Design Support for e-Commerce Information Systems using Goal, Business and Process Modelling

Ananda Edirisuriya

November 2009

A Dissertation submitted to the Stockholm University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Department of Computer and Systems Sciences
Stockholm University and Royal Institute of Technology
Forum 100
SE-164 40 Kista
Stockholm
Sweden
Dedicated to the

Loving Mother and Father
Abstract

Enterprises use various models to find design solutions to their e-Commerce information systems. Goal, business and process models are parts of a chain of models used for this purpose. Business modelling requires structured methods to support design and traceability. Process modelling also needs structured methods to manage design complexity, traceability and flexibility. The thesis proposes several artifacts to address these challenges.

It proposes a method to design business models from goal models. We start by analyzing strategic goals and express goal model notions using business notions. A set of guidelines is proposed to design a business model. The method provides structured business model design and traceability of the decisions from business to strategic level.

The thesis also discusses a set of mappings to relate goal modelling language constructs to process modelling notions. We use BMM, i* and KAOS for goal modelling. The mappings are used to identify how these techniques can support process design. In addition, a set of mappings is proposed to relate business modelling language constructs to process modelling notions. We use the e³value model for business modelling. Based on these mappings a number of activities are identified to structure process activities from a business viewpoint.

We also propose a method to design process models using goal and business models as inputs. As a bridge between two input models and a process model, the notion of an activity dependency model is introduced. The transitions between models are performed by using guidelines. The method provides structured process model design, flexibility for process models and traceability of the decisions from operational to strategic and business levels.

These artifacts can be used for designing enterprise business and process models. We follow the design science research method used in information systems research. The evaluation builds on scenarios, implementations and a literature review.
Acknowledgements

First of all I would like to offer many sincere thanks to my academic supervisor, Prof. Paul Johannesson, for his excellent supervising and assistance given to me in gaining knowledge during my research work. He always been available to look at my research work and has helped me to restructure my thoughts whenever necessary.

I also would like to give many thanks to Birger Andersson, Maria Bergholtz, Prof. Hans Weigand and Tharaka Illaperuma for knowledge, I gained through discussions and working together. Special thanks go to Dr. Jelena Zdravkovic who helps my research studies in numerous ways during last stages. Also, many thanks go to the staff of the Department of Computer and Systems Sciences, Stockholm University and Royal Institute of Technology, for providing me with the necessary facilities to carry out my research studies.

I acknowledge SIDA/SAREC-IT Sri Lanka project for financing my studies through a split Ph.D. programme. My gratitude goes to Dr. Gihan Wikramanayake of the School of Computing, University of Colombo, Sri Lanka and Dr. Prasad Jawaweera of the University of Ruhuna, Sri Lanka for the help given to me from time to time as the local supervisors.

Finally, my gratitude goes to my wife Indramani, daughter Lumini and son Charindu for their patience and support during my stay in Sweden.
# Table of Content

## 1 Introduction

1.1 General Background ................................. 1  
1.2 Research Domain .................................. 5  
1.3 Problem Statement and Research Questions ........ 5  
1.4 Research Goals .................................... 7  
1.5 Research Purpose .................................. 8  
1.6 Research Methodology .............................. 9  
1.6.1 Research Approach .............................. 9  
1.6.2 Research Procedure ............................. 12  
1.7 Disposition ........................................ 21  
1.8 Scientific Publications ............................ 22  

## 2 Background Work

2.1 Requirements Engineering .......................... 29  
2.2 Running Scenarios .................................. 31  
2.2.1 Organizing Scientific Conference ............... 31  
2.2.2 Massively Multiplayer Online Game Playing .... 32  
2.2.3 Eye-Care Treatment .............................. 32  
2.3 Goal Modelling ...................................... 33  
2.3.1 The Business Motivation Model .................. 33  
2.3.2 The i* Goal Modelling Language ................. 37  
2.3.3 The KAOS Goal Modelling Language .............. 40  
2.3.3.1 Basic Constructs ............................... 40  
2.3.3.2 KAOS Requirements Engineering Process .... 43  
2.4 Business Modelling .................................. 44  
2.5 Business Process Modelling ......................... 50  
2.5.1 Research related to Business Process Design ... 50  
2.5.2 Business Process Modelling Languages .......... 53  
2.5.2.1 Business Process Types ....................... 54  
2.5.2.2 Business Process Modelling Notation ......... 54  

(BPMN): Core Elements
3 Goal Models to Business Models

3.1 Introduction
3.2 Formulating Goals and System Requirements
  3.2.1 Goal Formulation
  3.2.2 System Requirements Templates
  3.2.3 The KAOS Agent Dependency Model
3.3 Design e³value Model from KAOS Models
3.4 Method Application to Create a Goal-Based Business Model
  3.4.1 Method Application: Scientific Conference Scenario
  3.4.2 Method Application: Eye-Care Scenario
3.5 Conclusion

4 Exploring Goal Model support for Processes Design

4.1 Introduction
4.2 Mappings between Constructs of Goal Modelling Languages and Business Process Notions
  4.2.1 Organizational Perspective: Participant
  4.2.2 Functional Perspective: Activity
  4.2.3 Informational Perspective: Information Resource
  4.2.4 Behavioral Perspective: Control Flow
  4.2.5 Behavioral Perspective: Data Flow
4.3 Conclusion

5 Exploring Business Model support for Process Design

5.1 Introduction
5.2 Internal Structure of Value Object and Value Exchange 99
5.3 From Business Model to Process Design Framework 100
  5.3.1 Organizational Perspective: Participant 100
  5.3.2 Functional perspective: Activity 100
  5.3.3 Informational perspective: Information Resource 102
  5.3.4 Behavioral perspective: Control Flow 102
  5.3.5 Behavioral perspective: Data Flow 103
5.4 Transition Guidelines 103
5.5 Process Patterns 107
5.6 Conclusion 109

6  Goal and Value Driven Process Design 111
  6.1 Introduction 111
  6.2 Concepts and Notation for Activity Dependency Model 112
  6.3 Model Transition 116
    6.3.1 Goal and Business Model to Activity Dependency Model 116
      6.3.1.1 Guidelines to Design an Activity Dependency Model 116
      6.3.1.2 Method Application to Create an Activity Dependency Model 119
        6.3.1.2.1 Method Application: Scientific Conference Scenario 119
        6.3.1.2.2 Method Application: Massively Multiplayer Online Game Playing Scenario 122
    6.3.2 Activity Dependency Model to Process Model 126
      6.3.2.1 Guidelines to Design a Process Model 126
      6.3.2.2 Method Application to Create a Process Model 131
        6.3.2.2.1 Method Application: Scientific Conference Scenario 131
        6.3.2.2.2 Method Application: Massively Multiplayer Online Game Playing Scenario 134
  6.4 Processing Repetitive Actions 136
  6.5 Conclusion 138
7 Implementation Details and Comparison to Similar Research

7.1 Protégé and OWL Ontologies 141
7.2 Extensible StyleSheet Language Transformations (XSLT) 146
7.3 Comparison with Similar Research 150
7.4 Conclusion 153

8 Concluding Discussion and Future Research Directions

8.1 Concluding Discussion 155
8.2 Future Research Directions 160

References 161

Appendix I 169

Program Output I: XML description corresponding to the \( e^3 \text{value} \) model for the Scientific Conference Scenario presented in Figure 3.3 169

Program Output II: XML description of mappings of elements in the \( BMM \) to the process design framework for the Scientific Conference scenario 172

Program Output III: XML description of mappings of elements in the \( i^* \) language to the process design framework for the Scientific Conference scenario 173

Program Output IV: XML description of mappings of elements in the \( KAOS \) language to the process design framework for the Scientific Conference scenario 174
Program Output V: XML description of mappings of elements in the $e^3value$ model to the process design framework for the Scientific Conference scenario

Program Output VI: XML description of the process model corresponds to Figure 5.1

Program Output VII: XML description of the activity dependency model corresponds to Figure 6.2

Program Output VIII: XML description of the process model corresponds to Figure 6.7

Appendix II

Program I: XSLT source code to produce an $e^3value$ model from the KAOS models
List of Figures

1.1 Summary of the organization of research results 22
2.1 Main activities of RE process: requirements elicitation, specification and validation 30
2.2 An excerpt from a Business Motivation Model for the Scientific Conference scenario 36
2.3 An excerpt from an $i^*$ SD model for the Scientific Conference scenario 38
2.4 An example of a KAOS goal model for the Scientific Conference scenario 42
2.5 The UML class diagram of $e^3value$ business ontology 46
2.6 An $e^3value$ model for the Scientific Conference scenario 49
2.7 An example of a BPMN process model 55
2.8 A process meta-model based on the process design framework 59
2.9 ISO open-edi phases for business collaboration 61
3.1 An agent dependency diagram for the Scientific Conference scenario 67
3.2 An excerpt from a KAOS goal model for the Scientific Conference scenario 74
3.3 An $e^3value$ model for the Scientific Conference scenario 76
3.4 An excerpt from a KAOS goal model for the Eye-Care scenario 79
3.5 An agent dependency diagram for the Eye-Care scenario 79
3.6 An $e^3value$ model for the Eye-Care scenario 81
4.1 An excerpt from a process model for the Scientific Conference scenario 91
5.1 An excerpt from a process model for the Scientific Conference scenario 107
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>An activity dependency model to coordinate submissions in the Scientific Conference scenario</td>
<td>115</td>
</tr>
<tr>
<td>6.2</td>
<td>An activity dependency model to coordinate conference program in the Scientific Conference scenario</td>
<td>121</td>
</tr>
<tr>
<td>6.3</td>
<td>An excerpt from a <em>KAOS</em> goal model for the Massively Multiplayer Online Game Playing scenario</td>
<td>122</td>
</tr>
<tr>
<td>6.4</td>
<td>An agent dependency model for the Massively Multiplayer Online Game Playing scenario</td>
<td>123</td>
</tr>
<tr>
<td>6.5</td>
<td>An <em>e³value</em> model for the Massively Multiplayer Online Game Playing scenario</td>
<td>123</td>
</tr>
<tr>
<td>6.6</td>
<td>An activity dependency model to coordinate game access in the Massively Multiplayer Online Game Playing scenario</td>
<td>125</td>
</tr>
<tr>
<td>6.7</td>
<td>A process model to coordinate conference program in the Scientific Conference scenario</td>
<td>132</td>
</tr>
<tr>
<td>6.8</td>
<td>A process model to coordinate submissions, reviews and conference program in the Scientific Conference scenario</td>
<td>133</td>
</tr>
<tr>
<td>6.9</td>
<td>A process model for the Massively Multiplayer Online Game Playing scenario</td>
<td>135</td>
</tr>
<tr>
<td>6.10</td>
<td>An alternative process model for the Scientific Conference scenario</td>
<td>136</td>
</tr>
<tr>
<td>7.1</td>
<td>A part of the OWL code of the activity dependency model for the Scientific Conference scenario</td>
<td>144</td>
</tr>
<tr>
<td>7.2</td>
<td>A screenshot of the activity dependency model for the Scientific Conference scenario</td>
<td>146</td>
</tr>
<tr>
<td>7.3</td>
<td>A sample XSLT code of the Scientific Conference scenario</td>
<td>148</td>
</tr>
</tbody>
</table>
List of Tables

1.1 Summary of commonly used Research Methods in IS Research 11
1.2 Design Evaluation Methods 15
2.1 Summary of BPMN core elements 56
3.1 Summary of the design guidelines to construct an $e^3$value model from the KAOS models 71
3.2 Excerpt of objects relevant to the Scientific Conference scenario 74
3.3 Excerpt of objects relevant to the Eye-Care scenario 78
4.1 Summary of the mappings of constructs of goal modelling languages to process design framework 90
5.1 Summary of the mappings of $e^3$value model concepts to process design framework 101
5.2 Summary of the guidelines to construct a process model from an $e^3$value model 105
5.3 Primitive process patterns 108
6.1 Summary of the guidelines to construct an activity dependency model from an $e^3$value Model 118
6.2 Summary of the guidelines to construct a process model from an activity dependency model 130
6.3 Examples of compound process patterns constructed by combining primitive process patterns 137
7.1 Summary of the comparison of process design methods 152
1 Introduction

1.1 General Background

Advances in Internet Technology in recent times have provided enterprises with new ways of doing their business activities. As a result, enterprises can now use electronic means to sell, distribute, market and buy products and services. Doing business transactions including business-to-business and business-to-consumer utilising electronic means is known as Electronic Commerce (e-Commerce) (Hartman, 2000). Such an e-Commerce environment provides enterprises with an opportunity to work together as a business value network, i.e. a group of enterprises that together fulfil customer needs, each excelling in its own specific speciality, products and services (Tapscott, 2000). These e-Commerce activities rely on information systems connected through the Internet and accessed by one or many actors. The e-Commerce information systems are a ‘specific kind of information systems, interconnected via the Internet, and exploited by one or more actors, which support and enable the exchange of objects of economic value between various actors’ (Gordijn, 2002).

As the interest in e-Commerce activities increases, major challenges for research communities are to develop the methods and techniques necessary to analyse, design, implement and manage e-Commerce information systems. For this purpose, the first step is to elicit the system requirements for such e-Commerce information systems. The system requirements define what the system is required to do and the circumstances under which it is required to operate (Kotonya, 1998). Requirements Engineering
(RE) techniques can be employed to derive, document and validate system requirements for such e-Commerce information systems.

To develop these system requirements, analysts and designers employ different views. The goal, economic-value and process views are the ones commonly used. In Goal-Oriented Requirements Engineering (GORE), strategic goals are exploited as the basis for the requirements engineering process (Lamsweerde, 2001). Value-oriented requirements engineering explores the concept of economic value during the requirements engineering process (Gordijn, 2003). In the process view business use cases are used to identify functional requirements, which are activities and services to be executed by the system (UML, 2003). These different views are captured in different models. The strategic goals are captured in goal models, while economic values are captured in business models. The process view is captured in process models.

Goal models represent interests, intentions and strategies of different actors. The goal models can be employed as a driving force for eliciting business activities and alternative ways of doing them, giving thereby a motivation for certain business decisions (Raadt, 2005). Thus, a goal model defines the ‘why’ aspect in e-Commerce. For representing goals in this thesis, we use three established goal modelling techniques, Business Motivation Model (BMM) (BMM, 2005), i* (Yu, 1995) and Knowledge Acquisition in automated Specification (KAOS) (Dardenne, 1993). Business models identify who is offering what of economic value (i.e. values) to whom and expect what in return (Gordijn, 2002). The business models give a high-level view by focusing on the ‘what’ aspect in e-Commerce. The purpose here is to understand the economic reciprocity of value exchanges and evaluate economic sustainability for each participating actor. To capture business views, we employ the e³value business model (Gordijn, 2002). A process model depicts the behaviour of actors. A process model focuses on the ‘how’ aspect in e-Commerce by explaining operational and procedural details in terms of orchestration and message communications (Andersson, 2005). The purpose here is to represent the process knowledge of doing business, which is
necessary for designing e-Commerce information systems solutions in enterprises. We use the Business Process Modelling Notation (BPMN) to represent process views (BPMN, 2003).

The research and development in the area of business process technology have gained momentum in the last two decades. This is witnessed by the advent of new workflow and process management systems, new process modelling languages and other tools (Andersson, 2005). Despite their success, process technologies and in particular process modelling techniques, face a number of challenges that need to be addressed. These are managing design complexity, providing traceability and offering flexibility.

**Manage design complexity:** Business process design and redesign are complex and time-consuming tasks (Gong, 2005). A lot of time and effort is needed for this purpose. Therefore, we need to manage design complexity by making the designing process easier.

**Provide traceability:** Traceability is related to design decisions. Constructing a process model includes taking a number of design decisions that affect the structure of the model. It should be possible to trace these design decisions back to reasoning behind the design decisions.

**Offer flexibility:** Most of the time, the main structure of the process model is stable over time, but the details may vary from case to case. The flexibility of a process model means allowing changes to parts of the process structure without much alteration to its main structure. It should be possible to offer flexibility at both design time and run time to save time and effort.

To address these challenges, many attempts have been made in different research works. Many researchers have considered the process operational aspects by developing languages (YAWL, 2003) (UML, 2003) (BPMN, 2003), design patterns (UMM, 2005) (Aalst, 2002) and design frameworks (Curtis, 1992) for designing process models. Some have considered the use of enterprise goals as the basis for process design (Kueng, 1997) (Bider, 2002) (Soffer,
A process model based on goals provides a rationale for the operational choices made in the process level in terms of long-term intentions. Other researchers have considered the use of economic values as the ground for designing process models (Jayaweera, 2004) (Pijpers, 2007) (Andersson, 2006b). A process model that is based on a business model gives a business orientation to the process model. That means process model components take a business-oriented view.

The research and development in the area of business models have also gained wide attention during recent times. Some research is oriented towards designing languages and ontologies for business modelling (McCarthy, 1992) (Gordijn, 2002) (Fox, 1992) (Osterwalder, 2004). Other research concentrate on designing approaches and the use of enterprise goals as the basis for designing business models (Gordijn, 2006) (Raadt, 2005). A business model should conform to the strategic goals of the enterprise. That is, the realisation and the characteristics of the business model should be in accordance with the enterprise’s long-term interests. Otherwise, values being offered and exchanged are not motivated properly, neither involvement of certain business actors. In this situation, even if the process model is derived from the business model it would not conform to the intentional dimension of the business.

Therefore, in RE there is a strong need for structured ways that can be used to construct these enterprise models which we address in this research work.

This chapter is organised as follows. Section 1.2 discusses the research domain. The research questions are identified in Section 1.3. Section 1.4 provides the research goals that we expect to achieve. Section 1.5 discusses the research purpose. The research method is discussed in Section 1.6. The thesis flow is presented in Section 1.7. The chapter concludes by summarising research contributions in Section 1.8.
1.2 Research Domain

We summarise here our domain of interest and the scope of this research.

This research is about relating different enterprise models used in e-Commerce information systems. We primarily focus on goal models, business models and process models.

Our scope of interest is in designing business models and process models using other enterprise models.

1.3 Problem Statement and Research Questions

Enterprises use different models in developing their e-Commerce information systems solutions. Goal models, business models and process models are parts of a chain of models used for this purpose. The process modelling is a difficult task and several challenges need to be dealt with. There is a need for structured design methods that reduce the complexity of designing process models. The design decisions taken during designing, which affect the structure of the process model, should be justifiable by providing traceability. The process modelling techniques must offer flexibility, by allowing changes to process model structure. The research and motivations on the importance of process model flexibility can be found in (UMM, 2005) (Aalst, 2002).

On the other hand, business modelling also requires structured approaches and methods to capture business ideas. The business decisions made explicit in business models should be justifiable by going backwards and thereby providing traceability. The motivations for providing business model traceability can be found in (Raadt, 2005).
In this research work, we investigate how these challenges can be addressed by relating enterprise goal models, business models and process models.

The research question we are going to address is:

*How can the design of business models and process models be supported by the use of other kinds of enterprise models?*

This main question is divided into the following four sub-questions.

**Question 1:** *How can goal modelling be used to support the design of business models?*

The business model constructed should conform to the goal model in the sense that participating in a value network based on the business model will support an actor in fulfilling the goals of the goal model (Raadt, 2005). If the business model does not conform to the strategic goals, business decisions made in the business model are not properly motivated.

**Question 2:** *How can goal modelling be used to support the design of process models?*

Goals can be used as a starting-point for designing process models (Kueng, 1997) (Bider, 2002) (Soffer, 2005) (Nurcan, 2005). A process model that is based on goals motivates operational choices made in the process level in terms of long-term intentions. Nevertheless, different goal modelling languages offer different support to this design task. Therefore, it is important to explore how different goal modelling languages support the design of process models.

**Question 3:** *How can business modelling be used to support the design of process models?*

Business models can be used as a basis for designing process models (Gordijn, 2000). A process model that is based on economic values provides a business orientation for operational choices made at the process level. Thus, it is important to analyse business model components and investigate their support for designing process models.
Question 4: How can goal modelling and business modelling together be used to support the design of process models?

The goal and economic value are two views that are used in designing process models. A process model that is based on both goals and economic values makes operational choices in line with long-term intentions and at the same time provides a business orientation to them.

In chapters to come, we address these questions by exploring concepts used in goal models, business models and process models.

1.4 Research Goals

As discussed in Section 1.3, business modelling and process modelling techniques face a number of challenges that need to be addressed. Any methodology, technique, framework or tool support that has the capability to address these challenges would be welcome. The main goals of the research work presented in this thesis can be formulated as follows.

Corresponding to the main research question, the overall goal of this research work is: ‘To construct methods that support the design of business and process models by using other kinds of enterprise models’.

We decompose the main goal into the following sub-goals which are to be achieved as the work progresses.

Sub-goal 1: Construct a method to design a business model from a goal model.

To answer the first question, we build a method to construct a business model from a goal model. The method can be used to design a business model that conforms to the enterprise goals.

Sub-goal 2: Create mappings between goal modelling language constructs and process modelling notions.
To respond to the second question, the relationships between concepts of goal modelling languages and process notions are established. We consider here three established goal modelling languages (BMM, i*, KAOS) used in requirements engineering. The business processes are described by means of Curtis’s process design framework.

Sub-goal 3.1: Create mappings between business modelling language constructs and process modelling notions.

Sub-goal 3.2: Construct a method to design a process model based on a business model.

The third question is answered in two steps. First the relationships between concepts of a business modelling language and process notions are identified. The e³value model is used as a basis for business modelling. Curtis’s process design framework is used to represent business processes. During the second step, a method is developed to construct a partial process model based on a business model.

Sub-goal 4: Construct a method to design a process model based on a goal model and a business model.

To answer the fourth question, a method is developed to construct a process model based on a goal model and a business model. Such a process model conforms to the enterprise goals and at the same time meets the enterprise’s economic values.

1.5 Research Purpose

The target users of the proposed methods are modelling experts, information system developers, business managers, business consultants and decision-makers.

The methods proposed can be applied in the design of business models and process models. Thus, the intended use of the methods can be summarised as follows.
• To facilitate the construction of business models that conforms to the strategic goals of an enterprise.

• To facilitate the construction of process models that conforms to the strategic goals and economic values of an enterprise.

1.6 Research Methodology

1.6.1 Research Approach

Research is the process of discovering new knowledge based on a certain scientific study or some investigation on a critical issue. The research in information systems (IS) is basically performed with the aim of improving the effectiveness and efficiency of organisations by the application of information technology (IT).

The IS research is conducted in two complementary paradigms, behavioural science and design science (Hevner, 2004). The behavioural science research tries to build and verify new theories to describe some organisational phenomena or human phenomena. Design science research is concerned with applying knowledge and inventing new artifacts for human purposes.

March at el. (1995) outline a research framework that can be used for design science research. The framework has four design artifacts and two design processes which are based on the output produced and the activities carried out in research. The four design artifacts cover constructs, models, methods and instantiations.

• **Construct**: The constructs or concepts are used to form the vocabulary to define and communicate the problems and proposed solutions (Schon, 1993).

• **Model**: A model is a set of propositions used to represent a real-world situation, the design problem and solution statement by use of constructs (Simon, 1996).
• **Method:** A method can be an algorithm or guideline used to solve the designated problem (Nolan, 1973). Methods use constructs and models when outlining the solution.

• **Instantiation:** An instantiation is the implementation of constructs, models and methods in a working system.

The two design processes involved in the framework are *build* and *evaluate*.

• **Build:** Building is the process of constructing an artifact.

• **Evaluate:** Evaluation of an artifact refers to the process of assessing its output performance through a criterion.

The research approaches are categorised in different ways. A commonly accepted classification is the distinction between qualitative and quantitative research (Myers, 1997). Qualitative research uses qualitative data and focuses on understanding some social and cultural phenomena (Dawson, 2002). Quantitative research focuses on studying quantitative properties, phenomena and their relationships (Dawson, 2002).

Knowledge should be collected in a scientific way using appropriate research methods. Palvia et al. (2003) outline thirteen different research methods that are commonly used in IS research. These include speculation/commentary, frameworks, conceptual models, library research, literature analysis, case study, survey, field study, field experiment, laboratory experiment, interview, secondary data and qualitative research (Palvia, 2003). These research methods are briefly explained and captured in Table 1.1.

Action research and grounded theory are examples of qualitative research. Surveys and laboratory experiments are examples of quantitative research. A research can employ single or multiple methodologies depending on the research area, research question, research goals, researcher’s background and target audience (Palvia, 2003) (Robson, 2002). The utilisation of multiple methods provides greater confidence in the research findings (Palvia, 2003).
<table>
<thead>
<tr>
<th>Table 1.1 Summary of commonly used Research Methods in IS Research (Palvia, 2003)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speculation/Commentary</td>
</tr>
<tr>
<td>Frameworks and Conceptual Models</td>
</tr>
<tr>
<td>Library Research</td>
</tr>
<tr>
<td>Literature Analysis</td>
</tr>
<tr>
<td>Case Study</td>
</tr>
<tr>
<td>Survey</td>
</tr>
<tr>
<td>Field Study</td>
</tr>
<tr>
<td>Field Experiment</td>
</tr>
<tr>
<td>Laboratory Experiment</td>
</tr>
<tr>
<td>Interview</td>
</tr>
<tr>
<td>Secondary Data</td>
</tr>
</tbody>
</table>
The research goals in this thesis are about developing methods to support the design of enterprise business models and process models using other enterprise models. Thus, given the above background, research problems and design-oriented research goals, we choose to proceed with the design science philosophy as proposed by March at el. (1995).

### 1.6.2 Research Procedure

In Section 1.6.1, we described the design science framework used in this thesis work. Hevner et al. (2004) provide a conceptual framework that can be used to evaluate design science research for its usefulness, applicability and effectiveness. The framework consists of seven guidelines that the design science research must
satisfy. We use these guidelines as the basis for validating our work.

- **Design as an Artifact:** The output of design science research must be a viable artifact. The artifact can be a construct, a model, a method or an instantiation (Hevner, 2004).

As outlined in our research goals, we focus our attention on constructing structured methods to design enterprise business models and process models. To this end, we propose the following artifacts to achieve our defined goals.

1. A method to design a business model from a goal model.
2. A set of mappings to relate goal modelling language constructs and process modelling notions.
3. A set of mappings to relate business modelling language constructs and process modelling notions.
4. A method to design a process model based on a business model.
5. A method to design a process model from a goal model and a business model.

The methods that can be used to construct business models and process models from other enterprise models are clearly artifacts.

In addition, the mappings that relate goal modelling language (BMM, i*, KAOS) constructs and business modelling language (e3value) constructs with process model notions are also artifacts. These mappings belong to the construct category in the March classification (March, 1995).

- **Problem Relevance:** This requires a technology-based solution to an important real-world problem (Hevner, 2004).

