THESIS PROPOSAL

A SURVEY OF TRANSFORMATION OF BUSINESS MODELS IN BUSINESS NETWORKING

By

Che-Chun Lin

Supervisors:

David Robertson,
John Kingston

Division of Informatics
University of Edinburgh

November 2000
TABLE OF CONTENTS

1. Introduction

2. Business modelling (Literature review)
   2.1 ER Modelling (Entity Relationship Modelling)
   2.2 RADs (Role Activity Diagrams)
   2.3 BSDM
   2.4 Representation and formalization of business models

3. Representing transformation of business models (My approach)
   3.1 Business Networking
   3.2 The representation of the change in the business model
   3.3 Basic notation and semantics

4. Automatic transformation of the business models
   4.1 Example 1: The automation transformation of role replacement
   4.2 Example 2: The automation transformation of multi-agent replacement
   4.3 Support for reasoning ability in the transformation of business models

5. Reasoning about transformation of business models (A More realistic example)
   5.1 An example - book order from an Internet bookstore
   5.2 Attributes of the entities in the process
   5.3 Use of properties during transformation
   5.4 Summary

6. Research to Date and Future Targets

Appendix

Bibliography
Chapter 1

INTRODUCTION

Recent developments in business have been concerned with the means by which enterprises can achieve networkability and adjust their business models to be successful in the Internet context. Following the needs of coordination of business processes and adjustment of the business operations, the main concern in this document is to expand the capabilities of business modelling to account for this transformation and its automation.

Networkability, as the ability to cooperate internally as well as externally, refers to resources, business processes and business units, and is “of strategic importance for every business wishing to become part of a business network” [Osterle et al. 00]. In order to respond to such imperatives of the information age, enterprises need to establish efficient and advantageous business models relevant to the Internet environment. Getting involved in the context of networkability of the Internet environment, most companies would like to adjust part of this operation to benefit from the synergy of business networking and avoid putting them at a competitive disadvantage.

However, it may not be straightforward or easy to change an organisation’s operation into a brand new practice that can make use of today’s Internet technology. To address this problem, an organisation normally creates a model based on its current operation, upon which it bases new kinds of practices. We are interested in the extent to which this process of deriving new practices may be understood in knowledge engineering terms, and in the ways in which formalisation help us to automate model transformation in the domain.
We focus on the above model transformation problems and use logic in a lightweight style as a common language relevant to number methods to study the problems and explore automation in a retail aspect domain of business networking. There exist many competing business modelling methods. In order to maintain relevance to traditional methods of business modelling, we examine the basic concepts of three modelling methods (ER, RADs and BSDM) related to our research.

Instead of adopting a specific modelling method in all its complexity, we use more basic logical notation to represent typical business scenarios in this document to illustrate our approach. At the same time, we analyse basic methods and rationales we used to build those models. Based on these models, we will formalize our representation and develop an analytical framework.

This document describes examples of our preliminary work on the concepts and methods we use to form our conceptual models, explore the domain, build simple mechanisms for transformation of business models, and give scenarios of their use.

To accomplish the work, we must answer some key questions:

1. What are the characteristics of business networking and business models?
2. What are the problems of business transformation modelling?
3. Which are the appropriate methods (and/or methodologies) for transforming business models? How will they be used?
4. Why choose logical methods to handle issues such as automatic model transformation?
5. Is it feasible to automate the transformation of business models?
Chapter 2

BUSINESS MODELLING (LITERATURE REVIEW)

In this chapter, we survey and compare several business modelling methods, discuss the concepts of formalization of business models and put forward the strategies to our research on the transformation of business models.

2.1 ER Modelling (Entity Relationship Modelling)

Entity relationship modelling has its origins in conceptual modelling of databases but is used more generally. Before beginning our discussion of business modelling, we give a scenario to exemplify the ideas of entity relationship modelling: a customer places orders which specify the books being ordered and each order is billed by an invoice, which is received by the customer who order the books. In this statement, four entities and relationship between them are introduced. From the perspective of data modelling, when a customer places a book order, the representation of what happens in business operation depends on the analysis of the abstraction of enterprise data. The Entity-Relationship Diagram (ERD) in figure 1 represents a business model with a place-order data structure.

