “IS Analysis Model” concept. They consist of different types of mental constructs. The requirements consists of mental constructs that are conditions for expressing the desires of the stakeholders. In comparison, the conceptual database schema consists of mental constructs (e.g., entity type, relationship type, and attributes) for representing the structure of the domain.

2.7.2.5. Examples of Concepts in the “Context” Category

In this example project, the requirements and the conceptual database schema are developed by the same group of stakeholders for the same IS in the same domain. The two models differ in their purposes and focal aspects. In Table 6, we compare the two models for similarities and differences, with respects to the concepts in the “context” category of the generalized view.

Table 6. Examples of Concepts in the “Context” Category

<table>
<thead>
<tr>
<th></th>
<th>Requirements</th>
<th>Conceptual Database Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Information System (IS)</strong></td>
<td>The family tree information system.</td>
<td></td>
</tr>
<tr>
<td><strong>Stakeholder</strong></td>
<td>The family members.</td>
<td>The developers hired to develop the IS.</td>
</tr>
<tr>
<td></td>
<td>The social environment of the family.</td>
<td></td>
</tr>
<tr>
<td><strong>Domain</strong></td>
<td>To express stakeholders’ desires about the domain, that will be satisfied by the IS.</td>
<td>To model the static aspects of the domain that will be represented by the IS.</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Conditions expressing stakeholders’ desires about the domain.</td>
<td>entity types, relationship types, attributes</td>
</tr>
<tr>
<td><strong>Focal Aspects</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.7.3. Summary of the Example Project

In summary, in this section, we have presented an example project and two IS analysis models of this project: requirements and conceptual database schema. We have discussed the two models to illustrate the abstract concepts in the generalized view with concrete examples.

In addition, we have demonstrated using concepts in the generalized view to compare different IS analysis models for similarities and differences. As we have elaborated at the end of §2.2, unlike our work, no previous work on comparing modelling products was tailored for the particular scope of IS analysis, and thus no previous work can be used to compare all types of modelling products in IS analysis.

2.8. CONCLUSION OF CHAPTER

2.8.1. Summary

In summary, in this chapter, we have proposed the generalized view of modelling in IS analysis, by studying the relevant concepts and relationship types. Unlike previous generic or limited views of modelling proposed or adopted by existing publications in IS analysis, the generalized view is specially tailored to the context of IS analysis, and at the same time is applicable to all types of modelling products in IS analysis. The work in this chapter mainly consists of the following parts.
In the first part (§2.3), we pointed out that the state of the art definition of conceptual modelling in IS analysis suffers from being generic, that it applies to many types of conceptual models, including but not limited to those produced in IS analysis. To remediate, we studied the concepts and relationship types that can be used to distinguish models produced in IS analysis versus in a generic context. We used them to explain why certain models should not be regarded as modelling products of IS analysis.

In the second part (§2.4), we were concerned that the area-specific views of modelling in IS analysis suffer from being limited, that they may not apply beyond the limited scope of their corresponding areas. To tackle this issue, we reviewed the definitions of modelling and model that are proposed or adopted by the literature related to IS analysis, and extracted from them the common characteristics of IS analysis models. We demonstrated that some of them can be used to compare different types of IS analysis models for similarities and differences.

In the third part (§2.5), we studied the concepts and relationship types that are relevant to the representation and interpretation of IS analysis models. First, we clarified the distinction between models and scripts, and explained why the distinction matters in the general scope of IS analysis. Second, we studied the mapping between model and scripts. Third, we studied the mapping between model and (perception of) domain.

In the fourth part (§2.6), we built the generalized view of modelling in IS analysis, by putting together the concepts and relationship types studied in the first three parts.

Finally, in the fifth part (§2.7), we presented an example project to provide concrete examples for the abstract concepts in the generalized view, and to demonstrate applying the generalized view to compare different IS analysis models.
2.8.2. Academic Contributions

The generalized view consists of concepts and relationship types that are useful for understanding conceptual models produced in the specific yet general context of IS analysis. For the general context of IS analysis, the existing views of conceptual models suffer from being either generic for not tailored to the specific context of IS analysis, or limited for not applicable to all types of IS analysis models. In contrast, the generalized view is constructed to be neither generic nor limited. Therefore, the generalized view is specific enough for guiding modelling in the specific context of IS analysis, and at the same time is general enough for being applicable to all types of IS analysis models.

The goal of the generalized view in this chapter is not set to incorporate every single concept applicable to IS analysis models\(^8\), but is set to summarize the concepts studied in depth in this chapter that are particularly important for improving the state of the art of general understanding of IS analysis models (e.g., concepts such as model, script, purpose, and focal aspects), along with some relevant basic concepts related to IS analysis models that have already been explicated in the existing literature (e.g., concepts such as domain, syntax, and semantics). In Table 7, we summarize the issues that have been addressed in this chapter to improve the general understanding of IS analysis models.

The generalized view is a summary of the relevant concepts and relationship types that are helpful for solving these issues. It has made the following additional contributions on top of the preceding parts of the chapter. First, it classified the relevant concepts into

---

\(^8\) For example, we explained earlier in this chapter that “modelling method” is a concept relevant to IS analysis models. However, we have not included it in the generalized view, because we have decided to focus on the “product” aspect of conceptual modelling, while “modelling method” is more relevant to the “process” aspect of conceptual modelling.
five categories (i.e., context, mental product, external product, theory, and language). Second, it organized the relevant concepts and relationship types in a whole picture, and visualized the connections among the relevant concepts. Third, as we discussed in the previous section, it can be used as an analytical framework to compare IS analysis models, while no existing framework was proposed for exactly the same scope and purpose.

Table 7. A Summary of Improvements on the General Understanding of IS Analysis Models

<table>
<thead>
<tr>
<th>Issue</th>
<th>Remedy</th>
<th>Contribution</th>
<th>Concepts &amp; Relationship Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>The state of the art general definition of conceptual modelling is</td>
<td>We proposed a supplement with an explicit statement of the IS analysis context.</td>
<td>We showed that the supplement distinguishes IS analysis models from other types of conceptual models.</td>
<td>Purpose, Information System, IS Analysis Model. See Figure 2 for relevant relationship types.</td>
</tr>
<tr>
<td>generic, and does not distinguish IS analysis models from other types of conceptual models.</td>
<td>(§2.3.3)</td>
<td>(§2.3.4)</td>
<td></td>
</tr>
<tr>
<td>Inconsistency among specific but limited views of conceptual modelling from different areas of IS analysis.</td>
<td>We searched for the common characteristics for comparing IS analysis models for similarities and differences.</td>
<td>We showed that the common characteristics can be used to understand the similarities and differences among different types of IS analysis models.</td>
<td>Purpose, Focal Aspects. See Figure 3 for relevant relationship types.</td>
</tr>
<tr>
<td>The existing general definitions do not agree on whether IS analysis models are mental or external in nature.</td>
<td>We clarified that IS analysis models are mental, and are represented by external scripts.</td>
<td>We showed that in certain IS analysis research problems, it is important to bear this distinction in mind.</td>
<td>Model, Script, Mental Construct, Syntactic Expression. See Figure 4 for relevant relationship types.</td>
</tr>
<tr>
<td>Semantic domain and ontology are two similar concepts. Their connection has not been explicitly discussed in existing literature.</td>
<td>We pointed out that a semantic domain is a type of ontology for defining things that a language is designed to describe.</td>
<td>We explicated the similarity, difference, and connection between semantic domain and ontology.</td>
<td>Semantic Domain, Ontology. See Figure 4 for relevant relationship types.</td>
</tr>
</tbody>
</table>
3.1. INTRODUCTION

3.1.1. Motivation

We surveyed the literature and found three areas of IS analysis: requirements engineering, conceptual schema modelling, and domain understanding. Several modelling methods have been proposed in each area. We found that the modelling methods in different areas focus on describing different aspects of the reality. In contrast, different modelling methods proposed in the same area focus on describing the same aspects of the reality in different ways. For example, modelling methods in the requirements engineering area focus on describing “the real-world goals for, functions of, and constraints on software systems” (Zave, 1997), whereas modelling methods in the conceptual schema modelling area focus on describing “the possible states of affairs of the universe of discourse [i.e., the domain] including the classifications, rules, laws, etc., of the universe of discourse [i.e., the domain].” (van Griethuysen, 1982, §1.3). Because of such commonality among different modelling methods in the same area, in some areas, general terminologies – that
are independent of any specific method – have been proposed as foundations to help researchers and practitioners to understand, use, and create specific modelling methods in that area. For example, Zave and Jackson (1997) proposed a general terminology for requirements engineering that is independent of any specific method.

However, we found that the general terminologies in existing literature are limited to the scope of their corresponding areas. The connections among different areas have not been discussed in detail. Such isolation creates difficulty for readers to obtain a whole picture of modelling in IS analysis.

Moreover, a general terminology widely accepted in an area may be inconsistent with one from another area. For example, the “Domain Knowledge” concept in requirements engineering is called “Domain Conceptual Schema” in conceptual schema modelling.

Therefore, in this chapter, we are motivated to consolidate the relevant general terminologies proposed separately in different areas related to IS analysis into one consolidated view of modelling in IS analysis, so as to provide a whole picture of modelling in IS analysis and to reconcile the inconsistencies among these areas. Since IS analysis requires expertise in multiple areas, we think that the consolidation will be helpful for improving inter-area collaboration and the overall IS analysis process.

### 3.1.2. Overview of Method

To perform the consolidation, we will start with a survey of the literature so as to find out which areas are related to modelling in system analysis. For each related area, we
will choose or – if necessary – propose a general terminology for that area. We will summarize the general terminology for each area related to system analysis as an area-specific perspective that consists of the relevant concepts and relationship types.

Subsequently, we will consolidate the area-specific perspectives into a consolidated view that can be used as a common terminology for all the relevant areas of IS analysis. The consolidation mainly consists of two operations: adaptation and union.

First, adaptations are made to reflect the scope of this chapter and to resolve inter-area inconsistencies. An information system is a specific type of system. Some of the area-specific perspectives are applicable to system analysis in general, including but not limited to IS analysis. They will be adapted to the more specific scope of modelling in IS analysis. For example, the requirements engineering perspective is applicable to the analysis of a fax machine, which is not an IS because it does not interpret the handled information as the state of any domain (Olivé, 2007, p. 3). Hence, the requirements engineering perspective is adapted to the more specific scope of modelling in IS analysis before integrated into the consolidated view. In addition, adaptations will be made to resolve inter-area inconsistencies, such as different naming of the same concept in different areas.

Second, once the adaptations are made, we will create a “union” (similar as the union operation in set theory) of the area-specific perspectives, and call it the consolidated view of modelling in IS analysis. The set of concepts and relationship types in the consolidated view is formed by taking the union of the sets of concepts and relationship types from all area-specific perspectives, after the adaptations. The purpose of creating this “union” is to create a consistent terminology for all relevant areas of IS analysis: requirements engineering, conceptual schema modelling, and domain understanding.
3.1.3. Outline of Chapter

The remaining sections of this chapter are organized as follows. First, we will survey the related work. Second, we will summarize the perspectives of modelling in related areas as area-specific perspectives. Third, we will consolidated the area-specific perspectives into the consolidated view of modelling in IS analysis. Fourth, we will present an example project to illustrate the concepts in the consolidated view with concrete examples, and to demonstrate applying the consolidated view in coordinating the use of multiple modelling methods in IS analysis. Fifth, we will apply the consolidated view to evaluate and compare two agent-oriented modelling methods. Finally, we will summarize this chapter and discuss its academic contributions.

3.2. RELATED WORK

In this section, we survey the related work and compare them to this chapter. We classify the related work into a few categories and discuss each category in a subsection.

3.2.1. General Terminologies of Areas Related to IS Analysis

In this chapter, we are motivated to build the consolidated view of modelling in IS analysis by consolidating the general terminologies of multiple areas related to IS analysis. The consolidated view differs from these general terminologies not in nature but in scope. In particular, the scope of our consolidated view includes terms useful for understanding modelling products in IS analysis. These terms may come from multiple related areas. In contrast, the scope of a general terminology of a related area includes terms useful for understanding various concepts of that particular area, including but not limited to
modelling products. A general terminology is one that is not tied to any specific method. For example, Zave and Jackson (1997) proposed a general terminology of the requirements engineering area, because it is applicable to all requirements engineering methods, rather than being tied to any specific method (e.g., the GBRAM method (Anton, 1996)).

We find three areas related to the scope of this chapter: requirements engineering, conceptual schema modelling, and domain understanding. In the next section, we will define the scope of this chapter and survey the general terminologies of these areas according to the scope. Thus, we will not dive into further details here.

### 3.2.2. Evaluation Frameworks of Modelling Grammars and Methods

As we will show in a subsequent section, the consolidated view can be applied to evaluate modelling methods regarding their support for multiple specific types of modelling products in IS analysis. Although previous work (such as the ones that we will survey later in this subsection) had been done on the topic of modelling grammar and method evaluation, none of them had addressed evaluating the support for specific types of modelling products in IS analysis. In this subsection, we survey the common evaluation topics with a few examples.

The first evaluation topic is the support for multiple types of modelling products. For example, many evaluation frameworks – such as Arazy and Woo (2002), Sturm and Shehory (2004), and Dam and Winikoff (2004) – had addressed evaluating the support for
three generic types of modelling products, i.e., analysis, design, and implementation models.

Our work on applying the consolidated view to evaluate modelling methods belongs to this topic. However, unlike our work, none of the previous frameworks had addressed evaluating the support for all of the specific types of modelling products in IS analysis, i.e., domain understanding, requirements, conceptual schema, functionality specification, and domain knowledge.

The second evaluation topic is on modelling grammars’ expressiveness, or the ability to represent multiple types of real world phenomena, by using certain theories as benchmarks. Some examples of the theories used in such evaluations include Bunge’s ontology (Wand & Weber, 1993), Searle’s social ontology (March & Allen, 2014), and concept theory (Parsons, 1996).

A model is created as a collection of selected types of phenomena in a domain for a specific purpose (Stachowiak, 1973). Since a system is more than the sum of its individual elements, the first evaluation topic on support for multiple types of models is different from the second evaluation topic on support for multiple types of phenomena.

The third evaluation topic is on the use of modelling grammars and methods in empirical studies. Some example frameworks are Gemino and Wand (2004), Burton-Jones et al. (2009), Kitchenham et al. (2002), and Bera, Burton-Jones, and Wand (2011).

In the first and second evaluation topics, researchers conduct theoretical evaluation of grammars and methods by comparing them against certain theoretical benchmarks, such as software process models or ontological theories. In contrast, in the third evaluation topic,
researchers conduct empirical evaluation of grammars and methods by analyzing data collected from the use of the grammars and methods in empirical studies. Our work on applying the consolidated view to evaluate modelling methods belongs to the first topic and evaluates IS analysis methods by comparing them to our consolidated view of modelling in IS analysis as a theoretical benchmark.

3.2.3. Multi-Product Development Methods

Some development methods are proposed to address multiple or all types of products in system development, such as analysis, design, and implementation models. Two examples of these methods are Gaia (Wooldridge et al., 2000) and Tropos (Castro et al., 2002; Giorgini, Mylopoulos, & Sebastiani, 2005). The classical development methods taught in software engineering textbooks – such as waterfall, prototyping, spiral development, and rapid application development – also belong to this category.

Similar to these methods, the consolidated view proposed in this thesis addresses all types of modelling products in IS analysis. However, these methods are different from the consolidated view. Each of them is essentially a particular way to produce multiple types of modelling products in IS analysis. When two such methods cover the same types of modelling products, they are substitutes to each other and are not meant to be used together. For example, waterfall and spiral development are two software engineering methods. They are two different ways to produce all the modelling products in a software project. In a project, developers may choose either of them, but not both at the same time.
In contrast, the consolidated view is essentially a general terminology of modelling in IS analysis. It is not associated with any particular method, and is applicable to all modelling methods related to IS analysis. In §3.5.4, we will demonstrate using the consolidated view along with other software engineering methods in an IS development project. In §3.6, we will demonstrate applying the consolidated view in evaluating and comparing two agent-oriented methods.

3.3. AREA-SPECIFIC PERSPECTIVES OF MODELLING IN SYSTEM ANALYSIS

Several academic areas are related to the study of modelling in IS analysis. Different areas have different goals and hence formed different perspectives of modelling. In this section, we seek to summarize the perspectives of modelling in three areas related to IS analysis: requirements engineering, conceptual schema modelling, and domain understanding. In the first subsection, we introduce our method for producing the summaries. In each subsequent subsection, we summarize an area-specific perspective.

3.3.1. Method for Summarizing Area-Specific Perspectives

We introduce our method for summarizing the area-specific perspectives in three parts: scope, presentation, and other considerations.

3.3.1.1. The Scope of the Summaries

We made the following decisions regarding the scope of the summaries.
Modelling can be studied from four aspects: product (or modelling script), grammar, method, and context (Wand & Weber, 2002). Our first decision is to focus on the product aspect. More specifically, we focus on the models that are commonly produced in the analysis phase of a typical system development project. This is because different areas are connected through such models: development activities studied in one area produces certain models as products that will be subsequently used as inputs for development activities studied in other areas.

However, models representing theories of IS analysis and design – such as the works of Wand, Monarchi, Parsons, and Woo (1995) and Wand and Weber (1990b) – are beyond the scope of this chapter because they are not produced in specific system development projects. When they are used in a specific system development project, they are used as inputs for providing theoretical guidance, rather than produced as products. The same reason applies to the exclusion of models for conducting empirical evaluations of IT systems – such as the Technology Acceptance Model (Davis, 1989; Davis et al., 1989), and the exclusion of models for conducting empirical evaluations of conceptual modelling methods – such as the one proposed by Gemino and Wand (2004). The reason for exclusion is that none of these models are produced as products in specific development projects.