With inter-organisational e-Commerce activities growing rapidly, there is a need for artifacts that can be used to analyse, design, implement and manage e-Commerce information systems. Different models are used to capture the requirements of these systems. The goal models capture the ‘why’ aspect of e-Commerce,
while business models capture the ‘what’ aspects in e-Commerce systems. The process models express ‘how’ e-Commerce activities can be carried out. The process modelling techniques face a number of challenges that need to be addressed. These include managing the complexity involved in designing process models, providing traceability for operational choices made and offering flexibility to process models. The business modelling also benefits by having structured design methods and providing traceability to business decisions made explicit in business models. In RE, there are needs for structured methods that are able to address these challenges. The aim of this thesis work is to construct structured methods that can be used to design enterprise business models and process models.

- **Design Evaluation:** The usefulness of the artifact must be evaluated in terms of utility, quality and efficiency (Hevner, 2004).

As explained by Hevner et al. (2004), evaluation is an essential part of the research process. The business environment sets up the requirements for evaluation of the designed artifact. The artifacts can be evaluated by use of quality attributes such as functionality, usability, flexibility, simplicity, completeness, accuracy, reliability and others. Hevner et al. (2004) discuss the methods available in the knowledge base that can be used to evaluate the designed artifact. These methods are summarised in Table 1.2.

We assess our work using qualitative descriptive methods: scenarios and informed arguments.

**1) Evaluate using Scenarios:**

The applicability of the mappings and methods are evaluated by applying them to different scenarios from different domains with different functional requirements. We use three scenarios for this purpose.
### Table 1.2: Design Evaluation Methods (Hevner, 2004)

<table>
<thead>
<tr>
<th>Category</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observational</td>
<td>Case Study</td>
<td>The artifact is studied in depth in a business setting</td>
</tr>
<tr>
<td></td>
<td>Field Study</td>
<td>Use of the artifact is monitored in multiple projects</td>
</tr>
<tr>
<td>Analytical</td>
<td>Static Analysis</td>
<td>The structure of the artifact is examined for static qualities (e.g. complexity)</td>
</tr>
<tr>
<td></td>
<td>Architecture Analysis</td>
<td>The artifact is studied for its fit into technical IS architecture</td>
</tr>
<tr>
<td></td>
<td>Optimisation</td>
<td>Demonstrate artifact optimal properties or provide optimality bounds on its behaviour</td>
</tr>
<tr>
<td></td>
<td>Dynamic Analysis</td>
<td>The artifact is studied for dynamic qualities (e.g. performance)</td>
</tr>
<tr>
<td>Experimental</td>
<td>Controlled Experiment</td>
<td>Examine the artifact in controlled environment for its qualities (e.g. usability)</td>
</tr>
<tr>
<td></td>
<td>Simulation</td>
<td>The artifact is executed with artificial data</td>
</tr>
<tr>
<td>Testing</td>
<td>Functional Testing</td>
<td>The artifact interfaces are executed to identify failures and defects</td>
</tr>
<tr>
<td></td>
<td>Structural Testing</td>
<td>Perform coverage testing to identify criterions that should be satisfied when testing an artifact</td>
</tr>
<tr>
<td>Descriptive</td>
<td>Informed Argument</td>
<td>Use information in the relevant literature to form convincing arguments to show the artifact’s utility</td>
</tr>
<tr>
<td></td>
<td>Scenarios</td>
<td>Use detailed scenarios (examples) to demonstrate the artifact’s utility</td>
</tr>
</tbody>
</table>
**Scenario 1**: This business scenario is about organising a scientific conference. It involves four actors, a program committee, a steering committee, an author and a reviewer. For the purpose of handling the conference a set of activities have been organised to receive submissions, allocate reviewers, evaluate submissions and organise the conference.

**Scenario 2**: The second business scenario is about online game playing. This scenario involves four actors: a game provider, an Internet service provider, a shipper and a customer. A game provider is responsible for producing the game contents as well as selling and distributing its software CDs to the customers. An Internet service provider is responsible for game hosting and Internet provisioning. A shipper provides the shipping service. A customer buys games access from a game provider and in return pays a fee.

**Scenario 3**: The third business scenario is from the Swedish health care domain and concerns eye treatments (REMS, 2007). The case encompasses three actors: a patient, a primary care physician and an eye-care specialist. A primary care physician provides basic treatments to patients. If more advanced treatments are necessary, the patient will be referred to an eye-care specialist. In this case, the referral information will be sent to the eye-care specialist by the primary care physician.

(2) **Evaluate using Informed Arguments:**

To demonstrate the formalisation, we have automated the mappings and guidelines using available software tools. We discuss these automation details in Chapters 7.1 and Chapter 7.2.

The artifacts are evaluated for their effectiveness using informed arguments by comparing them with other similar artifacts available in the literature. This is done with the intention of describing similarities and differences and explains the advances of our artifacts. We discuss these comparisons in Chapter 7.3.
Research Contribution: The design artifact must be novel, i.e. it introduces an improved or innovative approach to solving the designated problem. The contributions of the research must be clear and verifiable (Hevner, 2004).

As mentioned in the goal section, our primary aim is to construct methods that support the design of business models and process models.

The first goal of our research is to construct a method to design a business model from a goal model. For this purpose, the components in the goal model are expressed in terms of business notions. Thereafter, we propose a set of guidelines to design a business model using a goal model as input. The representation of goal model components using business notions and guidelines are an original contribution of this research. These results are published in publication 1 and publication 2.

The second goal is to create a set of mappings to relate goal modelling language constructs and process modelling notions. For this purpose, we explore three goal modelling languages (BMM, i*, KAOS) used in requirements engineering with respect to Curtis’s process design framework. A set of mappings is proposed to relate goal modelling language constructs to the process design framework. The mappings are original contribution of this work. The results are published in publication 3.

The third goal is to create a set of mappings to relate business modelling language constructs and process modelling notions. To achieve this goal, we have analysed a well-known business model (e³value model) with respect to Curtis’s process design framework. A set of mappings is proposed to relate business modelling language constructs to the process design framework. The mappings are original contribution of this research.

The next goal is to create a method to design a process model based on a business model. Based on the mappings created under the third goal, a set of guidelines is proposed to design a partial process model based on a business model. The guidelines are
original contribution of this research. The results are published in 
*publication 4.*

The final goal is to construct a method to design a process model based on a goal and a business model. This is done through an intermediary model. We use well-known goal and business modelling languages to model enterprise goals and business views. The intermediary model captures activities needed to coordinate, create and transfer values. Thereafter, a set of guidelines is proposed to map input models (goal and business) to an intermediary model, and then an intermediary model to a process model. The intermediary model and mapping guidelines are clear contributions. The results are published in *publication 5* and *publication 6.*

- **Research Rigour:** In design science research, rigorous methods should be employed during development and evaluation of the artifact (Hevner, 2004).

The artifact must be develop and evaluate using rigorous methods. It should be represent formally, logical and internally consistent. According to Hevner et al. (2004) rigour is achieved by using existing knowledge bases in design research.

The work presented in this thesis has a theoretical foundation in requirements engineering (Loucopoulos, 1995), value theory (Porter, 1985) and value constellation theory (Normann, 1993). We employ prior research for goal, business and process modelling which has strong relations to the above areas.

For goal modelling, we employ *BMM, i*, *KAOS* languages. *KAOS* is a semi-formal, mathematically-grounded modelling language used in requirements engineering for goal acquisition, definition, refinements and reasoning (Letier, 2001). To capture the business view we use *e³value* model. The work in *e³value* modelling is inspired by concepts in value theory and value constellation theory (Gordijn, 2002). For these languages comprehensive meta models can be found in *BMM* (BMM, 2005), *i* (Gemma, 2006), *KAOS* (Objectiver, 2007) and *e³value* model (Gordijn, 2002).
To represent the process view of enterprises we exploit the Business Process Modelling Notation (BPMN) (BPMN, 2003). The BPMN has strong relations to XML Process Definition Language (XPDL, 2005). The XPDL is based on standard XML technology (XPDL, 2005).

During implementation, we use the Protégé knowledge base system and Extensible Stylesheet Language Transformations (XSLT). Protégé uses standard OWL technology as its base and XSLT uses standard XML technology. Protégé can verify the consistency of the knowledgebase itself. Formalisation is demonstrated through the implementations. The meta models of BMM, i*, KAOS and e3value are implemented using Protégé. The mappings and guidelines are automated by developing XSLT transformation programs. Our artifacts are compared with existing ones available in the knowledge. This provides research rigour for the evaluation of the work.

- **Design as a Search Process:** The search for a useful artifact needs to employed available means to reach the desired ends by satisfying laws and regulations in the problem environment. In this process laws and regulations in the problem environment should not be violated. The means are the course of actions and available resources to construct the artifact. The ends are the goals and constraints of the solution (Hevner, 2004).

Design science is basically an iterative process. The design is an incremental activity. Feedback will be provided from the evaluation phase to the development phase to improve the quality of the artifact. This is done until the artifact meets the specified requirements (Hevner, 2004).

The other research methods we used are speculations/commentary, discussions and literature studies. The guidelines and mapping rules are developed by applying them to target scenarios iteratively until our goals are fulfilled.

**Speculation/commentary:** We use speculations/commentary to build the artifacts.
**Literature studies:** We use literature studies as a data collection method to accumulate existing knowledge on goal, business and process modelling researches. The scenarios that we use to test and evaluate the validity of the artifacts are based on literature studies.

As the research is focused on requirements engineering, we first collect relevant literature from requirements engineering using forward and backward tracing. This includes mainly goal, business and process modelling literature. As the research progresses, we collect other relevant literature from other areas, such as research methods and scientific communications according to the need.

**Discussions:** Discussions are used as a method to build the artifacts. Extensive discussions are held, especially with fellow academic researchers and comments from peer reviews are used.

- **Communication of Research:** The results of design science research must be effectively communicated to both technical audiences and management audiences (Hevner, 2004).

The works presented are communicated to both audiences. The guidelines are presented at a level that makes it easy for technical and management audiences to understand them. We automate the mappings and guidelines using available software tools. The source codes carry the technical details required by the technical programmers for implementing them. The graphical models are used to present the results of goal, business and process modelling. In addition, we use scenarios to explain the application of the artifacts.
1.7 Disposition

The thesis is structured as follows:

Chapter 2 provides an overview of the background literature required for the work. A brief overview of requirements engineering is presented. The scenarios used to describe the applications of the different artifacts are also presented. In addition, we discuss goal modelling and business modelling languages used in this thesis. The chapter also provides a brief overview of process modelling.

Our first research question, ‘how can goal modelling be used to support the design of business models?’ is answered in Chapter 3. We discuss how to express KAOS goal model components using business model notions. The chapter also provides some guidelines on how to construct a business model from the KAOS models.

The answer to the second research question, ‘how can goal modelling be used to support the design of process models?’ is discussed in Chapter 4. The chapter explores three goal modelling languages used in requirements engineering, BMM, i* and KAOS with respect to Curtis’s process design framework. Some mappings are discussed to relate goal modelling language constructs to process design framework.

In Chapter 5 we provide answers to our third research question, ‘how can business modelling be used to support the design of process models?’. The concepts in e³value model are explored with respect to Curtis’s process design framework. Some mappings are discussed to relate e³value model components to the process design framework. We also present some guidelines to construct a partial process model using an e³value model.

Our fourth research question on ‘how can goal modelling and business modelling be used to support the design of process models?’ is answered in Chapter 6. In this chapter we discuss a method to build a process model using an input goal and business model. This is done through constructing an intermediary model.
The chapter also provides some guidelines for constructing the intermediary model from input goal and business models, as well as a process model from the intermediary model.

Chapter 7 discusses the implementation details. We also compare our work with similar research studies. Finally, concluding discussions are presented in Chapter 8 with some discussion on future research work.

### 1.8 Scientific Publications

The different parts of the research results from work related to the thesis have been accepted and published in several forums. These publications are listed below. The research results are organized and presented as shown in Figure 1.1.

![Figure 1.1 Summary of the organization of research results](image)

Figure 1.1 Summary of the organization of research results
Publication 1

This paper discusses a method to construct a business model from a goal model. We use well-established techniques for presenting the two views: KAOS for goal view and $e^3$value for business view. The method starts by eliciting and analysing strategic business goals using a goal model. We have formulated the main elements of the KAOS language, such as goals and system requirements, in a uniform way and in terms of business notions. Four goal patterns are discussed and a set of templates is proposed for formulating system requirements. We have also proposed a set of guidelines for creating a business model that conforms to the goals.

In this paper Ananda Edirisuriya is the main contributor in all sections. This includes formulation of goals, system requirement templates and construction of design guidelines.

Publication 2

This paper discusses the alignment of business models with goal models. We use BMM for goal modelling and $e^3$value model for business modelling. The method starts with a BMM goal model and as-is $e^3$value business model and ends with a goal-aligned business model. The method relies on the existence of a link
between goal and business models, which is primarily provided through the notion of means. We have formulated goals and means in terms of business model notions. A set of rules is discussed to create a to-be business model using a goal model and an as-is $e^3 value$ model.

>Ananda Edirisuriya contributed in all sections of this paper. He gave the main contributions in goal and mean formulations.

**Publication 3**


In this paper we have addressed the problem of exploring goal modelling languages support for process design. We consider the use of three established goal modelling languages in requirements engineering, BMM, i* and KAOS. We explore language constructs of these goal languages with respect to different perspectives in the modelling of business processes. A set of mappings is discussed which relates the elements of BMM, i* and KAOS goal modelling languages to the elements that constitute business processes.

>Ananda Edirisuriya is the main contributor in all sections of this paper.

**Publication 4**


In this paper, we have contributed to a method for constructing a partial process model from a business model. We use $e^3 value$
model as the basis for business modelling. The method is grounded in a renewed analysis of the value object notion. Based on the analysis a set of processes is identified to be included in a process model.

*Ananda Edirisuriya contributed in all sections of this paper. He gave the main contribution in identification of different processes that are to be included in a process model.*

**Publication 5**


In this paper, we have discussed a method for constructing a process model from a business model. For business modelling, we use $e^3value$ model. The method uses a notion of an activity dependency model to bridge the gap between an $e^3value$ model and a process model. A set of process patterns is discussed for designing process models. These patterns are grounded on renewed analysis of value exchange. Finally, a set of guidelines is presented to construct an activity dependency model from an $e^3value$ model, also a process model from an activity dependency model.

*Ananda Edirisuriya is the main contributor in all sections of this paper. This includes construction of process patterns and design guidelines.*
Publication 6


The paper contributes to a method for constructing a process model from a business model. We introduce a notion of an activity dependency model to bridge the gap between a business model and a process model. The paper then discusses a set of guidelines to go from a business model to an activity dependency model and from an activity dependency model to a process model.

In this paper, Ananda Edirisuriya mainly contributed to the constructing guidelines for designing an activity dependency model from a business model.

Other Publications

Other works that this author has contributed to, but not are parts of this thesis.


2 Background Work

This chapter presents the background work required for the thesis. Section 2.1 provides a brief overview of requirements engineering. The running scenarios use to explain the applications of the different artifacts are discussed in Section 2.2. Section 2.3 and Section 2.4 discuss the goal and business modelling languages used. An overview of business process modelling is provided in Section 2.5. Section 2.6 concludes the chapter with a summary.

2.1 Requirements Engineering

Requirements Engineering (RE) is an approach used by information systems developers to develop system requirements. The system requirements are defined in the early stages of a system development in order to build specifications of what should be implemented.

According to Loucopoulos et al. (1995) ‘Requirements engineering is the process of developing requirements through an iterative co-operative process of analyzing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained’.

The main activities of the RE process involve requirements elicitation, specification and validation. These activities are summarised in Figure 2.1.

Elicitation

The purpose of requirements elicitation is to discover candidate requirements of the system to be developed, based on the weakness
of the current system as they emerge from domain understanding. The requirements are discovered through consultation with stakeholders, users, domain experts, system documents and so on. Different techniques are used to discover requirements, such as interviews, goal analysis, observation and scenario analysis (Lamsweerde, 2008). The output of the requirements elicitation process is a series of conceptual models that constitutes the knowledge required to understand the problem domain well.

![Diagram of RE process](image)

**Figure 2.1** Main activities of RE process: requirements elicitation, specification and validation (Loucopoulos, 1995).

**Specification**

The requirements specification phase has two main purposes:

- **Decide requirements**: The requirements are analysed in detail and stakeholders negotiate to decide which requirements are to be accepted. This is necessary to identify conflicting requirements, incomplete requirements and other constraints.
• **Document requirements:** Further requirements are documented at an appropriate level of detail. The requirements are documented in natural language and diagrams.

**Validation**

During the requirements validation phase specifications are checked to see whether they are in accordance with the stakeholders’ intentions. The requirements documents are carefully checked for consistency and completeness of requirements before they are used as a basis for the system development.

To develop the requirements of e-Commerce information systems, analysts and designers employ different views. They are goal view, economic-value view and process view. In Goal-Oriented Requirements Engineering (GORE) strategic goals are used as the foundation (Dardenne, 1993). Value-oriented requirements engineering uses the concept of economic value (Gordijn, 2002), while process view employed business use cases to identify functional requirements (UML, 2003).

### 2.2 Running Scenarios

This section describes the running examples used in this thesis to explain and illustrate different stages of the proposed artifacts.

#### 2.2.1 Organising a Scientific Conference

The Scientific Conference scenario (SC) is about organising a scientific conference. For this purpose a set of activities have been arranged to acquire submissions, assign reviewers, decide on papers and organisation of the conference.

This business scenario involves four actors: program committee, steering committee, author and reviewer. The task of the program
committee is to organise the conference. The authors send submissions to the program committee. The program committee assigns reviewers to each paper and sends papers for reviewing. The reviewers submit their review reports to the program committee. Upon receiving the review reports the program committee decides on acceptance of papers. The decision is sent to the authors together with evaluations. The program committee is responsible for providing the conference program. The steering committee is responsible for funding the conference.

2.2.2 Massively Multiplayer Online Game Playing

The Massively Multiplayer Online Game (MMOG) is about playing computer games through the Internet. There are four actors: game provider (GP), Internet service provider (ISP), shipper and customer. A GP is responsible for creating the game contents, selling and distributing game software on CDs to the customers. To sell and distribute games, a GP obtains the services from an ISP, who in turn receives a payment as compensation. To deliver software on CDs, a GP obtains the service of a shipper. The shipper receives a payment for shipping service. The customers access the game servers in order to play games. A GP is being paid for this service by a customer.

2.2.3 Eye-Care Treatment

The Eye-Care case is from the Swedish eye-care domain (REMS, 2007). The case encompasses three actors: patient, primary care physician and eye-care specialist. When a patient needs eye treatment, s/he goes to a primary care physician. After diagnosis the primary care physician provides treatment to the patient, getting a fee in return. If more advanced treatments are necessary, the patient will be referred to an eye-care specialist. In this case, the referral information will be sent to the eye-care specialist by the
primary care physician. The eye-care specialist provides advanced treatment to the patient and gets a fee in return.

2.3 Goal Modelling

Goals are an important aspect used to represent business strategies and decision support. They direct the enterprise toward concrete actions and as a consequence the actions are firmly based on a business justification. Goal models capture states (i.e. conditions) that enterprise stakeholders wish to achieve. As such they can be utilised as a driving force for elaborating requirements for business analysis (BMM, 2005) and system requirements (Dardenne, 1993) (Yu, 1995).

Research on goal-driven approaches has led to the development of a number of goal modelling techniques and languages, such as Business Motivation Model (*BMM*) (BMM, 2005), i* (Yu, 1995) and Knowledge Acquisition in automated Specification (*KAOS*) (Dardenne, 1993). The *BMM* was developed to build business plans in an enterprise. The i* framework is an intentional, agent-oriented approach for reasoning within an organisational environment and its information systems. The *KAOS* model was developed to collect system requirements specifications using organisational goals for automated systems.

2.3.1 The Business Motivation Model

The *BMM* was developed with the purpose of providing a structure for developing, communicating and managing business plans in an enterprise (BMM, 2005). A business model can be developed using elements in the business plan. Such a business model can be used as a foundation model for IS design phase. The *BMM* is constructed with respect to the enterprise for which a business plan is formulated.
According to BMM, there are two major areas in a business plan: (1) ends and means; (2) influencers and assessments.

- **End**: The Ends are the states the enterprise wishes to achieve. An end can be a *vision* or a *desired result*.
  - **Vision**: A vision provides an overall picture of where the enterprise wishes to go in the long run.
  - **Desired Result**: The desired results are more specific *goals* and *objectives* that the enterprise intends to achieve. A goal is a statement about a state or a condition that the enterprise wishes to achieve or sustain. A goal amplifies a vision. An objective provides an attainable, time-dependent and measurable statement about a goal. The objectives quantify goals.

- **Means**: Means is something that enterprises use to achieve the end, such as technique, capability or instrument. The means can be differently categorised into *mission*, *course of action* and *directive*.
  - **Mission**: A mission describes the overall function of the enterprise through the ongoing operational activities. A mission makes a vision operative.
  - **Course of Action**: A course of action contains the basic elements of the general plan or overall solution that the enterprise will take to achieve its desired results. A course of action channels efforts towards desired results. The BMM identifies two types of course of actions, *strategy* or *tactic*. A strategy represents the course of action necessary to achieve the goals. A mission is planned by means of strategies. A tactic describes the strategy in more detail and implements the strategy.
  - **Directive**: The directives control the behaviour of course of actions. A directive could be a *business policy* or a *business rule*. The business policies are general practices established within the company. They are not fully structured and are less formally expressed rules. The aim of business rules is
to formulate atomic, structured and expressive statements about the courses of action. The business rules can be derived from business policies. A directive governs course of action.

- **Influencer**: An influencer is anything that may impact on the achievement of means or achievement of ends. These influencers can be either internal or external to the enterprise. The infrastructure and internal resources are examples of internal influencers. Competitors, customers, environment, partners, regulations, suppliers and technologies are external influencers.

- **Assessment**: An assessment judges the effect of an influencer in achieving ends by using means. A commonly accepted approach for assessments is SWOT (SWOT, 2007). In SWOT, assessments are categorised as strength, weakness, opportunity and threat. Strength or weakness is used to judge internal influencers. Opportunity or threat is used to judge external influencers.

Figure 2.2 provides an excerpt from a *BMM* for the Scientific Conference scenario that illustrates the relationship between certain elements. The *BMM* is constructed with respect to a program committee. The goal that is to be achieved by this business transformation is ‘Publish high quality scientific papers’. This goal is quantified by the objective ‘Keep the acceptance rate between 10% - 15%’ in order to test its achievement. The strategy adopted to achieve this goal is ‘Strict reviewing procedure’. This strategy is implemented by the tactic ‘Review submissions through good reviewers’. These courses of actions are governed by the directive ‘Ask a committee member to review submissions when there is a tie’. The overall vision of the conference is ‘Be a popular scientific conference’. This vision makes operative by the mission ‘Coordinate submissions, reviewing, evaluations, conference and prepare proceedings’. The influencer ‘Availability of scientific researches’ may have an impact on the achievement of these means and ends. Assessment ‘Increasing trend for cooperate academic
work’ provides an opportunity to judge the impact of this influencer on these means and ends.

Figure 2.2 An excerpt from a Business Motivation Model for the Scientific Conference scenario
2.3.2 The \textit{i*} Goal Modelling Language

The \textit{i*} goal modelling language was developed originally with the purpose of intentional modelling and reasoning within an organisational environment (Yu, 1995). The main modelling concepts used in \textit{i*} are actors, goals, tasks, resources and soft-goals.

- **Actor:** An actor is an active entity who is capable of carrying out a certain action to achieve a certain goal, such as a program committee or an author.

- **Goal:** A goal is a condition or a state of affairs in the world that the actor would like to achieve. For example, ‘Conference Organised’ is a state that a program committee would like to achieve. How the goal is to be achieved is not specified, allowing alternatives to be considered.

- **Task:** A task specifies a particular course of action that produces a desired effect. For example, task ‘Assign Reviewer’ means assigning a reviewer to a paper.

- **Resource:** A resource is a physical or an information entity that is regarded as valuable by some actors. An actor views a resource as valuable because s/he can use it for producing other resources, accomplish a goal or a soft-goal or carry out some tasks. The resource ‘Review Report’ is used to take a decision on a paper.

- **Soft-goal:** A soft-goal is typically a non-functional condition with no clear-cut criteria for achieving it, for example, ‘Quality Reviewing’. From a soft-goal, one can decide why one alternative is chosen over the other.

The \textit{i*} provides two main models, the Strategic Dependency (SD) model and the Strategic Rationale (SR) model. In this work, we use the Strategic Dependency model as it can be used to model agent dependencies.

**Strategic Dependency Model:** The SD model is used to capture dependency relationships among involved actors. We call the
depending actor the *depender* and the actor who is depended upon the *dependee*. The object involved in the dependency relationship is called the *dependum*. The dependum can be a goal, a soft-goal, a task, or a resource.

- **Goal Dependency**: In a goal dependency, a condition for a state to be achieved is set by a depender. The dependee has the freedom to take any decision and choose a way to achieve the goal. A failure of the dependee to get the goal satisfied can result in the depender failing to get one of his own goals satisfied.

- **Soft-Goal Dependency**: A soft-goal dependency is similar to a goal dependency. The difference lies in the fact that the goal involves a soft-goal that is not objectively defined.

- **Task Dependency**: A task dependency expresses that an actor depends on a task that is performed by another actor. Here, the depender specifies how the task should be executed. The decision is made by the depender.

![Diagram](image)

**Legends:**
- Actor
- Task
- Resource
- Goal
- Dependency
- Task decomposition link
- Means-Ends link

**Figure 2.3** An excerpt from an *i* SD model for the Scientific Conference scenario
• **Resource Dependency**: In a resource dependency an actor depends on a resource supplied by another actor. Here, the issue of decision does not arise. The resource is usually an output of a certain deliberate action process.

Figure 2.3 provides an excerpt from an i* SD model for the Scientific Conference scenario. There are four actors, ‘Program Committee’, ‘Steering Committee’, ‘Author’ and ‘Reviewer’. The ‘Program Committee’ depends on the resource ‘Submission’ provided by the ‘Author’ and is responsible for providing the resource ‘Evaluation’. Further, the ‘Program Committee’ depends on the resource ‘Submission’ provided by the ‘Author’ and is responsible for providing the resource ‘Evaluation’. Further, the ‘Program Committee’ depends on the ‘Steering Committee’ in achieving the goal (state) ‘Payment to be Received’. Here, the program committee does not mention how this goal should be achieved. The steering committee can decide how the goal should be achieved. For example, the steering committee can provide the funding all at once or in instalments. In addition, the ‘Program Committee’ depends on the ‘Reviewer’ for the task ‘Review Report to be Provided’ to be performed. Here, the program committee specifies how to perform the task. For example, the program committee might ask the reviewer to upload the review report in a server or use an email service.