![Figure 1 Example entity relationship diagram](image-url)
Because entity relationship model maps the enterprise onto a conceptual model of data, it is obvious that the representation from an ERD does not include any activities and agents responsible for the activities in business modelling [Levene & Loizou 99]. Providing a high level model, an ERD has three components: entity types, relationship type and attributes. Here, we describe the concepts at the root of Entity-Relationship model and adjust to an approach to a transformation of business models:

- Entity and aggregation: A type of abstraction forming a class of objects is called aggregation, such as a customer entity or an order entity in figure 1. By abstraction, generalisation and specialisation, we can form objects with more structure, and commence to represent objects in business model. In the same way, when a transformation from a business model to another model is performed, in order to keep the original functionality while adapting new functions, the entities and relationships between them involved in the change should be captured. For example in the figure 1 case, if planning to add the other products- music and media to the order items, we can adapt the new entities- music entity and media entity and construct new relationship to our model. From the perspective of a transformation of business model, when we represent the entities in the business model, the type of abstraction about the kind of entity and its attribute should be paid more attention about the concepts of adaptive ability. For example, in the book order case, if we plan to outsource the book information to suppliers and add an agent to check the connection and consistency of different data sources from suppliers, we may need to add an agent entity, a process entity and a datastore entity to our model. After new entities are added and some of the entities are replaced, the new model should retain the business of processes of the original model, and the new functions of entities should work accordingly. However, the type of entity
should be also dealt with our declarative model in more structure for fitting into a transformation process.

- Relationships and their functionality: In ERD, there may be more than one relationship type between two entity types. In the transformation of the models this may cause problems, for instance if one of the entities is removed and the replaced entity could not discern the different relationships. Furthermore, in doing transformation, if we try to represent the relationship types as our component type, like in an ERD, not only do we need to match the functions from entity to entity, but also between entity and relationship. Initially, a simpler way to formalize the transformation is using two tiers (Entity-entity) rather than three tiers (Entity-entity-relationship), because the latter introduces problems in differentiating the property of a relationship from the property of an agent or a process.

- Attribute and Domain: In ER modelling, “the domain of an attribute of an entity type is a set of constraint values associated with that attribute.” [Levene & Loizou 99]. In a similar way, we need a set of constraint variables and values associated with the attribute of an entity for constructing an adaptation design for a transformation of the models. For example, we can add a variable to the attribute of an agent to signify what kind of functions it provides to what process, and use this variable of the attribute to check the connection with the connectors for verifying the consistency of the information. We will give an example of this point in chapter 5.
2.2 RADs (Role Activity Diagrams)

There are several ways of describing formally Role Activity Diagrams. Here we choose to examine Role Activity Diagrams (RAD) from [Ould 95]. In RADs, the process models are represented in the form of Role Activity Diagrams. A RAD shows the roles, their component activities and their interactions, together with external events [Ould, 95]. It is intended to capture the possible behaviour of a process in an organization (from a viewpoint of role abstraction), and it has the ability to simulate a process in the meanwhile. The main points of interest in RADs are as follows:

- Role, Role type: A role in RADs is a kind of pattern for a procedure, abstracted from the business processes of the enterprise, and is composed of a set of activities to achieve a goal. For example, in the process of publishing a book, we might identify roles such as Designing, Editing, Planning, Producing, Marketing, and so on. Roles can take many forms, such as a unique functional group, a unique functional position, a job title, a replicated functional group, a replicated functional position, and an abstraction (e.g. Progress Chasing). [Ould, 95] In RADs, a business is depicted in terms of roles, that are types or a class of role, and any roles are independent of other roles. A role may communicate with other roles. Therefore, RADs conceptual models represent the business processes of an enterprise in a component-based style. Its unique viewpoint comes from the consolidation of sequential activities into the role framework, which could be adjusted to include a group, a position, a job, a person, a project, or subset of the tasks. From this viewpoint, having its responsibility and structure in organization, the role defined in RADs appropriately represents the conventional operation of the business of the enterprise.
• Activities and states: In the process of placing a book order we might find activities such as Enquire about a Book, Place an Order, Confirm the order, Pay the fees, Obtain the book. In RADs, “activities are what actors do as ‘individuals’ in their role.” [Ould, 95]. Like roles, activities are types that have instances. An activity is defined by the following characteristics:

  o what makes it start and what makes it stop;

  o what state the world is in when it starts;

  o what state the world is in when it stops.

A RAD uses the triggering condition, the pre-condition and the post-condition to create and capture the states of an activity instance. In order to capture the materials involved in an activity, a RAD uses input and output entities. Furthermore, in an activity, it also builds the properties: sequence, condition and concurrence to capture the relationship between activities. The purpose of these properties is to represent various patterns of the activities that connect activities in different ways.