Our second decision is to summarize each area by relying – if possible – mainly on one highly-cited peer-reviewed publication whose goal is to provide an overview of modelling products in that area. We may also supplement the summary by referring to other publications holding a compatible view, particularly those written or cited by the same authors of the main publication. This decision is based on several considerations. First, since the goal of an area-specific perspective is to obtain a whole picture of modelling in
one area, we have chosen publications with the same goal, rather than those focusing on a particular modelling method or a narrow aspect of modelling in an area. Second, peer-reviewed publications, particularly those highly-cited ones, are more likely to reflect a widely accepted view of that area. Third, even within the scope of an individual area, different publications may have inconsistent terminologies and perspectives. Resolving such *intra*-area inconsistency is different from our research goal of resolving *inter*-area inconsistency, and is not feasible given our resource. Suppose we resolved such *intra*-area inconsistency, then we must have introduced some new interpretations by us or an expert panel. But then data regarding the extent to which these new interpretations are accepted by others – suppose it would be feasible to solicit them – may not be as reliable as the citation data readily available for existing publications. Thus, we choose to rely mainly on one highly-cited publication for each area.

We believe that, at the time our research is conducted, the chosen publications are the best ones for capturing the dominant perspectives of their corresponding areas. But, we concede the possibility that future research may do a better job in capturing the dominant perspectives of these areas, the dominant perspectives may change overtime, and new areas may emerge in the future. However, in that case, researchers can still use the method described in this subsection to summarize the updated area-specific perspectives, and then use the method described in the next section to produce the updated consolidated view.

Through our study of the relevant publications, we have made our third decision to study the related areas which we categorized as having two categories of concepts and three categories of relationship types. In Table 8 below, we define these categories, and explicate
them with examples from the requirements engineering perspective (which will be presented in the next subsection).

Table 8. Categories of Concepts and Relationship Types in the Consolidated View

<table>
<thead>
<tr>
<th>Category</th>
<th>Definition</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conceptual Model</td>
<td>For a concept in this category, its instances are models produced in a project.</td>
<td>Requirements, Specifications, and Domain Knowledge.</td>
</tr>
<tr>
<td>System Being Modeled</td>
<td>For a concept in this category, its instances are systems being modeled in a project.</td>
<td>Environment (modeled by the requirements and the domain knowledge), and Machine (whose functions are modeled by the specifications).</td>
</tr>
<tr>
<td>Relationship Types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conceptualization</td>
<td>For a concept in this category, its instances are relationships between a conceptual model and a system, so that the model conceptualizes the system.</td>
<td>An example instance is specifications indicate the functions of the machine, i.e. a conceptualization relationship links the specifications and the machine.</td>
</tr>
<tr>
<td>Composition</td>
<td>For a concept in this category, its instances are relationships between either two systems or two models, so that one system or model is a part of the other system or model.</td>
<td>An example instance is a composition relationship between an environment and a machine, such that the machine becomes part of the environment at the end of the project.</td>
</tr>
<tr>
<td>Production</td>
<td>For a concept in this category, its instances are relationships between models, one of which is created by using one or more other models as input.</td>
<td>An example instance is a production relationship links domain knowledge, requirements, and specifications, in which the specifications are produced by refining the requirements with the domain knowledge.</td>
</tr>
</tbody>
</table>
3.3.1.2. The Presentation of the Summaries

We present each area-specific perspective as follows.

First, we introduce the corresponding area and how it is related to modelling in IS analysis. Second, we visualize the relevant concepts and relationship types in the area. Third, we define the concepts and relationship types. The definition of each relationship type in the “conceptualization” category is implied by the definition of the associated concept in the “conceptual model” category. Thus we only need to define the other relationship types, i.e., those in the “composition” and “production” categories, along with all the concepts. For example, in the conceptual schema modelling perspective, the functionality specification is defined as a concept whose instances are models that define the functions of IS’s. This implies a conceptualization relationship type between the functionality specification concept and the IS concept. This also implies that the relationship type can be defined as one whose instances are relationships between an instance of functionality specification and an instance of IS, that the instance of functionality specification conceptualizes the functions of the instance of IS. Fourth, additional discussions are given at the end if necessary.

3.3.1.3. Other Considerations for the Summaries

One thing we strive for in this section is to summarize the concepts, relationship types, and their definitions the same as they are in the original literatures, while minimizing our own interpretations as much as possible, except when necessary. Readers may notice that quotes are used quite intensively. The intention is to enable the readers, as well as the
authors themselves, to use the provided quotes as evidence to verify that the perspectives presented here are consistent with the existing literatures.

Nonetheless, we do not deny the importance of synthesizing the existing literature. In fact, this is the main contribution of this chapter. Before we present the synthesis in the next section as the consolidated view, however, we hope to build a solid foundation here.

3.3.2. The Requirements Engineering Perspective

“Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families.” (Zave, 1997, p. 315). In this subsection, we summarize the requirements engineering perspective based on Zave and Jackson (1997). (A preliminary version of this summary was presented in one of our workshop presentations (Sun et al., 2011).) They proposed the principle that “All the terminology used in requirements engineering should be grounded in the reality of the environment for which a machine is to be built.” According to this principle, they proposed and defined a common terminology of requirements engineering that “applies to all software development projects”. In this terminology, there are three products of requirements engineering: requirements, domain knowledge and specifications. Regarding each of these products as a model, we summarize this perspective in Figure 6.
The concepts in this perspective are defined as follows:

- **Environment**: “The environment is the portion of the real world relevant to the software development project” (Zave & Jackson, 1997, p. 24), whose “current behavior is unsatisfactory in some way” (Zave & Jackson, 1997, p. 7).

- **Machine**: “The machine is a computer-based machine that will be constructed and connected to the environment, as a result of the software development project” (Zave & Jackson, 1997, p. 24), “in such a way that the behavior of the environment becomes satisfactory” (Zave & Jackson, 1997, p. 7).

- **Requirements**: The collection of all statements describing phenomena of “the environment as we would like it to be because of the machine”, which are “intended to express the desires of the customer concerning the software development project”. (Zave & Jackson, 1997, pp. 24-25). Such described phenomena should be “observable at the interface between the environment and the machine, and nothing else about the machine.” (Zave & Jackson, 1997, p. 5). The requirements are not satisfied in the environment without the machine, but are expected to be satisfied in the environment with the machine.
• **Domain Knowledge**: The collection of all statements describing phenomena in “the environment as it would be without or in spite of the machine”, which are “intended to be relevant to the software development project.” (Zave & Jackson, 1997, pp. 24-25).

• **Specifications**: The refinement of the Requirements that not only still describes “the environment as we would like it to be because of the machine” as described by the requirements, but also are “intended to be directly implementable and to support satisfaction of the requirements” (Zave & Jackson, 1997, pp. 24-25). A statement is considered directly implementable if it can “be guaranteed or effected by a computer alone”. (Zave & Jackson, 1997, p. 19). Like in the requirements, phenomena described in the specifications should be “observable at the interface between the environment and the machine, and nothing else about the machine.” (Zave & Jackson, 1997, p. 5).

Thus, the specifications describe only external behavior of the machine.

The definition of each relationship type in the “conceptualization” category is implied by the definition of the associated concept in the “conceptual model” category. These relationship types are listed as follows:

• The Requirements *are satisfied by* the Environment, *if and only if with the Machine*.

• The Domain Knowledge describes *project-relevant machine-independent phenomena of* the Environment.

• The Specifications *is a directly implementable description of the external behavior of* the Machine.

The other relationship types are defined as follows:

• The Machine *is connected to and affects behavior of* the Environment: This is implied
by the definition of the machine concept.

- The Specifications *refine* and *imply satisfaction of* the Requirements *with support of* the Domain Knowledge: “The primary role of domain knowledge in requirements engineering is in supporting refinement of requirements to implementable specifications. Correct specifications, in conjunction with appropriate domain knowledge, imply the satisfaction of the requirements.” (Zave & Jackson, 1997, p. 2).

“Some requirements are already implementable, so they can be copied directly from” the set of requirements to the set of specifications. (Zave & Jackson, 1997, p. 20). Otherwise, the not directly implementable requirements are refined into directly implementable specifications with the support of the domain knowledge. (Zave & Jackson, 1997, §5).

Zave and Jackson (1997, p. 2) prescribed that “It is not necessary or desirable to describe (however abstractly) the machine to be built. Rather, the environment is described in two ways: as it would be without or in spite of the machine and as we hope it will become because of the machine.” Zave and Jackson (1997, p. 5) further explained that all the three products (requirements, domain knowledge, and specifications) are supposed to describe phenomena in the environment that are “observable at the interface between the environment and the machine”, and “to say anything else about the machine is regarded as implementation bias.” In particular, describing states of the machine “introduces serious implementation bias”, because they “are internal [to the machine] and not directly observable at the interface between the machine and its environment.” (Zave & Jackson, 1997, p. 6). Therefore, as shown in Figure 6, the requirements and the domain knowledge conceptualize the environment, rather than the machine; and the specifications
conceptualize the external behavior of the machine, rather than anything internal to the machine.

### 3.3.3. The Conceptual Schema Modelling Perspective

In this subsection, we summarize the conceptual schema modelling perspective mainly based on Olivé (2004), with supplement from Olivé (2007) and van Griethuysen (1982). (A preliminary version of this summary was presented in one of our workshop presentations (Sun et al., 2011).) “The main purpose of conceptual [schema] modelling is to elicit the conceptual schema of the corresponding information system.” (Olivé, 2007, p. 22). The conceptual schema must be defined for the subsequent development of information systems (Olivé, 2007, §1.2.6). This perspective is visualized in Figure 7.

**Figure 7. The Conceptual Schema Modelling Perspective**

The concepts in this perspective are defined as follows:

- **Domain**: “The collection of all objects (entities) that ever have been, are, or ever will be in a selected portion of a real world or postulated world of interest that is being described.” (van Griethuysen, 1982, §1.3). There are relationships between these objects. The objects and relationships are classified into concepts. (Olivé, 2007, p. 10).
“An information system ... must refer to the state of a certain domain, such as an organization. The term “domain” is also called “universe of discourse” (van Griethuysen, 1982, §1.2), “enterprise” (Burns et al., 1986, §2.2), or “object system” (Olivé, 2007, p. 3, §1.1) in related literature.

- **Information System (IS):** “An information system is a designed system that collects, stores, processes, and distributes information about the state of a domain.” “The usual constraint on the kind of information handled by an information system is that it must refer to the state of a certain domain.” (Olivé, 2007, p. 3).

- **Conceptual Schema:** “The description of the possible states of affairs of the universe of discourse [i.e., the domain] including the classifications, rules, laws, etc., of the universe of discourse [i.e., the domain].” (van Griethuysen, 1982, §1.3). Two guidelines have been prescribed for the scope of the conceptual schema. First, “All relevant general static and dynamic aspects, i.e. all rules, laws, etc., of the universe of discourse [i.e., the domain] should be described in the conceptual schema. The information system cannot be held responsible for not meeting those described elsewhere, including in particular those in application programs.” (van Griethuysen, 1982, §3.1, 100 Percent Principle). Second, “A conceptual schema should only include conceptually relevant aspects, both static and dynamic, of the universe of discourse [i.e., the domain], thus excluding all aspects of (external or internal) data representation, physical data organization and access as well as all aspects of particular external user representation such as message formats, data structures, etc.” (van Griethuysen, 1982, §3.1, 100 Percent Principle)."

---

9 The nature of this domain does not influence the definition of an IS. For many IS’s, the domain is an organization, but the definition does not exclude other domains, such as a vehicle, the atmosphere, or a chess game.” (Olivé, 2007, p. 3).
§3.1, Conceptualization Principle).

- **Domain Conceptual Schema**: “The DCS [i.e., domain conceptual schema] defines the knowledge about the domain, independent of the IS”. For example, the DCS may include definitions of the entity types, attributes, relationship types, integrity constraints, and domain event types in the domain (Olivé, 2004, p. 21).

- **Functionality Specification**: The functionality specification defines the functions that the IS must perform. (Olivé, 2004, p. 20). For example, this may include derivation rules, query event types, generating conditions, and “the delimitation of the relevant fragment of the DCS [i.e., domain conceptual schema] that must be represented in the IS.” (Olivé, 2004, p. 22).

The definition of each relationship type in the “conceptualization” category is implied by the definition of the associated concept in the “conceptual model” category. These relationship types are listed as follows:

- The Conceptual Schema is a complete conceptualization of the Domain.
- The Domain Knowledge conceptualizes the IS-independent knowledge about the Domain.
- The Functionality Specification conceptualizes the functions of the IS.

The other relationship types are defined as follows:

- An IS represents, informs about, and changes the state of the Domain: An IS becomes part of the domain at the end of a project and is “considered to have three main functions: 1) Memory: to maintain a representation of the state of a domain. 2) Informative: to provide information about the state of a domain. 3) Active: to perform
actions that change the state of a domain.” (Olivé, 2007, p. 3).

- The Domain Conceptual Schema is part of the Conceptual Schema; the Functionality Specification is part of the Conceptual Schema: “The DCS and the FS are the two components of a CS.” “Every piece of knowledge in the CS is part of either the DCS or the FS.” “Conceptual Schema = Domain Conceptual Schema + Functionality Specification.” (Olivé, 2004, pp. 20-21).

- The Functionality Specification includes delimiting the part represented by the IS in the Domain Conceptual Schema: The functionality specification includes “the delimitation of the relevant fragment of the DCS [Domain Conceptual Schema] that must be represented in the IS.” (Olivé, 2004, p. 22). For example, a particular domain conceptual schema may include three entity types – Supplier, Customer, and Visitor. Say that only Supplier and Customer must be represented in the IS to support the IS’s functions. Then Supplier and Customer are also included in the functionality specification, while Visitor is not.

3.3.4. The Domain Understanding Perspective

In conceptual schema modelling, the stakeholders produce the conceptual schema to model the new domain that they want to create at the end of the project so as to satisfy their requirements. Moreover, before the requirements are known, the stakeholders may also produce other types of conceptual models to understand certain aspects of the domain so as to facilitate eliciting the requirements. For examples of the relevant conceptual
modelling methods, readers may refer to $i^*$ (Yu, 1993, 1997) & (Yu, 2010, §2.4) and e³-value (Gordijn & Akkermans, 2001; Gordijn & Akkermans, 2003).

Since requirements are desired conditions not satisfied by the domain at the beginning of the project, understanding the domain at the beginning of the project provides the foundation for expressing the requirements as such conditions, and thus is arguably an indispensable step for eliciting the requirements. For example, before stating the requirement that “the IS notifies the store owner about low stock items”, the stakeholders have to understand the relevant things and attributes in the domain such as items, store owner, and stock quantity.

However, we have not found any peer-reviewed academic publication that generalizes about conceptual models created to understand the domain at the beginning of the project so as to elicit the requirements, without being tied to any specific modelling method (e.g., $i^*$ or e³-value).

Therefore, in this subsection, we define “Domain Understanding” as the concept whose instances are conceptual models created to understand the domain at the beginning of a certain project, from which the requirements can be elicited. Within the scope of IS analysis, this concept is similar to but different from the “Domain Knowledge” concept in the requirements engineering perspective, aka the “Domain Conceptual Schema” concept in the conceptual schema modelling perspective. In a project, a domain understanding and a domain knowledge both conceptualize knowledge of the same domain excluding the IS’s functions. However, the former is conceptualized before the IS’s functions are conceptualized, for the purpose of eliciting the requirements. In contrast, the latter is conceptualized together with the IS’s functions in the “Conceptual Schema”, for the
purpose of designing a domain that satisfies a given set of requirements. Some knowledge may be useful for both purposes and thus may be copied from the former to the latter.

To summarize the above discussions, we present the domain understanding perspective in Figure 8. It is applicable to – but not limited by – the scope of IS development, because the stakeholders may plan to change the domain by IS development as well as other means, such as business process reengineering without IS.

**Figure 8. The Domain Understanding Perspective**

The concepts in this perspective are defined as follows:

- **Domain**: A portion of a social or physical world that the stakeholders are motivated to understand and change in the project.

- **Domain Understanding**: A model conceptualizing the stakeholders’ understanding of the domain at the beginning of the project. For the domain at the end of the project, some of these knowledge – included in the domain understanding – may no longer apply, while other new knowledge – not included in domain understanding – may apply.

- **Requirements**: A model conceptualizing certain conditions expressing how the stakeholders would like the domain to behave at the end of the project. This set of conditions is not satisfied by the domain at the beginning of the project.
• **Conceptual Schema**: A model conceptualizing stakeholders’ design of the structure and behavior of the domain at the end of the project. The design may or may not include an IS, along with other things. The content of the model may include – for example – entity types, relationship types, event types, and laws of the domain.

The definition of each relationship type in the “conceptualization” category is implied by the definition of the associated concept in the “conceptual model” category. These relationship types are listed as follows:

- The Domain Understanding *conceptualizes stakeholders’ understanding of* the Domain, *at the beginning of the project.*
- The Requirements *conceptualizes conditions expressing stakeholders’ desire of* the Domain.
- The Conceptual Schema *conceptualizes the stakeholders’ design of the structure and behavior of* the Domain, *at the end of the project.*

The other relationship types are defined as follows:

- The Requirements *are elicited using* the Domain Understanding: The stakeholders may elicit the requirements from the domain understanding.
- The Conceptual Schema *refines and implies satisfaction of* the Requirements: The conceptual schema maps (or refines) the requirements to a design of the structure and behavior of the domain at the end of the project, so that a domain designed accordingly satisfies the requirements.
3.4. A CONSOLIDATED VIEW OF MODELLING IN IS ANALYSIS

In the previous section, we have summarized a number of area-specific perspectives of modelling. In this section, we build a consolidated view that is the union of all these perspectives, within the scope of modelling in IS analysis.

The method (goals and process) for building the consolidated view is explained in the first subsection. The process consists of four steps: determining the set of concepts, determining the set of relationship types, visualizing the consolidated view, and defining the concepts and relationship types. The result of each step is presented in a subsequent subsection. Some comments are made in the last subsection.

3.4.1. Method for Building the Consolidated View

In this subsection, we describe the method at a high level in two parts. First, we describe the scope and goals of the consolidated view. Second, we describe the steps of the consolidation process.