**Strategic Rationale Model**: The i* supports modelling relationships among intentional elements internal to actors. These internal relationships are called means-ends links, task-decomposition links and contribution links. They are shown in a SR model.

• **Means-Ends**: A means-ends link describes how to realise an end through a certain means. An end can be a goal, a soft-goal, a task or a resource, and means is usually a task.

• **Task-Decomposition**: A task-decomposition link describes how a task can be decomposed into different intentional elements, such as goals, soft-goals, resources, or other tasks.

• **Contribution-Link**: A contribution-link explains to what extent a means contributes towards achievement of an end. The contribution can be positive or negative. A positive value
indicates that a means supports the achievement of an end and a negative value indicates means support is negative.

2.3.3 The KAOS Goal Modelling Language

The Knowledge Acquisition in Automated Specification (KAOS) language is used in goal-oriented requirements engineering to collect system requirements using organisational goals. KAOS has been extended in (Lamsweerde, 2008) to introduce agent dependencies to its modelling mechanism.

2.3.3.1 Basic Constructs

We briefly discuss here the concepts of KAOS used in this thesis. The main concepts of KAOS are goals, agents, objects and operations.

- **Goal**: A goal is a prescriptive assertion of intent that the system should satisfy with the cooperation of some agents. Goals can be of different types, such as functional and non-functional. The functional goals concern with the services to be provided by the system. Non-functional goals concern with the quality of a service offered by the system, such as safety, security.

- **Object**: An object is a ‘thing’ of interest in the system, whose instances can be distinctly identified (from goal statements) and may evolve from state to state. An object can be an entity, an agent, an event or a relationship.

  - **Entity**: An entity is an autonomous object, whose instances may exist in the system independently of instances of other objects. They cannot control the behaviour of other object instances. For example, we can model the ‘Submission’ and ‘Review Report’ objects in our running scenario as entities.

  - **Agent**: An agent is an autonomous active object, who is capable of performing some specific role in a goal satisfaction.
They can control the behaviour of instances of other objects. An agent can be an organisation, a human, a device or a software.

❖ **Event**: An event is an instantaneous object (i.e. an object active in one state only) that triggers an operation performed by an agent. For example, an instance of the object that represents the event ‘Request for submission of scientific papers’ exists only in a state where a program committee has made a request for paper submissions.

❖ **Relationship (Association)**: A relationship is a conceptual object that links some other objects. For example, an intended relationship between an author and a submission relates the agent ‘Author’ to the entity ‘Submission’.

- **Operation**: An operation is an action performed by an agent. They can create objects and provoke object state transitions. The operations can have pre-, post- and trigger conditions. An event could be used to trigger or stop the execution of an operation.

In addition to the above constructs, **KAOS** defines certain relationships between these constructs.

- **System requirement**: A system requirement is a low-level goal that can be placed under the responsibility of a single agent.
- **Refinement**: The high-level goals can be refined into low-level system requirements by means of refinement relations.
- **Responsibility**: A responsibility relationship is used to assign a system requirement to an agent.
- **Operationalization**: An operationalization relationship is used to indicate an operation that realises a system requirement.
- **Performance**: A performance relation describes an agent who executes an operation.
- **Input**: An input link describes an object that is being utilised by an operation.
- **Output**: An output link describes an object that is being produced by an operation.
Figure 2.4 An example of a KAOS goal model for the Scientific Conference scenario

Figure 2.4 provides an excerpt from a KAOS goal model for the Scientific Conference scenario. The high level goal Achieve[Conference Organised] is refined into sub-goals Achieve[Submission Handled], Achieve[Submission Reviewed] and Achieve[Submission Evaluated]. The later is refined into the system requirement Achieve[Submission Selected]. The system requirement Achieve[Submission Selected] is assigned to the agent ‘Program Committee’ and is realised by the operation ‘Decide On Acceptance’. This operation is performed by the agent ‘Program Committee’ and it takes the objects ‘Review Report’ as the inputs and produces the object ‘Evaluation’ as the output.

The KAOS language uses several models for its modelling task. They include goal, object, responsibility, operation, agent interface and agent dependency models (Lamsweerde, 2008). To present the work in this thesis, we only need the goal model and agent dependency model.
Goal Model: The KAOS goal model is a set of interrelated goal diagrams. A goal diagram captures the business strategic goals, system requirements, agents, operations and their interrelationships. The high-level goals are decomposed into low-level system requirements which can be fulfilled by a single agent.

Object Model: The goal formulations may refer to specific objects. These objects are captured in detail in the object model.

Responsibility Model: The responsibility model comprises all the responsibility diagrams. A responsibility diagram captures all the system requirements that have been assigned to an agent.

Operation Model: The operational model captures the operations that agents must perform to achieve the goals. An operation works on the objects defined in the object model. Operations can create objects, trigger objects state transitions and activate other operations by sending events.

Agent Interface Model: An agent interface model declares the objects that are monitored and controlled by each agent.

Agent Dependency Model: KAOS uses i* notion of dependency to describe agent dependencies (Lamsweerde, 2008). An agent depends on another agent on a certain dependum to be satisfied. Here, dependum is defined as a goal. The agent dependency model is similar to the i* SD model, except in KAOS the dependum is a goal.

2.3.3.2 The KAOS Requirements Engineering Process

We briefly explain here the KAOS requirements engineering process. The first activity involves identifying goals and constructing the goal refinement graph. The high-level goals are refined into sub-goals until each sub-goal cannot be further refined and can be kept under the responsibility of a single agent. These low-level goals, also known as system requirements, drive the identification of objects and operations. The responsibility, agent interface and operation models are gradually obtained from the
goal model. Once the objects have been identified, the system requirements can be assigned to agents. The operations use input objects and produce output objects. These operations are responsible for realising goals.

2.4 Business Modelling

Enterprises engage in activities that create economic values, which are then exchanged with other business partners. The business models are used to facilitate communication between partners engaged in business activities. Business models are also used to explore new business ideas (Gordijn, 2003), as a starting-point in business process design (Gordijn, 2000). A business model of an enterprise describes ‘what’ economic values the enterprise offers to one or several customers and the architecture of the enterprise with its network of partners in creating, marketing and delivering value. It does not show any order of executing its business activities.

Research and development in the area of business modelling have gained wide attention in the recent past. Different researches introduce a number of languages and ontologies to model enterprise businesses. Among them, three widely used and well-established ontologies are Resource-Event-Agent (REA) (McCarthy, 1982), Business Modelling Ontology (BMO) (Osterwalder, 2004) and e³value model (Gordijn, 2002). The REA ontology was originally developed to manage double-entry bookkeeping in accounting information systems. Lately, REA has expanded and been applied in e-Commerce frameworks (UMM, 2003) and enterprise IS architectures (Hruby, 2006). The BMO provides a way to describe organisational business using its value propositions, infrastructure management, customer interface and financial aspects. The e³value business ontology was originally proposed to model multi-actor business collaborations.

Other research has focused towards developing methods to design enterprise business models. The relationship between goals
and business models is discussed in (Raadt, 2005) to identify goal-aligned economically profitable business services. A set of guidelines are proposed to transform an \(i^*\) goal modelling framework to an \(e^3\text{value}\) model. The approach put the emphasis on exploring web services using enterprise strategic goals and business modelling.

In this thesis, we use \(e^3\text{value}\) business ontology as the basis for business modelling. The following factors motivate us to do so. It provides a rich set of software tools to design and analyse multi-actor business collaborations, including a simple graphical tool. It also provides a minimal set of concepts and relations, thus making it easier to understand for all involved stakeholders.

The \(e^3\text{value}\) Business Model Ontology

The \(e^3\text{value}\) business ontology was originally proposed to model value networks of cooperating business partners (Gordijn, 2002). The ontology aims at identifying exchanges of objects of economic value (value objects) between the involved actors in business collaboration. The \(e^3\text{value}\) model also provides a graphical tool that enables visualising of multi-actor business collaborations. It provides tools to perform profitability analysis to determine whether a certain value network is sustainable or not. The ontology is based on well-known business concepts such as value chain, value constellation and investment theory (Gordijn, 2002). The \(e^3\text{value}\) model has been extended in (Andersson, 2006b), (Weigand, 2006) and (Pijpers, 2007) to incorporate other business notions which are not typically captured in business models.

The main concepts used in the \(e^3\text{value}\) model are: actor, value object, value port, value offering, value interface, value exchange, value transaction, market segment and value activity. The ontology is shown graphically in Figure 2.5 (Gordijn, 2002) using UML notations.

- **Actor:** An actor is an economically independent entity. An actor is perceived by its environment as often, but not necessarily, a
legal entity. A program committee is an example of an actor. An actor must be able to take economic decisions and be responsible for profits and losses. In a sustainable business model, each actor should be able to make a profit or create a value that in financial terms exceeds the costs of producing it.

- **Value object**: A value object is something that one actor transfers to another actor. It has an economic value for at least one actor involved in the transfer. A review report is a value object for the program committee, as it can be used to make a decision on a paper.

- **Value port**: A value port is used to provide or receive value objects to or from other actors. It has a direction, *in* or *out*. An *in-port* is used to receive a value object (e.g. receive a ‘Review Report’). An *out-port* is used to provide a value object (e.g. provide an ‘Evaluation’).

![Figure 2.5 The UML class diagram of e³value business ontology](image)

- **Value offering**: A value offering describes what an actor offers to (an out-going offering) or requests from (an in-going offering) another actor.
offering) the surroundings. A value offering is a set of equally directed value ports belonging to one actor. All ports in a value offering should exchange value objects or, if not, no port should exchange value objects. A value offering models the idea that value objects can only be requested or offered in combination. Offering or requesting combinations of value objects may sometimes be more profitable than a single value object. As an example, a telephone company offers a mobile phone (value object) with a subscription (value object).

- **Value interface**: A value interface is used to group value ports. It consists of in-port(s) and out-port(s) that belong to the same actor. Value interfaces are used to model economic reciprocity, also known as duality. This means an actor is willing to offer to another actor if s/he receives adequate compensation in return. The exchange of value objects across one value interface is atomic. That means all value objects in the value offering should be transferred. If it is not possible to transfer all value objects, then none should be transferred.

- **Value exchange**: A value exchange is a pair of value ports of opposite directions belonging to different actors. When an actor gives up something, another actor takes it up. It represents one or more potential trades of value objects between these value ports.

- **Value transaction**: A value transaction is a set of economic reciprocal value exchanges between one or more actors. A value transaction aggregates all value exchanges. If a transaction occurs, all value exchanges in it should occur, or none at all.

- **Value activity**: A value activity is an operation that produces value objects. For example, ‘Assign Reviewer’ is an activity carried out by the program committee. A value activity can be performed economically independently of other value activities. It must yield profits or should increase economic values for the performing actor. A value activity can be decomposed into smaller activities, but still these small activities should yield
profits. This works as a decomposition stopping rule for value activities.

- **Market segment**: A market segment is a group of value interfaces belonging to actors, who may value exchanging economic objects equally. Note: a market segment does not group actors.

### Additional $e^3$ value model constructs

The above constructs can be used to relate actors in a value network. An actor can, however, have multiple value interfaces. In order to relate these multiple value interfaces within the same actor, some additional constructs need to be introduced.

- A dependency element is a start stimulus, end stimulus, AND-fork, AND-join, OR-join, OR-fork, or value interface.
- A connection element is a pair of dependency elements.
- A scenario path is a sequence of dependency elements connected by connection elements. A start stimulus represents a consumer needs and triggers the exchange of value objects. An end stimulus represents a model boundary and terminates a value exchange. A scenario path shows the dependencies between value interfaces. The execution of a value interface in a scenario path will result in execution of other value interfaces connected to this value interface. It does not show the order of the value exchanges (Gordijn, 2003).
- An AND-fork connects a dependency element to one or more other dependency elements, while an AND-join connects one or more dependency elements to one other dependency element. An AND-fork splits a scenario into more sub-scenarios, while AND-join merges sub-scenarios into one scenario.
- An OR-fork connects a dependency element to one or more other dependency elements, while an OR-join connects one or more dependency elements to one other dependency element. An OR-fork models a continuation of a scenario in one direction
to be chosen from a number of alternatives. An OR-join merges two or more sub-scenarios into one.

An $e^3$ value model corresponding to the Scientific Conference scenario is shown in Figure 2.6. The business model shows various value exchanges between the actors. The legends give the meaning to the $e^3$ value model components. The names of value objects are attached as labels to the value exchanges. There are two actors, ‘Program Committee’ and ‘Steering Committee’, and two market segments, ‘Author’ and ‘Reviewer’. The value object an author provides to the program committee is ‘Submission (paper)’ and in return s/he gets the ‘Evaluation’. The value activity ‘Decide on Acceptance’ determines whether the paper can be accepted based on the review reports submitted by reviewers.

![Figure 2.6 An $e^3$ value model for the Scientific Conference scenario](image)
2.5 Business Process Modelling

The purpose of a business process model is to describe the internal business processes of an enterprise. They are used to steer day-to-day business activities of enterprises. A business process ‘is a specific ordering of work activities across time and place, with a beginning, an end, and clearly defined inputs and outputs; a structure for action’ (Davenport, 1992). There are many other definitions given for a business process. The main idea behind all of these definitions is that a business process is a sequence of activities that create value by transforming an input into a valuable output.

Research and development in the area of business process modelling have gained momentum during the last two decades. According to the literature, process design techniques, process modelling languages, frameworks and design patterns have proposed to analyse, model, describe and document business processes.

In the sections to follow, we describe research related to process design techniques, process modelling languages, Curtis process design conceptual framework and ISO open-edi business transaction phases.

2.5.1 Research related to Business Process Design

To handle the increased process design complexity in the form of control flows and data flows, there is a need for structured approaches to process design that merge the IT and business perspectives. Therefore, enterprise goals and business values have been recognised in a number of research works as the two major grounds for business process design.

Some research has considered the goal perspective as a starting-point for process design and redesign. An informal method for
designing process models from goal statements is proposed in (Kueng, 1997). The method tries to identify process concepts such as activities, roles and resources from the functional goals. In this method goals are captured first and represented graphically. In addition to goal definition, a measurement criterion (non-functional goals) is defined to assess the goal fulfilment. The high-level goals are decomposed into low-level goals until they can be realised in terms of activities. The input and output resources for activities are identified and activities are assigned to roles. The roles and resources are identified from goal statements. Finally, business process models are evaluated by measurement criteria to check that the designed business process fulfils the defined goals. The method does not discuss how to orchestrate activities.

Nurcan et al. (2005) have proposed a map model for modelling flexible business processes. The approach provides a representation system for process models based on intentions and strategies. A map is a directed graph with nodes and edges. The nodes are connected through edges. In the map model each intention is depicted as a node and each strategy is represented by an edge. A business intention is a goal and can be realised by executing a business process. A strategy is an approach to achieve an intention. A map consists of a number of sections. Each section is a triplet with source intention (I_i), target intention (I_j) and strategy (S_{ij}). A strategy S_{ij} shows a way to achieve the state change from I_i to I_j. A business map consists of a number of paths from ‘start’ state to ‘end ‘state’, each of them representing a business process model.

A state-oriented approach for process modelling is discussed in (Bider, 2002). The approach has its roots in mathematical system theory (Kalman, 1969). A state is defined by a number of state variables accepting real values. A business process is a trajectory in the space of all possible states, i.e. state flow. Each business process has a goal. A goal is defined as a set of conditions that needs to be achieved. The state that satisfies goal conditions is the final state of the process. The goal conditions define a surface in the state space of the process. The business process is driven
forward towards the goal state through activities which are executed by roles.

A process modelling framework and criteria for evaluating validity of process models is discussed in (Soffer, 2004). The validity is defined as the capability of the process to achieve a defined goal. The authors discuss how invalidity of processes can occur and ways to avoid such situations.

Other research has considered organisational values and constructive results have been obtained from analysis of the business models of organisations. A framework for designing process models is discussed in (Jayaweera, 2004). The framework uses the speech acts theory (Austin, 1975) and language action perspectives (Dietz, 2000) as its foundation. Based on these theories a set of contract-negotiation and contract-execution process patterns are proposed for use as building blocks in designing process models. It uses some action dependencies to order the activities. Using these process patterns and action dependencies, it proposes a set of production rules and uses a structured query session to move from a business model to a process model.

A methodology to chain a process model with a business model is proposed by Andersson et al. (2006b). The method starts by analysing the transfers of economic resources (i.e. values) among the actors involved in a business value network. A value transfer is an aggregation of three components; rights on a resource, custody of a resource and an evidence document. Based on this analysis, a set of processes is elicited, which could be used for constructing complex process models.

A concept of an $e^3$transition model is introduced in (Pijpers, 2007) to bridge the gap between an $e^3$value business model and a process model. The $e^3$transition model captures various business notions, which are not typically captured in an $e^3$value business model. In a value exchange in an economic $e^3$value model, the value objects (an object with economic value) and their rights are transferred together. In an $e^3$transition model the rights of a value
object and actual instance of value object can be transferred independently. Two important rights, ownership rights and possession rights have been identified. The ownership right is the right to use and claim the possession of a value object (Pijpers, 2007). The possession right is the right to have the possession of a value object (Pijpers, 2007), but gives no rights to use or consume it. If an actor has the ownership right s/he can claim it later. A set of steps have proposed to go from an $e^3$ value model to an $e^3$ transition model and then to a process model. The steps do not, however, provide any heuristics to order activities.

2.5.2 Business Process Modelling Languages

According to the literature there are many languages and tools that have been proposed to model business processes. Among them, Data Flow Diagrams, Unified Modelling Language (UML), Business Process Modelling Notation (BPMN) and Yet Another Workflow Language (YAWL) are the widely used process modelling languages. The Data Flow diagrams describe the flow of data between various units within an organisation (Stevens, 1974). The UML activity diagrams (UML, 2003) and BPMN (BPMN, 2004) provide standardised graphical notation to describe the business processes of an organisation. YAWL (YAWL, 2003) is a formally defined language based on Petri nets (Aalst, 2002) to describe complex control flow relations between business processes.

In this thesis, the process models are expressed in BPMN notation. BPMN provides a graphical notation to facilitate human communication between involved stakeholders of complex business processes, which might further be converted to an executable process language. There are two reasons for choosing BPMN as the modelling notation. (1) Simplicity: BPMN notations are easy to understand for a wide spectrum of stakeholders ranging from business domain experts to technical developers. (2) XML Process Definition Language (XPDL) (XPDL, 2005) and XML syntax can be employed to define business processes modelled in
BPMN notation. But UML or YAWL could also have been employed to illustrate our artifacts.

### 2.5.2.1 Business Process Types

In business process modelling a wide variety of information is being communicated to a wide range of stakeholders. To serve this purpose, business processes can be defined at three different levels of abstraction, namely private, public and collaboration (BPMN, 2004).

**Private (Internal) Business Processes:** These are generally known as workflows and describe business processes which are internal to a specific enterprise.

**Public (Abstract) Business Processes:** These represent the interaction between a private business process and another process or participant. Internal activities of the private business process are hidden.

**Collaboration (Global) Business Processes:** These processes model the interaction between two or more business parties. The internal activities of the private business process are detailed in addition to the message exchanges between them. The business processes we modelled in this thesis are collaboration business processes.

### 2.5.2.2 Business Process Modelling Notation (BPMN): Core Elements

This section discusses a selected set of core BPMN elements used in presenting the work in this thesis. These BPMN elements are *activity, pool, lane, data object, sequence flow, message flow, gateway* and *event*. These elements are summarised in Table 2.1. The full description can be found in the specifications for BPMN (BPMN, 2004).
To illustrate these modelling elements we show an example of a process model in Figure 2.7. The process model depicts activities carried out by two actors, ‘Author’ and ‘Program Committee’. Each actor is represented by a pool. The activities carried out by an actor are shown within a pool. The sub-process ‘Receive Submission’ in the ‘Program Committee’ pool is for receiving a submission (paper) from an author. An arrow from one sub-process to another represents a sequence flow, which shows the order of execution of activities. In each pool there is a ‘Start’ event to start the business process and an ‘End’ event to terminate the business process. The dashed arrows show the message exchanges between the actors.

![Figure 2.7 An example of a BPMN process model](image-url)
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>BPMN Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>A process definition consists of one or more activities. An activity represents a unit of work, which is performed by a participant or computer application. An activity can be a task or sub-process.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>A task is an atomic activity included within a process. A task is used when the work done in a process cannot be further decomposed into more details.</td>
<td></td>
</tr>
<tr>
<td>Sub-Process</td>
<td>A sub-process is used to execute a set of smaller activities.</td>
<td></td>
</tr>
<tr>
<td>Pool</td>
<td>A pool represents the participant(s) in a process. A pool contains activities and transitions between them. Lanes are partitions of a pool. A pool can contain one or more lanes.</td>
<td>Pool</td>
</tr>
<tr>
<td>Lane</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Object</td>
<td>Data Objects are used for annotation purposes. They provide information about inputs and outputs of activities.</td>
<td></td>
</tr>
<tr>
<td>Sequence Flow</td>
<td>A sequence flow shows execution order of the activities.</td>
<td></td>
</tr>
<tr>
<td>Message Flow</td>
<td>A message flow shows flow of the messages between participants.</td>
<td></td>
</tr>
</tbody>
</table>
| Gateway | A *gateway* is used for routing purposes to converge or diverge incoming sequence flows (paths). There are four types of gateways for branching, merging, forking, and joining of paths.  

Branching (Decision): Exclusive-XOR splits a single path into alternative multiple paths.  
Merging: Exclusive-XOR can also merge multiple alternative paths into a single path.  
Forking: AND-Fork splits a single path into multiple parallel paths.  
Joining: AND-Join synchronises multiple parallel paths into a single path. |
|---|---|
| Event | There are three basic event types, start, intermediate and end.  
*Start* event starts execution of the activities in a business process.  
*End* event terminates the flow of a business process.  
*Intermediate* event occurs between a start event and an end event. This is an event that occurs after a process has been started. |
2.5.3 Business Process Design Framework

A conceptual framework is proposed by Curtis et al. (1992) to classify different aspects that constitute a process model. In this framework, the components in a process model are viewed from four perspectives: organisational, functional, informational and behavioural.

- **Organisational perspective**: This perspective describes the distribution of the responsibility for executing process activities. The main focus here is on the notion of the actor. Korherr (2008) discusses three types of participant actors who can execute a process activity, organizational unit, role and software. If a participant is an organizational unit its members are performing the activity. A role is an abstract characterisation of responsibilities and behaviour of actors within a context, such as a reviewer. Software services can be used to automatically execute activities. Using the organisational perspective, it is possible to dedicate and control responsibilities of parties engaged in a process.

- **Functional perspective**: The functional perspective describes how a process can be decomposed into activities that are to be executed. An activity can be either atomic or composite. A composite activity can recursively be refined to atomic activities.

- **Informational perspective**: Here the main concern is the informational entities produced or manipulated by a process (Curtis, 1992). There are two main elements in informational perspective, resource and event (Korherr, 2008). A resource is an entity produced or consumed by an atomic activity (Korherr, 2008). A resource can be either traditional or informational. Products and services are traditional resources, while data and software are informational resources (Korherr, 2008). An event triggers an activity.
• **Behavioural perspective**: This perspective concerns the flows of activities and data within a process. A flow can be either a *control flow* or *data flow*. A *control flow* expresses when an activity is to be executed in relation to others. For specification of coordination rules among activities, process specifications rely on control flow patterns: sequence flow, parallel execution (AND-Split), conditional branching (OR-Split, XOR-Split) and synchronisation (AND-Join, OR-Join, XOR-Join, N-Out-of-M-
Join) (Korherr, 2008). These patterns make it possible to specify
decisions made according to certain business rules.

A data flow describes the flow of information resource from one
activity to another and message exchanges among actors.

For each perspective, we consider the elements that constitute a
process model. The elements include participant (actor),
informational resource (resource, event), activity, control flow and
data flow. The relationships between these elements are captured in
a business process meta-model, as shown in Figure 2.8. The figure
is inspired by the business process meta model presented in
(Korherr, 2008).

2.5.4 ISO open-edi Phases and Process
Patterns

Designing and creating process models are a complicated and time-
consuming task. A good design practice to overcome these
difficulties is to use already provided solutions. Process patterns are
a suitable candidate for this purpose. A pattern is a ‘description of
a general solution to a specific analysis and design problem, when
to apply the solution, and when and how to apply the solution in a
new context’ (Larman, 2004). It is up to a designer to compare the
alternatives and select the most attractive one given his/her
business goals. The ISO open-edi initiative (UMM, 2003) discusses
a set of phases to be used in business transactions. These phases
provide a fundamental set of process patterns to model business
collaborations. The phases include planning, identification,
negotiation, actualisation and post-actualisation. Figure 2.9 shows
these phases in a BPMN diagram. In the diagram, each phase is
represented by a sub-process.

Planning: In this phase customer and provider are engaged in
activities to identify the actions needed for selling or purchasing
goods and services. As an example, a distributor sends catalogues
to the potential customers.
**Identification**: This phase involves activities necessary to exchange data among providers and potential customers regarding selling or purchasing goods and services. For example, a provider sends a quotation to a customer.

![Diagram of ISO open-EDI phases for business collaboration](image)

**Figure 2.9** ISO open-edi phases for business collaboration

After these two phases the provider and customer have identified themselves to a certain level and the nature of the goods and services for purchase or sale.

**Negotiation**: In the negotiation phase contracts are proposed and accepted. The detailed specification of goods and services, quantity, price, terms and conditions are determined in this phase. If required, the parties involved may make bids and put forward counter offers. As an example, a customer sends an order request to a provider and in return the provider sends an order confirmation to a customer.

**Actualisation**: This phase includes all activities necessary to exchange goods and services between involved actors as agreed during negotiations. As an example, a provider sends the shipment of goods and a supplier makes the payment.

**Post-Actualisation**: This phase contains all activities and associated exchanges of information between involved actors after the goods and services are provided. Examples are after sales services and warranty coverage.
2.6 Summary

This chapter provided a brief description of the background researches. An overview of the requirements engineering was presented first. We also discussed the running scenarios used to explain applications of the different artifacts. The chapter also discussed the goal modelling techniques, BMM, i* and KAOS used for our study. In addition, we explained the $e^3$value business model ontology, which is used as the basis for business modelling. The researches relevant to the business process design and BPMN were too discussed. The Curtis’s process design framework which can be used to express the main elements that constitute a process model were also presented. Finally, a brief description of ISO open-edi phases that can be used in business collaborations were provided.
3 Goal Models to Business Models

3.1 Introduction

The goal view and the economic value view are two approaches used in Requirements Engineering (RE) to derive system requirements. The goal view is captured in goal models, where it is represented as interests, intentions and strategies of different actors. The economic values are considered in business models. Business models identify actors, economic resources (i.e. values), value exchanges among actors and also how values are created and offered. Goal models can be employed as a driving force for eliciting business activities and alternative ways of doing them, thereby giving explanations for the motivations for certain business decisions being pursued.