• Interaction: All activities and interaction take place within roles. Roles use interaction to communicate to each other. In the Book-Enquiry-Order process, the role of Finding and Order a Book needs to interact with the role of Stock Enquiry and the role of Payment Check, and the latter two roles can be initiated by the previous role. To complete a business process, a RAD composes a group of role instances with interaction among them. Therefore, when adding a new role, we should consider the requirements for deriving the interaction to complete the process goals.
• Process goals: Each role diagram defines a point at which a terminating state has been achieved. If the goal is a composite goal, it can be recognised as a combination of states in a number of roles across the process. Thus, we can roughly refer to a role as the composition of the parts (threads, activities, state indication and thread switch controls) in a structured process description, such as condition and sequence among activities. Some business processes need several roles together to achieve the goal. From the perspective of a process, although a role has its meaning in its responsibility for the activities, we can view a role as a function for completing the process goal. How to coordinate different functions to execute the business goal is the main concern of business modelling and business transformation modelling.

2.3 BSDM

BSDM Business System Development Method (BSDM) is a development method for building systems that support the business [IBM.a 92]. It uses “Business Mapping” as a kind of metaphor to describe the methods and rules to build business model. It uses the concept of entity to capture the fundamental element of the business and constructs the relationships between entities to represent a business process. Its overall structure is described in [Chen-Burger et al. 95]. The essential representational concepts are as follows:

• Entity and dependence: Entities are types, like roles in RAD. An entity type is a class of things that business needs to manage. Thus, entities could be generalized and classified into a hierarchical structure to represent an entity model of a business domain. In an entity model, link between entities is named dependence, which is used to describe that occurrences (instances) of some entities cannot come into existence unless the corresponding occurrences of certain
other entities (called parent entities) also exist [IBM.b 92]. Intuitively, BSDM entities try to capture the business elements and a certain relationship between them, such as figure 2.

Figure 2. Prices over Time. This describes how an entity diagram would allow for several product prices being held for separate periods. [IBM.b 92]

In order to represent the business behaviour “the price offered for products is available over a certain period”, we need to build “a period entity type” and “a product type entity type”, and then define the price “a product price entity type”. When activities happen, occurrences (instances) of entities are identified for updating the attributes of business data. Therefore, for embedding a business process, not only must the entities within the scope of the process be updated but also these dependent entities outside the scope.

- Process: The procedure BSDM uses to build a business model includes: identifying processes, establishing process scope (identify entities within a process), defining processes and defining attributes. The method of process description in BSDM is to consolidate a group of entities and relate them in certain procedure logic to execute the business activities, and also to define the entities resulted from a process in its procedure. BSDM uses structured English to represent business rules and process-entity matrix to capture the relation of entity with process (Figure 3, Figure 4). BSDM starts with
an entity diagram like the one in figure 2 and ends with process definitions like the one in figure 3. There are various intermediate stages not discussed here.

Rules:
Check existence of CUSTOMER ORDER
A. CUSTOMER ORDER does not exist:
   Take new CUSTOMER ORDER
B. CUSTOMER ORDER exists, and
   a. CUSTOMER ORDER Status closed
      Take new CUSTOMER ORDER
   b. CUSTOMER ORDER Status not closed
      1. For each PRODUCT TYPE required:
         If PRODUCT TYPE exists
            Add PRODUCT ON ORDER
               - Quantity required
               - Date added to order
         Otherwise
            Report mistaken order
      2. If at least one PRODUCT ON ORDER
         Change CUSTOMER ORDER
            - Status = pending

Figure 3. Process rule definition [IBMc 92]

Figure 4. Process-entity table for a university degree scenario [Robertson & Agusti99] (Labels in the cells describe relationships between entity and process: ‘*’ denoting the entity type is originated
by the process, ‘X’ denoting the entity type is changed as part of the primary purpose of the process, ‘>’ denoting the entity type is a strict perquisite for this process, and ‘—’ denoting the entity type is not a strict perquisite for the process.)