3.4.1.1. Scope and Goals for Building the Consolidated View

The area-specific perspectives – from the previous section – provide the foundation for defining the scope of the consolidated view. We define its scope by defining and striving for the following four goals.

- **IS Focus**: Modelling for system analysis is the common theme among all area-specific perspectives in the previous section. Some perspectives (e.g., requirements engineering) are applicable to not only IS analysis but also system analysis in general. However,
since the focus of this chapter is on IS, the concepts and relationship types from the area-specific perspectives will be adapted to the situation where they are applied in IS analysis, if necessary.

- **Solid Foundation**: The consolidated view should be well grounded on the area-specific perspectives. This goal will be achieved by ensuring that every concept or relationship type in the consolidated view can be mapped to at least one concept or relationship type in at least one of the area-specific perspectives.

- **Comprehensiveness**: The consolidated view should include all concepts and relationship types from all area-specific perspectives. This goal will be achieved by ensuring that every concept or relationship type from every area-specific perspectives can be mapped to at least one concept or relationship type in the consolidated view.

- **Compatibility**: The definition of a concept or relationship type in the consolidated view should not contradict the definitions of the corresponding concepts or relationship types in the area-specific perspectives.

Moreover, we will also bear the following **additional goals** in mind:

- **Consistency**: The area-specific perspectives may be inconsistent with one another. For example, the “Domain Knowledge” concept in the requirements engineering perspective is called the “Domain Conceptual Schema” concept in the conceptual schema modelling perspective. The consolidated view should reconcile any inter-area inconsistency.

- **Parsimony**: As we explained in the previous section, in defining the area-specific perspectives, we intensively employed quotes and avoided interpretation for sake of
fidelity and other considerations. In defining the consolidated view, we will not hesitate to rephrase for sake of parsimony. The quotes and citations from the area-specific perspectives are the sources for definitions of the consolidated view.

3.4.1.2. **Process for Building the Consolidated View**

We build the consolidated view in four steps.

The first step is to determine the set of concepts in the consolidated view. Since the consolidated view is a union of all the area-specific perspectives, we start with all concepts from all perspectives, adapt them according to the goals of the consolidated view, and merge concepts that are the same within the scope of the consolidated view.

The second step is to determine the set of relationship types in the consolidated view. The procedure is the same as in the first step.

The third step is to visualize the concepts and relationship types in the consolidated view. This is done in the same way as we did for visualizing the area-specific perspectives.

The fourth step is to define the concepts and relationship types in the consolidated view, based on the original definitions from the area-specific perspectives. The original definitions are rephrased, if necessary, according to the goals of the consolidated view. However, the definitions in the consolidated view should not contradict the corresponding definitions in the area-specific perspectives.

3.4.2. **Concepts in the Consolidated View**

In this subsection, we present the result of the first step – determining the set of concepts in the consolidated view. The concepts in the consolidated view and their mapping to concepts in the area-specific perspectives are presented in Table 9. In the third
column of this table, an “N / A” indicates that a concept in the consolidated view (referenced in the first column of the same row) cannot be mapped to any concept in an area-specific perspective (referenced in the second column of the same row). This reflects not a contradiction, but a difference in the scope between the consolidated view and the area-specific perspectives.

Table 9. Concepts in the Consolidated View

<table>
<thead>
<tr>
<th>Concept in the Consolidated View</th>
<th>Concept in the Area-Specific Perspectives</th>
<th>Concept in the Area-Specific Perspectives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requirements Engineering</td>
<td>Conceptual Schema Modelling</td>
</tr>
<tr>
<td>Domain</td>
<td>Environment</td>
<td>Domain</td>
</tr>
<tr>
<td>Information System (IS)</td>
<td>Machine</td>
<td>Information System (IS)</td>
</tr>
<tr>
<td>Domain Understanding</td>
<td>N / A</td>
<td>N / A</td>
</tr>
<tr>
<td>Requirements</td>
<td>Requirements</td>
<td>N / A</td>
</tr>
<tr>
<td>Conceptual Schema</td>
<td>N / A</td>
<td>Conceptual Schema</td>
</tr>
<tr>
<td>Domain Knowledge</td>
<td>Domain Knowledge</td>
<td>Domain Conceptual Schema</td>
</tr>
<tr>
<td>Functionality Specification</td>
<td>Specifications</td>
<td>Functionality Specification</td>
</tr>
</tbody>
</table>

We found that the mapping of all concepts are straightforward, after adapting some concepts from the area-specific perspectives to the situation of IS development. In particular, the requirements engineering perspective is applicable to the development of IS as well as other types of machines\(^\text{10}\), and the domain understanding perspective is applicable to projects with as well as without machine development\(^\text{11}\). Concepts from these two perspectives are adapted to the more specific situation of IS development.

In the literature, the mapping between the requirements engineering and conceptual schema modelling perspectives had been discussed in (Olivé, 2004), among others.

\(^\text{10}\) For example, the requirements engineering perspective is applicable to the development of fax machines, which are not IS because they do not interpret the handled information as representing the state of any domain (Olivé, 2007, p. 3).

\(^\text{11}\) For example, some business process reengineering projects may not involve any machines.
According to them, the “domain knowledge” and “specifications” in requirements engineering map to the “domain conceptual schema” and “functionality specification” in conceptual schema modelling, respectively.

### 3.4.3. Relationship Types in the Consolidated View

In this subsection, we present the result of the second step – determining the set of relationship types in the consolidated view.

Since there is one-to-one correspondence between relationship types in the “conceptualization” category and concepts in the “conceptual model” category, the mapping of the former can be directly derived from that of the latter, which has been presented in the previous subsection. To save space, we do not present these relationship types and their mapping here. They are included in the visualization and definition of the consolidated view in the next two subsections.

The other relationship types in the consolidated view and their mapping to relationship types in the area-specific perspectives are presented in Table 10. In the third column of this table, “N / A” indicates that a relationship type in the consolidated view (referenced in the first column of the same row) cannot be mapped to any relationship type in an area-specific perspective (referenced in the second column of the same row). This reflects not a contradiction, but a difference in the scope between the consolidated view and the area-specific perspectives.
Table 10. Relationship Types in the Consolidated View, Except Those in the “Conceptualization” Category

<table>
<thead>
<tr>
<th>Relationship Types in the Consolidated View</th>
<th>Relationship Types in the Area-Specific Perspectives</th>
<th>Requirements Engineering</th>
<th>Conceptual Schema Modelling</th>
<th>Domain Understanding</th>
</tr>
</thead>
<tbody>
<tr>
<td>The IS represents, informs about, and changes the state of the Domain</td>
<td>The Machine is connected to and affects behavior of the Environment</td>
<td>An IS represents, informs about, and changes the state of the Domain</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>The Requirements are elicited using the Domain Understanding</td>
<td>N/A</td>
<td>N/A</td>
<td>The Requirements are elicited using the Domain Understanding</td>
<td></td>
</tr>
<tr>
<td>The Conceptual Schema refines and implies satisfaction of the Requirements</td>
<td>The Specifications refine and imply satisfaction of the Requirements with support of the Domain Understanding</td>
<td>N/A</td>
<td>The Conceptual Schema refines and implies satisfaction of the Requirements</td>
<td></td>
</tr>
<tr>
<td>The Functionality Specification is the IS design relevant subset of the Conceptual Schema</td>
<td>N/A</td>
<td>Functionality Specification is part of the Conceptual Schema &amp; includes delimiting the part represented by the IS in the Domain Conceptual Schema</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>The Domain Knowledge is the IS-independent subset of the Conceptual Schema</td>
<td>N/A</td>
<td>The Domain Conceptual Schema is part of the Conceptual Schema</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

3.4.4. A Visualization of the Consolidated View

In this subsection, we present in Figure 9 the result of the third step – visualizing the concepts and relationship types in the consolidated view.
The relevant definitions are given in the next subsection.

### 3.4.5. Definitions of the Consolidated View

Earlier in this section, we have determined and visualized the concepts and relationship types in the consolidated view. In this subsection, we present the result of the fourth step – defining these concepts and relationship types based on the original definitions of their corresponding concepts and relationship types in the area-specific perspectives and the goals of the consolidated view.

In §3.4.2, we have determined the set of concepts in the consolidated view. In this subsection, we define them as follows. The definitions are adapted from the definitions of the corresponding concepts in the area-specific perspectives in §3.3, according to the goals listed in §3.3.1.

- **Domain**: “The collection of all objects (entities) that ever have been, are, or ever will be in a selected portion of a real world or postulated world” (van Griethuysen, 1982, §1.3) that the stakeholders are motivated to understand and change.
• **Domain Understanding**: A model conceptualizing the stakeholders’ understanding of the domain at the beginning of the project. For the domain at the end of the project, some of these knowledge – included in the domain understanding – may no longer apply, while other new knowledge – not included in the domain understanding – may apply.

• **Information System (IS)**: “A designed system that collects, stores, processes, and distributes information about the state of a domain.” (Olivé, 2007, p. 3).

• **Requirements**: A model conceptualizing conditions expressing how the stakeholders would like the domain to behave at the end of the project because of the IS and others changes made to the domain. This set of conditions is not satisfied at the beginning, but is expected to be satisfied at the end, of the project.

• **Conceptual Schema**: A model conceptualizing a design of the structure and behavior of the domain at the end of the project. This design includes an IS, other things, and their interactions (e.g., entity types, relationship types, attributes, integrity constraints, domain event types, action request event types, derivation rules); but excludes anything internal to the IS (for example data representation, data structure, and message format). The description of the IS’s interactions should be directly implementable\(^\text{12}\).

• **Functionality Specification**: A model conceptualizing all the functions the IS performs in the domain. Each described function should be directly implementable, i.e., each can be effected by an IS alone. Its content may include – for example – action

---

\(^{12}\) The term “directly implementable” will be explicated in the definition of the Functionality Specification.
request event types, derivation rules, and the fragment of the domain knowledge that must be represented by the IS.

- **Domain Knowledge**: The part of the conceptual schema that is independent of the IS. Its content may include – for example – entity types, relationship types, attributes, integrity constraints, and domain event types.

  In §3.4.3, we have determined the set of relationship types in the consolidated view. The definition of each relationship type in the “conceptualization” category is implied by the definition of the associated concept in the “conceptual model” category. These relationship types are listed as follows:

  - The Domain Understanding *conceptualizes stakeholders’ understanding of* the Domain, *at the beginning of the project*.
  - The Requirements *conceptualizes conditions expressing stakeholders’ desire of* the Domain.
  - The Conceptual Schema *completely conceptualizes stakeholders’ design of the structure and behavior of* the Domain, *at the end of the project*.
  - The Functionality Specification *is a directly implementable and complete conceptualization of the functions of* the IS.
  - The Domain Knowledge *conceptualizes the IS-independent part of stakeholders’ design of the structure and behavior of* the Domain, *at the end of the project*.

  We define the other relationship types as follows. The definitions are adapted from the definitions of the corresponding relationship types in the area-specific perspectives in §3.3, according to the goals listed in §3.4.1.
• The IS *represents, informs about, and changes the state of* the Domain: At the end of the project, the IS becomes part of the domain. It then affects the behavior of the domain by performing three types of functions: representing, informing about, and changing the state of the domain.

• The Requirements *are elicited using* the Domain Understanding: The stakeholders may elicit the requirements from their understanding of the domain at the beginning of the project, as described in the domain understanding.

• The Conceptual Schema *refines and implies satisfaction of* the Requirements: The conceptual schema maps the desired changes in the domain (described in the requirements) to the design of the IS’s functions and other aspects of the domain at the end of the project. The stakeholders will validate that the design satisfies the requirements.

• The Functionality Specification *is the IS design relevant subset of* the Conceptual Schema: The functionality specification is extracted from the conceptual schema and subsequently used as input to IS design.

• The Domain Knowledge *is the IS-independent subset of* the Conceptual Schema: This is implied by the definition of domain knowledge. It may not be necessary to extract it from the conceptual schema, because – unlike the functionality specification – it does not need to be subsequently used as input to IS design.
3.4.6. Comments about the Consolidated View

Finally, in this subsection, we make some comments about the consolidated view.

The consolidated view is essentially a consolidation of theoretical guidelines proposed in various areas for designing modelling methods in these areas. The “conceptual model” concepts in the consolidated view indicate which types of models are produced in IS analysis. The “production” relationship types indicate, for each type of modelling products in IS analysis, which other types of modelling products are used as input in their production.

The consolidated view is applicable regardless of the process aspect of the IS analysis method. IS analysis methods may differ in their processes. For examples, one method may prescribe a waterfall style process where one modelling product is finished before work on the next starts, whereas another method may prescribe an iterative style process where parts of all modelling products are gradually added in each of a sequence of iterations. However, according to the consolidated view, all complete IS analysis processes should produce the same types of modelling products, namely domain understanding, requirements, conceptual schema, functionality specification, and (optionally) domain knowledge. Moreover, the same “production” relationship types apply to these types of modelling products in all IS analysis processes, namely requirements are elicited using domain understanding, the conceptual schema refines and implies satisfaction of the requirements, and the functionality specification is the IS design relevant subset of the conceptual schema.
3.5. AN EXAMPLE PROJECT

In this section, we present an example project of an inventory IS to explicate the abstract concepts in the consolidated view with concrete examples, and to demonstrate applying the consolidated view in coordinating the use of multiple modelling methods in IS analysis. Since we want readers of this section to focus on understanding the consolidated view rather than the example project itself, we make the example project fictitious rather than realistic, as simple as possible, and just complex enough for explicating the concepts in the consolidated view. This example project is similar to a case study, except that case studies are based on observations of real-world phenomena in their natural settings (Benbasat et al., 1987, p. 371).

The rest of this section is organized as follows. First, we introduce the background of the example project. Second, we present a story on how the stakeholders applied the consolidated view in choosing IS analysis modelling methods for this example project. Third, we present the modelling products of the example project, as concrete examples to explicate the abstract concepts in the consolidated view from the previous section. Finally, we generalize about applying the consolidated view in coordinating the use of multiple modelling methods in IS analysis.

3.5.1. Background of the Example Project

In this subsection, we introduce the background of the (fictitious) example project.
The Greenway Store has opened since July 1st, 2014 as a local family business. It sells various items from various suppliers to various consumers. The store has an inventory, whose access is controlled by a door inside the store. The door is a smart door powered by the latest technology and is connected to the local network of the store. The door can be opened or closed by using the correct password with a remote procedure call (RPC) via the network. The smart door comes with a smartphone app for end users and an application programming user interface (API) for developers to control the state of the door via RPC.

The inventory stocks various items, and each item may have zero or more units in stock. The stock of the inventory may be changed by two types of events. First, a sale is an event where one or more units of one or more items are sold to a customer at a certain time. Second, a replenishment is an event where one or more units of one or more items are purchased from a supplier at a certain time.

On August 1st, 2014, one month after the store has opened, the store owner has decided to develop a simple inventory IS so as to automate certain aspects of the current business process so that the store can operate more efficiently. Thus, a group of software developers has been hired for system development.

3.5.2. Choosing IS Analysis Modelling Methods for the Example Project

In this subsection, we present a story on how the developers applied the consolidated view in choosing the IS analysis modelling methods.
The hired developers have just graduated from the college and started their own software development company. Although inexperienced, they have been hired for offering a much lower price than the competitors, and for convincing the store owner of their ability by achieving top academic performance in college.

Since this was their first project in the real-world, the developers were so excited for having the opportunity to put all their software engineering knowledge into real use. Hence, once they obtained the contract, they could not wait to start the project. The first thing they needed to do was to choose the appropriate modelling methods in IS analysis. They have been well educated on two IS analysis modelling methods in college:

- The $i^*$ early phase conceptual modelling method proposed by (Yu, 1997, 2010).

However, it was not until they try to put these two methods into real use that they found the need to answer the following questions before they could proceed:

- Are the two methods substitutes to each other, so that the use of one eliminates the need for the other?
  - If this is the case, is there a way to evaluate which method is better?
- Or, instead, are the two methods complements to each other, so that using them together is better than using only one of them?
  - If this is the case, how to use them together in a project?
- Is using one or both of these two methods enough for them or do they need to use other methods as well?
The developers had different opinions and could not convince each other. Being confused, one of them consulted the college professor who taught their system analysis course. The professor pointed him to the publication of the consolidated view of modelling in IS analysis (i.e., this chapter).

They studied the consolidated view in the chapter. They learned that a complete IS analysis process produces at least four types of modelling products: domain understanding, requirements, conceptual schema, and functionality specification. (Separating the domain knowledge from the conceptual schema is optional.) By mapping the products of these modelling methods to concepts in the consolidated view, they found that:

- The *i* early phase conceptual modelling method proposed by Yu (1997) focuses on the domain understanding.

Thanks to the consolidated view, they reached a consensus on answering the questions that none of them could answer correctly and confidently earlier:

- The two methods are complements to each other because they focus on different IS analysis modelling products. Therefore they can be used together.
- Guided by the consolidated view, it becomes clear how to use them together. First, they use the *i* early phase conceptual modelling method to produce a domain understanding (Yu, 1997, 2010). Second, they elicit the requirements based on the domain understanding. Third, given the requirements, they use the conceptual
schema modelling method proposed by Olivé (2004) to produce the conceptual schema and functionality specifications.

- They may benefit from having a third method on modelling the requirements, because neither of the two methods they already know covers this topic. However, given the time constraint of the project, they decided to start the project with the two methods they know, and just model the requirements informally in natural language based on their developers’ intuitions. After finishing this project, they plan to search for a formal method on requirements modelling that they can use in future projects.

3.5.3. Modelling Products of the Example Project

In this subsection, we present the modelling products – as well as the systems they model – of the example project, as concrete examples to explicate the abstract concepts in the consolidated view from the previous section.

3.5.3.1. Information System and Domain

The major stakeholders of this project include the store owner, store staff, and software developers. Their goal is to develop a software system to represent, inform about, and change the state of the store inventory. This means that this system will maintain certain information and interpret them as the state of the portion of the world around the
inventory. Therefore, this system is an **Information System** and this portion of the world is its **Domain**.