Ideally, a business model should conform to the enterprise goal model. That means the realisation and the characteristics of the business model should be in line with the enterprise's long-term intentions. Otherwise, the economic values exchanged and the involvements of certain business actors are not properly motivated. In this situation, even if the system model is derived from the business model it does not conform to the intentional dimension of the business. Therefore, in the RE process there is a need for structured methods to construct enterprise business models that conform to strategic goals.

In this chapter, we investigate the relation between goal models and business models. We use KAOS for goal modelling, a well-established goal modelling language used in system requirements engineering. The KAOS goal modelling language was selected as it
provides rich set of concepts to model system requirements. The \( e^3 \text{value} \) model is used as our business model ontology. The \( e^3 \text{value} \) model was chosen as it provides a rich set of concepts for modelling multi-actor business collaborations. This chapter addresses the research question: \textit{How can goal modelling be used to support the design of business models?} To address the described problem, we explain goal model components using business model notions. Our goal is to build a method to design a business model from a goal model. Using such a method ensures that an enterprise business model conforms to the enterprise strategic view point.

The chapter is structured in the following way. In Section 3.2 we discuss our approach. In this section, we discuss how goals and system requirements should be formulated. The transition steps are discussed in Section 3.3. Section 3.4 illustrates how to apply the method to two running scenarios. The chapter concludes with a discussion of the results in Section 3.4.

### 3.2 Formulating Goals and System Requirements

\textit{KAOS} is a goal-oriented requirement elaboration method used to derive system requirements for future systems from high-level goals. The \( e^3 \text{value} \) model is developed with the purpose of modelling value networks of cooperating business partners. In this section, we set the focus on how to express to goals and system requirements using business model notations.

#### 3.2.1 Goal Formulation

The formulation of goals is a difficult task, as goals can vary from day-to-day operational goals to strategic goals of enterprises. To overcome this problem, we suggest how to formulate goals in a uniform way using business model notions. \textit{KAOS} provides a set of
goal patterns to be used as guidelines for the acquisition and definition of goals. These patterns are based on the temporal behaviour required by the goals. KAOS discusses four such goal patterns, namely: Achieve, Cease, Maintain and Avoid (Dardenne, 1993).

We formulate a goal as a desired condition (state) on one or more features (property) of a resource (entity). Here, resource is an entity type KAOS object. The patterns used in KAOS are extended with the introduction of temporal behaviour to this goal expression.

Achieve goals: Goals requiring a condition on one or more features of an entity eventually holds.

Cease goals: Goals requiring a condition on one or more features of an entity eventually to stops hold.

Maintain goals: Goals requiring a condition on one or more features of an entity to always hold.

Avoid goals: Goals requiring a condition on one or more features of an entity never to hold.

For example, in the Scientific Conference scenario, goal ‘Conference Organized’ can be reformulated as Achieve[Organized (condition) a Conference (entity) to enhance the Knowledge (feature)]. The objective of a program committee is to organize (condition to meet) a scientific conference (entity) to enhance the knowledge (feature) of a certain research community. The goal modellers find, it is easy to formulate goals in this way and it serves to make goal expressions more uniform.

3.2.2 System Requirement Templates

Goal models can be constructed using the goal formulation described above. Once a goal model is constructed, high-level goals should be refined to sub-goals and then further to the level of system requirements.
Here a question arises – how can these system requirements be modelled? In the context of enterprise business, where actors exchange resources and money in a value network, most of the actions are about acquisition, production or provision of resources. Therefore, requirements in a value network can be modelled using these business model notions. In other words, requirements can be expressed by using the basic entities of business models describing the actors, resources and transfer of resources as well as the activities required to produce these resources. Thus, we can use a small number of templates to formulate the requirements involved in modelling value networks. A template has three parts: an operation, an entity (resource) and an agent, i.e. Template: [Operation, Entity, Agent].

We propose the following set of templates to construct the system requirements of value networks. As system requirements are low-level goals, we append the name of the goal pattern to the beginning of the template.

The templates are:

Template 1: Provide (Operation) Entity by Agent  
Template 2: Receive (Operation) Entity by Agent  
Template 3: Produce (Operation) Entity by Agent

For example, in the Scientific Conference example, the system requirement ‘Take a decision on a submission’ can be formulated as ‘Achieve[Decide (operation) on a Submission (entity) by Program Committee (agent)]’.

3.2.3 The KAOS Agent Dependency Model

As discussed in Section 2.3.3.1, KAOS uses $i*$ notion of dependency to describe agent dependencies. An agent depends on another agent for a certain dependum to be satisfied. KAOS defines dependum as a goal (Lamsweerde, 2008). We extend this further to accommodate objects and operations.
The dependum can be a goal, an object or an operation. In an object dependency, an agent depends on an entity (resource) supplied by another agent. In an operation dependency an agent depends on an operation performed by another agent. Here, the depender agent specifies how the operation should be executed.

Figure 3.1 shows a KAOS agent dependency diagram for the Scientific Conference scenario. The ‘Steering Committee’ depends on ‘Program Committee’ for the goal Achieve[Organized a Conference to enhance the Knowledge] to be achieved. The ‘Program Committee’ depends on ‘Reviewer’ for the entity ‘Review Report’ to be supplied. In Figure 3.1, we represent a goal by an angled rectangle, an entity by a rectangle and an agent by a hexagon.
3.3 Design $e^3$value Model from KAOS Models

In this section, we discuss how to construct an $e^3$value model from KAOS models. To summarize, KAOS goal model complements the $e^3$value model by exposing the strategic reasoning behind its value creating and value transferring activities. The notions of goals, system requirements, objects, responsibility assignments, operations and agent dependencies of the KAOS models provide a rich set of concepts to model the range of interests of actors involved in a value network.

We propose the following detailed guidelines (G) for constructing an $e^3$value model from KAOS models. These guidelines are summarized and presented in Table 3.1. Let $D$ be an agent dependency model in KAOS.

**Actor (Market Segment):**

$G1$: For each agent in $D$, a corresponding legal agent becomes an actor or market segment in an $e^3$value model depending on whether the involved agent is a single agent or multiple agents (who assign a same value to objects).

*Justification:* An agent in KAOS is an active object who is able to perform some manual or automated operation. An actor or market segment in an $e^3$value model is an economically independent entity who is capable of performing some activities to create and transfer values. Furthermore, actors and market segments are responsible for taking economic decisions resulting in profits and losses. Thus, a legal agent who is responsible for a KAOS agent can be mapped to actor or market segment in the $e^3$value model. For example, if the KAOS agent is a coffee machine, then the company who owns the coffee machine becomes the actor in $e^3$value model.

**Value Object:**

$G2$: An entity (resource) $R$, involved in a dependency in $D$, becomes a value object in the $e^3$value model if $R$ has an
economic value for at least one of the agents involved in the dependency.

*Justification:* An entity in KAOS is any autonomous object. This encompasses a broad class from informational and tangible to non-tangible resources. Value objects in the $e^3$value model are products, money, services and consumer experiences which actors think are valuable for them. Thus, only entities with economic values become value objects in $e^3$value model.

**Value Exchange:**

*G3:* A dependency in $D$ becomes a value exchange in the $e^3$value model, if the corresponding entity involved in the dependency becomes a value object (G2) in $e^3$value model.

*Justification:* As explained in Section 3.2.3, an agent depends on another agent for a certain dependum to be satisfied. The dependum can be a goal, an object (entity), or an operation. This can be interpreted as a value exchange in an $e^3$value model, if the dependency involves an economic resource.

**Value Interface:**

*G4:* A group of dependencies between two agents in $D$ can be mapped to value interfaces of corresponding actors (market segments) in the $e^3$value model, if the associated dependencies are mapped to value exchanges (G3) in $e^3$value model.

*Justification:* In KAOS, dependencies reflect relationships between agents. In $e^3$value model, actors exchange value objects through value interfaces. Therefore, a set of dependencies between two agents can be translated into value interfaces of corresponding actors or market segments, if the dependencies are mapped to the value exchanges.

**Value Activity:**

*G5:* An operation performed by an agent that produces (template 3) an entity (with an economic value) in KAOS becomes a value
activity of the corresponding actor or market segment (G1) in the $e^3\text{value}$ model.

**Justification:** An operation in KAOS is an activity executed by an agent and operates on objects. An operation may use input objects and may produce output objects. A value activity in the $e^3\text{value}$ model is an operation that produces value objects. Thus, an operation that produces entities (template 3) in KAOS becomes a value activity of the relevant actor (or market segment).

**Value Port:**

**G6:** An operation (template 1 and template 2) performed by an agent that provides or receives an entity in KAOS becomes a value port of the corresponding value interface (G4) in the $e^3\text{value}$ model.

**Justification:** According to the template 1 and template 2 an operation can be either provide or receive. A value port in $e^3\text{value}$ model is used to provide or receive value objects. Thus, operation (provide/receive) in KAOS is a value port of the relevant value interface.

**Value Exchange between two Internal Value Activities:**

**G7:** Two operations (performed by the same agent) linked by input and output links become value exchanges between corresponding value activities (G5) of the corresponding actor or market segment (G1).

**Justification:** This guideline is for modelling value exchange between value interfaces of two value activities of the same actor (market segment). Consider the two operations $op1$ and $op2$ in KAOS that are performed by the same agent. If the output of $op1$ is an input in $op2$ and $op1$ and $op2$ become value activities in an $e^3\text{value}$ model, then input and output relations of $op1$ and $op2$ become a value exchange between the two value activities.
Value Exchange between a Value Activity and a Value Interface of Actor (Market Segment):

G8: If an entity involved in a certain dependency in D becomes a value object (G2) and is an input or output of a certain KAOS operation (op) and op becomes a value activity (G5), then there will be a value exchange between value interface of the corresponding value activity and value interface (introduced by G4) of the corresponding actor or market segment.

Justification: This guideline is for modelling value exchange between a value interface of a value activity and a value interface of the corresponding actor (market segment). An agent dependency might involve an entity which might be an input or an output of a certain KAOS operation. If the operation is mapped to a value activity and an agent dependency is mapped to a value exchange, we can draw a value exchange between the corresponding value interfaces.

Table 3.1 Summary of the design guidelines to construct an e³value model from the KAOS models

<table>
<thead>
<tr>
<th>KAOS Models</th>
<th>e³value Business Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legal Agent</td>
<td>Actor or Market Segment</td>
</tr>
<tr>
<td>Entity (Resource)</td>
<td>Value Object</td>
</tr>
<tr>
<td>Agent Dependency</td>
<td>Value Exchange</td>
</tr>
<tr>
<td>Group of Agent Dependencies</td>
<td>Value Interface</td>
</tr>
<tr>
<td>Operation</td>
<td>Value Activity / Value Port</td>
</tr>
<tr>
<td>Operations related with Input – Output link</td>
<td>Value Exchange between two Value Activities</td>
</tr>
<tr>
<td>Agent Dependency AND (Operation Input link OR Operation Output link)</td>
<td>Value Exchange between a Value Interface of a Value Activity and a Value Interface of an Actor (Market Segment)</td>
</tr>
</tbody>
</table>
3.4 Method Application to Create a Goal-Based Business Model

This section discusses how to use the above method. The method involves taking an input goal model and agent dependency model and constructing a business model that conforms to the goal model. The main tool used here is the template design for modelling the requirements of a value network and the design guidelines discussed in Section 3.3.

To apply this method, the goal modeller first needs to construct a goal model. In constructing the goal model, the goal modeller can use the goal patterns discussed in Section 3.2.1. To construct the requirements, the modeller can use the templates discussed in Section 3.2.2. In addition, the goal modeller needs to construct the agent dependency model. Finally, using the guidelines discussed in Section 3.3, we can construct an e³value business model. The method can be summarized as follows:

1. The goal modeller constructs a goal model using the goal patterns and requirements templates.
2. The goal model will be extended by adding objects, agent responsibilities and operations. Furthermore, the goal model will be complemented with the agent dependency model.
3. The guidelines provided in Section 3.3 are applied to construct an e³value business model from scratch.

3.4.1 Method Application: Scientific Conference Scenario

Here, we apply our method to the Scientific Conference scenario. First, the goal model is constructed from the goal patterns and requirements templates in Section 3.2.
Figure 3.2 shows an excerpt from a KAOS goal model designed for the scientific conference scenario. The figure describes the top goals, sub-goals, system requirements, objects, operations and their relationships. The leaf nodes in the tree are system requirements. To formulate the goals we have used goal patterns as discussed in Section 3.2.1. The top goal Achieve[Organized a Conference to enhance the Knowledge] is refined into sub-goals Achieve[Submissions (resource) are Received (condition) to access the quality of Scientific Contributions (feature)], Achieve[Submissions are Reviewed for better Scientific Contributions] and Achieve[Review Reports are Evaluated for best Scientific Contributions]. The latter is further refined into system requirements Achieve[Evaluate Review Reports by Program Committee] and Achieve[Provide Evaluation by Program Committee]. The system requirement Achieve[Evaluate (operation) Review Reports (resource) by Program Committee (agent)] is constructed using template 3. Using the system requirements a modeller can identify the objects, agents and operations. The goal model is complemented with an agent dependency model to show agent dependencies.

**Identifying Objects**

The objects can be identified using leaf nodes (system requirements) in the goal tree. KAOS objects include entities, agents, events and relationships. Out of these constructs, it is sufficient to identify entities and agents to construct business models. The events are more important in designing process models. To identify entities and agents we use template 1 and template 2. The objects identified from the goal model in Figure 3.2 are shown in Table 3.2.
Identifying Agent Responsibilities

Once the agents have been identified, the system requirements can be assigned to the agents. An agent is responsible for a particular requirement. For example, requirement Achieve[Evaluate Review Reports by Program Committee] is assigned to the agent ‘Program Committee’.
Identifying Operations

The next step is to identify the operations, operations input and output objects and assignment of operations to agents. For example, from the requirement Achieve[Evaluate Review Reports by Program Committee], we can identify the operation ‘Evaluate (Decide on Acceptance)’. This operation uses the input resources ‘Review Reports’ and produces output resource ‘Evaluation’. This operation is assigned to the agent ‘Program Committee’. It is the responsibility of goal modeller to identify such information and show them in the goal model using input, output and performance links.

Construct the $e^3$ value model

The guidelines provided in Section 3.3 will help to construct the $e^3$ value model. An $e^3$ value model drawn for the Scientific Conference scenario is shown in Figure 3.3.

$G1$: We identify four agents from our running example, ‘Program Committee’, ‘Steering Committee’, Author’ and ‘Reviewer’. They are described as actors or market segments in $e^3$ value model (see ① in Figure 3.3).

$G2$: In the agent dependency model in Figure 3.1, goal dependency ‘Achieve[Organized a Conference to enhance the Knowledge]’ contains the entity ‘Conference’, which has some economic value for the program committee and steering committee. Therefore, the entity ‘Conference’ becomes a value object (‘Conference Program’) in $e^3$ value model (see ② in Figure 3.3).

$G3$: The object dependency ‘Review Report’ from ‘Program Committee’ to ‘Reviewer’ in the agent dependency model implies that former is depending on the latter for the entity ‘Review Report’. This dependency maps to a value exchange in $e^3$ value model, as the entity ‘Review Report’ is mapped to a value object. (see ③ in Figure 3.3).
Figure 3.3 An $e^3$value model for the Scientific Conference scenario

G4: Each group of dependencies between two agents is mapped to value interfaces. For example, two object dependencies ‘Submission’ and ‘Evaluation’ between ‘Author’ and ‘Program Committee’ can be mapped to the value interfaces of ‘Author’ and ‘Program Committee’. This is because both dependencies have been mapped to value exchanges in the $e^3$value model (see ④ in Figure 3.3).

G5: The operation ‘Decide on Acceptance’ produces the entity ‘Evaluation’, which has some value for ‘Author’. Such an operation can be mapped to a value activity of ‘Program Committee’ in the $e^3$value model (see ⑤ in Figure 3.3).

G6: The operation ‘Provide’ offers the entity ‘Submission’ and is performed by the author. Such an operation can be mapped to a value port of ‘Author’ in the $e^3$value model (see ⑥ in Figure 3.3).
G7: We use an example to describe this guideline. Consider the two operations ‘Decide on Acceptance’ and ‘Assign Reviewer’ performed by the ‘Program Committee’. The operation ‘Decide on Acceptance’ uses the output (‘Review Report’) of ‘Assign Reviewer’ as its input. If both operations are mapped to value activities of ‘Program Committee’ then ‘Review Report’ can be modelled by a value exchange between two value activities (see 7 in Figure 3.3).

G8: The operation ‘Decide on Acceptance’ performed by ‘Program Committee’ uses the ‘Submission’ and produces the entity ‘Evaluation’ (see Figure 3.2). The two object dependencies ‘Submission’ and ‘Evaluation’ between ‘Author’ and ‘Program Committee’ involve these two entities. The operation ‘Decide on Acceptance’ is mapped to a value activity (G5) and two dependencies introduce a value interface (G4) in ‘Program Committee’. Thus, we can draw value exchanges between the value interface of the value activity ‘Decide on Acceptance’ and value interface of the ‘Program Committee’ (see 8 in Figure 3.3).

The guidelines presented in Section 3.3 are implemented by developing the XSLT transformation programs. The program output I in Appendix I shows the XML description corresponding to the e³value model for the Scientific Conference scenario presented in Figure 3.3. The XSLT source code can be found in Appendix II program I.
3.4.2 Method Application: Eye-Care scenario

In this section, we apply our method to an Eye-Care scenario. Figure 3.4 shows an excerpt from a goal model designed for the Eye-Care scenario. To formulate goals we use the goal patterns discussed in Section 3.2.1. For example, the goal Achieve[Primary Treatment (resource) for better (condition) Eyesight (feature)] is formulated using the Achieve goal pattern. This explains that the condition ‘better’ eyesight (feature) holds for the resource ‘Primary Treatment’ at some point in time. To construct system requirements the templates given in Section 3.2.2 are used. For example, system requirement ‘Provide Treatment by Primary Care Physician’ is constructed using template 1. The system requirements can be used to identify the objects, agents and operations. The goal model is complemented with the agent dependency model shown in Figure 3.5.

Identifying Objects

The objects entities and agents can be identified from the leaf nodes (system requirements) in the goal model (Figure 3.4). These objects are shown in Table 3.3.

<table>
<thead>
<tr>
<th>Object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient</td>
<td>Agent</td>
</tr>
<tr>
<td>Primary Care Physician</td>
<td>Agent</td>
</tr>
<tr>
<td>Eye-Care Specialist</td>
<td>Agent</td>
</tr>
<tr>
<td>Fee</td>
<td>Entity</td>
</tr>
<tr>
<td>Treatment</td>
<td>Entity</td>
</tr>
<tr>
<td>Advanced Treatment</td>
<td>Entity</td>
</tr>
<tr>
<td>Voucher to get Advanced Treatment</td>
<td>Entity</td>
</tr>
</tbody>
</table>
Figure 3.4 An excerpt from a KAOS goal model for the Eye-Care scenario

Figure 3.5 An agent dependency diagram for the Eye-Care scenario
Identifying Agent Responsibilities

After agents have been identified, the system requirements need to be assigned to the agents. For example, the requirement ‘Specialized Examination for Advanced Treatment by Eye-Care Specialist’ is assigned to the agent ‘Eye-Care Specialist’.

Identifying Operations

The operations, their inputs and output objects need to be identified. In addition, these operations need to be assigned to the agents. For example, from the requirement ‘Examination (operation) for Treatment (resource) by Primary Care Physician (agent)’, we can identify the operation ‘Primary Treatment (Examination)’. This operation uses the input resource ‘Fee’ and produces the output resource ‘Treatment’. It is performed by the ‘Primary Care Physician’. The goal modeller should identify such pieces of information and show them in the goal model.

Construct the $e^3$ value model

The guidelines discussed in Section 3.3 can be used to construct the $e^3$ value model. An $e^3$ value model drawn for the Eye-Care scenario is shown in Figure 3.6.

G1: We can identify three agents in the Eye-Care scenario: ‘Patient’, ‘Primary Care Physician’ and ‘Eye-Care Specialist’. They are described as actors or market segments in the $e^3$ value model (see ① in Figure 3.6).

G2: In the agent dependency model, the goal dependency ‘Achieve[Primary Treatment (resource) for better Eyesight]’ contains the resource (entity) ‘Primary Treatment’, which has some economic value for the patient. Therefore, the entity ‘Primary Treatment’ becomes a value object in the $e^3$ value model (see ② in Figure 3.6).
G3: The object dependency ‘Fee’ from ‘Primary Care Physician’ to ‘Patient’ in the agent dependency model implies that former depends on the latter for the entity ‘Fee’. This dependency maps to a value exchange in the $e^3value$ model, as the entity ‘Fee’ is mapped to a value object (see ③ in Figure 3.6).

Figure 3.6 An $e^3value$ model for the Eye-Care scenario
G4: Each group of dependencies between two agents is mapped to value interfaces. For example, two dependencies ‘Achieve[Primary Treatment for better Eyesight]’ and ‘Fee’ between ‘Patient’ and ‘Primary Care Physician’ can be mapped to the value interfaces of ‘Patient’ and ‘Primary Care Physician’. This is because both dependencies have mapped to value exchanges in the e³-value model (see ③ in Figure 3.6).

G5: The operation ‘Primary Treatment’ produces the entity ‘Treatment’, which has some value for ‘Patient’. Such an operation can be mapped to a value activity of ‘Primary Care Physician’ in the e³-value model (see ⑤ in Figure 3.6).

G6: The operation ‘Receive’ collects the entity ‘Fee’ by the ‘Eye-Care Specialist’. Such an operation can be mapped to a value port of ‘Eye-Care Specialist’ in the e³-value model (see ⑥ in Figure 3.6).

G7: Consider the two operations ‘Diagnose’ and ‘Special Treatment’ performed by the agent ‘Eye-Care Specialist’. The operation ‘Special Treatment’ uses the output (‘Result’) of ‘Diagnose’ as its input. If both operations are mapped to value activities of ‘Eye-Care Specialist’ then ‘Result’ can be modelled by a value exchange between two value activities (see ⑦ in Figure 3.6).

G8: The operation ‘Primary Treatment’ performed by ‘Primary Care Physician’ uses the entity ‘Fee’ and produces the entity ‘Primary Treatment’. The two dependencies ‘Achieve[Primary Treatment for better Eyesight]’ and ‘Fee’ between ‘Patient’ and ‘Primary Care Physician’ involve these two entities. The operation ‘Primary Treatment’ is mapped to a value activity (G5) and two dependencies introduce a value interface (G4) in ‘Primary Care Physician’. Thus, we can draw value exchanges between the value interface of the value activity ‘Primary Treatment’ and the value interface of the ‘Primary Care Physician’ (see ⑧ in Figure 3.6).
3.5 Conclusion

Business models, representing actors, economic values and value exchanges make it easier to justify design decisions at the IT level and trace them back to business level. In this chapter we have discussed a requirement engineering method to design an enterprise business model that conforms to an enterprise goal model. The method starts by eliciting and analyzing strategic business goals using a goal model, and ends using a set of transformations, with the creation of a goal-aligned business model. We have used well-established techniques for presenting the two models, namely, KAOS for goal modelling and e3value for business modelling.

The proposed method provides a way to formulate main elements of KAOS models, such as goals and system requirements in a uniform way using business notions. It offers a structured way to design an enterprise business model and relates components in the KAOS models to those of a business model. The method allows complete traceability of decisions made from the business level to the strategic level. The method is simple to understand. Hence, following the method construction of a unique goal model and a business model would be an easy task.
4 Exploring Goal Model Support for Business Process Design

4.1 Introduction

Goal models are used in the early phases of Requirements Engineering (RE) for system design, where they help to capture the interests, intentions and strategies of different actors. The BMM, i* and KAOS are goal modelling techniques widely used for this purpose. The BMM is a modelling approach used to develop, communicate and manage business plans of enterprises. The i* is an intentional agent oriented approach for reasoning about organisational environment and its information system design. The KAOS provides a schema to collect system requirements and thereby to construct system models using enterprise goals in developing automated systems. We described these goal modelling languages in Section 2.3. The process models show the behaviours of actors. In particular, the process models show activities, actors who execute the activities, flow of resources, order of execution of the activities and data flows among activities. The goal models enable reasoning about operational choices made in process models and thereby give a motivation (i.e. ‘why’) for decisions made in process models.

Thus, there is a need to be able to describe and analyse, in a structured way, the interests of actors in order to design processes that conform to the strategic goals of the enterprises. The chapter answers the research question: How can goal modelling be used to support the design of process models? Our goal is to establish the relationships between concepts of the goal modelling languages and business process notions. To meet this need, we choose three
different goal modelling languages, BMM, i* and KAOS, used in goal-oriented requirements engineering. We then explore their support and propose some mappings between goal modelling language constructs and process model notions. These mappings help to find support of each of these languages for designing process models.

The chapter is organised as follows. In Section 4.2, we discuss the mappings between concepts in goal layer and process layer. Section 4.3 concludes the chapter with a summary of the study.

4.2 Mappings between Constructs of Goal Modelling Languages and Business Process Notions

Neiger et al. (2004) propose a framework that uses Curtis’s process modelling framework to compare goal modelling languages. Curtis’s process modelling framework is discussed in Section 2.5.3. We use Curtis’s process design framework to describe business process notions. For the four perspectives, organisational, functional, informational and behavioural, we consider the elements that compose the process model. These elements are participant actor, activity, resource, event, control flow and data flow (Korherr, 2008). As stated in Section 2.5, a business process is a sequence of activities that create value by transforming an input into a valuable output. An activity can be a composite activity or atomic activity. A participant performs an atomic activity. The control flow determines the order of executing the activities. The data flows describe the flow of informational resources between atomic activities. An event triggers an activity.

The relations between constructs of goal modelling languages and the elements in the process design framework are explored. The result is a set of mappings between concepts in the goal layer
and the process layer. The purpose here is to explore the support of the three goal modelling languages for the business process design. These mappings are presented informally in Table 4.1 and discussed in the text following. Part of the process model for a Scientific Conference scenario, expressed in BPMN, is shown in Figure 4.1.

### 4.2.1 Organisational Perspective: Participant

The main concern of this perspective, as described in Section 2.5.3, is the participating actors who perform process activities.