2.4 Representation and formalization of business models

Because models are now so ubiquitous, it is almost impossible to examine exhaustive lists of business models. However, there are general and deep concerns for us in the way the models are constructed to represent the intentional part of the reality, the structures of their representation, the aspect and degree of their representation, and the relation between formalisation and their structure. Here, we would like to discuss these points based on the concepts introduced above and complemented with other models or formal languages. Afterwards, we would like to introduce issues across the spectrum of knowledge representation, and accordingly we would introduce the possible strategies and procedures for our research. Then we would propose a concept of “the axis of a wheel” as the metaphor for our form of knowledge engineering.

The viewpoint of a business model. It is easy to discern the point of view held by the models above: ER from data abstraction, RAD from role activity abstraction and BSDM from entity abstraction. Thus, a conceptual model that tries to capture the concepts at this high level of abstraction has its particular viewpoint on the world. Its advantages and limitations also follow from its view. Although Eriksson Penker classifies four different views of a business that is blend with UML modelling, business vision, business process, business structure and business behaviour, usually the construction of business models are interwoven with changes across these views [Eriksson & Penker 00]. For example, RAD modelling could be used to consider all of the views, although it would not be ideal for all of them. A
conceptual model is motivated by its objectives and its construction will naturally be driven by these concerns. Besides, not only do there exist different viewpoints in business models but also the concepts and terminology in models are intermixed and inconsistent. When we try to represent the transformation of models from as-is to to-be under the influence of business networking, mostly we only pay attention to the part of the model which needs to change internal and the part which needs to be adjusted for the other model. An exchange protocol between different models would not be our concern in this research.

**The construction and structure of a business model.** A business model defines the important concepts used in a domain. The model establishes a vocabulary for all the concepts used in modelling, including its components, relationship, mapping rules and infrastructure, and even modelling procedure. For example, ER modelling uses entity type as a way of data abstraction. Constructing the relationship between entities, we build a model to describe a event or activity in the context of business, such as a Entity Relationship Model of Issuing Ticket System which is constructed by a group of entities: passenger, ticket, coupon, flight, airline route, airline and airport, and by relationships among them: ticket on passenger, coupon on flight, flight of airline route, etc. Each entity has attributes, such as a ticket’s attribute: date of issue, fare, currency [Barker 89]. With the components, structure and rules appropriate to this structure, we can analyse a model and represent in a formal language and enhance its power beyond the ability of notations and operations instituted in the model, such as to verify the model. But not every conceptual modelling method can be formalised without redefining the concepts and clarifying the rules by which the concepts operate to build a model. For example, even with a simple conceptual modelling method and domain, a modeller may lay out a sequence of actions and decisions, and the resulting model may be different from that of other modellers, without considering that there exist delicate differences between each modeller’s perception of the meaning of the
components. When we consider the concepts across several conceptual models, the problem becomes serious. For example, an input may be at the start of a process in one model but may be within the process in another model, or a start condition may be an external trigger in one model but an internal trigger in another model [Bock 99]. When embedded within a new process, we must consider how to adapt the model to function under the new requirements and constrains.

**The aspect and degree of the representation of a business model.**
Different business modelling notations are tuned for particular aspects of the descriptive problem but are less well able to cope with other aspects. For example, a RAD uses a “role type” concept to capture the activities in an enterprise and represent these in a model. When trying to introduce a new business process in an enterprise, if there is not a similar process in our existing model, we need to analyse the activities of the new business process, decompose and match the activities with roles, and construct an interaction among the roles to implement the new process. When we consider the transformation of business models, the scope must be limited, because designing a heuristic algorithm for this specific domain is difficult and may not be reusable in other domains.

Also, each conceptual modelling language has deficiencies. For example, the BSDM method provides no formal method for detailed process description. BSDM does help a modeller identify processes, establish process scope, define processes and define attribute processing based on entities, and defines the functions in entity to signify the temporal relationship with a process [IBM.c 92]. But it has no method to classify processes and define the relationship between them.

**The structure and formalisation of a model.** “The conceptual models that are constructed in a modern Knowledge Engineering process are typically defined using a combination of natural language and graphical
elements” [Harmelen 95]. Most high-level conceptual models including the models above with natural language and graphical notation are defined informally or semi-formally. Often the formal aspects of the language provide an efficient and convenient way of keeping track of the many relationships and variables operating in a modelling domain [Aben 94] [Harmelen 95]. Although a conceptual model could be independent of any particular implementation formalism, usually there exists a relationship between conceptual models and the language used to implement them. In our research, we will incrementally structure the problem description, translate the specification into formal interpretation, construct formal model in iteration and explore behaviours of models.