We will further analyze the things in the domain later in §3.5.3.4. The things in the domain may change overtime. For example, the quantities of items in the inventory may be increased by replenishments and decreased by sales.

### 3.5.3.2. Domain Understanding

At the beginning of the project, the stakeholders decided to understand the domain by using the *i* early phase conceptual modelling method (Yu, 1997, 2010; Yu & Mylopoulos, 1994) to create a domain understanding model for their inventory IS project, so as to facilitate eliciting the requirements later. They analyzed the current business process of the store and produced the model in Figure 10.

**Figure 10. Example Project – Domain Understanding**
In this chapter, we do not go into the details of the \( i^* \) modelling language. For readers not familiar with \( i^* \), we briefly explain the meanings of this model as follows:

- The store Staff depend on Customers for Demand for Items, and depend on Suppliers for Supply of Items.
- In the current business process, the store Staff’s task of Manage Inventory consists of the sub-goal of Inventory Access Be Secured, the sub-task of Manage Item Stock, and the sub-task of Handle Sales of Items. In addition, in this task of Manage Inventory, the Staff tries to achieve a soft goal of Minimize Cost (in terms of effort).
- Currently, the goal of Inventory Access Be Secured is achieve by the Staff’s task of Manage Door Access (i.e., open / close the door when the business hour begins / ends). Since this requires lots of efforts from the Staff, the task of Manage Door Access – as a means to achieve the goal of Inventory Access Be Secured – contributes negatively to the soft goal of Minimize Cost.
- The task of Manage Item Stock requires the sub-task of Replenish Low Stock Item and the resource of Item Stock Information, which is obtained by the task of Manage Item Stock Information. Since this requires lots of efforts from the Staff, the task of Manage Item Stock Information – as a means to obtain the resource of Item Stock Information – contributes negatively to the soft goal of Minimize Cost.
3.5.3.3. Requirements

Based on the domain understanding model discussed earlier, the stakeholders went through a few meetings and agreed on the following requirements of the inventory IS. (They made the requirements simple due to limited budget.)

- **R1:** At any time, a staff should be able to query the IS about the inventory stock quantity of a given item at present or at any given time in the past.
- **R2:** The door to the inventory should be automatically opened at the beginning and closed at the end of each business day.

3.5.3.4. Conceptual Schema

To satisfy the requirements, the stakeholders worked out a design of the structure and behavior of the domain at the end of the project. The design is based on the domain understanding and is documented in the conceptual schema.

The stakeholders followed the conceptual schema modelling method proposed by Olivé (2004), and the content of the conceptual schema modelled using this method consists of entity types, relationship types, domain event types, attributes, integrity constraints, derivation rules, and action request event types. For readers not familiar with this method, we define these concepts in Table 11.
Table 11. Definitions of concepts regarding the content of the conceptual schema in the conceptual schema modelling method proposed by Olivé (2004)

<table>
<thead>
<tr>
<th>Concept</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>concept</td>
<td>A mental construct formed by generalizing from perceptions of certain things in a domain.</td>
<td>See examples for entity type, relationship type, domain event type, and action request event type.</td>
</tr>
<tr>
<td>entity</td>
<td>An individual, identifiable object.</td>
<td>A person, an item, a door, etc.</td>
</tr>
<tr>
<td>entity type</td>
<td>A concept whose instances are entities.</td>
<td>The stakeholders may define “Customer” as a concept whose instances are individual customers.</td>
</tr>
<tr>
<td>relationship</td>
<td>A relationship is an association among two or more entities.</td>
<td>An association between a staff and an inventory, in which the staff manages the inventory.</td>
</tr>
<tr>
<td>relationship type</td>
<td>A concept whose instances are relationships.</td>
<td>The stakeholders may define “Manages” as a concept whose instances are associations between a staff and an inventory that the staff manages the inventory.</td>
</tr>
<tr>
<td>attribute</td>
<td>A characterization of certain aspects of an entity, relationship, event, etc.</td>
<td>The name of an item or person. The time when an event happened.</td>
</tr>
<tr>
<td>integrity constraint</td>
<td>A condition whose establishment increases stakeholders’ confidence that the IS represents all relevant facts, and only true facts, of the domain.</td>
<td>The stock quantity of an item in the inventory must be a nonnegative integer.</td>
</tr>
<tr>
<td>domain event</td>
<td>A change of the entities and/or relationships of the domain.</td>
<td>A group of items are sold to a customer.</td>
</tr>
<tr>
<td>domain event type</td>
<td>A concept whose instances are domain events.</td>
<td>The stakeholders may define “Sale” as a concept whose instances are domain events when a group of items are sold to a customer.</td>
</tr>
<tr>
<td>action request event</td>
<td>A request for the IS to perform certain actions, such as to query about or to change the state of the Domain.</td>
<td>Query about the current stock quantity of a selected item. Close a door at a certain time.</td>
</tr>
<tr>
<td>action request event type</td>
<td>A concept whose instances are action request events.</td>
<td>The stakeholders may define “Get Current Stock Quantity” as a concept whose instances are action request events that a user queries the IS about the current stock quantity of a selected item.</td>
</tr>
<tr>
<td>derivation rule</td>
<td>A defined way to use certain facts to infer certain other facts of the domain.</td>
<td>Age of a person = current date – date of birth of the person.</td>
</tr>
</tbody>
</table>
Entity Types, Relationship Types, Domain Event Types, and Attributes: The entity types, relationship types, domain event types, and attributes are represented in Figure 11 as an entity-relationship diagram.

Figure 11. Example Project – Entity-Relationship Diagram for the Conceptual Schema

Given the background introduced earlier and the simplicity of the example, readers should have no difficulty in understanding the meaning of the elements of this diagram. Thus, we omit the definitions for the elements.

Integrity Constraints: The stakeholders decided that the integrity constraints include the following:

- **IC1**: The door state is either open or closed.
- **IC2**: The stock, replenish, and sale quantities are always nonnegative integers.
- **IC3**: A replenishment immediately increases the stock quantities of the corresponding items by the corresponding replenish quantities.
• IC4: A sale immediately decreases the stock quantities of the corresponding items by the corresponding sale quantities.

**Derivation Rules:** The stakeholders decided that the derivation rules include the following:

• DR1: The stock quantity of any item at a given past time = its current stock quantity – all its replenish quantities after the given past time + all its sale quantities after the given past time.

**Action Request Event Types:** The action request event types are obtained by refining the requirements. If a statement in the requirements is directly implementable, it can be directly copied into the conceptual schema as an action request event type. Otherwise, it needs to be refined into a directly implementable action request event type, which – together with the other parts of the conceptual schema – implies its satisfaction.

As shown above, the requirements contain two statements, R1 and R2. Because the required information is represented by the IS, R1 is directly implementable because a query is considered something that can be done by an IS alone. Therefore, R1 can be directly copied into the conceptual schema to form an action request event type:

• ARET1: At any time, staff should be able to query the IS about the inventory stock quantity of a given item at present or at any given time in the past.

In contrast, R2 is not directly implementable, because physically opening or closing a door is not typically considered something that can be done by an IS alone. Therefore, the stakeholders decided to refine R2 into the following action request event type:
ARET2: Use the smart door API to initiate a Control RPC event using the correct door password as the control password with “open” as the control state at the beginning of each business day, and with “close” as the control state at the end of each business day.

When the required information are represented by the IS, ARET2 is directly implementable because initiating a RPC is considered something that can be done by an IS alone. In addition, ARET2 and the “Control State Of” relationship type\textsuperscript{13} imply the satisfaction of R2. Therefore it is correct to refine R2 into ARET2.

Comment: Some readers may be concerned that, since ARET2 refers to implementation details such as RPC and API, it may introduce an implementation bias and hence it may not be appropriate to include ARET2 in the conceptual schema. However, when such implementation details originate from the part of the domain that is independent of the IS being developed, they are considered part of the design decision rather than implementation bias (Straub & Zelkowitz, 1992). In our particular example project, because the door is not part of the IS but part of its domain, these “details” origins from the part of domain that is independent of the IS being developed. Therefore, in our example project, ARET2 is a design decision rather than an implementation bias.

3.5.3.5. Functionality Specification

The functionality specification is the subset of the conceptual schema that is relevant to IS design. It conceptualizes the IS’s functions. In the conceptual schema

\textsuperscript{13} This relationship type is defined as follows: When a Control RPC event is initiated to control the state of the door, the door state of the door will immediately become the control state of the event if the door password of the door matches the control password of the event, and the door state remains unchanged if otherwise.
modelling method proposed by Olivé (2004), it includes all the action request event types and derivation rules of the conceptual schema.

In addition, it also includes the other part of the conceptual schema that needs to be represented by the IS to support the IS’s functions. In our example project, this part includes all the integrity constraints in the conceptual schema, as well as the entity types, relationship types, domain event types, and attributes in Figure 12. Note that the content of Figure 12 is a subset of the content of Figure 11, because typically it is not necessary for the IS to represent everything in the domain.

Figure 12. Example Project – Entity-Relationship Diagram for the Functionality Specification

3.5.3.6. Domain Knowledge

The domain knowledge is the subset of the conceptual schema that is independent of the IS. In the conceptual schema modelling method proposed by (Olivé, 2004, 2007), this includes all the entity types, relationship types, domain event types, attributes, and
integrity constraints in the conceptual schema. However, unlike the functionality specification, it does not need to be subsequently used as input to IS design. Hence it may not be necessary to extract it from the conceptual schema.

In our example project, the domain knowledge includes the following:

- All the entity types, relationship types, event types, and attributes of the conceptual schema that are visualized in Figure 11.
- All the integrity constraints of the conceptual schema.

3.5.3.7. Summary of Modelling Products of the Example Project

In this subsection, we have used concrete examples from an example project to explicate the seven abstract concepts in the consolidated view: domain, IS, domain understanding, requirements, conceptual schema, functionality specification, and domain knowledge.

In this example project, the Domain is the portion of the business world around the Greenway store inventory, and the stakeholders want to develop an IS (i.e., the Greenway Inventory IS) to automate certain business processes in the domain.

The other five concepts in the consolidated view correspond to five types of IS analysis modelling products. We presented the concrete IS analysis modelling products of this example project as examples of these five abstract concepts. A summary of these modelling products is presented in Table 12. Each row of this table corresponds to a specific modelling product of the example project, which has been presented in this subsection as an example of a “conceptual model” concept in the consolidated view.
Finally, we have a few comments about this example project. First, the particular modelling languages (i.e., $i^*$, entity-relationship model, and English) and methods (i.e., the $i^*$ early phase conceptual modelling method (Yu, 1997, 2010), and the conceptual schema modelling method proposed by Olivé (2004)) are only used as examples, and stakeholders of other projects may choose different languages and methods, and may produce models with different contents. Second, to make the example project easy to understand, we have describe a sequential process through which the models are developed. This process is only used as an example and stakeholders of other projects may choose different processes, such as an iterative process.
3.5.4. Applying the Consolidated View in Coordinating the Use of Multiple Modelling Methods in IS Analysis

Earlier in this section, we presented a story on how the developers applied the consolidated view in choosing the modelling methods in IS analysis. In this subsection, we generalize about the utility of the consolidated view in coordinating the use of multiple modelling methods in IS analysis.

Different modelling methods in IS analysis have different focuses. Although there are some integrated methods (such as Tropos (Castro et al., 2002) that we will discuss in the next section) designed to support all types of modelling products of IS analysis, in some situations it may be beneficial or necessary to use multiple modelling methods in IS analysis. For example, one possible situation may be the case when a new modelling method (such as i* in the era before Tropos was created) focusing on a particular type of modelling product has significant advantages over the old methods with respect to the same type of modelling product, but is not yet integrated into any integrated modelling method that supports all types of modelling products in IS analysis. Another possible situation may be the case when method designers are trying to propose integrated modelling methods that support all types of modelling products in IS analysis, by synthesizing multiple modelling methods with narrower focuses.

Before the consolidated view in this chapter was created, as shown in our example project, the stakeholders facing such a situation may be puzzled by the following questions:
• Are the multiple modelling methods substitutes or complements to each other? (For substitute methods, using one of them eliminates the need for using the others. In contrast, complementary methods are better used together.)

• For modelling methods that are complements to each other, how to coordinate their use?

• When multiple modelling methods are used together, do they support all types of modelling products in IS analysis?

• For modelling methods that are substitutes to each other, how to evaluate and compare them?

As shown in our example project, the consolidated view is useful for answering these questions. In particular, the stakeholders can find the answers to these questions by using the “conceptual model” concepts in the consolidated view to analyze these modelling methods regarding which types of IS analysis modelling products do they support, and then apply the following principles:

• Modelling methods that support the same types of modelling products are substitutes to each other regarding these types of modelling products. Conversely, modelling methods that support different types of modelling products are complements to each other regarding these types of modelling products.

• In order to coordinate the use of modelling methods that are complements to each other, use each modelling method for the types of modelling products supported by it. The connections among these modelling products are explained by the “production” relationship types in the consolidated view.
• When multiple modelling methods are used together, they support all types of modelling products in IS analysis if and only if they support all types of “conceptual model” concepts in the consolidated view.

When the area-specific perspectives are used instead of the consolidated view, it is more challenging to apply these principles to answer these questions. Before applying the above principles, the specific IS analysis modelling products of the specific modelling methods under investigation have to be mapped to IS analysis modelling product types in a commonly accepted terminology. No individual area-specific perspective covers all IS analysis modelling product types. Hence, when analyzing a specific modelling method using an individual area-specific perspective as the terminology, the stakeholders may not be able to map all of its modelling products to an element of the terminology. For example, it might be difficult to apply the requirements engineering perspective to analyze the conceptual schema modelling method proposed by (Olívé, 2004, 2007), because the “conceptual schema” modelling product in the method cannot be directly mapped to an element of the perspective. One may suggest applying all the area-specific perspectives at the same time. However, this would also be more challenging than applying the consolidated view because there is inconsistency among the area-specific perspectives.

In the example project of this section, the two modelling methods discussed as examples are complements to each other. Hence we are only able to discuss the above three questions within this section. In the next section, we will tackle the last question on evaluating and comparing modelling methods that are substitutes to each other.
3.6. AN EXAMPLE APPLICATION

The consolidated view provides a whole picture for understanding all types of modelling products in IS analysis. It can be applied to evaluate and compare modelling methods designed to support all types of modelling products in IS analysis. In this section, we demonstrate its utility in such applications, by using it to conduct a simple evaluation and comparison of Gaia and Tropos, two agent-oriented software engineering methods.

The first subsection (§3.6.1) introduces Gaia and Tropos as the background. The second subsection (§3.6.2) presents a simple evaluation and comparison of the two by using the consolidated view. Finally, the third subsection (§3.6.3) summarizes this section.

3.6.1. Background: Gaia and Tropos

Agent-oriented methods are a new family of software engineering methods, in which agent-oriented concepts (e.g., agent, goal, interaction, and etc.) are used throughout the entire development process, including the analysis, design, and implementation phases.

Gaia and Tropos are among the earliest and most well-known agent-oriented methods in the academic literature. In this section, we conduct a simple evaluation and comparison of the two regarding their support for the IS analysis modelling products. While several improvements for both have been proposed overtime in the literature, we only focus on their original versions as described in Wooldridge et al. (2000) and Castro et al. (2002), respectively, in this section. This is because the purpose of this section is not to comprehensively evaluate and improve these two methods (which might require substantial amount of future work), but to demonstrate how to apply the consolidated view to evaluate
and compare modelling methods. Gaia and Tropos are only used as example modelling methods in this section.

In the rest of this subsection, we briefly review the support of IS analysis modelling products in Gaia and Tropos. To help readers who are not familiar with them, we provide examples of their IS analysis modelling products, by applying each of them to our example project (which has been introduced in the previous section). However, we do not discuss the details of their modelling languages. Interested readers may refer to Wooldridge et al. (2000) and Castro et al. (2002) for such details. Finally, we also map their modelling products to the “conceptual model” concepts in the consolidated view.

Readers who are not interested in the details of the evaluation and comparison process can skip the rest of this subsection, and resume reading from §3.6.2.

3.6.1.1. The Support of IS Analysis Modelling Products in Gaia

In applying the Gaia method, the stakeholders start with a given set of requirements, and goes through two phases: analysis and design. Only the analysis phase belongs to the scope of IS analysis and thus is studied in this section. (Wooldridge et al., 2000, §2).

The analysis phase of Gaia does not support eliciting or modeling the requirements. Rather, it is assumed that the requirements are already given. In this phase, the stakeholders produce a role model and an interaction model, so as to define the organization of the domain at the end of the project. (Wooldridge et al., 2000, §3).

The *role model of Gaia* defines the roles in the domain and declares the interactions each role participate in. (The interactions will be later defined in the interaction model.) Each role is defined by the following attributes (Wooldridge et al., 2000, p. 289):
• **Responsibilities:** There are two types of responsibilities:
  
  o **Liveliness properties:** Conditions expressing that “something good happen”.
  
  o **Safety properties:** Conditions expressing that “something bad do not happen”.

• **Permissions:** Resources available to the role for realizing its responsibilities.

• **Activities:** Actions performed by the role itself, without interacting with other roles.

• **Protocols:** Patterns of interaction among the role and other roles.

  For example, if Gaia was used in our example project (introduced in the previous section), the stakeholders may produce a role model that consists of three roles: staff, door, and inventory IS. These roles may be defined in a modelling script like in Figure 13.