**BMM**: In BMM, the business plan is formulated with respect to the organisational unit that is considered to be the principal actor. BMM does not explicitly support actor roles. The BMM influencer is something that may impact on the achievement of means or ends. BMM influencers include suppliers, customers and competitors. Thus, the actors’ involvement can be obtained from the described concept of the organisational unit or from an influencer (see Section 2.3.1).

**i***: The i* language supports the actor element directly. The i* strategic dependency model represents the strategic dependencies among actors. An actor depends on another actor on a certain dependum. The dependum can be a goal, a soft goal, a resource or a task. Therefore, the participant element corresponds to the strategic actors involved in the strategic dependency model.

**KAOS**: An agent in KAOS is an autonomous active object who is capable of performing some specific role in goal satisfaction. A system requirement can be placed under an agent. An agent is responsible for executing operations in achieving goals. An agent can be an organisation, a human, a device or software. Thus, the participant element is a list of agents who are responsible for carrying out operations to achieve given requirements.

In the BPMN diagram such an actor can be shown as a lane or a pool (see Figure 4.1).
4.2.2 Functional Perspective: Activity

The main concern in the functional perspective is the unit of work, that is, activity, which is performed by an actor during a business process execution.

**BMM:** BMM does not support the concept of activity explicitly. In BMM, the course of action is an approach or a plan executed to achieve some desired results. A course of action can be a strategy (i.e. a plan or overall solution) or a tactic (an operative task). A tactic implements a strategy. Ultimately, a course of action can be realised by a business process. Thus, process activity originates from a strategy or a tactic.

**i*: The i* language defines a task as an action carried out by an actor to achieve a goal. Therefore, a process activity is obtained from a task performed by i* strategic actor.

**KAOS:** In KAOS, an agent performs an operation to realise a system requirement. An operation can create other objects and provoke object state transitions. Thus, an activity is an operation in KAOS.

In the BPMN diagram such an activity can be represented by a task or sub-process. The sub-process ‘Decide on Acceptance’ can represent a tactic, a task or an operation (see in Figure 4.1).

4.2.3 Informational Perspective: Informational Resource

As explained in Section 2.5.3, the informational perspective includes resources and events. A resource is an entity that is produced or consumed by an atomic activity. A resource can be an informational or a traditional resource. Products, money and services are examples of traditional resources. Data repositories and software applications are informational resources. An event can invoke an activity.
**BMM:** The *BMM* does not represent the concept of informational resources (resources and events) explicitly. Nevertheless, a *BMM* influencer includes the enterprise resources, which can be money, material, stocks, staff and assets. Therefore, *resource* elements can be extracted from *influencers*.

*i*:* The *i* considers resource as an entity that is required to complete a hard goal, a soft goal or some task or produce other resources. A resource can be a physical or an informational entity. Therefore, the *resources* in *i* correspond to the *resource* element in the information perspective.

**KAOS:** An object is a thing of interest in the system to be modelled whose instances can be identified and change from state to state. A *KAOS* operation can produce objects in realising system requirements. *KAOS* objects can be an entity, an agent, an event or a relationship. An entity is an autonomous object that does not rely on other objects. An agent is an active object who can perform operations to achieve goals, whereas an event object is used to trigger operations. Relationship objects define relations between other objects. In our scenarios, ‘Submission’, ‘Review Report’, ‘Evaluation’, ‘Eye Treatment’ and ‘Fee’ are all traditional resources. Therefore, *entities* correspond to *resources* in the process design framework. An *event* object corresponds to an *event* element in the process design framework.

In a *BPMN* diagram, a resource can be represented by data objects. These data objects provide information about what activities are required to perform a certain action or what output they produce. The data object can be attached to an atomic activity (task in BPMN). The task ‘Decide on Acceptance’ takes the resource ‘Review Report’ as input and produces ‘Evaluation’ as output (see figure 4.1). An event can be represented using *BPMN* event element (e.g. Start event) (see figure 4.1).
Table 4.1 Summary of the mappings of constructs of goal modelling languages to process design framework

<table>
<thead>
<tr>
<th>Perspective</th>
<th>Process Model Element</th>
<th>BMM</th>
<th>i*</th>
<th>KAOS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Organisational</strong></td>
<td>Participant (Actor)</td>
<td>Organisational unit (Principal actor), subset of Influencers</td>
<td>Strategic actors</td>
<td>Agent</td>
</tr>
<tr>
<td><strong>Functional</strong></td>
<td>Activity</td>
<td>Course of Action (Strategy, Tactic)</td>
<td>Task</td>
<td>Operation</td>
</tr>
<tr>
<td><strong>Informational</strong></td>
<td>Resource Event</td>
<td>Subset of Influencers</td>
<td>Resource</td>
<td>Entity</td>
</tr>
<tr>
<td><strong>Behavioural</strong></td>
<td>Control Flow</td>
<td>Business Rules</td>
<td>Resource Dependency</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Goal Dependency</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Task dependency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data Flow</td>
<td>Business Rules</td>
<td>Resource Dependency</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Operations Input-Output relations</td>
<td></td>
</tr>
</tbody>
</table>

4.2.4 Behavioural Perspective: Control Flow

The control flow describes the order of execution of activities. This is determined by the various dependency relations that may exist between activities.

_BMM_: In _BMM business rules_ can be used to determine _control flows_. For example, consider the business rule ‘The eye treatment should be provided only after the fee is received’. This business
rule requires a fee to be received before the eye treatment. Therefore, the activity that acquires the fee (‘Receive Fee’) needs to be executed before the activity ‘Provide Treatment’.

Figure 4.1 An excerpt from a process model for the Scientific Conference scenario

\( i^* \): The \( i^* \) language supports modelling of goal, resource, task and soft goal dependencies. The goal, resource and task dependencies can be used to derive causal relations among activities to a certain extent. The soft goals are used to select between alternative ways of achieving things (goals, tasks and resources).

- **Resource Dependency**: An actor depends on a resource provided by another actor. This resource could be used for realising a certain goal or a task. For example, a program committee depends on the resource ‘Submission’ provided by an author (see Figure 2.3). Then the submissions are assigned to reviewers for reviewing. This resource dependency can introduce a sequence flow between two activities that acquire the resource ‘Submission’ and use the resource ‘Assign Reviewer’ (see 5 in Figure 4.1).
Goal dependency: An actor depends on another actor for a certain state to be achieved. A program committee cannot organise a conference unless the steering committee funds the conference. That means the state ‘Conference Organised’ can be achieved only if the activity of providing funds is completed. This can lead to a sequence flow from the activity ‘Provide funds’ to the state ‘Conference Organised’, particularly to the activity that achieves this state.

Task dependency: An actor depends on another actor for a certain task to be performed. A program committee depends on a reviewer for the task of providing a review report to evaluate a submission. This may introduce a sequence flow from the activity that provides a review report to the activity that evaluates a submission.

KAOS: The KAOS operation model represents the behaviours of agents which are necessary to fulfil the requirements. The behaviours are expressed by means of operations performed by agents. An operation can create new objects, change object states and trigger other operations by means of events. KAOS maintains control flow through events (Objectiver, 2007).

In addition to this, KAOS goal dependency can be used to derive control flows in some way as discussed under i* goal dependency.

4.2.5 Behavioural Perspective: Data Flow

A data flow describes the flow of information resources between two atomic activities. Here, the main concern is to identify flow of data and associated data artifacts to activities.

BMM: The business rules in BMM can be used to determine data flows. For example, consider the business rule ‘An author must be provided an evidence document (Receipt) with the evaluation’. This rule requires the information resource ‘Receipt’ to be processed by certain activities, which could be a data flow (see © in Figure 4.1).
i*: The i* resource dependency can be used to identify data flow in a process model if the resource involved is an information resource. There will be a data flow between the two activities involved in the dependency that produce the resource and consume the resource.

KAOS: The KAOS operations have input links and output links. An operation takes input objects and produces output objects. KAOS maintains data flows through operation input and output relations in the operational model (the output of an operation becomes the input of another operation) (Objectiver, 2007).

We implement these mappings by developing the XSLT transformation programs. The program output II, III and IV in Appendix I shows the XML description of mappings of BMM, i* and KAOS to the process design framework for the Scientific Conference scenario.

4.3 Conclusion

Goal models and process models are two important models used for designing and structuring enterprise business. Goal models can be used to elicit the reasons behind the operational choices made in process models. Business process re-engineering is an expensive and time-consuming task. Therefore, it is important to design process models in such a way that they conform to the enterprise business goals and thereby justify the process evolution from a strategic perspective.

In this chapter we explored the gap between goal models and process models. We used BMM, i* and KAOS intentional modelling languages for this purpose. Then, by emphasising the different perspectives in the modelling of business processes, we set out some mappings between concepts in the two layers. The aim was to provide a comprehensive basis for relating the concepts of BMM, i* and KAOS goal models to the elements that constitute
business processes. To summarise, these mappings, as discussed in Section 4.2 and collected in Table 4.1, have revealed the following:

**Organisational perspective:** The strategic actor in $i^*$ and an agent in $KAOS$ were mapped to the participant element in the process design framework, thus covering the organisation perspective properly. $BMM$ focuses mainly on the enterprise as a whole when designing enterprise business plans and goals. Thus, the $BMM$ organisational unit maps to the participant element. In addition, involvement of participant actors can be extracted from the $BMM$ influencers.

**Functional perspective:** The goal languages explored support the functional view similarly. Each language, and its components, covers the functional aspect, in the form of means that realise the intentions; the means (named differently in different languages) are aligned with the activity element in business process models.

**Informational perspective:** The $i^*$ and $KAOS$ languages cover the information perspective to a certain extent; however, neither of the two languages distinguishes between the different types of resources. The $i^*$ resource maps to the resource element in the information perspective. The $i^*$ does not have a notion called event. The $KAOS$ entity and event type objects are mapped to the resource and event elements respectively in the process design framework. The resources in $BMM$ language can be captured from the influencers. The $BMM$ does not support the notion of event.

**Behavioural perspective:** The $BMM$ supports the control flow aspect in a certain way. The business rules in $BMM$ can be used to orchestrate, that is, to determine the constraints on the control flow of process activities. The business rules may be used in designing static process models as well as for modelling dynamic bindings of activities, which are executed by a rule engine. In $KAOS$ and $i^*$ languages the dependencies between participant actors can be mapped to sequence flows between involved activities, thus covering the control flows partially.

The $BMM$ business rules and $KAOS$ input and output relations of operations can be used to decide on data flows. The $i^*$ language
supports data flows partially that may be discovered from resource dependencies.

A clear benefit of these mappings is that they help to identify the relations between concepts in the three goal modelling languages to the process layer. Therefore, these mappings can be used in developing a method to design process models.
5 Exploring Business Model Support for Process Design

5.1 Introduction

The primary aim of business enterprises is to create economic values and exchange them with other collaborating partners and customers. Business models are used to capture ideas of business enterprises in terms of economic values. They express the values an enterprise offers to others and how these values are created and offered. More detailed business models show enterprises’ infrastructure management, financial aspects and marketing aspects. The REA, $e^3\text{value}$ model and BMO are such business modelling languages. The REA business ontology is proposed to model the change of values of accounting information systems in an enterprise. The $e^3\text{value}$ business model has been developed with the intention of modelling value networks of cooperating business partners. The BMO provides a way to model enterprise value propositions, infrastructure, customer interface and financial aspects. A process model shows the behaviours of actors, in particular the activities, roles, flow of resources and order of execution of the activities.

The design of process models is a complex and time consuming task as there are many decisions to be taken regarding the structuring and ordering of activities. As discussed in Chapter 1, process modelling techniques face a number of challenges that need to be addressed, namely manage process design complexity and provide traceability and flexibility. To address these challenges there is a need for structured approaches to process design that merge
with the enterprise IT infrastructure. Business models are used as a good starting point to construct process models (Gordijn, 2000). A process model based on a business model provides a business orientation, thereby enabling reasoning about operational choices made in process models in terms of business notions. The chapter answers the research question: How can business modelling be used to support the design of process models? Our goal is twofold: (1) Establish the relations between the concepts of the business models and process modelling notions (2) Construct a method to design a process model based on a business model. To meet this end, we choose $e^3$ value model as our business model. The $e^3$ value model is chosen as it provides a rich set of concepts for modelling value networks. We consider Curtis’s four perspective process design framework and map components in $e^3$ value model to these four perspectives. A set of guidelines is proposed for the construction of a partial process model based on an $e^3$ value model. The mappings help to identify the relations between $e^3$ value model concepts and business process notions. The guidelines help to construct a process model that conforms to the enterprise business model. A business model is not sufficient to construct a unique process model. Some additional knowledge is required to order the activities.

This chapter is arranged as follows. In Section 5.2, we discuss the notion of value object and value exchange in the $e^3$ value model. Section 5.3 discusses how business model concepts can be mapped to process model notions. Section 5.4 discusses a set of guidelines to identify process model elements from an $e^3$ value model. In Section 5.5, we discuss a set of process patterns for handling process flexibility. Section 5.6 terminates the chapter with conclusions.
5.2 Internal Structure of Value Object and Value Exchange

A value object is something that actors exchange and has an economic value. The obvious candidates for value objects are products, money and information. When a product is transferred from one actor to another, what is actually transferred is the ownership to the product (resource). The ownership of a resource is seen as a bundle of rights to the resource. When someone has an ownership of a resource, s/he can enjoy it according to its right. Thus, a value object is a certain right on a resource (Andersson, 2006a).

A value exchange involves a value object. Therefore, in a value exchange both the resource being transferred and the right on the resource must be specified. Furthermore, resources should be made available to enjoy the rights. For example, when someone buys a product through Amazon, the product should be delivered to a desired destination. This is known as transferring the custody (enabling) of the resource. When a buyer has custody of a resource, s/he can enjoy the rights on that resource. An actor has custody of a resource if s/he has immediate control of the resource. A shipper may have custody of the product, but s/he is not allowed to use the product for purposes other than shipping. A shipper may have the rights to transfer the product to a desired destination. Hence, providing custody of a resource is needed in a value exchange (Andersson, 2006a).

A value exchange may also include transfer of some documentary evidence to certify that a buyer has a certain right on the resource (Andersson, 2006a). An example is an e-voucher that certifies its owner has the right on the resource.

Thus, a value exchange is an aggregation of three components: rights on a resource, custody of a resource and an evidence document (Andersson, 2006a). The first component is compulsory, while the last two are optional. For example, in a simple
transaction, where there is instantaneous exchange of goods for money, the transfer of right is performed tacitly and no evidentiary document is provided. When buying a piece of land, the buyer is typically not given the custody of the resource.

5.3 From Business Model to Process Design Framework

In this section, we discuss how the $e^3value$ model concepts are mapped to Curtis’s process design framework. The details of process design framework are discussed in Section 2.5.3. The summary of the mappings are given in Table 5.1 and details are discussed in the following text.

5.3.1 Organisational Perspective: Participant

The main concern of the organisational perspective is the distribution of work activities among the actors.

Business Model: In an $e^3value$ model, actors or market segments are responsible for creating values and transfer of values. Therefore, actor and market segment are mapped to a participant element in the process design framework.

5.3.2 Functional Perspective: Activity

The main concern here is the identification of the activities that constitute a process model.

Business Model: From an $e^3value$ model we can discover five types of activities: value activity, value port, value offering, value exchange and value transaction. In an $e^3value$ model a value activity produces a value object. Value ports are used to provide or receive value objects. Some time value objects are offered or
requested in combination. This is done through value offering. A value offering is what an actor offers or requests to or from his/her environment. It is a set of equally directed value ports belonging to one actor. In a value exchange one actor transfers a value object to another actor. Thus, value activity, value port, value offering and value exchange map to the activity element in the business process framework. A value exchange is an aggregation of the three components: rights, custody and evidence document. At the activity level, we need one activity for sending (using out-port) and one for receiving (using in-port) each of these components. Further, a value transaction is a set of value exchanges. As value exchange is mapped to a number of activities, a value transaction is also mapped to a more complex activity in the process framework.

**Table 5.1** Summary of the mappings of $e^3$value model concepts to process design framework

<table>
<thead>
<tr>
<th>Process Design Framework Perspective</th>
<th>$e^3$value Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organisational (Participant)</td>
<td>Actor</td>
</tr>
<tr>
<td></td>
<td>Market Segment</td>
</tr>
<tr>
<td>Functional (Activity)</td>
<td>Value Activity, Value Port, Value Offering Value Exchange (Transfer Rights, Transfer Custody, Transfer Evidence Document) Value Transaction</td>
</tr>
<tr>
<td>Informational (Resource)</td>
<td>Rights of Resource Custody of Resource Evidence Document</td>
</tr>
<tr>
<td>Informational (Event)</td>
<td>Start Stimulus</td>
</tr>
<tr>
<td></td>
<td>End Stimulus</td>
</tr>
<tr>
<td>Behavioural (Control Flow)</td>
<td>None</td>
</tr>
<tr>
<td>Behavioural (Data Flow)</td>
<td>Value Port</td>
</tr>
<tr>
<td></td>
<td>Value Offering</td>
</tr>
<tr>
<td></td>
<td>Value Exchange</td>
</tr>
</tbody>
</table>
5.3.3 Informational Perspective: Informational Resource

The focus here is on the resources that are manipulated by an activity and events that trigger the activities. There are two types of resources: traditional and informational (see Section 2.5.3).

**Business Model:** In an $e^3$ value model, a value object is something that one actor gives to another actor that is regarded as valuable by at least one actor. According to the value object analysis in Section 5.2, a value object is a right on a certain resource. In addition, a value exchange involves a right on the resource, custody of the resource and some documentary evidence. The rights and evidence documents are informational in nature. The custody of a resource means having the control of a resource (traditional or informational). Thus, **rights**, **custody** and **evidence document** are mapped to the **resource** element in the information perspective.

A start stimulus represents an event (Gordijn, 2002) caused by an actor and triggers the exchange of value objects. An end stimulus is used to represent the model boundary and terminates a value exchange. Therefore, **start stimulus** and **end stimulus** map to the **event** element in the information perspective.

5.3.4 Behavioural Perspective: Control Flow

The control flow considers the order of execution of process activities. It is determined by the various dependency relations that may exist between activities.

**Business Model:** The $e^3$ value model does not model time dimension (Pijpers, 2007) or show any causal relations among its components. Therefore, we cannot derive any control flow information from it.
5.3.5 Behavioural Perspective: Data Flow

The data flows describe the flow of information resources within a process. Here, the main concern is to identify flow of data and associated data artifacts to activities.

**Business Model:** Rights and evidence documents are informational in nature and they can be realised as data flows in processes. The out-port and in-port involved in a value exchange can be used to decide the direction of the data flow.

These mappings are implemented by developing the XSLT transformation programs. The program output V in Appendix I shows the XML description of mappings of elements in the $e^3 value$ model to the process design framework for the Scientific Conference scenario.

5.4 Transition Guidelines

In this section, we formulate a set of design guidelines to construct a process model from an $e^3 value$ model using the mappings presented in the previous section. The method starts with an input $e^3 value$ model and produces a process model upon the application of guidelines. A business model does not show any causal relations among its value activities and value exchanges. From the process design perspective, we cannot employ an $e^3 value$ model to orchestrate flow of value exchanges in the form of activities. Thus, we cannot design a unique process model and only able to construct a partial process model without orchestrations. The purpose of these guidelines is merely to understand what we can absorb from business models in designing process models. The organisational, functional and informational perspectives of process design framework can be used for this purpose. The behavioural view can be used to a certain extent to identify data flows.
Since a business model involves several actors and a number of value creating and value exchanging activities, one way to present the final process model is to show it as the collaboration of smaller process models. This can be done by focusing on one actor in a business model and the dualities involving this actor. We define a base actor as the actor with respect to whom we view business collaborations. Thus, a smaller process model can be seen as activities carried out by a base actor related to one duality in the $e^3$value model. We will therefore use the BPMN pool element to represent the responsibilities of a single actor (base actor) in a particular duality. As we discussed in Section 2.5.4, a value transaction may consist of several phases. In this chapter, we focus on the execution phase where actors perform activities in order to fulfil their promises.

Let $D$ be a duality in an $e^3$value model and $P$ be a pool in the BPMN model. We formulate the following guidelines (G) to construct a partial process model from an $e^3$value model. These guidelines are summarized in Table 5.2.

For every duality $D$ and its base actor:

$G1$: Introduce a pool $P$ in the process model.

*Justification*: The pool $P$ can be used to show all the activities that are required to carry out the responsibilities in the corresponding duality, in terms of acquiring, producing and provisioning resources.

$G2$: For each value activity of the base actor introduces an optional sub-process in $P$ if this value activity is needed to produce a resource required by $D$.

*Justification*: An out-going value offering requires resource(s) in order to provide them to the environment. This sub-process is added to produce the resource(s) required for the out-going value offering.
G3: For each value offering in $D$ where the base actor is involved in provisioning (or receiving) of a resource(s) to (from) other actor (or market segment), introduce a sub-process in $P$ to provide (or receive) the resource(s).

Justification: An out-going value offering in a duality contains one or more value out-ports belonging to the same actor. An in-coming value offering in a duality contains one or more value in-ports belonging to the same actor. The purpose of this sub-process is to provide (or receive) the custody of the resource(s) involved in out-going (or in-coming) value offerings.

G4: For each value offering in $D$ where the base actor is involved in provisioning (or receiving) of an evidence document to (from) other actor (or market segment) introduce an optional sub-process in $P$ to provide (or receive) the evidence document.

G5: For each value offering in $D$ where the base actor is involved in provisioning (or receiving) of rights on a resource to (from) other actor (or market segment), introduce a sub-process in $P$ to provide (or receive) the rights on the resource.

Justification: The guidelines G4-G5 are added to provide (or receive) the evidence documents and rights corresponding to the resource involved in value offerings in G3.

Table 5.2 Summary of the guidelines to construct a process model from an $e^3$value model (with respect to a base actor and a one duality)

<table>
<thead>
<tr>
<th>$e^3$value Model</th>
<th>Process Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duality</td>
<td>Pool</td>
</tr>
<tr>
<td>Value Activity</td>
<td>Sub-process: To produce resource</td>
</tr>
<tr>
<td>Outgoing Value Offering</td>
<td>Sub-processes: To provide custody, evidence document and rights of the resource</td>
</tr>
<tr>
<td>Incoming Value Offering</td>
<td>Sub-processes: To receive custody, evidence document and rights of the resource</td>
</tr>
</tbody>
</table>
Method Application

Here we explain how elements in a process model can be constructed from an $e^3$value model. To illustrate the method, the guidelines are applied to the Scientific Conference scenario (see Figure 2.6 for the business model). An excerpt of the process model is shown in Figure 5.1. The BPMN process of the program committee (base actor) coordinates the activities related to the duality ‘Evaluation’ shown in the top pool. We also show the BPMN process of the author (base actor) coordinating the activities related to the duality ‘Submission’ in the bottom pool.

**G1**: A pool is added for each duality of the base actor. The base actor ‘Program Committee’ drives the process diagram given in the pool ‘Coordinate Evaluation’ (see ① in Figure 5.1). This pool depicts all activities related to the duality ‘Evaluation’.

**G2**: The sub-process ‘Decide on Acceptance’ is added to produce the resource ‘Evaluation’ as required by the duality evaluation (see ② in Figure 5.1). This sub-process corresponds to the value activity ‘Decide on Acceptance’ in the $e^3$value model. The purpose of this sub-process is to produce the resources required for value offering evaluation.

**G3**: The sub-process ‘Provide Evaluation’ is added to provide the resource ‘Evaluation’ to the ‘Author’ (see ③ in Figure 5.1). This corresponds to the value offering ‘Evaluation’ in the $e^3$value model.

**G4**: The sub-process ‘Provide an E-mail Notification’ is added to provide an evidence document for the value offering ‘Evaluation’. This email notification can be used as a proof for the resource ‘Evaluation’ (see ④ in Figure 5.1).

**G5**: Rights are not tangible, so the transfer of a right can be done communicatively. To do this, we can introduce sub-processes in the corresponding pools to provide and receive the rights. For illustration purposes, we model transfer of rights using BPMN message flows (data flows). The message flow from ‘Provide Submission’ to ‘Receive Submission’ is added to provide the rights on the submission to the program committee (see ⑤ in Figure 5.1).
The above guidelines are implemented by developing the XSLT transformation programs. The program output VI in Appendix I shows the XML description corresponding to the process model presented in Figure 5.1.

### 5.5 Process Patterns

In process models it is sometimes necessary to handle repeating activities. Such a repeating activity executes the same set of actions once or a specified number of times. To meet this requirement, in this section we discuss a set of process patterns that can be used to handle repeating actions.

A value exchange is an interaction between two actors. In a single exchange, an actor may interact with one or more actors (same type). As discussed previously a value exchange is an aggregation of rights on a resource, custody of a resource and an evidence document. We call each of these components a business object (BO). During a value exchange, a provider actor must offer...
these business objects to one or more recipient actor(s) who receive them. Thus, to model these repeating activities, we introduce a set of process patterns and they are summarised in Table 5.3.

To explain these patterns we use the following template.

**Template:** \(<\text{Action}, \text{Business Object (BO)}, \text{Cardinality}>\)

where,

- \(\text{Action} \in \{\text{Send/Provide, Receive, Produce}\}\)
- \(\text{Business Object} \in \{\text{Right, Custody, Evidence Document}\}\)
- \(\text{Cardinality} \in \{1, \ast\}\).

**Table 5.3 Primitive process patterns**

<table>
<thead>
<tr>
<th>No</th>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(&lt;\text{Provide, BO, 1}&gt;)</td>
<td>An actor provides a BO to another actor</td>
</tr>
<tr>
<td>2</td>
<td>(&lt;\text{Receive, BO, 1}&gt;)</td>
<td>An actor receives a BO from another actor</td>
</tr>
<tr>
<td>3</td>
<td>(&lt;\text{Provide, BO, \ast}&gt;)</td>
<td>An actor provides a BO to many other actors (same type)</td>
</tr>
<tr>
<td>4</td>
<td>(&lt;\text{Receive, BO, \ast}&gt;)</td>
<td>An actor receives a BO from many other actors (same type)</td>
</tr>
<tr>
<td>5</td>
<td>(&lt;\text{Produce, Resource, 1}&gt;)</td>
<td>An actor produce a resource</td>
</tr>
<tr>
<td>6</td>
<td>(&lt;\text{Produce, Resource, \ast}&gt;)</td>
<td>An actor produce a resource many times</td>
</tr>
</tbody>
</table>

These patterns can be used to handle repeating actions. In BPMN, such repeating actions can be modelled with cyclic loops. In Figure 5.1, ‘Receive Submission’ is a loop sub-process, which means the sub-process is used to collect custody of the submissions from one or many authors (see © in Figure 5.1).