Secondly, because formal and executable specification forces a detailed formulation of the modelling, we could use this to check the structuring, hierarchization and modularisation of the specification in modelling, and then reengineer visual representation of the problem, for instance by inventing informal diagrammatic and textual notation, to help us understand the match of the formal model with the domain. Furthermore, we can use this knowledge to evaluate the other methods to help reengineering our representation of our conceptual model, such as adjusting the methods of PIF or PSL to reengineer the conceptual model based on BSDM. For us, to formalize the specification of a conceptual model not only gives more opportunities to validate and verify it in a domain but also helps us to reengineer a model properly.

Thirdly, We will use logic to describe the problem formally. When using logic to small parts of problems, [Robertson & Agusti99] points up “Given that so many of the factors which determine the success of modelling are unquantifiable, why bother to use logic? Ironically, it is the very fact that the problem is so ill formed which makes logic, when appropriately applied, such a valuable tool.” Following this argument, [Robertson & Agusti99] lists six main features that are required of conceptual models as follows:
- **Idealised** It does not pretend a perfect mirror of reality.
- **Germane** It should represent all concepts which are strictly relevant to the problem.
- **Precise** It should be possible for those other than the designer of a model to derive predicated consequences from it in a methodical way.
- **Arguable** It should be possible for those other than the designer to draw consequence from the model which the designer may not have anticipated.
- **Traceable** The origin of each element of the model can be justified in terms of its relationship to the problem itself or to a feature of some other model which is related to the problem.
- **Communicable** It should be possible, at acceptable cost, to explain the model to those in the domain to which it applies.
- **Methodical** Built in a stepwise manner which allows the style of construction to be inspected by other modellers.

By representing the problem descriptions formally, we give more attention to satisfying these requirements of conceptual modelling for those aspects of problems.

**The methods of our conceptual modelling.** For us, in addition to helping the analysis of required expertise of KSB, formal methods also contribute to tracking, validating and refining the required knowledge of a conceptual model. To connect to this more general use of methods, we use logic as our modelling language, and Horn-clauses will be a basic language in our research.
Figure 5. Five aspects of knowledge engineering continually involved with our conceptual modelling.

Following our argument above, we introduce our research in three parts. Firstly, in figure 5, we emphasize five aspects of knowledge engineering continually cultivated by conceptual modelling. For instance, in the representation level, we use informal language or graphic notation to represent our conceptual model, and after it is structured, formalized and given a mechanism for drawing inferences from it we may find it has deficiencies in capturing the concepts we intended to represent. We will then reengineer the representation level of the conceptual model. The spiral line represents the idea that this is a spiral development model, where other aspects of knowledge engineering will be reengineered after the previous aspect is refined.

Secondly, we use structural correspondence as the core concept to check the relationships among “the conceptual model”, “the construction of formal specification” and “informal stated requirements” (Figure 6). After we use a formal language to formalize the specification of the conceptual model, in addition to validating it against the requirements, we also need to check
whether the conceptual model captures the problem properly and whether it satisfies the requirements of the problem or not. After validating against the requirements, the construction of a formal specification translated from the conceptual model also endows the conceptual model with the ability to be more precise, traceable and further refined.

![Diagram](image)

*Figure 6. Structural correspondence among the conceptual model, the formal specification and requirements*

If we take the first theme as a viewpoint from the process of our knowledge engineering and the second theme as a constraint on our style of knowledge engineering, we would like introduce a third theme as the goal of our knowledge engineering: to explore a system which has a ability to transform a business model to a new model under the requirements of business operation involved in a business networking context. Roughly speaking, the structure of our KBS will be divided into three components: formal representation, inference engineering and translation rules (*figure 7*). Having used different types of scenarios to build our conceptual models, formalise
them and track the problems in the domain, we would abstract a higher conceptual model to integrate the ability to deal with the problems in a system. The actions of decomposition and composition may be repeated across the structure of the model to provide a sequence of revisions to its overall structure.

*Figure 7* The structure of our system
Chapter 3

REPRESENTING TRANSFORMATION OF BUSINESS MODELS

“Entrepreneurs are looking for a business model of the information age which will allow them to identify real options and to assess the consequences of their decisions.” [Osterle, 00, p19] In our research, we do not assume our modeling accurately specifies whole real systems, but apply logic to part of transformational modeling of business models. We will introduce the domain of our research: the transformation of business operations in a business network. Then we will describe basic ideas about the representation of change of business model and basic notation and semantics we would like to use.