**Figure 13. An Example Role Model in Gaia**

<table>
<thead>
<tr>
<th>Role Schema</th>
<th>Staff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>An employee who works for the store and manages the inventory.</td>
</tr>
<tr>
<td>Protocols &amp; Activities</td>
<td></td>
</tr>
<tr>
<td>Protocols</td>
<td>Query Stock, Sell, Replenish</td>
</tr>
<tr>
<td>Activities</td>
<td></td>
</tr>
<tr>
<td>Permissions</td>
<td></td>
</tr>
<tr>
<td>reads</td>
<td>supplied itemName</td>
</tr>
<tr>
<td>changes</td>
<td>saleQuantity, replenishmentQuantity, stockQuantity</td>
</tr>
<tr>
<td>Responsibilities</td>
<td></td>
</tr>
<tr>
<td>Liveliness</td>
<td>Staff = (Sell. Replenish. Query Stock)(\omega)</td>
</tr>
<tr>
<td>Safety</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Role Schema</th>
<th>Door</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The smart door that can be opened or closed via a network protocol. It controls the physical access to the inventory.</td>
</tr>
<tr>
<td>Protocols &amp; Activities</td>
<td></td>
</tr>
<tr>
<td>Protocols</td>
<td>Door State Control</td>
</tr>
<tr>
<td>Activities</td>
<td></td>
</tr>
<tr>
<td>Permissions</td>
<td></td>
</tr>
<tr>
<td>reads</td>
<td>supplied door</td>
</tr>
</tbody>
</table>
The interaction model of Gaia defines the interactions among the roles in the domain, in the form of protocols. A protocol is a pattern of interaction among multiple roles. Each protocol is defined by the following attributes (Wooldridge et al., 2000, p. 294):

- **Purpose**: A brief textual description of the nature of the interaction.
- **Initiators**: The roles that start the interaction.
- **Responders**: The other roles that participate in the interaction.
- **Inputs**: Information used by the initiators in the interaction.
- **Outputs**: Information provided by the responders in the interaction.
• **Processing:** A brief textual description of the initiators’ processing activities performed in the interaction.

For example, if Gaia was used in our example project (introduced in the previous section), the stakeholders may produce an interaction model that consists of four protocols: replenishment, sale, query stock, and control door state. The protocols may be defined in a modelling script as in Figure 14. (Note: Wooldridge et al. (2000) represents Gaia interaction models in a different but equivalent format. We use the following alternative format to improve the readability of the script for readers who are not familiar with Gaia.)

**Figure 14. An example interaction model in Gaia**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Replenish</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose</strong></td>
<td>Increase stock quantities of items when a replenishment arrives from a supplier.</td>
</tr>
<tr>
<td><strong>Initiators</strong></td>
<td>Staff</td>
</tr>
<tr>
<td><strong>Responders</strong></td>
<td>Inventory IS</td>
</tr>
<tr>
<td><strong>Inputs</strong></td>
<td>List of items and replenishment quantities</td>
</tr>
<tr>
<td><strong>Outputs</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td><em>When a replenishment arrives:</em> Record the replenishment transaction in the database. For each item in the replenishment, increase stock quantity by the corresponding replenishment amount. <em>Otherwise:</em> Do nothing.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Sale</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose</strong></td>
<td>Decrease stock quantities of items when a sale is made to a customer.</td>
</tr>
<tr>
<td><strong>Initiators</strong></td>
<td>Staff</td>
</tr>
<tr>
<td><strong>Responders</strong></td>
<td>Inventory IS</td>
</tr>
<tr>
<td><strong>Inputs</strong></td>
<td>List of items and sale quantities</td>
</tr>
<tr>
<td><strong>Outputs</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td><em>When a sale is made:</em> Record the sale transaction in the database. For each item in the sale, decrease stock quantity by the corresponding sale amount. <em>Otherwise:</em> Do nothing.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Query Stock</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Purpose</td>
<td>Find the stock quantity of an item at present or any given time in the past.</td>
</tr>
<tr>
<td>Initiators</td>
<td>Staff</td>
</tr>
<tr>
<td>Responders</td>
<td>Inventory IS</td>
</tr>
<tr>
<td>Inputs</td>
<td>Item Name, A given time in present or past</td>
</tr>
<tr>
<td>Outputs</td>
<td>The stock quantity of the corresponding item at the given time.</td>
</tr>
<tr>
<td>Processing</td>
<td>When there is a need to query item stock: Query the database and return the stock quantity of the database. The present stock quantity of an item is directly recorded in the database. The stock quantity of any item at a given past time = its current stock quantity – all its replenish quantities after the given past time + all its sale quantities after the given past time. Otherwise: Do nothing.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Door State Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Open the door at the beginning of each business day, and close the door and the end of each business day.</td>
</tr>
<tr>
<td>Initiators</td>
<td>Inventory IS</td>
</tr>
<tr>
<td>Responders</td>
<td>Door</td>
</tr>
<tr>
<td>Inputs</td>
<td>Current Date, Current Time.</td>
</tr>
<tr>
<td>Outputs</td>
<td></td>
</tr>
<tr>
<td>Processing</td>
<td>At the beginning (or end) of a business day: the Inventory IS automatically send a command to the Door to change its state to open (or closed). At other moments: Do nothing.</td>
</tr>
</tbody>
</table>

In summary, Gaia supports IS analysis through two types of models: the role model and the interaction model. The two models define the organization of the domain at the end of the project. Therefore, together they correspond to the conceptual schema concept in our consolidated view.

### 3.6.1.2. The Support of IS Analysis Modelling Products in Tropos

The Tropos method spans four phases: early requirements, late requirements, architectural design, and detailed design. The first two phases belong to the scope of IS analysis and will be studied in this section. In both of the first two phases, the modelling
products are represented by using the $i^*$ framework. The third and fourth phases belong to
the scope of IS design and are beyond the scope of this section. (Castro et al., 2002, p. 366).

In the early requirements phase of Tropos, the stakeholders focus on an
organizational setting of interest (Castro et al., 2002, p. 366), and produce two models: a
strategic dependency model and a strategic rational model (Castro et al., 2002, §3).

The strategic dependency model in the early requirements phase of Tropos
describes stakeholders in the organization setting as actors, along with the high level
interactions among them as strategic dependencies. For example, if Tropos was used in our
example project (introduced in the previous section), in the early requirements phase, the
stakeholders may produce a strategic dependency model as in Figure 15.

**Figure 15. An Example Strategic Dependency Model in the Early Requirements Phase of Tropos**

![Figure 15](image)

The strategic rationale model in the early requirements phase of Tropos extends
the former strategic dependency model by describing the goals of the actors, along with
the possible ways of fulfillment of the goals through other actors. For example, if Tropos
was used in our example project (introduced in the previous section), in the early
requirements phase, the stakeholders may produce a strategic rationale model that is the
same as the domain understanding model in Figure 10 from our example project.
In the *late requirements phase* of Tropos, the stakeholders focus on studying the functions and qualities of the system-to-be in its operational environment (Castro et al., 2002, p. 366), and produce two models: a strategic dependency model and a strategic rationale model (Castro et al., 2002, §4).

The *strategic dependency model in the late requirements phase* of Tropos describes the system-to-be as an actor, along with its interactions with other actors that were identified in the early requirements phase as strategic dependencies. These strategic dependencies define the functional and non-functional requirements of the system. For example, if Tropos was used in our example project (introduced in the previous section), in the late requirements phase, the stakeholders may produce a strategic dependency model as in Figure 16.

**Figure 16. An Example Strategic Dependency Model in the Late Requirements Phase of Tropos**

The *strategic rationale model in the late requirements phase* of Tropos extends the former strategic dependency model by mapping the strategic dependencies of the system-to-be into its functions. For example, if Tropos was used in our example project (introduced in the previous section), in the late requirements phase, the stakeholders may produce a strategic rationale model as in Figure 17.
In summary, Tropos supports IS analysis in its early and late requirements phases. The first phase focuses on the organization setting, whereas the second phase focuses on the system-to-be. In each phase, the stakeholders produce a strategic dependency model and a strategic rationale model. We analyzed these four types of modelling products with our consolidated view. We present the analysis results in Table 13.

**Table 13. Analysis Results of the IS Analysis Modelling Products in Tropos**

<table>
<thead>
<tr>
<th>Tropos Phase</th>
<th>Tropos Modelling Product</th>
<th>Content</th>
<th>Corresponding Concept in the Consolidated View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early Requirements</td>
<td>Strategic Dependency Model</td>
<td>Stakeholders. The strategic dependencies among the stakeholders.</td>
<td>Domain Understanding</td>
</tr>
<tr>
<td>Phase</td>
<td>Strategic Rationale Model</td>
<td>Stakeholders’ goals. The possible ways of fulfillment of the goals through other actors.</td>
<td></td>
</tr>
<tr>
<td>Late Requirements</td>
<td>Strategic Dependency Model</td>
<td>The requirements of the IS in its operational environment.</td>
<td>Requirements</td>
</tr>
<tr>
<td>Phase</td>
<td>Strategic Rationale Model</td>
<td>The functions of the IS. The mapping of these functions to the requirements.</td>
<td>Conceptual Schema</td>
</tr>
</tbody>
</table>
3.6.2. Evaluation and Comparison

Recall that the consolidated view consists of two categories of concepts (systems and conceptual models) and three categories of relationship types (conceptualization, composition, and production). In this subsection, we will demonstrate how to use the “conceptual model” concepts to evaluate and compare modelling methods that are designed to support all types of modelling products in IS analysis, so as to demonstrate the utility of the consolidated view.

Recall that the consolidated view includes five “conceptual model” concepts: domain understanding, requirements, conceptual schema, functionality specification, and domain knowledge. The first four correspond to four types of modelling products that should be produced in a complete IS analysis method. It is optional to separate the domain knowledge from the conceptual schema, as explained in the relevant definitions in the consolidated view. Thus, a method – designed to support all types of modelling products in IS analysis – can be evaluated regarding the extent to which it provides modelling product support in IS analysis, by checking whether it supports representing each type of modelling product as modelling scripts.

We applied these concepts to evaluate and compare Gaia and Tropos, and the results are presented in Table 14. (The mapping between the “conceptual model” concepts in the consolidated view and the method specific modelling products of Gaia and Tropos can be found in the previous subsection.)
Table 14. Evaluating and Comparing Gaia and Tropos Regarding Modelling Product Support

<table>
<thead>
<tr>
<th>Modelling Product Type in the Consolidated View</th>
<th>Gaia</th>
<th>Tropos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Understanding</td>
<td>Unsupported</td>
<td>Supported</td>
</tr>
<tr>
<td>Requirements</td>
<td>Unsupported</td>
<td>Supported</td>
</tr>
<tr>
<td>Conceptual Schema</td>
<td>Supported. But, contrary to the definition of the conceptual schema in the consolidated view, statements of the IS’s functions are not explicitly required to be directly implementable.</td>
<td></td>
</tr>
<tr>
<td>Functionality Specification</td>
<td>Not directly supported, because the functionality specification is not separated(^\text{14}) from the rest of the conceptual schema.</td>
<td></td>
</tr>
<tr>
<td>Domain Knowledge (optional)</td>
<td>The domain knowledge is not separated from the rest of the conceptual schema. But this separation is optional according to the consolidated view.</td>
<td></td>
</tr>
</tbody>
</table>

The above results indicate that Tropos is relatively more complete than Gaia in modelling product support, but neither is complete when evaluated using the consolidated view as a theoretical benchmark.

### 3.6.3. Summary of the Example Application

In this section, we have conducted a simple evaluation and comparison of Gaia and Tropos, as an example application of the consolidated view. The evaluation and comparison method we have used can also be applied to other modelling methods that are designed to support all types of modelling products in IS analysis. The evaluation and comparison results are useful for modelling method designers to propose improvements to

\(^{14}\) According to its definition in the consolidated view, the functionality specification is the part of conceptual schema that is directly relevant to IS design, and thus needs to be separated from the rest of the conceptual schema before provided to the system designers. However, neither Gaia nor Tropos explicitly discussed how to separate the functionality specification from the rest of the conceptual schema.
the evaluated methods, and for developers to choose one from several of such methods (that are substitutes to each other) based on their completeness in modelling product support.

Such evaluation and comparison are more challenging if the area-specific perspectives are used instead of the consolidated view. The inconsistency among the area-specific perspectives exists as a barrier for directly using them together at the same time in such evaluation and comparison. However, no area-specific perspective is complete and hence none of them can be used to evaluate and compare modelling methods regarding their completeness in IS analysis modelling product support.

3.7. CONCLUSION OF CHAPTER

In this section, we summarize this chapter, and discuss its academic contributions and applications.

3.7.1. Summary

In summary, this chapter has studied the modelling products in IS analysis by examining the relevant concepts and relationship types. It mainly consists of four parts.

In the first part, we have summarized three area-specific perspectives of modelling. Each perspective consists of two categories of concepts (system being modeled and conceptual model) and three categories of relationship types (composition,
conceptualization, and production). The elements of each perspective are visualized and defined.

In the second part, we have built a consolidated view that incorporates adapted versions of all elements from all area-specific perspectives summarized in the first part. Since this chapter focuses on IS, one major adaptation is that, while some perspectives apply to system development in general, they are adapted to the more specific situation of IS development, before incorporated into the consolidated view.

In the third part, we have presented an example project to explicate the abstract concepts in the consolidated view with concrete examples from the example project, and to demonstrate the utility of the consolidated view in coordinating the use of multiple modelling methods in IS analysis.

Finally, in the fourth part, we have applied the consolidated view to evaluate and compare Gaia and Tropos, two agent-oriented modelling methods that are designed to support all types of modelling products in IS analysis.

3.7.2. Academic Contributions and Applications

In this chapter, we have made a number of academic contributions with both the area-specific perspectives and the consolidated view.

3.7.2.1. Contributions of the Area-Specific Perspectives

The area-specific perspectives provide an overview of the basic knowledge about modelling products in the corresponding areas. As a foundation for building the consolidated view, we have selected the concepts and relationship types that are useful for
defining system analysis modelling products in three areas related to modelling in IS analysis: requirements engineering, conceptual schema modelling, and domain understanding.

For requirements engineering and conceptual schema modelling, we have selected the concepts and relationship types based on previous publications that study the general terminologies of these two areas. For domain understanding, where no such previous publication is available, we have explained the necessity of including it in this chapter, and proposed its set of concepts and relationship types based on our understanding.

For each of the three areas, the selected concepts and relationship types are independent of any specific modelling methods. Therefore, for researchers and practitioners specializing in modelling in system analysis, the area-specific perspectives provide a concise guided tour to the relevant basic knowledge in all the related areas.

3.7.2.2. Contributions of the Consolidated View

The consolidated view is a union of all the area-specific perspectives. As we can see by checking the tables that explicate the mapping between the consolidated view and the area-specific perspectives (i.e., Table 9 and Table 10), it has made its own contributions on top of the area-specific perspectives, by being a consistent and complete picture of modelling products in IS analysis. This contribution is elaborated as follows.

First, the consolidated view provides a consistent terminology for modelling products in IS analysis. In existing literature, these modelling products are studied in different areas, and thus are defined in area-specific terminologies that are inconsistent
with each other. The consolidated view remedies by reconciling such inter-area inconsistency.

Second, within the scope of this chapter, the consolidated view is complete for including all basic knowledge of modelling products in IS analysis, while none of the individual area-specific perspectives is complete.

3.7.2.3. Applications of the Consolidated View

Being consistent and complete, the consolidated view provides utility in three types of applications.

First, as we have demonstrated at the end of §0 (i.e., the section where we have presented the example project), the consolidated view can be used to determine whether two or more modelling methods are complements or substitutes to each other.

Second, as we have demonstrated at the end of §0 (i.e., the section where we have presented the example project), for multiple modelling methods that are complements to each other, stakeholders of IS development projects can apply the consolidated view to coordinate their use in a project, while method designers can use the consolidated view to propose new integrated modelling methods by integrating them together.

Third, as we have demonstrated in §3.6 (i.e., the section where we have presented the example application), for modelling methods designed to support all types of modelling products in IS analysis, the consolidated view can be used to evaluate and compare them regarding their completeness in modelling product support in IS analysis. The evaluation
results are useful for method designers to improve them, and for project stakeholders to make choices among them since they are substitutes to each other.

These applications are more challenging if the area-specific perspectives are used instead of the consolidated view, because each individual area-specific perspective is incomplete, while the inconsistency issue exists as a barrier for directly applying them together. Elaborated discussions have been presented at the end of the §0 and §3.6.
Chapter 4: Joint Application

Analyzing the GRAPPLE Modelling Method

4.1. INTRODUCTION

4.1.1. Background

Earlier in §2.3.3, we have defined the IS analysis model as a concept whose instances are conceptual models produced in the process of developing a specific new IS by representing selected focal aspects of its domain, for certain purposes related to analyzing the requirements of the IS to be developed in the domain.

To improve the understanding of IS analysis models, we have proposed the generalized and consolidated views of modelling in IS analysis. The generalized view (proposed in Chapter 2) explicates certain concepts that apply to all types of IS analysis models. We demonstrated that it can be applied to answer the following questions:

- How to distinguish IS analysis models from other types of models?
- What are some of the concepts that apply to all types of IS analysis models?
- How to compare different IS analysis models for similarities and differences?
- What is the difference between model and script?
- What is the connection between semantic domain and ontology?
The consolidated view (proposed in Chapter 3) presents a typology of different types of IS analysis models, and explicates the connections among them. We demonstrated that it can be applied to answer the following questions:

- Whether two modelling methods are complements or substitutes to each other?
- How to coordinate the use of multiple modelling methods that are complements to each other?
- How to compare multiple modelling methods that are substitutes to each other?
- How to evaluate a modelling method regarding its adequacy in supporting all types of IS analysis models?

4.1.2. Motivation

In this chapter, we want to test the two views regarding their generalizability, by applying them to answer the above questions for a modelling method that is not studied in previous chapters.

Moreover, in previous chapters, we have only demonstrated applying the two views individually. We demonstrated that they both can be applied to analyze modelling products or methods in IS analysis. However, despite such similarity in their applications, we have not demonstrated how to apply the two view together, and have not explained the differences and relationships between the two views in such applications. Therefore, in this chapter, we are also motivated to answer the following questions:

- How to apply the two views together to analyze a modelling method?
- What are the differences between the roles of the two views in such application?
- What are the relationships between the roles of the two views in such application?

Last but not least, based on the analysis results, we want to propose improvements to the modelling method being analyzed in this chapter.
4.1.3. Overview of Analysis Method

As in earlier chapters of this dissertation, we continue focusing on modelling products in our analysis. We propose the following three step analysis method for analyzing modelling methods using both the generalized and consolidated views.