In Section 6.4, we discuss how process flexibility can be achieved by using these process patterns and repeating actions.
5.6 Conclusion

Business models and process models are two important models used for structuring enterprise business activities. Business models are made in order to show who the actors are in a business scenario and explain their relations, which are formulated in terms of activities that create and offer values. Process models contain the required process knowledge, which can ultimately be realised by a set of automated activities and services. For process-oriented IT systems, it is important to design a process model that conform to a business model and thereby justify the process evolution from a business perspective.

In this chapter, we have investigated the gap between high-level business models and operative realisation in the form of process models. For this purpose, we have used a process modelling framework that encompasses all the major aspects of process design, i.e. organisational, informational, functional and behavioural. We have performed a transformation analysis for each of the process design aspects. The results of the analysis reveal that the $e^3$ value model supports organisational, functional and informational perspectives in a comprehensive way. In $e^3$ value model activities are modelled at different level of granularity (value ports, value offerings, value exchanges and value transaction). The $e^3$ value model cannot alone orchestrate process activities in terms of control flows. Some additional knowledge from the problem domain is required to orchestrate activities. The data flows can be deduced to a certain extent by using the direction of value exchanges (out-port to in-port). The proposed guidelines enable the obtaining and structuring of process model elements from business model elements. It is possible to trace the design decisions made in constructing the process model and motivate them using business notions, thereby providing traceability.
6 Goal and Value Driven Process Design

6.1 Introduction

Business enterprises use different models in deriving their information systems requirements. The goal models, business models and process models are parts of chains of models used for this purpose. Each model is used for a distinct purpose and describes different aspects of business. A goal model helps in clarifying interests, intentions and strategies of different stakeholders of the business. A business model captures a high level view of the business activities taking place within and between enterprises. It captures actors in business collaboration, economic values and how these values are created and offered. A process model deals with operational and procedural aspects by describing activities, roles, resources, control flows and data flows.

In this chapter, we discuss how goal models and business models may be used in a transition method for designing process models. The chapter provides answers to the research question: How can goal modelling and business modelling be used to support the design of process models? Our goal is to build a method to design a process model that conforms to the both enterprise goal model and a business model. The main benefit of the method is that it offers a structured way to design process models. At the same time it provides a way to manage traceability by allowing the tracing of the design decisions made in constructing the process model and also a way of motivating them using goal notions and business notions.
The method starts with an input goal model and a business model that are already aligned. The method uses an intermediary model to bridge the gap between two input models with an output process model. We chose KAOS goal modelling language as the basis for goal modelling. But any other goal modelling language that supports agent modelling, such as i* for example, could also have been employed. For the business modelling, we use $e^3 value$ model. The $e^3 value$ model is chosen as it is conceptually rich for modelling value networks.

The chapter is structured in the following way. In Section 6.2 we discuss the concepts involved in the activity dependency model that is the intermediary model. Section 6.3.1 discusses the transition steps required to go from a goal model and a business model to an activity dependency model. Section 6.3.2 presents the transition steps required to go from an activity dependency model to a process model. Section 6.4 presents how to offer process flexibility by using process patterns and repetitive actions. Finally, Section 6.5 wraps up the chapter with our conclusions.

### 6.2 Concepts and Notation for Activity Dependency Model

It is evident from the mappings presented in Chapter 4 and Chapter 5 that both goal models and business models provide useful information to construct process models. Business models provide useful knowledge about business activities. Thus, with respect to the process design framework, we can employ business models to extract organizational, informational and functional aspects and goal models to extract behavioural related information to a certain extent.

The purpose of an activity dependency model is to describe, on a high level, the activities needed to carry out the value exchanges specified in a business model and the casual relationships among
them. A business model and a goal model are combined together to form an activity dependency model. The activity dependency model performs the role of an intermediary model in constructing a process model. To construct a process model, an activity dependency model extracts knowledge about activities from a business model as well as knowledge needed to orchestrate these activities to a certain extent from a goal model. The missing knowledge that is further required to construct a process model is then added to the activity dependency model before constructing a process model.

An activity dependency model provides more detail than a business model by identifying, classifying and relating activities needed for executing and coordinating value exchanges. On the other hand, an activity dependency model is less detailed than a process model, as it abstracts away from control and message flow aspects. An activity dependency model is always constructed from a particular actor’s perspective, called the base actor. This means that an activity dependency model focuses on one actor in a business model and the dualities involving this actor. We define a base actor as the actor with respect to whom we view business collaborations.

Structurally, an activity dependency model can be seen as a graph with three kinds of nodes, representing activities, and four kinds of directed edges, representing relationships between activities. The three kinds of activities are:

- **Value Transfer Activity**: A value transfer activity transfers a resource from one actor to another.
- **Production Activity**: In a production activity, the base actor produces a resource required for a value transfer activity.
- **Coordination activity**: A coordination activity coordinates the value transfer activities within one duality as well as additional production activities.
The four kinds of relationships between activities are:

- **Duality Dependency**: A duality dependency from a coordination activity to a value transfer activity expresses that the latter is included in the duality of the former; recall that each coordination activity corresponds to one duality.

- **Flow Dependency**: A flow dependency (Malone, 1999) from one activity to another expresses that the resource obtained by the first activity is needed as input to the second activity. As an example, a program committee has to obtain a submission from an author before delivering it to a reviewer.

- **Trust Dependency**: A trust dependency (Jayaweera, 2004) between two value transfer activities within the same duality expresses that the first activity has to be carried out before the second one as a consequence of low trust between the involved actors. Informally, a trust dependency states that one actor wants the other actor do her work before doing his own work. An example could be a program committee requiring a payment from a steering committee before delivering a conference program.

- **Trigger Dependency**: A trigger dependency from a coordination activity to a production activity expresses that the production activity is to be initiated and managed by the coordination activity.

The components of an activity dependency model have a clear business motivation, i.e. they can be explained and motivated in business terms. At the same time, an activity dependency model also provides some motivation for some operational choices (flow and trust dependencies) made in process models. This makes the activity dependency model a useful instrument for constructing a more detailed process model.

An example of an activity dependency model to coordinate submission is shown in Figure 6.1, which is based on the Scientific Conference scenario. The base actor is the program committee. The diagram shows two columns of coordination and value transfer
activities corresponding to the two dualities, submission and reviewing in the e³ value model. There are two production activities ‘Assign Reviewer’ (for assigning a reviewer to a paper) and ‘Decide on Acceptance’ (for deciding whether to select a paper), and a number of flow and trigger dependencies. For example, there is a flow dependency from ‘Assign Reviewer’ to ‘Receive Review’ meaning that an assignment of a reviewer to a paper must exist before a review of that paper can be obtained. The trigger dependency from ‘Coordinate Submission’ to ‘Decide on Acceptance’ means the latter is initiated and managed by the former. Furthermore, cardinalities (as in UML) have been attached to some of the dependencies. For example, the star on the flow dependency from ‘Receive Submission’ to ‘Assign Reviewer’ means that for each submitted paper, several assignments of reviewers may exist.

![Diagram](image-url)

**Figure 6.1** An activity dependency model to coordinate submissions in the Scientific Conference scenario
6.3 Model Transition

This section is dedicated to introducing the transition steps. In Section 6.3.1 we discuss how to construct an activity dependency model from a goal model and a business model. Section 6.3.2 discusses the transition from an activity dependency model to a process model.

6.3.1 Goal and Business Model to Activity Dependency Model

An activity dependency model can be partially derived from the KAOS models and a business model. In Section 6.3.1.1, we discuss some design guidelines that can be used for this purpose. The guidelines are applied to the Scientific Conference scenario and Massively Multiplayer Online Game playing scenario in Section 6.3.1.2.

6.3.1.1 Guidelines to Design an Activity Dependency Model

Let BM denote an $e^3$ value model and AM denote an activity dependency model. These guidelines are summarized in Table 6.1.

$G1$: For each duality $D$ of the base actor in a BM there is one coordination activity in an AM.

*Justification:* A value interface in an $e^3$ value model consists of in-port(s) and out-port(s) that are belonging to the same actor. Value interfaces are used to model economic reciprocity (duality). This means an actor is willing to offer a value object to another actor if s/he receives adequate compensation in return. We introduce a coordination activity to coordinate activities in one duality.

$G2$: For each value offering of the base actor in a BM there is one value transfer activity in an AM to provide or receive the resource(s).
**Justification:** A value offering models what an actor offers to, or requests from, his/her environment. A value offering is a set of equally directed value ports belonging to one actor. Either all ports or no ports should exchange value objects in a value offering. Therefore, there will be a value transfer activity in an AM to provide or receive resource(s) involved in a value offering, to (from) environment.

**G3:** For each value offering in a duality of the base actor introduces a duality dependency in AM from the corresponding coordination activity to the corresponding value transfer activity.

**Justification:** The incoming and outgoing value offerings in one value interface (duality) are related to the corresponding coordination activity and value transfer activities by a duality dependency.

**G4:** For each value activity of the base actor introduces an optional production activity in AM if this value activity is needed to produce resource required by an outgoing value offering.

**Justification:** A value interface offers value object(s) to its environment. These value objects need to be produced. Thus, for each duality we need to add activities to produce resource(s) required by a value transfer activity.

The components of the activity dependency model discussed above are directly derived from a business model. But flow, trust and trigger dependencies cannot be obtained from a business model. Some of the flow dependencies can be obtained from the KAOS agent dependency model.

The task of designing process models cannot be fully automated. Model designers need to intervene at some point. Thereby, the designer has two options:

1. Use the KAOS models to obtain dependencies as much as possible and use domain knowledge for the rest. The use of KAOS models to uncover such dependencies provides an
opportunity to motivate ‘why’ certain operational choices are made in process models from a strategic perspective.

(2) Use the domain knowledge completely to discover flow, trust and trigger dependencies. In this case, the designer draws these dependencies in the AM using the domain knowledge.

**Goal Model to Discover Dependencies:**

We can use the *KAOS* agent dependency model to find flow dependencies to a certain extent. An example of a *KAOS* agent dependency model can be found in Figure 3.1. Let DM be a *KAOS* agent dependency model.

G5: For each object dependency in DM where the base actor depends on an entity provided by another actor, draw a flow dependency from the activity that acquires this entity to the activity (let’s say ‘Produce’) that uses this entity as an input, if ‘Produce’ is included in the AM.

*Justification:* Input resources that are necessary for producing other resources need to be collected from other actors before they are actually processed. This idea motivates us in formulating this guideline.

**Table 6.1** Summary of the guidelines to construct an activity dependency model from an *e³value* model (with respect to a base actor)

<table>
<thead>
<tr>
<th><em>e³value Model</em></th>
<th>Process Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duality</td>
<td>Coordination Activity</td>
</tr>
<tr>
<td>Value Offering</td>
<td>Value Transfer Activity</td>
</tr>
<tr>
<td>Value Offering</td>
<td>Duality dependency from a Coordination Activity to a Value Transfer Activity</td>
</tr>
<tr>
<td>Value Activity</td>
<td>Production Activity</td>
</tr>
<tr>
<td>Object Dependency</td>
<td>Flow dependency</td>
</tr>
</tbody>
</table>
G6: For each object dependency in DM where the base actor is responsible for providing an entity to another actor, draw a flow dependency from the activity (let’s say ‘Produce’) that produces this resource to the activity that provides this resource, if ‘Produce’ is included in the AM.

Justification: Output resources are first necessary to be produced before being delivered to the environment. This inspired us in forming this guideline.

6.3.1.2 Method Application to Create an Activity Dependency Model

In this section, we explain how an activity dependency model can be constructed from KAOS models and an e3value model by applying the guidelines to the two scenarios Scientific Conference and Massively Multiplayer Online Game Playing.

6.3.1.2.1 Method Application: Scientific Conference Scenario

The input models used are KAOS goal model, an agent dependency model and an e3value model. An activity dependency model to coordinate conference program is shown in Figure 6.2. The base actor is the program committee. An excerpt from a KAOS goal model can be found in Figure 3.2. The agent dependency model and e3value model corresponding to the Scientific Conference scenario can be found in Figures 3.1 and 2.6 respectively.

G1: The base actor has three dualities (value interfaces) to handle submissions, reviews and the conference program. Thus, there are three coordinating activities in the AM to coordinate activities related to these dualities (see ⊗ in Figure 6.2).

G2: The value offerings become value transfer activities in the AM. The base actor provides or receives six value objects to (from) the environment (e.g. provide ‘Evaluation’). Each of
these value offerings becomes value transfers in the AM (see ② in Figure 6.2).

**G3:** The value interfaces are used to bundle reciprocal value offerings. For example, the base actor receives submissions from authors and in return gives evaluations. The value offerings become value transfers in the AM. Thus, value transfer activities corresponding to the value offerings submission and evaluation can be related to the coordination activity ‘Coordinate Submission’ through the duality dependencies (see ③ in Figure 6.2).

**G4:** The two value activities ‘Decide on Acceptance’ and ‘Assign Reviewer’ are mapped to two production activities as they are required to produce the resource ‘Evaluation’ (see ④ in Figure 6.2).

To discover some flow dependencies a designer can use *KAOS* models.

**G5:** In Figure 3.1, the ‘Program Committee’ depends on the ‘Author’ for the entity ‘Submission’. In the goal model (Figure 3.2) we see ‘Submission’ is an input for the operation ‘Assign Reviewer’. Thus, we draw a flow dependency from the activity (‘Receive Submission’) that acquires the entity ‘Submission’ for the activity (‘Assign Reviewer’) that uses the entity ‘Submission’ (see ⑤ in Figure 6.2).

**G6:** In Figure 3.1, the ‘Author’ depends on the ‘Program Committee’ for the entity ‘Evaluation’. In the goal model (Figure 3.2) we see ‘Evaluation’ is an output of the operation ‘Decide on Acceptance’. Thus, we draw a flow dependency from the activity (‘Decide on Acceptance’) that produces the entity ‘Evaluation’ to the activity (‘Provide Evaluation’) that offers the entity ‘Evaluation’ (see ⑥ in Figure 6.2).
Figure 6.2 An activity dependency model to coordinate conference program in the Scientific Conference scenario

A designer can use the domain knowledge to find other flow dependencies as well as trust and trigger dependencies. The flow dependency from ‘Assign Reviewer’ to ‘Receive Review’ is drawn from the knowledge; a review report can be obtained only after assigning a reviewer. The program committee is waiting for the payment to be received from steering committee in order to provide the conference program. This introduces a trust dependency from ‘Receive Payment’ to ‘Provide Conference Program’. The trigger dependency from ‘Coordinate Conference Program’ to ‘Assign Reviewer’ means the latter is initiated and managed by the former.
We implement these guidelines by developing the XSLT transformation programs. The program output VII in Appendix I gives the XML description corresponding to the activity dependency model presented in Figure 6.2.

### 6.3.1.2.2 Method Application: Massively Multiplayer Online Game Playing Scenario

In this section, we apply the guidelines to the Massively Multiplayer Online Game Playing Scenario. We use the KAOS goal model (Figure 6.3), the agent dependency model (Figure 6.4) and the e³value model (Figure 6.5) as inputs. An activity dependency model drawn to coordinate game services is shown in Figure 6.6. The base actor is the game provider.

**Figure 6.3** An excerpt from a KAOS goal model for the Massively Multiplayer Online Game Playing scenario
Figure 6.4 An agent dependency model for the Massively Multiplayer Online Game Playing scenario

Figure 6.5 An $e^3$ value model for the Massively Multiplayer Online Game Playing scenario
G1: The base actor has three dualities for game access, hosting service and shipping service. Therefore, there are three coordinating activities in the AM to coordinate activities related to these dualities (see ① in Figure 6.6).

G2: The value offerings become value transfer activities in the AM. The base actor provides three value objects (e.g. provide ‘Game Access’) and receives three value objects (e.g. receive ‘Internet and Hosting Service’) to (from) the environment. Each of them becomes a value transfer in the AM (see ② in Figure 6.6).

G3: The value interfaces bundle reciprocal value offerings. For example, a game provider provides game access to customers and gets fee in return. The value offerings become value transfers in an AM. Thus, value transfer activities corresponding to the value offerings game access and fee can be related to the coordination activity ‘Coordinate Game Service’ through duality dependencies (see ③ in Figure 6.6).

G4: The value activity ‘Create Games’ is added to develop games required for the value offering game access (see ④ in Figure 6.6).

To discover some flow dependencies a designer can use KAOS models.

G5: In Figure 6.4 the ‘Game Provider’ depends on the ‘Internet Access Provider’ for the entity ‘Internet and Hosting Service’. In the goal model (Figure 6.3), we see ‘Internet and Hosting Service’ is an input for the operation ‘Create Games’. Therefore, we draw a flow dependency from the activity (‘Receive Internet and Hosting Service’) that receives the entity ‘Internet and Hosting Service’ to the activity (‘Create Games’) that uses the entity ‘Internet and Hosting Service’ (see ⑤ in Figure 6.6).

G6: In Figure 6.4, the ‘Customer’ depends on the ‘Game Provider’ for the entity ‘Game Access’. In the goal model (Figure 6.3) we see ‘Game Access’ is an output of the operation ‘Create
Games’. Thus, we draw a flow dependency from the activity (‘Create Games’) that produces the entity ‘Game Access’ to the activity (‘Provide Game Access’) that offers the entity ‘Game Access’ (see 6 in Figure 6.6).

The flow dependency from ‘Provide Game Access’ to ‘Receive Shipping Service’ is drawn from the domain knowledge that game CDs can be delivered only after providing game access. The game provider waits for the payment to be received from the customer in order to provide the game access. This introduces a trust dependency from ‘Receive Game Fee’ to ‘Provide Game Access’. The trigger dependency from ‘Coordinate Game Service’ to ‘Create Games’ means the latter is initiated and managed by the former.

Figure 6.6 An activity dependency model to coordinate game access in the Massively Multiplayer Online Game Playing scenario
6.3.2 Activity Dependency Model to Process Model

In this section, we discuss how a process model can be derived from an activity dependency model. In Section 6.3.2.1, we discuss some design guidelines that can be used for this purpose. In Section 6.3.2.2 the guidelines are applied to the two scenarios Scientific Conference and Massively Multiplayer Online Game Playing.

6.3.2.1 Guidelines to Design a Process Model

Moving from an activity dependency model to a process model is essentially about specifying the detailed control flows between activities and data flows (message exchanges in BPMN) between involved actors. The starting point is to let each coordination activity in the activity dependency model become a process defined within one pool. This means each pool models the view of the base actor in exchanging resources with another actor and production activities needed for this exchange. Such a pool contains a series of activities, gateways and events which are connected by sequence flows. The entire process model will consist of number of such processes with pools that communicate with each other.

A single pool essentially describes a binary collaboration between two partners. Such collaboration may consist of several phases as discussed in Section 2.5.4, including planning, identification, negotiation, actualisation and post-actualisation. In this chapter, we consider only the two phases, negotiation and execution. In the negotiation phase commitments for resource exchanges are proposed and accepted. The resource transfers occur and are acknowledged in the execution phase.

To present the guidelines, we relate components of the process design framework to the Business Process Modelling Notation (BPMN). A participant actor in the framework (organizational aspect) is mapped to a pool in the BPMN notation, while an activity (functional aspect) is mapped to a sub-process. The control
flows and data flows (behavioural aspect) are named as sequence flows and message flows respectively. The informational aspect can relate to BPMN artifacts.

Let AM denotes an activity dependency model and C be a coordination activity in AM. The AM is drawn with respect to C and would contain required information to coordinate the activities related to a particular duality D in the $e^3$ value model. Also, let PM denotes a process model and $P_C$ be pool in PM that corresponds to the coordination activity C. An activity A in C can be identified by C.A. We denote value transfer activities coordinated by C, where a base actor is responsible for receiving and providing resources from or to other actor by $In_i$ and $Out_j$ respectively.

For coordination activity C in AM:

$G1$: Introduce a pool $P_C$ in PM.

*Justification*: A single pool models the view of the base actor in exchanging resources with another actor in a single collaboration. Thus, a pool will model all the related activities required for these exchanges.

$G2$: Introduce an optional *negotiation* sub-process in $P_C$.

*Justification*: Typically, business collaboration contains a negotiation phase but not in all cases. This sub-process is introduced to negotiate commitments for resource exchanging.

$G3$: For value transfer activity $In_i$, introduce a sub-process $In_i$ in $P_C$.

$G4$: For value transfer activity $Out_j$, introduce a sub-process $Out_j$ in $P_C$.

*Justification*: The purpose of each of these sub-processes is to transfer the custody of the resource(s) involved in a value transfer.

$G5$: For each flow dependency from an activity $D.In_i$ (where $D$ is a coordination activity $\neq C$) to a value transfer activity $Out_j$, there is one sub-process $get(D.In_i)$ in $P_C$. The sub-process
repeats if the flow dependency is not injective (not one-to-one).

*Justification:* The sub-process $get(D.In_i)$ is added to acquire the resource(s) required for a value transfer activity, which are not internally produced. The resources obtained can be directly provided by the outgoing value transfer activity. The sub-process needs to be repeated if multiple actors need to be handled.

**G6:** For each trigger dependency from $C$ to a production activity $Production$, there is one sub-process $do(Production)$. The sub-process repeats:

(i) if there is a multi-valued flow dependency from a value transfer activity $D.In_i (D \neq C)$ to $Production$ and $get(D.In_i)$ is included in $P_C$

(ii) if there is a multi-valued flow dependency from $Production$ to a value transfer activity $D.Out_j (D \neq C)$ and $get(D. Out_j)$ is included in $P_C$

(iii) if there is a multi-valued flow dependency from $Production$ to $C.Out_j$.

*Justification:* The sub-process $Production$ is added to produce the resource(s) required for a value transfer. The sub-process needs to be repeated if multiple actors need to be handled.

**G7:** Let $Act$ is a value transfer activity and $Production$ is a production activity. For each flow dependency,

(i) from $Act$ to $Production$, or

(ii) from $Production$ to $Act$

such that the corresponding sub-process $do(Production)$ is included in $P_C$, there is one sub-process $do(Act)$ in $P_C$. The sub-process repeats if $do(Production)$ is repeating or if the flow dependency is not injective (not one-to-one).

*Justification:* The sub-process $do(Act)$ is introduced in order to acquire the resources needed for a production activity or provide
resources produced by a production activity. The sub-process needs to be repeated if multiple actors need to be handled.

**G8**: There is a sequence flow between two sub-processes if there is a flow dependency or a trust dependency between the corresponding activities in AM.

Now it is necessary to combine the pools with message flows. Two pools need to be connected if one pool requires a resource provided by the other.

**G9**: Add a *message flow* from the resource providing sub-process in the providing pool to the resource receiving sub-process in the requesting pool to transfer the rights and the (optional) evidence document.

*Justification*: A value exchange contains transfer of rights as well as an optional evidence document. These actions are communication actions. Thus, rights and evidence documents can be transferred using message exchanges. We can represent both components in a one message flow.

When sub-processes are constructed and linked together using these guidelines we will end up with chunks of sub-processes leaving ends on the left and right (optional) side open. There will be a number of leftmost sub-processes \( L_i \) with no incoming sequence flows and rightmost sub-processes \( L_j \) (optional) with no outgoing sequence flows. Add a gateway \( G \) to pool \( P_C \). If there is a *negotiation* sub-process, connects the *negotiation* sub-process to the gateway \( G \) by a sequence flow. Add sequence flows from the gateway \( G \) to each \( L_i \). If there are rightmost sub-processes with no outgoing sequence flows add another gateway to merge the paths \( L_j \). Connect each \( L_j \) to the gateway \( G \) by a sequence flow. Finally attach *start* and *end* events to complete the process diagram \( P_C \). These guidelines are summarized and presented in Table 6.2.
<table>
<thead>
<tr>
<th>Activity Dependency Model</th>
<th>Process Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordination Activity (C)</td>
<td>Pool ($P_C$)</td>
</tr>
<tr>
<td>Value Transfer Activity $In_i$ in C</td>
<td>Sub-process($In_i$)</td>
</tr>
<tr>
<td>Value Transfer Activity $Out_j$ in C</td>
<td>Sub-process($Out_j$)</td>
</tr>
<tr>
<td>Production Activity ($Production$) coordinated by C</td>
<td>Sub-process($Production$), sub-process repeats if the flow dependency is multi-valued</td>
</tr>
<tr>
<td>Flow Dependency from $D.In_i$ ($D \neq C$) to $Out_j$</td>
<td>Sub-process($D.In_i$), sub-process repeats if the flow dependency is multi-valued</td>
</tr>
<tr>
<td>Flow Dependency from $D.In_i$ ($D \neq C$) to $Production$</td>
<td>Sub-process($D.In_i$), sub-process repeats if the flow dependency is multi-valued or Sub-process($Production$) is repeating</td>
</tr>
<tr>
<td>Flow Dependency from $Production$ to $D.Out_j$ ($D \neq C$)</td>
<td>Sub-process ($D.Out_j$), sub-process repeats if the flow dependency is multi-valued or Sub-process ($Production$) is repeating</td>
</tr>
<tr>
<td>Flow Dependency, Trust Dependency</td>
<td>Sequence Flows between corresponding Sub-processes</td>
</tr>
</tbody>
</table>
6.3.2.2 Method Application to Create a Process Model

In this section, we explain how a process model can be constructed from an activity dependency model by applying the guidelines to the Scientific Conference and Massively Multiplayer Online Game Playing scenarios.

6.3.2.2.1 Method Application: Scientific Conference Scenario

A process model constructed to coordinate the conference program in Scientific Conference scenario using BPMN notation is shown in Figure 6.7. The input model used is the activity dependency model given in Figure 6.2. The base actor is the program committee. There are three coordination activities: ‘Coordinate Submission’, ‘Coordinate Review’ and ‘Coordinate Conference Program’. We apply the guidelines for the coordination activity ‘Coordinate Conference Program’.

\[ G1 \]: Add a BPMN pool to represent all the activities needed to coordinate conference program (see ① in Figure 6.7).

\[ G2 \]: The negotiation sub-process ‘Negotiate Conference’ is to negotiate commitments for exchanging the two resources conference program and payment (see ② in Figure 6.7).

\[ G3 \]: Add a sub-process (‘Receive Payment’) to receive the resource payment (see ③ in Figure 6.7).

\[ G4 \]: Add a sub-process (‘Provide Conference Program’) to provide the resource conference program (see ④ in Figure 6.7).

\[ G5 \]: The guideline 5 is applied when it is required that resource(s) which are not internally produced be acquired from another actor and be transferred directly by an outgoing value transfer activity.

\[ G6 \]: The two sub-processes ‘Assign Reviewer’ and ‘Decide on Acceptance’ are added due to the two trigger dependencies
for the corresponding production activities. They are required to produce resources needed for the provision of the conference program (see 5 in Figure 6.7). These subprocesses are repeating as the corresponding flow dependencies are multi-valued.