3.1 Business Networking

Because of business networking in Internet context, the actions and strategies to coordinate information, resources and processes (or functions) within and across companies happens continually. Although Business Process Reengineering (BPR) has been invented to reengineer the system of business processes in a company or a chain of industries, most companies are concerned to transform their part of a business in a convenient way that keeps their business advantages or empowers their competitiveness in the market without following strictly the analysis and procedure of BPR. However, when a company purchases different products to satisfy their new requirements, such as providing product information, procurement systems, and credit checking, it needs to adjust these new models to its conventional business models, which may involve different processes within organization. In this research, our intention is to deepen understanding of transformation of business models in a lightweight study, which means we will constrain

21
our modelling only to the most closely related part of a new model and of an existing model, but not to the whole business process. The aim is to provide a roadmap for a business to evaluate its operations within and across companies. In this section, we would like to picture the possible operations of business in the environment of business networking.

*Figure 8 Enterprise in Business Networks. [Osterle et al. 00]*

Imagine that, operating in Internet environment, a company decides to produce an “e-business bus” to provide and get the information, resources and processes to extend to business networking across organizational units, companies, industry and markets. Osterle introduced the framework of the business model of the information age as *figure 8*, and describes the characteristics of business involving in Internet environment: coverage, partnering, critical mass, position, focusing, process efficiency and networkability [Osterle et al. 00]. For instance, in the case of *figure 8*, a car dealer may take on the task of helping the customer with car selection,
financing, tax payment, navigation and resale, etc. A company must confront the problem of the coordination and prospects of its business in this environment, in which the reassembly or disassembly of the processes across companies evolves. Therefore, a company must adjust its processes in the interorganization to sharpen its relationship and responsibility with the processes driving the new organization. As an example (see figure 9), amazon.com integrates multiple catalogues from different publishers into a single information source to customer and cooperates with regional booksellers to deliver books, and builds a user profile to allow customized interaction with customer. In this case, we summarise two main points: firstly, new processes are created to achieve new configured goals; secondly, following new processes across organization and customer, a company must adjust its processes in the organization after reconfigured its relationships and interactions with organizations and customers.

* Books are only delivered if sold: no more returns
* Market info through advance sales

* Cooperation with regional booksellers
* Market research
* User profile: allows proactive actions
* Associates program

* User friendliness
* Customer-specific
* Completeness of service
* Orientation towards customer problem

**Figure 9. Networking at Amazon.com [Osterle et al. 00]**

In a business network, a company evolves through its adjustment of its business processes, such as process outsourcing, combining processes, building new processes, etc. A modelling method to be developed under
these circumstances must assist a company to figure out its problem and transform its business model.

3.2 The representation of the change in the business model

![Diagram](image)

*Figure 10* From As-is model to To-be model

When trying to represent the transformation of a business model, we first need to identify the conventional model (As-is) and the new model (To-be). Then we need to compare the two models and to consider how the components of the two models can be matched, also figuring out how the transformation can be made, and how to validate and certificate to changes in the new model. Before further exploring the detail of the components for the capability of transformation, here we create one of the possible prototypes of transformation between the As-is model and the To-be model and describe the way making the transformation (*Figure 10*). The diagram (a) composed of processes P1, P2 and P3 in sequence, which is planned for transforming into the diagram (b). As the diagrams show there are two possible ways for the
transformations, that is assumed to be satisfied with business needs here: either we combine three processes into P1′, or two processes into P1′ with no relation to the time sequence with P3. Of course, this is only one example of process transformation. Later we consider another in more detail.

On the whole, the system of transformation modelling must have the ability to check whether the new models can function to achieve the goals, whether they will lose some of the functions, whether the links, from agents or processes, of old processes are well adjusted into new processes, etc. It must provide reports about constraints and requirements of the To-be model transformation, and the information about comparison between the new models and the capability using this information to refine the structure of the new models in meeting a stepwise transformation or a leap transformation. Thus, in this case, we need to check the constraints and requirements of combining P1, P2 and P3 into P1′, which may refer to the functions, capabilities, action specification and conditions, and links to the people, agents, datastore or the other processes. Afterwards, some of these requirements may be cancelled, some may be substituted or modified to fit into P1′, and some are adding. Not only does P1′ has its ability to execute the functions endued with the new responsibility, but also must be qualified for the requirements of embedding into the whole business model.