In the first step, we investigate how do concepts in the generalize view apply to each type of modelling product in the modelling method. By doing so, we obtain a basic understanding of the modelling method regarding its modelling product types.

In the second step, based on the understanding obtained in the first step, we map the specific modelling product types in the modelling method to the general modelling product types in the consolidated view. We produce this mapping mainly by using the “purpose” and “focal aspects” concepts in the generalized view to analyze the method-specific modelling product types, and then by comparing the analysis results to definitions of the method-independent modelling product types in the consolidated view.

In the third step, based on the mapping obtained in the second step, we apply the consolidated view to evaluate the specific modelling method regarding its adequacy in supporting different types of modelling products.

In the steps above, the first step is similar to the application of the generalized view in Chapter 2, while the third step is similar to that of the consolidated view in Chapter 3. The second step is an intermediate step useful for applying the two views together.
4.1.4. Choosing a Modelling Method for the Analysis

To demonstrate the joint application of the generalized and consolidated views in analyzing modelling methods, we need to choose a modelling method to be used in the analysis. Because one of the motivations of this chapter is to test the generalizability of the two views in such applications, we want to choose a modelling method that has different characteristics from the ones we used in previous chapters.

In making this choice, we have the following specific criteria in mind.

The first criterion is that the modelling method should not be agent-oriented. Thus, this method would take a different modelling approach from Gaia and Tropos, the two agent-oriented methods that we analyzed in previous chapters.

The second criterion is that the modelling method should be mainly intended for practical use. Thus, this method would be different from the Gaia and Tropos methods we analyzed in previous chapters, which have been mainly used for academic research so far.

The third criterion is that the official publication that defines the modelling method should also present an example project and the relevant modelling products. In previous chapters, we developed our own example projects to demonstrate the applications of the two views. In contrast, in this chapter, we want to reuse an example project written by another author.

The fourth criterion is that the modelling method and its official example project should be simple enough so that we can present them in the length of a typical academic publication, without requiring our readers to study other publications (e.g., the official publication of the modelling method). We want to choose a simple modelling method so
that readers can focus their energy on understanding the original contribution we make in this chapter, rather than the modelling method and example project being analyzed. Moreover, by choosing a simple method that we can understand well given our limited time and resource, we can reduce the threat to internal validity caused by lack of good understanding of the modelling method being analyzed. Note: In previous chapters, we had also chosen to analyze simple modelling methods for the same reason. But because all modelling methods analyzed in this dissertation are simple, we cannot be certain that the generalized and consolidated views in this dissertation apply to complex modelling methods (e.g., the Unified Process (Kruchten, 2003)) as well as simple ones. This is a threat to the external validity of the study in this dissertation. Thus, in our future work, we want to apply the two views to analyze and evaluate complex modelling methods as well.

The fifth criterion is that the modelling method should cover both the analysis and design of information systems. Hence, we can demonstrate applying the generalized view to distinguish IS analysis models from other types of models in IS development.

Bearing the above criteria in mind, we have chosen the GRAPPLE modelling method (Schmuller, 2004). As we will elaborate in the next section, it satisfies all the above criteria. It is a UML based object-oriented modelling method intended for practical use. It covers both analysis and design of information systems. The official definition of the method (Schmuller, 2004) includes an example project with example modelling products of the method. The method and the official example project are fairly simple so that we can present them in a short length.
We have also considered the Rational Unified Process (Kruchten, 2003), a very well-known UML based object-oriented modelling method. However, we have not chosen it because it is very complex, and hence does not satisfy the fourth criterion above.

4.1.5. Outline of Chapter

The rest of this chapter is organized as follows. First, we introduce GRAPPLE, a modelling method that we will analyze in this chapter. Second, we use the generalized view to analyze GRAPPLE. This corresponds to the first step in our analysis method. Third, we use the consolidated view to analyze GRAPPLE. This corresponds to the second and third steps in our analysis method. Fourth, we clarify the differences and relationships between the roles played by the generalized and consolidated views in analyzing modelling methods, and discuss whether the related work from previous literature can play such roles as well as the two views. The last section concludes.

4.2. THE GRAPPLE MODELLING METHOD: AN OVERVIEW

In this section, we introduce the modelling products of the Guidelines for Rapid Application Engineering (GRAPPLE), a modelling method proposed in Schmuller (2004) for object-oriented system analysis and design, based on the Unified Modelling Language (UML). In subsequent sections, we will apply the generalized and consolidated views to analyze this method.
As introduced in Schmuller (2004, §15), GRAPPLE is an iterative method and consists of five segments: requirements gathering, analysis, design, development, and deployment. Unlike sequential method (such as waterfall) where one segment is finished before the next, each iteration of GRAPPLE may involve one or more of these segments. In this chapter, we focus on modelling products produced in the first two segments, because the remaining segments are beyond the scope of IS analysis.

We illustrate these modelling products with the relevant modelling products of an example project that is presented in Schmuller (2004, Part II. A Case Study). In the previous chapters, we developed our own example projects to demonstrate the applications of the two views. In contrast, in this chapter, we deliberately reuse an example project written by another author, so as to put our generalized and consolidated views to further test regarding their generalizability. This is because the modelling products of such an example project are not affected by our mindset formed based on our generalized and consolidated views.

The rest of this section is organized as follows. In the first subsection, we present a brief overview of the example project and the relevant modelling products. In the second subsection, we provide examples to each type of IS analysis modelling product in GRAPPLE. The last subsection summarizes the work in this section.

For sake of conciseness, we do not present all details of GRAPPLE. Instead, we only present enough content to help readers to understand each modelling product type in GRAPPLE. Interested readers may refer to Schmuller (2004, pp. 265-380, Part II. A Case Study) for the details.
4.2.1. GRAPPLE and the Example Project: An Overview

The background of the example project – as introduced in Schmuller (2004, §16) – is situated in a typical formal restaurant. The goal of the project is to improve the restaurant business by developing an information system. The major stakeholders of this project include the developers (who develop the IS to improve the restaurant business), customers (who purchase the foods from the restaurant), owners (who own the restaurant business), and employees (who work for the restaurant to earn income).

The major types of employees are managers (who manage other employees), servers (i.e., the waiters and waitresses who present the menu, take orders, and serve the foods), chefs (who cook and prepare the foods), bartenders (who make the drinks), bussers (who clean the tables), coat check clerks (who manage the coats of the customers and seat them), and assistants (to whom the servers and chefs delegate certain simple tasks).

The developers started with “requirements gathering”, the first segment of GRAPPLE. They interviewed various restaurant employees. Through these interviews, they discovered the existing business processes and modelled them as UML activity diagrams (Schmuller, 2004, §16), and analyzed the existing structure of the domain (i.e., static aspects such as entity types, relationship types, and attributes) and modelled it as UML class diagrams (Schmuller, 2004, §17).

In activities above, the stakeholders focus on understanding the existing domain, without envisioning the IS to be developed. After this, the stakeholders started conceptualizing the IS by analyzing the diagrams produced in the above activities. Through the analysis, they have found that communication is an important aspect of the business.
They have decided to develop an IS to improve – by speeding up – the communication in, without introducing major changes to the existing business processes.

Having made this decision, they started “analysis”, the second segment of GRAPPLE. They began this segment by conceptualizing the IS with UML package diagrams (of use cases) that list the IS’s functions as use cases and group them into packages (Schmuller, 2004, §18). Then they proceeded to add details to the use cases (Schmuller, 2004, §19). They produced UML use case diagrams to visualize the relationships between use cases and stakeholders. Each use case is accompanied by textual descriptions of its details (e.g., pre-conditions, post-conditions, and steps).

Moreover, they updated the UML class diagrams from the “requirements gathering” segment to support the use cases. For example, in Schmuller (2004, §19, Figure 19.3), the developers decided to update the class diagram, by replacing the “Assistant” class with two more specific classes: “Assistant Server” and “Assistant Chef”.

In this particular project, they have not updated the UML activity diagrams from the “requirements gathering” segment. However, we think that this may be necessary if the business processes of the domain would be significantly changed by the IS, and that this could be done in principle. For example, if the IS creates a new business process that allows restaurant customers to make reservations online, a new business process diagram need to and can be added. Essentially, this would differentiate the business process model at the end of the project from that at the beginning of the project.

At the end of the analysis segment, the developers model the components of the IS and the interactions among these components (Schmuller, 2004, §20). They used UML sequence diagrams to model the interactions among the components.
In addition, according to the specification of GRAPPLE in Schmuller (2004, §15), they could also have used UML state diagrams to analyze the change of states of the components, UML collaboration diagrams to analyze the interactions among the components, and UML deployment diagrams to analyze how the components of the IS being developed and other system are physically deployed on system hardware. However, for this particular example project, they have not presented any examples of UML state diagrams or UML collaboration diagrams. Moreover, they have not presented any examples of UML deployment diagrams until describing a subsequent segment of GRAPPLE in Schmuller (2004, §21).

In summary, the IS analysis modelling products of GRAPPLE include UML activity diagrams for describing business processes, UML class diagrams for describing domain structure, UML package diagrams (of use cases) for listing and classifying system functionalities, and UML use case diagrams (with textual descriptions) for describing system functions in detail. In the next subsection, we will present examples for these diagrams.

Although UML state diagrams, UML sequence diagrams, UML collaboration diagrams, and UML deployment diagrams are also used in the “analysis” stage of GRAPPLE, they are used to model the software components of the IS rather than to analyze the requirements of the IS. Thus, according to the definition of “IS analysis model” in §2.6.5, they do not belong to the scope of IS analysis and are not instances of the “IS analysis model” concept in the generalized view. So we do not discuss them in the rest of this chapter.
We consider it a potential issue for including these IS design modelling products in the “analysis” segment along with the IS analysis modelling products. This may confuse users of the GRAPPLE method regarding the focus of the “analysis” segment. While IS analysis should focus on the real world around the IS, IS design should focus on the internal software artifacts of the IS. We think it would be better to move these diagrams into the “design” segment of GRAPPLE.

### 4.2.2. Example IS Analysis Models of GRAPPLE

In this subsection, we illustrate the four types of IS analysis models of GRAPPLE with examples. Due to copyright concerns, we provide our own examples here for illustration, rather than directly copying from Schmuller (2004, Part II. A Case Study). Interested readers may refer to Table 15 for references to the corresponding original examples provided by Schmuller (2004, Part II. A Case Study). Replacing our own examples by the original ones would not change the analysis and evaluation of GRAPPLE in this chapter.

#### Table 15. A Summary of Examples IS Analysis Models of GRAPPLE

<table>
<thead>
<tr>
<th>IS Analysis Model Type of GRAPPLE</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>UML Activity Diagram for Modelling Business Processes</td>
<td>Figure 18, Schmuller (2004, p. 281, Figure 16.7)</td>
</tr>
<tr>
<td>UML Class Diagram for Modelling Domain Structure</td>
<td>Figure 19, Schmuller (2004, p. 302, Figure 17.15)</td>
</tr>
<tr>
<td>UML Package Diagrams (of Use Cases) for Listing and Classifying System Functions</td>
<td>Figure 20, Schmuller (2004, p. 322, Figure 18.8)</td>
</tr>
<tr>
<td>UML Use Case Diagrams &amp; Use Case Descriptions for Describing System Functions In Detail</td>
<td>Figure 21 &amp; Figure 22, Schmuller (2004, p. 332, Figure 19.2) &amp; Schmuller (2004, pp. 331-332)</td>
</tr>
</tbody>
</table>
4.2.2.1. UML Activity Diagram for Modelling Business Processes

In Figure 18 below, we present an example use of UML activity diagram to model business process in GRAPPLE. This diagram models the “welcome customer” business process, where the check clerk performs a sequence of actions in his or her job. For further details of UML activity diagram, readers may refer to Schmuller (2004, §11).

Figure 18. Example GRAPPLE UML Activity Diagram for Modelling Business Process

4.2.2.2. UML Class Diagram for Modelling Domain Structure

In Figure 19 below, we present an example use of UML class diagram to model domain structure in GRAPPLE. This diagram models the different types of stakeholders in
the restaurant, their attributes, and operations. For further details of UML class diagram, readers may refer to Schmuller (2004, §§3-5).

Figure 19. Example GRAPPLE UML Class Diagram for Modelling Domain Structure

![UML Class Diagram](image)

4.2.2.3. UML Package Diagrams (of Use Cases) for Listing and Classifying System Functions

In Figure 20 below, we present an example use of UML package diagram (of use cases) to list and classify system functions in GRAPPLE. This diagram lists and classifies the functionalities of the IS to be developed in the example project. For further details of UML package diagram, readers may refer to Schmuller (2004, §14).
Figure 20. Example GRAPPEL UML Package Diagram (of Use Cases) for Listing and Classifying System Functions

4.2.2.4. UML Use Case Diagrams and Use Case Descriptions for Describing System Functions In Detail

In Figure 21 below, we present an example use of UML use case diagram for describing the relationship between use cases and actors in GRAPPEL. This diagram describes the use case “get payment”, that it is related to the “server” actor, and includes another use case “pay for meal”, which is related to the “customer” actor.
In both UML and GRAPPLLE, each use case should be described in detail. In Figure 22 below, we present an example use case description for the use case “get payment”.

Figure 22. Example GRAPPLE UML Use Case Description.

Get Payment

Description
Using the IS, the server get payment from the customer, and registers the payment with the server’s name for bonus calculation purposes.

Assumptions
- Both the server and the paying customer are at the table.
- The server has an iPad which has a user-interface screen for displaying the total amount of money for the meal at the table.

Pre-conditions
- The customers at a table finish the meal and call the server for payment.

Post-conditions
- The payment is received and is registered with the server’s name.

**Steps**
1. The server use the iPad to show the customer the total amount of money for the meal.
2. The customer choose a payment method and pay, as modelled by the “pay for meal” use case.
3. The server registers his or her name with the payment.

For further details of UML use case diagram and use case description, readers may refer to Schmuller (2004, §§6-7).

### 4.2.3. Section Summary

In summary, in this section, we have introduced the GRAPPLE modelling method. Modelling activities in this method is organized into five segments. Only the first two segments (i.e., requirements gathering and analysis) overlaps with the scope of IS analysis. Thus we only introduced modelling products of the first two segments.

We have found that only some of these modelling products are within the scope of IS analysis. We have illustrated them with examples. For the other modelling products, we have explained that it may be confusing to organize them in the first two segments because they are beyond the scope of IS analysis. We have proposed to solve this issue by moving them to the design segment.
4.3. ANALYZING GRAPPLE USING THE GENERALIZED VIEW

In this section, we study how do concepts in the generalize view apply the IS analysis modelling products of GRAPPLE. Recall from Chapter 2 that the concepts in the generalized view are classified into five categories (i.e., mental product, external product, context, theory, and language). In each subsection, we conduct the study for a category and investigate how do concepts in the category apply to the four types of IS analysis modelling products (i.e., activity diagram, class diagram, package diagram (of use cases), and use case diagram (including textual descriptions)) in GRAPPLE.

In addition, we also use this analysis as a test of the generalized view regarding its generalizability to GRAPPLE.

4.3.1. Applying Concepts in the “External Product” Category

In the generalized view, there are two concepts in the external product category: script and syntactic expression. Recall that a syntactic expression is an expression written in a modelling language, while a script is a collection of syntactic expressions recorded on an external medium (e.g., paper).

The example GRAPPLE modelling scripts (e.g., the UML activity diagram in Figure 18) given in the previous section are instances of the script concept, while the elements of the modelling scripts (e.g., the labeled bubble symbols in Figure 18) are instances of the syntactic expression concept.
4.3.2. Applying Concepts in the “Mental Product” Category

In the generalized view, there are two concepts in the mental product category: IS analysis model and mental construct. Recall that mental constructs are what people form in their minds to describe certain phenomena they perceive, while an IS analysis model is a conceptual model formed of mental constructs that are conceived to analyze the requirements of a new IS to be developed.

The models represented by the example GRAPPLE modelling scripts (e.g., the business process model represented by the activity diagram in Figure 18) given in the previous section are instances of the IS analysis model concept, while the mental constructs represented by the syntactic expressions of the modelling scripts (e.g., the activities represented by labeled bubble symbols in Figure 18) are instances of the mental construct concept.

4.3.3. Applying Concepts in the “Context” Category

In the generalized view, there are five concepts in the context category: domain (the portion of the world relevant to the project), IS (the system being developed that represents, informs about, and changes the state of the domain), stakeholder (the people, groups, or organizations that will affect or be affected by the project outcome), purpose (the intended use of an IS analysis model), and focal aspects (the types of things in the domain that are selected to be represented by an IS analysis model).

The first three concepts – stakeholder, IS, and domain – do not apply to modelling product type of GRAPPLE directly, but apply to their instances, i.e., the specific modelling products of GRAPPLE. Note that, in a typical IS development project, all types of
modelling products (i.e., domain understanding, requirements, conceptual schema, functionality specification, and domain knowledge) are associated with the same stakeholder, IS, and domain. Across different IS development projects, the same type of modelling products (e.g., conceptual schema) may be associated with different stakeholders, domains, and IS’s. Thus, stakeholder, IS, and domain are characteristics of specific instances of IS analysis model (e.g., specific conceptual schemas in specific projects), rather than a type of IS analysis model (e.g., the conceptual schema concept as a type of IS analysis model).

For all the example modelling products given in the example project in the previous section, the domain is the business environment of the restaurant, the information system (IS) is the “Wireless Interactive Network (WIN)” system to be developed and deployed in this domain, the stakeholders include developers, customers, owners, and employees.

We applied the other two concepts – purpose and focal aspects – to analyze the four types of IS analysis modelling products in GRAPPLE. We present the results in Table 16.