\textbf{Figure 6.7} A process model to coordinate conference program in the Scientific Conference scenario

\textit{G7}: The sub-process ‘Receive Review Report’ is introduced to acquire the resources needed to produce the resource ‘Evaluation’. This sub-process is repeated since there could be multiple review reports per submission (see 6 in Figure 6.7).

The sub-process ‘Provide Evaluation’ is added to provide the resource ‘Evaluation’. This sub-process is repeated as there are multiple authors to be handled (see 7 in Figure 6.7).

\textit{G8}: There is a sequence flow from the sub-process ‘Receive Payment’ to the sub-process ‘Provide Conference Program’. This is due to the trust dependency between the corresponding activities in the AM (see 8 in Figure 6.7).

There is a sequence flow from the sub-process ‘Receive Submission’ to the sub-process ‘Assign Reviewer’. This is
due to the flow dependency between the corresponding activities in the AM (see ◊ in Figure 6.7).

Figure 6.8 A process model to coordinate submissions, reviews and conference program in the Scientific Conference scenario

For illustration purposes, we also add the process diagrams of author and reviewer coordinating collaborations submission and reviewing to the process diagram shown in Figure 6.7. The new process diagram is shown in Figure 6.8. The process of the program committee (base actor) coordinating the collaboration conference program is shown in the middle pool. The process of the reviewer (base actor) coordinating the collaboration reviewing is shown in the top pool. The process in the bottom belongs to the author (base actor) who coordinates the collaboration submission.

G9: The process diagram shown in Figure 6.8 shows numbers of message flows between pools. The message flow from the
sub-process ‘Provide Evaluation’ to the sub-process ‘Receive Evaluation’ is to transfer the rights to the evaluation report and, if necessary, an evidence document can also be included (see ® in Figure 6.8).

We develop XSLT transformation programs to implement these guidelines. The program output VIII in Appendix I shows the XML description of the process model presented in Figure 6.7.

**6.3.2.2 Method Application: Massively Multiplayer Online Game Playing Scenario**

A BPMN process model drawn for the Massively Multiplayer Online Game Playing scenario is shown Figure 6.9. There are four coordination processes to coordinate ‘Game Service’, ‘Internet and Hosting Service’, ‘Payment’ and ‘Shipping Service’. The top pool shows the customer (base actor) coordinating the collaboration ‘Payment’. The second pool from top shows the process of the Internet service provider (base actor) coordinating the collaboration Internet service provisioning. The game provider (base actor) coordinating the collaboration game access is shown in the third pool. Finally, the process at the bottom belongs to the shipper (base actor) who coordinates the collaboration shipping service.

For illustration purposes, we apply the guidelines for the coordination activity ‘Coordinate Game Access’ coordinated by the game provider. The activity dependency model for this coordination process is given in Figure 6.6.

**G1:** A BPMN pool is added to show all the activities required to coordinate game services (see Î in Figure 6.9).

**G2:** The sub-process ‘Negotiate Game Access’ is added to negotiate commitments for resource exchange game access and game fee (see © in Figure 6.9).
G3: The sub-process (‘Receive Game Fee’) is added to receive the resource game fee (see ③ in Figure 6.9).

G4: Add a sub-process (‘Provide Game Access’) to offer the resource game access (see ④ in Figure 6.9).

G6: The sub-process ‘Create Games’ is added as there is a trigger dependency for the corresponding production activity. The purpose here is to produce the resources needed to provide the game access (see ⑤ in Figure 6.9).

**Figure 6.9** A process model for the Massively Multiplayer Online Game Playing scenario

G7: The sub-process ‘Receive Internet and Hosting Service’ is introduced to acquire the resources needed to produce the resource ‘Game Access’ (see ⑥ in Figure 6.9).
G8: The two sub-processes ‘Receive Game Fee’ and ‘Provide Game Access’ are connected by a sequence flow as there is a trust dependency between the corresponding value transfer activities in the AM (see ⑦ in Figure 6.9).

G9: The message flow from the sub-process ‘Provide Shipping Fee’ to the sub-process ‘Receive Shipping Fee’ is to transfer the rights of the payment to the shipping service provider (see ⑧ in Figure 6.9).

6.4 Processing Repetitive Actions

If a process model PM conforms to an activity dependency model AM and PM’ is derived from PM as specified below, then PM’ conforms to AM. PM should contain two repeating sub-processes S and T joined by a sequence flow. PM’ is derived from PM by replacing the repeating sub-processes and their joining sequence flow by one repeating sub-process containing S and T joined by a sequence flow.

Figure 6.10 An alternative process model for the Scientific Conference scenario

The reason for this is to allow for more flexibility in the process design. Without this flexibility, there would be an unwanted restriction, namely on how repeating activities are treated. It would
be assumed that if there are two subsequent repeating activities, then there must be a repetition over the first activity followed by a repetition over the second. For example, in Figure 6.7 all submissions have to be received before any assignment is made. However, this assumption is too restrictive in many cases, an alternative is to have a single repetition over a sequence of the two activities. For example, in Figure 6.10 a received submission may be directly followed by an assignment.

By using the primitive process patterns discussed in Section 5.5, we can form more compound process patterns. The Table 6.3 shows three example compound process patterns that are constructed by combining primitive process patterns. Here, we have only considered the business object custody of resource. For example, the bigger sub-process in Figure 6.10 can be constructed using the compound process pattern 3 in Table 6.3. Here the resources involved are ‘Submission’ and ‘Reviewer’.

Table 6.3 Examples of compound process patterns constructed by combining primitive process patterns

<table>
<thead>
<tr>
<th>Primitive Process Patterns</th>
<th>Compound Process Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Receive Resource 1</strong> → <strong>Provide Resource 2</strong></td>
<td><strong>Receive Resource 1</strong> → <strong>Provide Resource 2</strong></td>
</tr>
<tr>
<td><strong>Provide Resource 1</strong> → <strong>Receive Resource 2</strong></td>
<td><strong>Provide Resource 1</strong> → <strong>Receive Resource 2</strong></td>
</tr>
<tr>
<td><strong>Receive Resource 1</strong> → <strong>Produce Resource 2</strong></td>
<td><strong>Receive Resource 1</strong> → <strong>Produce Resource 2</strong></td>
</tr>
</tbody>
</table>
6.5 Conclusion

Goal models, business models and process models are used for structuring enterprise business. Goal models are used to model intentions and strategies of different stakeholders. Goal models enable reasoning about new customer needs and business opportunities, thereby providing a motivation for operational choices. Business models show the actors, economic values, how these economic values are created and offered. For process-oriented IT systems it is important to design process models in such a way that they conform to the long term enterprise goals and desired economic values and, thereby justify the process evolution from a strategic and a business viewpoint.

In this chapter, we have introduced the notion of an activity dependency model. According to the mappings provided in Chapter 5, it is revealed that business models cover organization as well as functional and informational perspectives of process design framework. The study of goal modelling languages constructs in Chapter 4 shows that goal languages (BMM, i* and KAOS) can be used as a basis for retrieving behavioural information to a certain extent. We use KAOS for goal modelling and the e3value model for business modelling. The purpose of an activity dependency model is to bridge the gap between goal, business and process models. An activity dependency model identifies, classifies and relates activities needed for executing and coordinating value transfers specified in a business model. The relations between activities are captured through activity dependencies. We have discussed four such dependencies; namely flow, trust, trigger and duality dependencies. Some flow dependencies can be derived from KAOS models. The duality dependencies can be derived from e3value model. The guidelines help to construct an activity dependency model and a process model from their input counter parts.
We believe that this approach effectively addresses the challenges introduced in Chapter 1.1.

Manage Design Complexity: The overall approach provides a structured way to design process models and, thereby reduce the complexity of designing process models.

Provide Traceability: A process model is based on an activity dependency model, which is in turn based on a goal model and a business model. This means that components in a process model can be motivated by back tracing to the strategic intentions and business oriented notions.

Offer Flexibility: The transition from goal model and business model to activity dependency model to process model gives the main structure of a process. However, the approach allows for flexibility by letting sub-processes be based on patterns. The atomic process patterns discussed in Chapter 5.5 can be used to make more compound patterns as explained in Section 6.4. The use of open-EDI phases as a fundamental process pattern provides a way to use other process patterns, for example negotiation process patterns as discussed in Jayaweera (2004). This means that the lower-level details of a process model can be tailored to the situation at hand by selecting appropriate patterns from a repository.
7 Implementation Details and Comparison to Similar Research

This chapter is devoted to introducing implementation details and to comparing our work with similar research. In section 7.1, we introduce the Protégé knowledge-base management tool. Here, we discuss the basic Protégé concepts required for the implementation. In section 7.2, we introduce the Extensible Stylesheet Language Transformations (XSLT). We use XSLT to implement the transition guidelines and mappings. Section 7.3 compares our work to other similar research. The chapter ends with a conclusion in Section 7.4.

7.1 Protégé and OWL Ontologies

Protégé is an integrated software tool used by knowledge engineers and domain experts to develop knowledge-based systems (OWL, 2004). It provides a graphical and interactive ontology design and knowledge based development environment. Applications developed with Protégé can be used in problem solving and decision making in particular domain of interest. The knowledge acquisition tool allows problem domain knowledge to be entered. The resulting knowledge base can be used with a problem solving method to answer the questions. Protégé domain ontologies and problem solving methods can be reused. This can reduce the development and program maintenance time. Protégé uses component-based architecture. This enables system developers to add new functionality by creating suitable plug-ins. We use the Protégé-OWL plug-in for creating OWL ontologies.
OWL Ontologies

Web Ontology Language (OWL) ontologies are used to capture knowledge about some domain of interest. We implement the BMM, i*, KAOS, e³value model and activity dependency model using the Protégé-OWL plug-in. A part of the OWL code and a screenshot of the activity dependency model corresponding to a Scientific Conference scenario are shown in Figures 7.1 and 7.2 respectively.

Basic Elements in OWL

Here, we describe some basic OWL concepts used during the implementation stage. OWL ontology consists of three basic elements called Classes, Individuals and Properties.

Class: A Class describes some concept in the domain (OWL, 2004). For example, Coordination_Activity, Value_Transfer_Activity and Production_Activity describe some concepts (activities) in our selected domain (see ① in Figure 7.1). It is possible to define sub-classes within a class to further classify the base class.

Individual: An Individual is an object in the domain of interest and is a specific instance of a class (OWL, 2004). ‘Conference_Program’ is an instance of the class Coordination_Activity (see ② in Figure 7.1).

Properties: The Properties are binary relations between individuals (OWL, 2004). That means properties link two individuals together. There are three main types of properties: object properties, datatype properties and annotation properties.

- An object property links an individual to an individual. For example, Duality_Dependency object property relates an instance of a Coordination_Activity class to an instance of a Value_Transfer_Activity class (see ③ in Figure 7.1).
• A *datatype property* links an individual to an XML schema datatype value or an RDF literal.

• An *annotation property* can be used to add information to classes, object properties, datatype properties and individuals.

A property may have a domain and a range. A property links the individuals from a domain to the individuals from a range. For example, the property *Trigger_Dependency* links the individuals belonging to the class *Coordination_Activity* to the individuals belonging to the class *Production_Activity* (see ④ in Figure 7.1).

The Object and Datatype properties can be further restricted using the complex property characteristics: *Functional, Inverse Functional, Transitive and Symmetric*.

• A *Functional* property relates a given individual to at most one individual via the property. For example, the property *Trust_Dependency* is functional as it relates an individual of *Value_Transfer_Activity* to only one individual of *Value_Transfer_Activity* (see ⑤ in Figure 7.1).

• An *Inverse Functional* property relates a given individual to more than one individual via the property. For example, the property *Duality_Dependency* is inverse functional as it relates an individual of *Coordination_Activity* to two individuals of *Value_Transfer_Activity* (see ⑥ in Figure 7.1).

• A Transitive property relates an individual \(a\) to an individual \(b\), and also an individual \(b\) to an individual \(c\), then an individual \(a\) to an individual \(c\) via the property.

• A Symmetric property relates an individual \(a\) to an individual \(b\), then individual \(b\) to an individual \(a\) via the property.
<owl:Class rdf:ID="Value_Transfer_Activity">
  <owl:disjointWith>
    <owl:Class rdf:ID="Production_Activity"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Coordination_Activity"/>
  </owl:disjointWith>
</owl:Class><owl:Class rdf:about="#Coordination_Activity">
  <owl:disjointWith rdf:resource="#Value_Transfer_Activity"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Production_Activity"/>
  </owl:disjointWith>
</owl:Class><owl:Class rdf:about="#Production_Activity">
  <owl:disjointWith rdf:resource="#Coordination_Activity"/>
  <owl:disjointWith rdf:resource="#Value_Transfer_Activity"/>
</owl:Class><owl:ObjectProperty rdf:ID="Trust_Dependency">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="#Value_Transfer_Activity"/>
  <rdfs:domain rdf:resource="#Value_Transfer_Activity"/>
</owl:ObjectProperty><owl:ObjectProperty rdf:ID="Trigger_Dependency">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdfs:range rdf:resource="#Production_Activity"/>
  <rdfs:domain rdf:resource="#Coordination_Activity"/>
</owl:ObjectProperty><owl:ObjectProperty rdf:ID="Duality_Dependency">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdfs:range rdf:resource="#Coordination_Activity"/>
  <rdfs:domain rdf:resource="#Value_Transfer_Activity"/>
</owl:ObjectProperty>
Figure 7.1 A part of the OWL code of the activity dependency model for the Scientific Conference scenario
Figure 7.2 A screenshot of the activity dependency model for the Scientific Conference scenario

7.2 Extensible Stylesheet Language Transformations (XSLT)

The Extensible Stylesheet Language Transformations (XSLT) (Clark, 1999) is a language used to transform and process XML documents. We use XSLT as the language to develop transformation programs to implement the mapping rules and guidelines presented in Chapter 3, Chapter 4, Chapter 5 and Chapter 6. The transformation process requires an XML document, an XML parser and an XSLT style sheet. The XML document and the XSLT style sheet work as the input for the parser. The XML
document in our work could be for example, OWL representation of the \( e^3 \) value model or the activity dependency model of the Scientific Conference scenario. An XSLT style sheet is itself an XML document consisting of one or more sets of rules that are called templates. Each template contains rules to apply when a specified node is matched. During the transformation, XSLT transforms an XML source-tree into an XML result-tree. The parser reads the input XML document into memory, creating a tree depiction of the input XML document based on its hierarchical structure. This structure is then compared against the XSLT style sheet for any matching templates or chunks of code that should be applied to a specified section of the XML document.

Here, we briefly describe some basic XSLT constructs used during the implementation using a sample XSLT code given in Figure 7.3.

- **<xsl:stylesheet>**

  The `<xsl:stylesheet>` element is always the top-level element of an XSLT style sheet. The name `<xsl:transform>` may be used as a synonym (see ① in Figure 7.3).

- **<xsl:template match="XPath Expression">**

  The `<xsl:template>` element is used to build templates. The `match` attribute is used to associate a template with an XML element in the input file. The value of the match attribute is an XPath expression. For example, `match="rdf:RDF"` defines the node `RDF` in the root node (see ② in Figure 7.3).

- **<xsl:element name="String constant">**

  The `<xsl:element>` is used to create an output element. The name of the output element can be specified in the name attribute. For example, `<xsl:element name="Pool">` creates an element called ‘Pool’ (see ③ in Figure 7.3).
Figure 7.3 A sample XSLT code of the Scientific Conference scenario
• `<xsl:attribute name="String constant">`

The `xsl:attribute` element is used to add an attribute value to an element. For example, `<xsl:attribute name="Name">` adds an attribute ‘Name’ to a given element (see ④ in Figure 7.3).

• `<xsl:apply-templates select= "XPath Expression">`

The `xsl:apply-templates` element applies a template to the current element or to the current element's child nodes. If we add the select attribute to the `xsl:apply-templates` element, it will process only the child elements that match the value of the attribute.

• `<xsl:value-of select= "XPath Expression">`

The `xsl:value-of` element is used to extract the value of a selected node and add it to the output stream of the transformation. For example, `<xsl:value-of select= "@rdf:ID"/>` selects the value of the attribute ‘ID’ of the current node (see ⑤ in Figure 7.3).

• `<xsl:for-each select= "XPath Expression">`

The `xsl:for-each` element iterates the statement over the nodes selected by select attribute in the XPath expression. It can be used as an alternative to `<xsl:apply-templates>` when the child nodes of the current node are known in advance. In `<xsl:for-each select= "Coordination_Activity[@rdf:ID='Conference_Program']">` the enclosed statements will iterate over the coordination activity `Conference_Program` node in the RDF node (see ⑥ in Figure 7.3).
7.3 Comparison with Similar Research

In this section, we compare our work with similar studies that have been carried out by others (Kueng, 1997) (Jayaweera, 2004) (Andersson, 2006b) (Pijpers, 2007) (Raadt, 2005). We use these comparisons to evaluate our methods with the intention of investigating how well they are in line with already existing methods. This is done in two steps:

1. Compare the method used to construct a business model as discussed in Chapter 3.

2. Compare the method used to construct a process model as discussed in Chapter 6.

Compare Business Model Design Method:

Raadt et al. (2005) discuss the relationship between goals and business models to identify goal-aligned economically profitable business services. The approach starts by eliciting strategic business goal that enterprises want to cooperate. It then formulates alternative ways of reaching these goals, by using modelling and evaluation techniques of i* (Yu, 1995). The approach then constructs an $e^3$ value business model for each alternative. Each alternative business model is evaluated to find out whether it is satisfactory for every stakeholder involved in the value network. The resulting business model is agreeable to every stakeholder in the value network and can be used as a good starting point for the design of business services. Some guidelines are discussed to transform i* goal modelling framework to an $e^3$ value model.

The guidelines proposed in Chapter 3 to move from a goal model to a business model have some similarities to those presented in Raadt (2005). But we use a different language for the goal modelling. The method starts by discovering and analysing strategic business goals of enterprises. We formulated the main elements in KAOS framework, such as goals and system requirements in terms of business notions. Four goal patterns are
discussed to formulate goals in a uniform way and a set of templates is proposed for formulating system requirements to fulfil needs in value networks. Finally, a set of guidelines is proposed for creating a business model that conforms to the goals.

**Compare Process Model Design Methods:**

As discussed in Section 2.5.1, various approaches have been proposed in (Kueng, 1997), Unified Framework (Jayaweera, 2004), Chaining Methodology (Andersson, 2006b) and e³transition model (Pijpers, 2007) for designing process models. Some of these use business view as the starting point whereas others use goal view. We use process design framework elements to compare these researches with our method. The results of the comparison are presented in Table 7.1. The information in the last column is related to the work presented in Chapter 6.

The goal oriented approach presented by Kueng (1997) makes an attempt to create process models from goals. It identifies process activities, responsibility roles, input and output resources of the activities from functional goals. This means with respect to the process design framework it covers the organizational and functional perspectives by identifying roles and activities. Informational perspective is covered to a certain extent by identifying the input and output resources of the activities. The research does not discuss the transfer of rights and evidence documents. Further, there is no discussion on orchestrating process activities.

Jayaweera’s (2004) Unified Framework uses business view as the starting point to design process models. It proposes a set of contract-negotiation and contract-execution process patterns to build process models by considering value transactions and roles played by actors in these value transactions. Some production rules are discussed to orchestrate the value exchanges involved in value transactions. Control flows and data flows are handled through these production rules. However, goals are not of primary interest in this work. Moreover, the transfer of rights and evidence
documents which are important components in value exchanges are not recognized. Therefore, the unified framework covers organizational, functional, behavioural perspectives and informational perspective to a certain extent.

**Table 7.1 Summary of the comparison of process design methods**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizational (Actor)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Functional (Activity)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Informational - Rights of resource</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Informational - Custody of resource</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Informational - Evidence document</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Behavioral</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The chaining methodology (Andersson, 2006b) uses e³ value model as the starting point in designing process models. It extends the e³ value model by capturing various business constraints which are not typically described in e³ value model. The method uses the value exchange analysis presented in (Andersson, 2006a) to elicit set of processes to include in the process model. However, the work does not discuss any mechanism to orchestrate these process activities. Further, methodology does not take into account the goal view for its consideration. The methodology covers organizational, functional and informational perspectives of the process design framework.

The concept of e³ transition model (Pijpers, 2007) provides a way to reduce the gap between economic e³ value business models and process models. The e³ transition model captures the notion of rights which are not modelled in e³ value models. A set of guidelines are proposed to go from one model to another. But these guidelines do not provide any heuristics to orchestrate activities in
a process model. This approach covers organizational and functional perspectives and informational perspective to a certain extent.

Our method bridges the gap between high-level, declarative models (i.e. goal and business models) and their operative realization given in the form of process models through an intermediary model. Regarding the different focus of goal and business models, we have tried to utilize both dimensions, i.e. goal-oriented and value-oriented, in order to capture them in the operational model. To structure the transitions we have used the process modelling framework that encompasses all the major aspects of process design, i.e. organizational, informational (including rights, custody and evidence document), functional and behavioural (control flow and data flow). The guidelines resulted in a set of transition rules that enable the obtaining and structuring of process model elements from goal model and business model components.

7.4 Conclusion

In this chapter we discussed the tools used during the implementation of our work and details of the implementation. Using Protégé-OWL plug-in we created OWL ontologies for BMM, i*, KAOS, e3 value model and activity dependency model for the Scientific Conference scenario. The mapping rules and guidelines presented in Chapter 3, Chapter 4, Chapter 5 and Chapter 6 were implemented by developing XSLT transformation programs as a proof of concepts. Through these transformation programs we have automated the derivation of the output models using its input counterparts.

The guidelines presented in Chapter 3 to construct a business model from a goal model have some similarities to those presented in Raadt (2005). The differences are that we propose a new way to model goals and system requirement to fulfil needs in value
networks and use a different language for the goal modelling. The similarity is that both approaches consider agent dependencies for the analysis.

The method proposed to design a process model in Chapter 6 use both goal view and economic value view. This is different from existing literature, where existing solutions use only either goal view or economic value view. This enables justify process evaluation from both strategic and economic value viewpoint. According to the Table 7.1, our method covers all aspects in the process design framework that includes organizational, functional, informational and behavioural perspectives.
8 Concluding Discussion and Future Research Directions

In this chapter we present conclusions and directions for further research. In Section 8.1, we revisit our research questions (see Section 1.3). We then discuss how and to which extent these questions have been addressed. Section 8.2 outlines directions for future research.

8.1 Concluding Discussion

The increase in e-Commerce activities has given rise to a need for requirements engineering methods which can analyze, design, implement and manage e-Commerce information systems. Such an e-Commerce environment involves multi-actor networks where each actor contributes its own specialty. Enterprises use different models for eliciting requirements of such e-Commerce information systems. Goal, business and process models are used in a chain for this purpose.

The goal models represent intentions and strategies of different actors. More detailed goal modelling techniques even consider the actor dependencies. The business models identify actors, economic resources as well as how economic resources are created and distributed. A process model depicts the behaviour of actors in terms of activities to be executed, roles, resource flows, activity ordering and data flows. Goal models enable reasoning about business choices made at the business level and operational choices made at the process level. Business models are also paving the way to reasoning about operational choices made at the process level in terms of business notions.
The design of process models is a difficult, expensive and time consuming task. Process modelling techniques face a number of challenges that need to be addressed. There is a need for structured design methods to reduce the complexity of process design. The design decisions made during the construction of the process should be traceable. The process design techniques should provide the flexibility to process models by allowing them to use already provided solutions in order to reduce the time and effort put into the designing task. In addition, there is a need for structured methods for designing business models. A business model should be designed in such a way that it conforms to the goals of the enterprise. Otherwise the components in a business model will not be in line with enterprise intentions.

Therefore, in this thesis we address an important question in requirements engineering: how can the design of business models and process models be supported by the use of other kinds of enterprise models? This research question is answered in several steps, by dividing it into sub-questions. We have used the design science research approach to find answers to the following questions.

- *How can goal modelling be used to support the design of business models?*

Our goal here is to construct a method to design an enterprise business model from an enterprise goal model. We have used the well-established techniques *KAOS* and *e³value* for goal modelling and business modelling, respectively. The method starts by eliciting and analyzing strategic business goals. We provide a uniform way to express strategic goals and system requirements of value networks using business notions. Finally, complementing a goal model with a *KAOS* agent dependency model, we build a set of guidelines to construct a business model that conforms to the enterprise business goals.

The proposed method offers a number of benefits. First, we have formulated the main elements of the *KAOS* language, such as goals and system requirements, in a uniform way and in terms of
business notions. The goal modellers find it is easy to formulate goals and system requirements using this approach. It also makes these expressions more uniform. Second, the proposed method provides a structured way of designing business models that are based on the goals and the needs of an enterprise as expressed in a goal model. Third, using our method, it becomes possible to relate the components of KAOS models to those of a business model, which gives a firm basis for establishing a complete traceability of decisions from the business to the strategic level. Using the method, a goal modeller finds it’s easy to construct a unique business model.

- How can goal modelling be used to support the design of process models?

Here, our goal is to construct a set of mappings between goal modelling language constructs and process modelling notions. For the study, we have chosen three well-established goal modelling languages BMM, i* and KAOS. To describe different aspects of process modelling, we employed Curtis’s process design framework. The framework consists of four perspectives: organizational, functional, informational and behavioural. The aim is to compare support of these goal modelling languages for process design. The mappings are summarized in Table 4.1 and have exposed the following.

Organisational perspective: The participant element can be obtained from i* strategic actor and KAOS agent. Thus, both these languages include participant element in the process design framework. A BMM business plan is constructed with respect to the interested enterprise. In addition, BMM influencers can also be used to extract the participant information. Thus, BMM supports participant element in the process design framework in an indirect way.

Functional perspective: The goal modelling languages considered support functional view similarly. Each language covers the functional view in the form of actions that are used to realize its
intentions. The actions named differently in each of this language aligned with the activity element in the process design framework.

**Informational perspective:** The \( i^* \) resource and \( KAOS \) entity notions are aligned with resource element in the information perspective. In \( BMM \) resources can be captured from the influencers to some extent. However, none of the languages distinguishes between different types of resources. The \( i^* \) and \( BMM \) does not discuss the event element in the information perspective. The \( KAOS \) event type object is mapped to the event element in the information perspective.

**Behavioural perspective:** The business rules in \( BMM \) can be used to design both static and dynamic process models. In \( KAOS \) and \( i^* \) languages, dependencies among participant actors can be used to determine the ordering of activities involved in those dependencies. Thus, all three languages support control flow in a certain way.

The business rules in \( BMM \) and input and output relations of operations in \( KAOS \) can be used to decide the data flows between involved activities. In \( i^* \) data flows can be partially discovered from resource dependencies.