Table 16. Applying the Purpose and Focal Aspects Concepts to Analyze the Four Types of IS Analysis Modelling Products of GRAPPLE

<table>
<thead>
<tr>
<th>IS Analysis Model Type of GRAPPLE</th>
<th>Purpose</th>
<th>Focal Aspects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Diagram</td>
<td>To represent business processes.</td>
<td>Actions and sequence of actions in business processes.</td>
</tr>
<tr>
<td>Class Diagram</td>
<td>To represent domain structure.</td>
<td>Classes (for representing entity types), associations (for representing relationship types), and attributes (for representing properties of entity types).</td>
</tr>
<tr>
<td>Package Diagram (of Use Cases)</td>
<td>To list and classify system functions.</td>
<td>Use cases (for representing system functions), and packages (for classifying use cases).</td>
</tr>
<tr>
<td>Use Case Diagram (including textual descriptions)</td>
<td>To describe the system functions in details.</td>
<td>Use cases, actors, relationships between use cases and actors, relationships between use cases, details of each use case.</td>
</tr>
</tbody>
</table>
4.3.4. Applying Concepts in the “Theory” Category

In the generalized view, there are two concepts in the theory category: ontology and ontological construct. Recall that an ontology is a model that articulates the nature and structure of the world, while an ontological construct is an element of an ontology used to articulate a certain type of phenomena in the world regarding its nature and structure.

Like many other modelling methods, GRAPPLE does not explicitly specify any ontology that the stakeholders should use to perceive the domain. However, since GRAPPLE is an object-oriented method, it is implicitly associated with the object-oriented ontology. The object-oriented ontology (as an example ontology) and its ontological constructs (e.g., object, class, attribute, and association) are introduced in Schmuller (2004, §2), to which interested readers may refer for further details.

4.3.5. Applying Concepts in the “Language” Category

In the generalized view, there are seven concepts in the language category: modelling language, syntax, syntactic construct, syntactic rule, semantics, semantic domain, and semantic mapping. Recall that a modelling language is a language in which models are represented as scripts. The other six concepts are used to define the elements of modelling languages.

In GRAPPLE, all IS analysis model types are written in UML, and are directly named after the corresponding type of UML diagrams. i.e., activity diagram is named after UML activity diagram, class diagram is named after UML class diagram, package diagram
is named after the UML package diagram, and use case diagram is named after the UML use case diagram.

Each type of UML diagram is written in a subset of the UML, and thus each type of IS analysis modelling product of GRAPPLE is written in the corresponding subset of UML. For the purposes of this chapter, we do not need to dive into the details regarding the elements of each subset of UML. Readers who are interested in such details may refer to Schmuller (2004, §11) for UML activity diagram, Schmuller (2004, §§3-5) for UML class diagram, Schmuller (2004, §14) for UML package diagram, and Schmuller (2004, §§6-7) for UML use case diagram.

However, we consider it an issue for naming modelling product types of GRAPPLE after the corresponding subset of the UML. This is because some types of UML diagrams (e.g., the UML class diagram) can be used to model both the domain (e.g., domain entity types) and the software artifact (e.g., software object types) in GRAPPLE. Without specifying the context, a reader of a “GRAPPLE class diagram” may be confused about whether the modelling script should be interpreted into an analysis model of the domain, or a design model of the software artifacts. Therefore, we propose to fix this issue by assigning a distinct name to each modelling product type in each segment of GRAPPLE. For example, in the first two segments of GRAPPLE, the “activity diagram” can be renamed as the “business process model”, the “class diagram” can be renamed as the “domain structure model”, the “package diagram” can be renamed as the “use case list”, and the “use case diagram” can be renamed as the “use case model”.

155
4.3.6. Section Summary

In summary, in this section, we have used the concepts in the generalized view to analyze the IS analysis modelling product types of GRAPPLE. We used this analysis to understand GRAPPLE. In addition, this analysis indicates that the generalized view is applicable to GRAPPLE.

We have found that the modelling product types of GRAPPLE are directly named after the corresponding UML diagram types. We have explained that this may be confusing, because the same type of UML modelling script may correspond to different types of GRAPPLE model. We proposed to solve this issue by assigning alternative names to the modelling product types of GRAPPLE.

4.4. ANALYZING GRAPPLE USING THE CONSOLIDATED VIEW

In this section, we use the consolidated view from Chapter 3 to analyze GRAPPLE. First, based on the analysis result in the previous section, we map the method-specific IS analysis modelling product types of GRAPPLE to the method-independent IS analysis modelling product types of the consolidated view. Then, as an application of the consolidated view, we use the mapping result to evaluate GRAPPLE regarding its adequacy in supporting different types of IS analysis modelling products.
In addition, we also use this analysis as a test of the consolidated view regarding its generalizability to GRAPPLE.

4.4.1. Mapping IS Analysis Modelling Product Types Between GRAPPLE and the Consolidated View

In Chapter 2, we learned that there are different types of IS analysis models (i.e., the requirements engineering, conceptual schema modelling, and domain understanding areas). The major difference among these model types is that they focus on different aspects of the domain to serve different purposes of the related areas. Therefore, to map the IS analysis modelling product types of GRAPPLE to that of the consolidated view, we shall start with an analysis of GRAPPLE using the “purpose” and “focal aspects” concepts in the generalized view. In the previous section, the analysis result has been presented in Table 16. By comparing this analysis result to the relevant definitions of the consolidated view from Chapter 3, and with a few of our own interpretations of GRAPPLE, we are able to complete the mapping. Below, we first present the mapping result in Table 17, and then provide the explanations.
Table 17. Mapping of IS Analysis Modelling Product Types between the Consolidated View and GRAPPLE

<table>
<thead>
<tr>
<th>In the Consolidated View</th>
<th>Segment</th>
<th>In GRAPPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Understanding</td>
<td>Requirements Gathering</td>
<td>UML Activity Diagram and UML Class Diagram.</td>
</tr>
<tr>
<td>Requirements</td>
<td>Analysis</td>
<td>UML Package Diagram (of Use Cases)</td>
</tr>
<tr>
<td>Conceptual Schema</td>
<td>Analysis</td>
<td>(Updated) UML Activity Diagram, (Updated) UML Class Diagram, and UML Use Case Diagram (including textual descriptions).</td>
</tr>
<tr>
<td>Functionality Specification</td>
<td>Analysis</td>
<td>UML Use Case Diagram (including textual descriptions).</td>
</tr>
<tr>
<td>Domain Knowledge (optional(^{15}))</td>
<td>Analysis</td>
<td>(Updated) UML Activity Diagram, and (Updated) UML Class Diagram.</td>
</tr>
</tbody>
</table>

We explain the mapping below.

In the requirements gathering segment of GRAPPLE, the UML activity diagrams and the UML class diagrams are used to model and understand the domain, so as to elicit the requirements at the end of the segment. Modelling in this segment is conducted before envisioning the IS. Therefore, in this segment, they correspond to the “domain understanding” in the consolidated view.

At the beginning of the analysis segment of GRAPPLE, the diagrams produced in the requirements gathering segment are analyzed to elicit the system functions as use cases. The UML package diagrams (of use cases) are used to list and classify the use cases, without concerning whether the use cases are directly implementable by the IS. Therefore,

\(^{15}\) Recall from Chapter 3 that, like the functionality specification, the domain knowledge is part of the conceptual schema. However, unlike the functionality specification, it is not necessary to use the domain knowledge as input to IS design, and hence it is optional to separate the domain knowledge from the rest of the conceptual schema.
the UML package diagram (of use cases) in the analysis segment of GRAPPLE corresponds to the “requirements” in the consolidated view.

Later in the analysis segment of GRAPPLE, the UML use case diagrams (including textual descriptions) are used to model the details of each use case. In other words, they are used to refine the UML package diagrams (of use cases). Therefore, they correspond to the “functionality specification” in the consolidated view, and thus are also part of the “conceptual schema” in the consolidated view.\textsuperscript{16}

Also in the analysis segment of GRAPPLE, while the functions of the IS are envisioned as use cases, the iterative nature of GRAPPLE allows the diagrams in the requirements gathering segment to be updated to support the envisioned functions of the IS. An example update of the class diagrams in GRAPPLE can be found in (Schmuller, 2004, §19, Figure 19.3). However, the update of the activity diagrams was neither explicitly discussed in the GRAPPLE method specification (Schmuller, 2004, §15), nor illustrated with any examples in the example project (Schmuller, 2004, Part II. A Case Study). This is because the overall business processes remained the same before and after the example project, and only certain details of the business processes (i.e., the communication methods) are changed by the project. However, we think that, in principle, the IS could significantly change the overall business processes. If this happens, the corresponding update of the UML activity diagrams would be necessary in the analysis segment of GRAPPLE. For example, if the IS creates a new business process that allows

\textsuperscript{16} Recall from Chapter 3 that the functionality specification is the IS design relevant subset of the conceptual schema.
restaurant customers to make reservations online, a new business process diagram need to be added.

Essentially, in the requirements gathering and analysis segments of GRAPPLE, the UML activity diagrams and the UML class diagrams are used to model the behavior and structure of the domain, except the functions of the IS to be developed. If the project will significantly change the behavior and structure of the domain, they may be updated accordingly. Otherwise, they may remain unchanged. In both cases, at the end of the analysis segment of GRAPPLE, they should model the IS-independent part of the behavior and structure of the domain at the end of the project, and correspond to the “domain knowledge” in the consolidated view, and thus are also part of the “conceptual schema” in the consolidated view\textsuperscript{17}.

4.4.2. Evaluating GRAPPLE for Adequacy in Supporting Different Types of IS Analysis Models

In Table 17 in the previous subsection, we mapped the IS analysis modelling product types between GRAPPLE and consolidated view. From the mapping, we have two observations. First, all IS analysis modelling product types of GRAPPLE can be successfully mapped to elements of the consolidated view. This indicates that the consolidated view is generalizable to GRAPPLE. Second, all general types of IS analysis

\textsuperscript{17} Recall from Chapter 3 that the domain knowledge is the IS-independent part of the conceptual schema.
modelling products in the consolidated view are covered by at least some modelling
product types of GRAPPLE. This seems to indicate that GRAPPLE is overall adequate in
IS analysis modelling product type coverage.

However, when we dive into the details of the definitions of the IS analysis model
types of the consolidated view, we find a few issues with GRAPPLE.

First, definitions of the consolidated view indicate that the “functionality
specification” should include a delimitation of the part of the “domain knowledge” that
must be represented in the IS. This is because usually only some parts of the “domain
knowledge” must be represented in the IS, and the delimitation of this part should be
explicitly given as input to IS design. However, this delimitation is not included in the
corresponding modelling product types (i.e., the UML use case diagrams (including textual
descriptions)) in GRAPPLE. This issue can be solved by producing a delimitation of the
part of the updated UML class diagram that must be represented by the IS, in the analysis
segment of GRAPPLE.

Second, according to the consolidated view, the “functionality specification”
should refine the “requirements” into directly implementable system functions. A
statement in a functionality specification (or a step in a use case in GRAPPLE) is directly
implementable if it can be effected by a computer alone (Zave & Jackson, 1997). In
GRAPPLE, while the “functionality specification” (i.e., the UML use case diagrams
(including textual descriptions)) contains greater details than the “requirements” (i.e., the
UML package diagrams (of use cases)), it is not explicitly required to be directly
implementable. In other words, neither GRAPPLE nor UML had specified what level of
detail is required for the textual descriptions of the use cases. Fortunately, the solution can
be found in the definition of the “functionality specification” in the consolidated view. Based on that, we propose two criteria regarding the desirable level of detail. First, to avoid implementation bias, the interactions among the components of the IS (e.g., between the user interface and the database) should not be described. Second, any description regarding the interaction between the IS and its environment should be directly implementable.

Third, the consolidated view indicates that the “domain understanding” and the “conceptual schema” are two different types of IS analysis modelling products. However, in GRAPPLE, UML activity diagrams and UML class diagrams are first produced in the requirements gathering segment as the “domain understanding”, and are later updated in the analysis segment as part of the “conceptual schema”. The issue is that GRAPPLE did not clarify that, before and after this update, these diagrams correspond to different types of IS analysis modelling products. We propose to solve this issue by clarifying this difference and requiring saving two sets of these diagrams, one set for the “domain understanding” and the other set for the “conceptual schema”.

4.4.3. Section Summary

In summary, in this section, we have mapped the IS analysis modelling product types between GRAPPLE and the consolidated view. As an application of the consolidated view, we analyzed the mapping and found that GRAPPLE covers all general types of IS analysis modelling product types in the consolidated view. This analysis indicates that the consolidated view is generalizable to GRAPPLE.
However, we have found some issues of GRAPPLE through a closer examination of the relevant definitions of the consolidated view.

First, GRAPPLE’s “functionality specification” does not include a delimitation of the part of its “domain knowledge” that must be represented in the IS. We have proposed to solve this issue by adding this delimitation.

Second, neither GRAPPLE nor UML specified the desirable level of details in the use case descriptions. We have proposed two relevant criteria to solve this issue.

Third, users of GRAPPLE may not realize that it is necessary to save two sets of UML activity diagrams and UML class diagrams, one for the “domain understanding” and the other for the “conceptual schema”. We have proposed to solve this issue by requiring users of GRAPPLE to do so, and explained the necessity to do so.

4.5. CLARIFYING THE ROLES OF THE TWO VIEWS IN THE JOINT APPLICATION

Earlier in this chapter, we have applied the generalized and consolidated views together to analyze the IS analysis models produced by GRAPPLE. In this section, we clarify the roles they play in such applications. First, we clarify the differences between the roles played by the two views. Second, we clarify the relationships between the roles
played by the two views. Third, we discuss whether related work from existing literature can play the roles as well as the generalized and consolidated views.

4.5.1. The Differences Between The Roles of The Two Views

The two views play different roles when applied to analyze modelling methods.

The generalized view is applied to analyze and understand each IS analysis model individually. In such analysis, each individual IS analysis model is analyzed using the relevant concepts in the generalized view, for purpose of understanding its characteristics. For example, in Table 16, we used the “purpose” and “focal aspects” concepts from the generalized view to analyze each of the UML activity, class, package, and use case diagrams of GRAPPLE, so as to understand the intended use of (i.e., purpose of) and the types of things selected to be represented by (i.e., focal aspects of) each model.

In contrast, the consolidated view is applied to analyze and evaluate the entire collection of IS analysis model types of a modelling method together, rather than each of them individually. In such analysis, the entire collection of IS analysis model types of the modelling method is compared to the collection of IS analysis model types of the consolidated view, for purpose of evaluating the modelling method’s adequacy in supporting all types of IS analysis models in the consolidated view. For example, in Table 17, we conducted this kind of evaluation for the GRAPPLE modelling method.
4.5.2. The Relationships Between The Roles of The Two Views

Although the roles played by the two views are different, they are also related. As we have seen in §4.4.1, the understanding obtained by applying the generalized view is useful for mapping each IS analysis model type of the modelling method (e.g., GRAPPLE) to a model type of the consolidated view, and thus also for conducting the evaluation by applying the consolidated view.

4.5.3. Discussions On Employing Related Work From Previous Literature For the Roles Instead of The Two Views

As we have already elaborated in Chapters 2 and 3, the generalized and consolidated views differ from the related work because of their designed scope. Unlike the two views, no related previous work was proposed for the entire scope of IS analysis. In contrast, the previous work were proposed for either a specific area of IS analysis (e.g., Zave and Jackson (1997) proposed a terminology for the requirements engineering area, but this terminology may not apply beyond the scope of the requirements engineering area), or a particular type of modelling methods (e.g., Monarchi and Puhr (1992) proposed a typology of object-oriented modelling methods, but this typology is not applicable to other types of modelling methods such as agent-oriented modelling methods).

Hence, unlike the generalized and consolidated view, no previous work can play their roles for all types of IS analysis modelling products of all types of modelling methods. For sake of argument, consider a hypothetical example where someone propose a new
hybrid modelling method in IS analysis that takes an agent-oriented approach to produce the domain understanding, a goal-oriented approach to produce the requirements, and an object-oriented approach to produce the conceptual schema, domain knowledge, and functionality specification. To evaluate this hybrid modelling method, no previous work proposed for a particular area (e.g., the requirements engineering area) includes all types of modelling products, while no previous work proposed for a particular type of modelling method (e.g., object-oriented methods) can be applied to all parts of this hybrid modelling method. For example, the terminology of requirements engineering proposed in Zave and Jackson (1997) only includes modelling products of requirements engineering (i.e., requirements, domain knowledge, and functionality specification) but not others (i.e., domain understanding and conceptual schema) of this hybrid method. The research typology of object-oriented modelling methods proposed in Monarchi and Puhr (1992) can only be applied to the object-oriented part of this hybrid method (i.e., the part for producing the conceptual schema, functionality specification, and domain knowledge), but not the agent-oriented part (i.e., the part for producing the domain understanding) and the goal-oriented part (i.e., the part for producing the requirements) of this hypothetical hybrid modelling method in this discussion.

In conclusion, the generalized and consolidated views make original contribution in the application of analyzing modelling methods in IS analysis, because they are applicable to all types of modelling products of all types of modelling methods in IS analysis, while no previous work has equivalent applicability.
4.6. CONCLUSION OF CHAPTER

4.6.1. Summary

In this chapter, we have applied both the generalized and consolidated views to analyze GRAPPLE, a UML based modelling method. We have answered a number of questions in this analysis, as summarized in Table 18 below.