A clear benefit of these mapping rules is that they help to identify the relations between concepts in the three goal modelling languages to the process layer.

- **How can business modelling be used to support the design of process models?**

The goals here are to: (1) Create mappings between business modelling language constructs and process modelling notions. (2) Construct a method to design a process model based on a business model. We have chosen the \( e^3 value \) model as our basis for business modelling. The process models are described using Curtis’s process design framework. The mappings reveal that the \( e^3 value \) model supports organizational, functional and informational perspectives. In \( e^3 value \) model activities are described to a different level of granularity. The \( e^3 value \) model does not expose any information regarding the orchestration of
Some additional knowledge from the problem domain is needed to order the process activities. By applying the guidelines we can construct a process model and structure process activities with a business viewpoint. The elements in the process model are aligned with those in the business model. This provides complete traceability for design decisions made in constructing the process model and motivate them using business notions.

- *How can goal modelling and business modelling be used to support the design of process models?*

Our goal is to construct a method to design a process model based on a goal model and a business model. Such a method facilitates the task of designing process models in a way that they are in line with strategic and business viewpoints. A key element of the approach is the notion of the activity dependency model, which works as a bridge between goal, business and process models. Regarding the different focus of goal and business models, we have tried to utilize both dimensions, i.e. intentional and economic value, in order to capture them in the operational model. The relationships between the activities within a process are captured through activity dependencies. Four kinds of dependencies have been utilized: flow, trust, trigger and duality. They have a clear strategic and business motivation and are used for the construction of the process model. The presented method addresses the challenges that are faced by process design techniques, namely: managing process design complexity, provide traceability and offer flexibility, which are discussed in Chapter 1.

The mapping rules and guidelines provided in this thesis work can be applied in an automated way. The modelling languages we considered, BMM, i*, KAOS and e3value are based on underlying ontologies. The four aspect process design framework shown in a core form in Figure 2.8 is a part of process ontology. Therefore, these transformation steps provide a basis for mapping between respective ontologies. Thereby, the transformation steps presented here form a basis for providing structured tool support for the development of respective models. The implementation steps discussed in Chapter 7 prove this is feasible.
As discussed in Chapter 1, the work presented in this thesis used design science research approach. To evaluate the usefulness, applicability and effectiveness of the proposed artifacts we used the Hevner et al. (2004) conceptual framework. The framework includes seven guidelines: design as an artifact, problem relevance, design evaluation, research contribution, research rigour, design as a search process and communication of research. The discussion on the validity of the results which is based on above guidelines can be found in Chapter 1.6.2 and Chapter 7 of the thesis.

8.2 Future Research Directions

There is more work to be done to gain real application benefits. Here we briefly discuss a few possible directions for future research.

1. The major task for any future work concerns the development of tool support to facilitate a minimal manual effort in the development of the respective models, in particular the business models and process models.

2. Another topic that is of interest is the validation of the methods by applying them to more complex case studies. Currently, we have not established the conditions that the goal model needs to fulfil in order to enable design support for business models and process models.

3. The study of overlapping in goal models and business models are also an interesting topic.

4. An exploration of whether additional templates are needed to model system requirements of value networks are an interesting issue. The investigation of whether additional kinds of activity dependencies are required in the activity dependency model is also an interesting topic. Another topic for further research is to study how to include more phases, in addition to the negotiation and execution phases, in the process models.
References


Appendix I

Program Output I: XML description corresponding to the $e^3value$ model for the Scientific Conference Scenario presented in Figure 3.3

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Business_Model>
  <Actors_Market_Segments>
    <Actor_Market_Segment>Reviewer</Actor_Market_Segment>
    <Actor_Market_Segment>ProgramCommitte</Actor_Market_Segment>
    <Actor_Market_Segment>Author</Actor_Market_Segment>
    <Actor_Market_Segment>SteeringCommittee</Actor_Market_Segment>
  </Actors_Market_Segments>
  <Value_Exchanges>
    <Value_Exchange>odAcknowledgement</Value_Exchange>
    <Value_Exchange>odReview_Report</Value_Exchange>
    <Value_Exchange>odSubmission</Value_Exchange>
    <Value_Exchange>odEvaluation</Value_Exchange>
    <Value_Exchange>odPayment</Value_Exchange>
    <Value_Exchange>Achieve_Organize_a_conference_to_enhance_the_acknowledge</Value_Exchange>
  </Value_Exchanges>
  <Value_Objects>
    <Value_Object>Acknowledge Contribution</Value_Object>
    <Value_Object>Review_Report</Value_Object>
    <Value_Object>Submission</Value_Object>
    <Value_Object>Evaluation</Value_Object>
    <Value_Object>Payment</Value_Object>
    <Value_Object>Conference_Program</Value_Object>
  </Value_Objects>
</Business_Model>
```
/Value_Objects>
  <Value_Interfaces>
    <Value_Interface>review</Value_Interface>
    <Value_Interface>paper</Value_Interface>
    <Value_Interface>fee</Value_Interface>
  </Value_Interfaces>
  <Value_Activities>
    <Value_Activity>AssignReviewer</Value_Activity>
    <Value_Activity>PrepareConferenceProgram</Value_Activity>
    <Value_Activity>DecideOnAcceptance</Value_Activity>
  </Value_Activities>
  <Value_Exchanges_Among_Value_Activities>
    <Value_Exchange Name="Submission" From="DecideOnAcceptance" To="AssignReviewer" />
    <Value_Exchange Name="ReviewReport" From="AssignReviewer" To="DecideOnAcceptance" />
  </Value_Exchanges_Among_Value_Activities>
  <Value_Exchanges_among_Actors_and_Value_Activities>
    <Value_Exchange Name="Acknowledgement">
      <Value_Interface>review</Value_Interface>
      <Value_Activity>AssignReviewer</Value_Activity>
    </Value_Exchange>
    <Value_Exchange Name="ReviewReport">
      <Value_Interface>review</Value_Interface>
      <Value_Activity>AssignReviewer</Value_Activity>
    </Value_Exchange>
    <Value_Exchange Name="Submission">
      <Value_Interface>paper</Value_Interface>
      <Value_Activity>DecideOnAcceptance</Value_Activity>
    </Value_Exchange>
    <Value_Exchange Name="Evaluation">
      <Value_Interface>paper</Value_Interface>
      <Value_Activity>DecideOnAcceptance</Value_Activity>
    </Value_Exchange>
    <Value_Exchange Name="Payment">
      <Value_Interface>fee</Value_Interface>
    </Value_Exchange>
</Value_Objects>
<Value_Activity> PrepareConferenceProgram </Value_Activity>
</Value_Exchange>
<Value_Exchange Name="ConferenceProgram">
<Value_Interface>fee</Value_Interface>
<Value_Activity> PrepareConferenceProgram </Value_Activity>
</Value_Exchange>
</Value_Exchanges_among_Actors_and_Value_Activities>
</Business_Model>
Program Output II: XML description of mappings of elements in the BMM to the process design framework for the Scientific Conference scenario

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Process_Model_Prospectives>
  <Organisational_Perspective>
    <Organiztioal_Unit Name="Program_Committee" />
    <Influencers>
      <Partner Name="Reviewer" />
      <Partner Name="Steering_Committee" />
      <Customer Name="Author" />
    </Influencers>
  </Organisational_Perspective>
  <Functional_Perspective>
    <Course_of_Action>
      <Tactic Name="Decide_on_acceptance" />
      <Tactic Name="Assigned_reviewer" />
    </Course_of_Action>
  </Functional_Perspective>
  <Informational_Perspective>
    <Influencers>
      <Resource Name="Money" />
    </Influencers>
  </Informational_Perspective>
  <Behavioral_Perspective>
    <Control_Flow>
      <Business_Rule>Use_review_reports_to_take_a_decision_on_a_paper</Business_Rule>
    </Control_Flow>
  </Behavioral_Perspective>
</Process_Model_Prospectives>
```
Program Output III: XML description of mappings of elements in the *i* language to the process design framework for the Scientific Conference scenario

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Process_Model_Prospectives>
  <Organisational_Perspective>
    <Actor Name="Author" />
    <Actor Name="Steering_Committee" />
    <Actor Name="Reviewer" />
    <Actor Name="Program_Committee" />
  </Organisational_Perspective>
  <Functional_Perspective>
    <Tasks>
      <Task Name="Assigned_Rewiewer" />
      <Task Name="Decide_on_Acceptance" />
    </Tasks>
  </Functional_Perspective>
  <Informational_Perspective>
    <Resources>
      <Resource Name="Submission" />
      <Resource Name="Acknowledge_Contribution" />
      <Resource Name="Review_Report" />
      <Resource Name="Evaluation" />
      <Resource Name="Payment" />
    </Resources>
  </Informational_Perspective>
  <Behavioral_Perspective>
    <Control_Flow>
      <Resource_Dependency Name="Submission_Dep" />
      <Resource_Dependency Name="Evaluation_Dep" />
      <Resource_Dependency Name="Acknowledge_Contribution_Dep" />
      <Resource_Dependency Name="Review_Report_Dep" />
      <Resource_Dependency Name="Payment_Dep" />
      <Goal_Dependency Name="Achieve_Organize_a_conference_to_enhance_the_knowledge" />
    </Control_Flow>
  </Behavioral_Perspective>
</Process_Model_Prospectives>
```
Program Output IV: XML description of mappings of elements in the KAOS language to the process design framework for the Scientific Conference scenario

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Process_Model_Prospectives>
  <Organisational_Perspective>
    <Agent Name="Author" />
    <Agent Name="Program_Committee" />
    <Agent Name="Reviewer" />
    <Agent Name="Steering_Committee" />
  </Organisational_Perspective>
  <Functional_Perspective>
    <Operations>
      <Operation Name="Decide_on_Acceptance" />
      <Operation Name="Assigned_Reviewer" />
    </Operations>
  </Functional_Perspective>
  <Informational_Perspective>
    <Entity Name="Payment" />
    <Entity Name="Acknowledge_Contribution" />
    <Entity Name="ReviewReport" />
    <Entity Name="Evaluation" />
    <Entity Name="Submission" />
  </Informational_Perspective>
  <Behavioral_Perspective>
    <Control_Flow>
      <Goal_Dependency Name="Achieve_Organize_a_conference_to_enhance_the_knowledge" />
      <Event Name="Send_a_reminder_to_reviewer" />
      <Event Name="Receive_submission_until_the_deadline" />
    </Control_Flow>
    <Data_Flows>
      <Data_Flow Name="ReviewReport" From="Assigned_Reviewer" To="Decide_on_Acceptance" />
    </Data_Flows>
  </Behavioral_Perspective>
</Process_Model_Prospectives>
```
Program Output V: XML description of mappings of elements in the $e^3$ value model to the process design framework for the Scientific Conference scenario

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Process_Model_Prospectives>
  <Organisational_Perspective>
    <MarketSegment Name="Author"/>
    <MarketSegment Name="Reviewer"/>
    <Actor Name="ProgramCommitte"/>
    <Actor Name="SteeringCommitte"/>
  </Organisational_Perspective>
  <Functional_Perspective>
    <Value_Activities>
      <Value_Activity Name="DecideOnAcceptance"/>
      <Value_Activity Name="AssignReviewer"/>
    </Value_Activities>
    <Outgoing_ValueOfferings>
      <Out_Port Name="outAuthorSubmission"/>
      <Out_Port Name="outPCVAEvaluation"/>
      <Out_Port Name="outReviewerReviewReport"/>
      <Out_Port Name="outSCPayment"/>
      <Out_Port Name="outPCAcknowledgement"/>
      <Out_Port Name="outPCConferenceProgram"/>
      <Out_Port Name="outPCEvaluation"/>
      <Out_Port Name="outPCVAConferenceProgram"/>
    </Outgoing_ValueOfferings>
    <Incoming_ValueOfferings>
      <In_Port Name="inAuthorEvaluation"/>
      <In_Port Name="inPCVAPayment"/>
      <In_Port Name="inSCConferenceProgram"/>
      <In_Port Name="inReviewerAcknowledgement"/>
      <In_Port Name="inPCVAREviewReport"/>
      <In_Port Name="inPCReviewReport"/>
      <In_Port Name="inPCPayment"/>
      <In_Port Name="inPCSubmission"/>
    </Incoming_ValueOfferings>
</Process_Model_Prospectives>
```
<In_Port Name="inPCVASubmission"/>
</Incoming_ValueOfferings>
<Value_Exchanges>
  <Value_Exchange Name="vePayment"/>
  <Value_Exchange Name="veConferenceProgram"/>
  <Value_Exchange Name="veSubmission"/>
  <Value_Exchange Name="veAcknowledgement"/>
  <Value_Exchange Name="veReviewReport"/>
  <Value_Exchange Name="veEvaluation"/>
</Value_Exchanges>
<Value_Transactions>
  <Value_Transaction Name="vtEvaluation"/>
  <Value_Transaction Name="vtConferenceProgram"/>
  <Value_Transaction Name="vtReviewreport"/>
</Value_Transactions>
</Functional_Perspective>
<Informational_Perspective>
  <right Name="Ownership"/>
  <resource Name="Recognition"/>
  <resource Name="Review_report"/>
  <resource Name="Conference_program"/>
  <resource Name="Money"/>
  <resource Name="Evaluation"/>
  <resource Name="Submission"/>
  <Evidence_Document Name="Invoice"/>
  <Evidence_Document Name="Acknowledgement"/>
  <Evidence_Document Name="EmailNotification"/>
</Informational_Perspective>
</Process_Model_Prospectives>
Program Output VI: XML description of the process model corresponds to Figure 5.1

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<ProcessModel>
  <Pool Name="Coordinate_Evaluation">
    <Provide>
      <Resource>Evaluation</Resource>
      <Right>Ownership</Right>
      <EvidenceDocument Name="Email_Notification" />
    </Provide>
    <Receive>
      <Resource>Submission</Resource>
      <Right>Ownership</Right>
      <EvidenceDocument Name="Email_Notification" />
    </Receive>
    <Production>AssignReviewer</Production>
    <Production>DecideOnAcceptance</Production>
  </Pool>
  <Pool Name="Coordinate_Submission">
    <Receive>
      <Resource>Evaluation</Resource>
      <Right>Ownership</Right>
      <EvidenceDocument Name="Email_Notification" />
    </Receive>
    <Provide>
      <Resource>Submission</Resource>
      <Right>Ownership</Right>
      <EvidenceDocument Name="Email_Notification" />
    </Provide>
  </Pool>
</ProcessModel>
```
Program Output VII: XML description of the activity dependency model corresponds to Figure 6.2

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Activity_Dependency_Model>
    <Coordination Activity="Submission">
        <Transfer_Activity Name="Receive_Submission"/>
        <Transfer_Activity Name="Provide_Evaluation"/>
        <Duality_Dependency From="Coordinate Submission" To="Receive_Submission"/>
        <Duality_Dependency From="Coordinate Submission" To="Provide_Evaluation"/>
    </Coordination>
    <Coordination Activity="Conference_Program">
        <Transfer_Activity Name="Receive_Payment"/>
        <Transfer_Activity Name="Provide_ConferenceProgram"/>
        <Duality_Dependency From="Coordinate Conference_Program" To="Receive_Payment"/>
        <Duality_Dependency From="Coordinate Conference_Program" To="Provide_ConferenceProgram"/>
        <Production_Activity Name="DecideOnAcceptance"/>
        <Production_Activity Name="AssignReviewer"/>
    </Coordination>
    <Coordination Activity="Review">
        <Transfer_Activity Name="Receive_ReviewReport"/>
        <Duality_Dependency From="Coordinate Review" To="Receive_ReviewReport"/>
        <Duality_Dependency From="Coordinate Review" To="Provide_Acknowledgement"/>
    </Coordination>
</Activity_Dependency_Model>
```
Program Output VIII: XML description of the process model corresponds to Figure 6.7

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Pool Name="Coordinate Conference_Program">
  <Sub_Processes>
    <Negotiate>Conference_Program</Negotiate>
    <Transfer_Activity>Provide_Conference_Program</Transfer_Activity>
    <Transfer_Activity>Receive_Payment</Transfer_Activity>
    <Production_Activity Type="Loop">Assign_Reviewer</Production_Activity>
    <Production_Activity Type="Loop">DecideOnAcceptance</Production_Activity>
    <Transfer_Activity Type="Loop">Receive_Review_Report</Transfer_Activity>
    <Transfer_Activity Type="Loop">Receive_Submission</Transfer_Activity>
    <Transfer_Activity Type="Loop">Provide_Evaluation</Transfer_Activity>
  </Sub_Processes>
  <Seqence_Flows>
    <Sequence_Flow From="Assign_Reviewer" To="Receive_Review_Report"/>
    <Sequence_Flow From="DecideOnAcceptance" To="Provide_Evaluation"/>
    <Sequence_Flow From="Provide_Evaluation" To="Provide_Conference_Program"/>
    <Sequence_Flow From="Receive_Review_Report" To="DecideOnAcceptance"/>
    <Sequence_Flow From="Receive_Submission" To="Assign_Reviewer"/>
    <Sequence_Flow From="Receive_Payment" To="Provide_Conference_Program"/>
  </Seqence_Flows>
</Pool>
```
Appendix II

Program I: XSLT source code to produce an $e^3\text{value}$ model from the KAOS models

```xml
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xs="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    exclude-result-prefixes="rdfs owl xs rdf">

    <xsl:output method="xml" indent="yes"/>

    <xsl:template match="/rdf:RDF">
        <Business_Model>

            <xsl:element name="Actors_Market_Segments">
                <xsl:for-each select="//Agent">
                    <xsl:element name="Actor_Market_Segment">
                        <xsl:value-of select="#@rdf:ID"/>
                    </xsl:element>
                </xsl:for-each>
            </xsl:element>

            <xsl:element name="Value_Exchanges">
                <xsl:for-each select="//Object_Dependency">
                    <xsl:element name="Value_Exchange">
                        <xsl:value-of select="#@rdf:ID"/>
                    </xsl:element>
                </xsl:for-each>
                <xsl:for-each select="//GoalDependency">
                    <xsl:element name="Value_Exchange">
                        <xsl:value-of select="#@rdf:ID"/>
                    </xsl:element>
                </xsl:for-each>
            </xsl:element>

        </Business_Model>
    </xsl:template>
</xsl:stylesheet>
```

<xsl:if test="$resource1=$resource2">
  <xsl:element name="Value_Exchange">
    <xsl:attribute name="Name">
      <xsl:value-of select="$resource1"/>
    </xsl:attribute>
    <xsl:attribute name="From">
      <xsl:value-of select="$va1"/>
    </xsl:attribute>
    <xsl:attribute name="To">
      <xsl:value-of select="$va2"/>
    </xsl:attribute>
  </xsl:element>
</xsl:if>
</xsl:for-each>
</xsl:for-each>

<xsl:for-each select="/rdf:RDF/Produce/output_va">
  <xsl:variable name="va1" select="../@rdf:ID"/>
  <xsl:variable name="resource1" select="substring-after(@rdf:resource,'#')"/>
  <xsl:for-each select="/rdf:RDF/Produce/input_va">
    <xsl:variable name="va2" select="../@rdf:ID"/>
    <xsl:variable name="resource2" select="substring-after(@rdf:resource,'#')"/>
    <xsl:if test="$resource1=$resource2">
      <xsl:element name="Value_Exchange">
        <xsl:attribute name="Name">
          <xsl:value-of select="$resource1"/>
        </xsl:attribute>
        <xsl:attribute name="From">
          <xsl:value-of select="$va1"/>
        </xsl:attribute>
        <xsl:attribute name="To">
          <xsl:value-of select="$va2"/>
        </xsl:attribute>
      </xsl:element>
    </xsl:if>
  </xsl:for-each>
</xsl:for-each>
</Value_Exchanges_Among_Value_Activities>
<Value_Exchanges_among_Actors_and_Value_Activities>
<xsl:for-each select="Group">
  <xsl:variable name="vi1" select="@rdf:ID"/>
  <xsl:for-each select="Group_Object_Goal_Dependency/Object_Dependency/ObjectDependency_involves_Entity">
    <xsl:variable name="resource1" select="substring-after(@rdf:resource,'#')"/>
    <xsl:for-each select="/rdf:RDF/Produce">
      <xsl:variable name="va1" select="@rdf:ID"/>
      <xsl:variable name="resource2" select="substring-after(input_actor/@rdf:resource,'#')"/>
      <xsl:if test="$resource1 = $resource2">
        <xsl:element name="Value_Exchange">
          <xsl:attribute name="Name">
            <xsl:value-of select="$resource1"/>
          </xsl:attribute>
          <xsl:element name="Value_Interface">
            <xsl:value-of select="$vi1"/>
          </xsl:element>
          <xsl:element name="Value_Activity">
            <xsl:value-of select="$va1"/>
          </xsl:element>
        </xsl:element>
      </xsl:if>
    </xsl:for-each>
  </xsl:for-each select="/rdf:RDF/Produce">
    <xsl:for-each select="/rdf:RDF/Produce">
      <xsl:variable name="va1" select="@rdf:ID"/>
      <xsl:variable name="resource2" select="output_actor/Entity/@rdf:ID"/>
      <xsl:if test="$resource1 = $resource2">
        <xsl:element name="Value_Exchange">
          <xsl:attribute name="Name">
            <xsl:value-of select="$resource1"/>
          </xsl:attribute>
          <xsl:element name="Value_Interface">
            <xsl:value-of select="$vi1"/>
          </xsl:element>
          <xsl:element name="Value_Activity">
            <xsl:value-of select="$va1"/>
          </xsl:element>
        </xsl:element>
      </xsl:if>
    </xsl:for-each select="/rdf:RDF/Produce">
</xsl:for-each select="Group">
</Value_Exchanges_among_Actors_and_Value_Activities>
<xsl:element name="Value_Activity">
  <xsl:value-of select="$va1"/>
</xsl:element>
</xsl:if>
</xsl:for-each>

<xsl:for-each select="/rdf:RDF/Produce/input_actor/Entity">
  <xsl:variable name="va1" select="../../@rdf:ID"/>
  <xsl:variable name="resource2" select="@rdf:ID"/>
  <xsl:if test="$resource1 = $resource2">
    <xsl:element name="Value_Exchange">
      <xsl:attribute name="Name">
        <xsl:value-of select="$resource1"/>
      </xsl:attribute>
      <xsl:element name="Value_Interface">
        <xsl:value-of select="$vi1"/>
      </xsl:element>
      <xsl:element name="Value_Activity">
        <xsl:value-of select="$va1"/>
      </xsl:element>
    </xsl:element>
  </xsl:if>
</xsl:for-each>

<xsl:for-each select="/rdf:RDF/Produce/output_actor">
  <xsl:variable name="va1" select="../@rdf:ID"/>
  <xsl:variable name="resource2" select="substring-after(@rdf:resource,'#')"/>
  <xsl:if test="$resource1 = $resource2">
    <xsl:element name="Value_Exchange">
      <xsl:attribute name="Name">
        <xsl:value-of select="$resource1"/>
      </xsl:attribute>
      <xsl:element name="Value_Interface">
        <xsl:value-of select="$vi1"/>
      </xsl:element>
      <xsl:element name="Value_Activity">
        <xsl:value-of select="$va1"/>
      </xsl:element>
    </xsl:element>
  </xsl:if>
</xsl:for-each>

185
<xsl:for-each select="Group_Object_Goal_Dependency/GoalDependency/GoalDependency_involves_Entity">
  <xsl:variable name="resource1" select="substring-after(@rdf:resource,'#')"/>
  <xsl:for-each select="/rdf:RDF/Produce">
    <xsl:variable name="va1" select="@rdf:ID"/>
    <xsl:variable name="resource2" select="output_actor/Entity/@rdf:ID"/>
    <xsl:if test="$resource1 = $resource2">
      <xsl:element name="Value_Exchange">
        <xsl:attribute name="Name">
          <xsl:value-of select="$resource1"/>
        </xsl:attribute>
        <xsl:element name="Value_Interface">
          <xsl:value-of select="$vi1"/>
        </xsl:element>
        <xsl:element name="Value_Activity">
          <xsl:value-of select="$va1"/>
        </xsl:element>
      </xsl:element>
    </xsl:if>
  </xsl:for-each>
</xsl:for-each>

<xsl:for-each select="/rdf:RDF/Produce/input_actor/Entity">
  <xsl:variable name="va1" select="../../../../@rdf:ID"/>
  <xsl:variable name="resource2" select="@rdf:ID"/>
  <xsl:if test="$resource1 = $resource2">
    <xsl:element name="Value_Exchange">
      <xsl:attribute name="Name">
        <xsl:value-of select="$resource1"/>
      </xsl:attribute>
      <xsl:element name="Value_Interface">
        <xsl:value-of select="$vi1"/>
      </xsl:element>
      <xsl:element name="Value_Activity">
        <xsl:value-of select="$va1"/>
      </xsl:element>
    </xsl:element>
  </xsl:if>
</xsl:for-each>

<xsl:for-each select="/rdf:RDF/Produce/input_actor/Entity">
  <xsl:variable name="val" select="../@rdf:ID"/>
  <xsl:variable name="resource2" select="@rdf:ID"/>
  <xsl:if test="$resource1 = $resource2">
    <xsl:element name="Value_Exchange">
      <xsl:attribute name="Name">
        <xsl:value-of select="$resource1"/>
      </xsl:attribute>
      <xsl:element name="Value_Interface">
        <xsl:value-of select="$vi1"/>
      </xsl:element>
      <xsl:element name="Value_Activity">
        <xsl:value-of select="$va1"/>
      </xsl:element>
    </xsl:element>
  </xsl:if>
</xsl:for-each>
<xsl:attribute>
    <xsl:element name="Value_Interface">
        <xsl:value-of select="$vi1"/>
    </xsl:element>
    <xsl:element name="Value_Activity">
        <xsl:value-of select="$va1"/>
    </xsl:element>
</xsl:if>
</xsl:for-each>

<xsl:for-each select="/rdf:RDF/Produce/output_actor">
    <xsl:variable name="va1" select="../@rdf:ID"/>
    <xsl:variable name="resource2" select="substring-after(@rdf:resource,'#')"/>
    <xsl:if test="$resource1 = $resource2">
        <xsl:element name="Value_Exchange">
            <xsl:attribute name="Name">
                <xsl:value-of select="$resource1"/>
            </xsl:attribute>
            <xsl:element name="Value_Interface">
                <xsl:value-of select="$vi1"/>
            </xsl:element>
            <xsl:element name="Value_Activity">
                <xsl:value-of select="$va1"/>
            </xsl:element>
        </xsl:element>
    </xsl:if>
</xsl:for-each>
</xsl:for-each>

</Value_Exchanges_among_Actors_and_Value_Activities>
</Business_Model>
</xsl:template>
</xsl:stylesheet>