Table 18. The Application of the Generalized and Consolidated Views in Analyzing GRAPPLE

<table>
<thead>
<tr>
<th>View</th>
<th>Application</th>
<th>Demonstration in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalized View</td>
<td>How to distinguish IS analysis models from other types of models?</td>
<td>At the end of §4.2.1, we applied the generalized view to distinguish IS analysis models of GRAPPLE from other types of modelling products of GRAPPLE.</td>
</tr>
<tr>
<td></td>
<td>What are some of the concepts that apply to all types of IS analysis models?</td>
<td>In §4.3, we applied the generalized view to analyze the modelling products of GRAPPLE.</td>
</tr>
<tr>
<td></td>
<td>How to compare different types of IS analysis models for similarities and differences?</td>
<td>In Table 16 of §4.3.3, we applied the generalized view to compare the different types of IS analysis models of GRAPPLE.</td>
</tr>
<tr>
<td></td>
<td>What is the difference between model and script?</td>
<td>In §4.3.5, we explained that it is confusing to name GRAPPLE model types after the corresponding UML modelling script types, because the same type of UML modelling script may correspond to different types of GRAPPLE model. We proposed to remediate by assigning alternative names to GRAPPLE models.</td>
</tr>
<tr>
<td>View</td>
<td>Application</td>
<td>Demonstration in This Chapter</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Consolidated View</td>
<td>What is the connection between semantic domain and ontology?</td>
<td>Not applicable in this chapter, because the GRAPPLE method is not explicitly associated with any ontology.</td>
</tr>
<tr>
<td></td>
<td>Whether two modelling methods are complements or substitutes to each other?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>How to coordinate the use of multiple modelling methods that are complements to each other?</td>
<td>Not applicable in this chapter, because we only analyze a single modelling method (i.e., GRAPPLE).</td>
</tr>
<tr>
<td></td>
<td>How to compare multiple modelling methods that are substitutes to each other?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>How to evaluate a modelling method regarding its adequacy in supporting all types of IS analysis models?</td>
<td>In §4.4.2, we applied the consolidated view to evaluate GRAPPLE.</td>
</tr>
<tr>
<td>When Applied Together</td>
<td>How to apply the two views together to analyze a modelling method?</td>
<td>In §4.1.3, we proposed a way to apply the two views together to analyze a modelling method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>We demonstrated applying the two views together to analyze GRAPPLE.</td>
</tr>
<tr>
<td></td>
<td>What are the differences between the roles of the two views in such application?</td>
<td>We have answered this question in §4.5.1.</td>
</tr>
<tr>
<td></td>
<td>What are the relationships between the roles of the two views in such application?</td>
<td>We have answered this questions in §4.5.2.</td>
</tr>
</tbody>
</table>
4.6.2. Academic Contributions

In this chapter, we have made the following contributions.

First, as we have explained in §4.1.4, by analyzing GRAPPLE, which was not analyzed in previous chapters, we have put our generalized and consolidated views to further test regarding their generalizability. We found that they are both applicable to GRAPPLE. In particular, we have successfully applied the generalized view to analyze the IS analysis models produced by GRAPPLE in an example project, and applied the consolidated view to evaluate GRAPPLE regarding its adequacy in supporting all types of IS analysis models. This finding increases our confidence about the generalizability of the two views. In previous chapters, the two views were applied to analyze other modelling methods with quite different characteristics.

Second, in §4.1.3, we have suggested a way to apply the generalized and consolidated views together in analyzing modelling methods. In previous chapters, they were applied individually.

Third, in §4.5, we have clarified the differences and relationships between the roles played by the generalized and consolidated views in analyzing modelling methods in IS analysis, and explained that the related work from previous literature cannot play such roles as well as the two views, mainly because – unlike the generalized and consolidated views – they were not designed to support the entire scope of IS analysis.

Fourth, through the analysis, we have identified a few issues of GRAPPLE, and proposed solutions to these issues. These issues and solutions have been summarized at the
end of §§4.2 – 4.4. They are helpful for improving GRAPPLE. In addition, as far as we know, the work in this chapter is the first that analyzes and evaluates GRAPPLE.

### 4.6.3. Limitations and Future Work

Due to the scope of the generalized and consolidated views, one limitation of the work in this chapter is that only the system analysis part of GRAPPLE has been analyzed. The other parts (e.g., system design) of GRAPPLE are beyond the scope of the two views. In the future, we are interested in applying the generalized and consolidated views to analyze other modelling methods, and propose improvements to them as well.
Chapter 5: Conclusion

In conclusion, in this dissertation, we have proposed the generalized and consolidated views of modelling in IS analysis, and applied them to analyze a few modelling methods used in IS analysis. In this chapter, we conclude this dissertation. First, we summarize its contributions. Second, we compare the generalized and consolidated views for similarities and differences. Third, we discuss the potential benefits of using the two views as parts of the standard terminology for IS analysis. Fourth, we discuss the limitations of this dissertation. Finally, we list a number of future work.

5.1. SUMMARY OF CONTRIBUTIONS

In this dissertation, we have proposed the generalized and consolidated views of modelling in IS analysis, and demonstrated their contributions in advancing the understanding of modelling in IS analysis. Both views consists of concepts and relationship types that are helpful for understanding IS analysis modelling products, independent of any specific – thus applicable to all – IS analysis modelling methods.

The generalized view in Chapter 2 focuses on concepts that are applicable to all types of IS analysis models. Previous views of modelling adopted in IS analysis suffer from
being either generic for not tailored to the specific context of IS analysis, or limited for not applicable to the entire scope of IS analysis. The generalized view remediates by being neither generic nor limited. It provides utility in distinguishing IS analysis models from other models, in comparing different types of IS analysis models, and in explaining the representation and interpretation of IS analysis models.

The consolidated view in Chapter 3 focuses on the connection among different types of IS analysis models. These types of models were studied in different areas in isolation with inconsistent terminologies, creating difficulties for inter-area collaboration. The consolidated view remediates by connecting all different types of IS analysis models in a comprehensive picture. It provides utility in helping stakeholders to determine whether two IS analysis modelling methods are substitutes or complements, in comparing and choosing from modelling methods that are substitutes, and in coordinating the use of multiple modelling methods that are complements.

The joint application in Chapter 4 demonstrates how to apply the generalized and consolidated views together in analyzing and evaluating modelling methods. The modelling method used in the evaluation in Chapter 4 (i.e., GRAPPLE) is different from the ones used in previous chapters (i.e., Gaia and Tropos). Thus, the work in Chapter 4 provides additional evidence for the generalizability of the generalized and consolidated views and thus also the external validity of the studies in previous chapters of this dissertation. Moreover, in Chapter 4, we have clarified the differences and relationships between the roles played by the generalized and consolidated views in analyzing modelling methods, and discussed whether the related work from previous literature can play such roles in place of the two views.
The generalized and consolidated views can both be applied to analyze IS analysis modelling methods (or products of such methods). In Chapters 2 and 3, we have demonstrated applying them individually. In Chapter 4, we have demonstrated applying them together. In these applications, we have evaluated three IS analysis modelling methods: Gaia (Wooldridge et al., 2000), Tropos (Castro et al., 2002), and GRAPPLE (Schmuller, 2004, §15). The evaluation results are useful for improving these modelling methods.

5.2. COMPARING THE GENERALIZED AND CONSOLIDATED VIEWS

In this section, we compare the generalized and consolidated views for similarities and differences.

The two views both focus on modelling products in IS analysis. They both consist of the concepts useful for defining modelling products in IS analysis, and relationship types useful for understanding the connections among the concepts.

Despite such similarity, the two views are built to represent different knowledge about modelling in IS analysis, and – as a result – the concepts and relationship types in them may have different applicability.

The generalized view is built to represent knowledge that apply to all IS analysis models. Therefore, every concept and relationship type in the generalized view is applicable to all IS analysis models. For example, the generalized view includes the
“purpose” concept, which is applicable to all IS analysis models because every IS analysis model has a purpose.

In contrast, the consolidated view is built to represent knowledge about the connections among different types of IS analysis models. Therefore, the concepts and relationship types in the consolidated view may only be applicable to certain types of IS analysis models. For example, the consolidated view includes the “domain understanding” concept. This concept is only applicable to instances of domain understanding such as $i^*$ models (Yu, 1997) and $e^3$-value models (Gordijn & Akkermans, 2003), but is not applicable to instances of other types of IS analysis models such as conceptual database models represented by entity-relationship diagrams.

Both views can be applied to analyze modelling products or methods in IS analysis, individually (see §2.7 and §3.6) or together (see §4). However, as we have already elaborated with examples in §4.5, they play different roles in such analysis. In particular, the generalized view is applied to understand each IS analysis model produced by a modelling method individually, while the consolidated view is applied to evaluate the modelling method’s adequacy in supporting all types of IS analysis models in the consolidated view.

5.3. POTENTIAL BENEFITS AS PART OF A STANDARD TERMINOLOGY FOR IS ANALYSIS

The generalized and consolidated views consist of concepts and relationship types that describe knowledge about modelling in IS analysis. They are applicable to all
modelling methods in IS analysis. Therefore, they can be used as terminologies for modelling in IS analysis. If they are adopted as part of the standard terminology for IS analysis, they may potentially bring significant benefits. In particular, they are helpful for alleviating the “Yet Another Modelling Approach (YAMA)” issue (Oei, van Hemmen, Falkenberg, & Brinkkemper, 1992).

The “YAMA” issue refers to the situation that researchers become frustrated with the proliferation of modelling grammars. Wand and Weber (1993) proposed to alleviate this issue by evaluating modelling grammars against a theoretical benchmark (e.g., the Bunge’s ontology). Hence, competing modelling grammars can be compared with respect to their evaluation results. As we have demonstrated in §3.6.2, the consolidated view has similar application in evaluating modelling methods. As we have elaborated in §3.2.2, unlike Wand and Weber (1993) that used the Bunge’s ontology to compare competing modelling grammars with respect to their ability to represent different types of phenomena, we used the consolidated view to compare competing modelling methods with respect to their ability to represent different types of IS analysis models.

The “YAMA” issue can not only be alleviated by providing theoretical benchmarks for evaluating and comparing modelling grammars and methods, but also – we think – by providing a standard terminology to bring more commonalities to different modelling grammars and methods. We think such commonality can help researchers and practitioners to more easily learn a new modelling method based on the ones they learned before.

In current literature, different modelling approaches use different terms for the same types of IS analysis models. This creates difficulty for people to learn new methods. But such difficulty can be alleviated if commonality is brought to different modelling
methods using the generalized and consolidated views as standard terminologies, because people who begin to learn a new modelling grammar or method can benefit from their existing knowledge of other modelling grammars or methods.

For example, we analyzed three modelling methods in this dissertation: Gaia (in §3.6.1.1), Tropos (in §3.6.1.2), and GRAPPLE (in §4.4.2). They all support producing the conceptual schema model type in the consolidated view. However, as summarized in Table 19, the three methods have different method-specific names for the conceptual schema. Say that a person who knows Tropos starts to learn Gaia, it would take him or her significant effort to determine that the strategic rationale model in late requirements phase of Tropos is equivalent to the role and interaction models in Gaia. In contrast, if both Gaia and Tropos name the conceptual schema model type as the “conceptual schema” after the consolidated view, the learning will become easier.

Table 19. Model Types That Correspond to the Conceptual Schema in Three Modelling Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Model Types That Correspond to the Conceptual Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaia</td>
<td>Role Model, Interaction Model</td>
</tr>
<tr>
<td>Tropos</td>
<td>Strategic Rationale Model (in late requirements phase)</td>
</tr>
<tr>
<td>GRAPPL</td>
<td>(Updated) UML Activity Diagram, (Updated) UML Class Diagram, and</td>
</tr>
<tr>
<td></td>
<td>UML Use Case Diagram (including textual descriptions).</td>
</tr>
</tbody>
</table>

Some readers may suggest that some similar previous research work in the literature can also be used as standard terminologies as well. However, it is important to realized that none of them is proposed for the entire scope of IS analysis. Therefore, none of them can
be used as the standard terminology for the entire scope of IS analysis. Some of such previous works are proposed for specific areas of IS analysis. For example, Zave and Jackson (1997) is proposed for, and thus can only be used as a standard terminology of, the requirements engineering area. Furthermore, as we have discussed in §3.6.3, the area-specific terminologies are inconsistent with each other and therefore simply putting them together (without consolidation) will not result in a consistent terminology, and thus cannot replace the consolidated view. The other such previous research works are proposed for specific types of modelling methods, such as agent-oriented methods (Arazy & Woo, 2002; Dam & Winikoff, 2004; Sturm & Shehory, 2004). As we have discussed in §2.2.2, they may not apply to other types of modelling methods in IS analysis, and thus cannot replace the generalized view.

5.4. LIMITATIONS

In this dissertation, for both the generalized and consolidated views, we have decided to focus on the product aspect of modelling within the scope of IS analysis. Because of this focus, the work in this dissertation has a number of limitations.

First, since this dissertation focuses on information systems only, the generalized and consolidated views may not be applicable to the development of systems other than information systems. For example, a fax machine is not an IS because it does not interpret the handled information as the state of a certain domain (Olivé, 2007, p. 3). While the consolidated view prescribes that the analysis modelling products of an IS should include
a conceptual schema, this prescription may not be applicable to the development of a fax machine which is not an IS.

Second, this dissertation focuses on the “product” aspect of modelling in IS analysis. It has not discussed other aspects of modelling in IS analysis, such as the “process” aspect. For example, say that two IS analysis modelling methods produce the same types of IS analysis models via two different processes, the generalized and consolidated views would not be useful for analyzing their differences.

Third, while IS development involves several types of activities such as analysis, design, and implementation, this dissertation focuses on analysis only. Thus, the generalized and consolidated views in this dissertation are not applicable to models other than IS analysis models. For example, external schema and internal schema are two types of modelling products discussed in the standard abstract architecture of information systems (van Griethuysen, 1982), but one should not apply the generalized and consolidated views to analyze them because they are not IS analysis models.

Fourth, while several types of models are involved in IS analysis (e.g., domain understanding, requirements, conceptual schema, functionality specification, domain knowledge, and theoretical ontology), this dissertation focuses on IS analysis models, i.e., those that are produced in the process of developing a specific new IS for analyzing its requirements, such as domain understanding, requirements, conceptual schema, functionality specification, and domain knowledge. The generalized and consolidated views are not applicable to analyze other types of models involved in IS analysis. For example, theoretical ontology (e.g., Bunge’s ontology (Wand & Weber, 1990a, 1993)) is a type of model widely studied in IS analysis literature. However, the generalized view
cannot be applied to compare different types of theoretical ontologies, and the consolidated view cannot be applied to explain the connections among different types of theoretical ontologies, because theoretical ontologies are not IS analysis models.

In addition, the generalized and consolidated views in this dissertation is dependent on our interpretation of the literature. For example, in the generalized view, we obtained some concepts from particular publications chosen by us. As a specific example, we relied on Harel and Rumpe (2004) to obtain most concepts related to modelling languages, because we think – in the existing literature – it provides the best explanation for the fundamental concepts related to modelling languages. In the consolidated view, we obtained the concepts of each area-specific perspective from a publication in that area chosen by us. As a specific example, we relied on Zave and Jackson (1997) to obtain the concepts in the requirements engineering perspective, because we think – in the existing literature – it provides the best explanation for the fundamental concepts in requirements engineering. It is possible that the publications chosen by us can be replaced with better alternatives (that we are not aware of when writing this dissertation). For example, a future publication might explain modelling languages better than Harel and Rumpe (2004) does, because of reasons such as future evolutions of modelling languages in IS analysis. However, if such better alternative publications are found or become available in the future, one can use the same research methods in this dissertation to produce updated versions of the generalized and consolidated views.
5.5. FUTURE WORK

5.5.1. Extension to IS Design Modelling Products

In our future work, we are particularly interested in extending both the generalized and consolidated views – which focus on the scope of IS analysis in this dissertation – to the scope of IS design. While IS analysis focuses on identifying and documenting the requirements of an IS to support organizational activities, IS design focuses on defining software artifacts (including architecture, components, modules, interfaces, and data) to satisfy the requirements specified during IS analysis (Iivari et al., 2006).

In such future work, we can set similar research goals and use similar research methods as in this dissertation. The details are discussed below.

A generalized view of modelling in IS design will consist of concepts that are applicable to all types of IS design models. Similar to the research goals we set for the IS analysis version in this dissertation, it should achieve the following research goals. First, it should distinguish IS design models from other models. Second, it should be useful for comparing different types of IS design models. Third, it should explain the representation and interpretation of IS design models. Fourth, we should demonstrate its application in analyzing and comparing different IS design modelling products of the same IS design modelling method.

Similar to the research method that we used in this dissertation to build the IS analysis version, we can start with a study of the concepts useful for achieving the research goals, and then synthesize the concepts together to form the generalized view of modelling in IS design.
A consolidated view of modelling in IS design will present a typology of IS design models and explicate the connections among different types of IS design models. Similar to the research goals we set for the IS analysis version in this dissertation, it should achieve the following research goals. First, it should include all general types of IS design model, and explicate their connections. Second, it should provide a consistent terminology for modelling in IS design. Third, we should demonstrate its application in helping stakeholders to determine whether two IS design modelling methods are substitutes or complements, in comparing and choosing from modelling methods that are substitutes, and in coordinating the use of multiple modelling methods that are complements. Fourth, we should demonstrate its application in evaluating and comparing modelling methods regarding their adequacy in supporting all types of modelling products in IS design.

Similar to the research method that we used in this dissertation to build the IS analysis version, we can start with a survey of the areas related to modelling in IS design, summarize the perspective of modelling in each area, and synthesize all the perspectives to form the consolidated view of modelling in IS design.

5.5.2. Specialization on Particular Modelling Orientations in IS Analysis

There are different orientations for modelling in IS development. The modelling methods in each orientation have certain distinct characteristics that distinguish them from other modelling methods. For example, in Chapter 3, we studied two agent-oriented modelling methods (i.e., Gaia and Tropos). The agent-oriented methods have the distinct
characteristics that they represent the world as agents, goals, actions, and etc. In Chapter 4, we studied an object-oriented modelling method (i.e., GRAPPLE). The object-oriented methods have the distinct characteristics that they represent the world as objects, attributes, classes, associations, and etc.

Thus, in our future work, we are interested in specializing the generalized and consolidated views for some of these particular orientations for modelling in IS development. Such specialization is similar to the extension that we discussed in the previous subsection. For example, say that we conduct such future work for agent-oriented modelling methods, we can build a generalized view of agent-oriented modelling in IS analysis, that explains the concepts applicable to all types of IS analysis models produced by agent-oriented modelling methods. We can also build a consolidated view of agent-oriented modelling in IS analysis, that explains the connections among all types of IS analysis models produced by agent-oriented modelling methods.


Schmuller, J. (2004). *Sams Teach Yourself UML in 24 Hours* (3rd ed.).


185