With a simple transformation description of a business model, it is reasonable to continually explore the structure of transformational modelling in detail and formalize the mechanism. But before doing that, we need to define the components of the model formally in order to express its behaviour.
3.3 Basic notation and semantics

With the intention of representing the transformation of the model, we first need to provide the basic notation and semantics that are suitable for describing business activities in general, and for describing the activities of the networking business in particular.

Certain concepts recur among the languages and notations of business modelling. Firstly, business activities, such as processes and executors etc. are classified and connected to others; for example, when a process is erased or moved, there is a kind of knowledge of what other processes or activities are related to it, so any links to and from the process can be updated. Secondly, in most cases the business activities are directly accessed and processed by an actor or machine, and the statuses are recorded, so after an actor is removed from the activity and his task absorbed by an agent. In addition, we need a notation for describing temporal aspects of the model.

Thus, with the ideas presented above, we use of three kinds of notations: nodes, links, and addition links. We can use this notation to represent the translation of business model.

1. Nodes

   a. Persons/agents/actors: We use agent, whether person or machine, to refer to an executor or a participant of processes, that could be sales person, customer or credit confirmation agent etc. We use an oval shape 🔄 to be an agent symbol. Each agent has an ability and responsibility to act on processes assigned to it.

   b. Datastore: To conceptualise the data-type in the model, which is the output from the process, the input to the process, and can be kept for a short period or for a long-
term period in a flexible or a strict form, we use “datastore” to substitute database. We use a cylinder as a datastore symbol.

c. Process: Generally speaking, process is an easy vague term that can be used in different ways. Here we define the process as the meaning more close to function and use a rectangle to indicate a process. We use figure 11 to clarify the idea of this sort of process. In the diagram, agent 1 makes a request and agent 2 responds it and produce a kind of information. This request-response process is constructed by the requirements of two agents and its function dealing with input and output (request-response) actions. Viewing this as a kind of the logic relation, the conceptual entity is defined as a process. We write the following expression to denote this ideas:

\[ \text{process(process\_name, [Inputs],[Outputs], process\_mechanism)} \]

![Figure 11](image) Request-response process

2. Links

a. Links between processes: There is relation between processes. We use an arrow line with broad width
to express a from-to single relation between processes. If a process has a return relation with another process, we use another from-to arrow line with opposite direction to represent.

b. Links between process and agent or datastore: Both of agent and datastore have ability to act on a process and a process also produces output to a datastore. We use an arrow line to capture the relation between them.

3. Additional Nodes and Links

a. Simulation operator node: Two symbol are used to describe the begin point and ending point of the process simulation.

Start  End

b. Simulation operator link: An arrow line with broad width denotes the link between the node of simulation operator node and process.

We can apply this notation with the semantics mentioned above to represent the transformation of the business model, that the diagram of the business model can be changed to another diagram representing new business model. If the models are transformed by hand to a different representation, the type of diagram and means of customisations provided are limited, in addition, high cost and high error chance are normally associated with them. Hence, after making the specification of the notations and its semantics, we describe the logic aspect for designing the transformation mechanism. The notation above is for the purpose of example. Real business modelling languages can be more complex.
Chapter 4

Automatic Transformation of the Business Models

After defining the notation and semantics at chapter 3, we give two simple examples to describe how to formalize and automate the transformation of the business models, according with our request.

When a company tries to shift its business to the internet, usually its first concern is how to make its products or services directly available for customers, with the result that customers can browse the catalogue at any time, checking the information such as products, specification, price, promotion, service, etc. To exemplify this we give two sample scenarios in which transformation is automated.

The first scenario is that a customer requests a book from bookstore but without assistance from sales person, so we need to reconstruct the model without the role of the sales person.

After a company shifts its business to the Internet, the barrier of the market entry is reduced. Just when anyone can enter a market or industry, the span of a market is being changed. For example, on the net, a hotel, not only provides its service to a travel agent but also directly to individual customer. On the net, a company, in order to service its customers may expand the market beyond its specialty, inducing new products from the other industries. In the face of these various situations, a company imperatively expands its abilities to do business with different kind of providers, intermediators and consumers, and fulfil diversified requirements from its customer. When a company provides these products or services to a customer, it means a customer has more options in making his decision. We give as a scenario a company modifying its business model to include the
ability for its customer doing business with different contractors, as our second example.

4.1 Example 1: The automation transformation of role replacement

Figure 12 A simplified book-order conceptual model

*Figure* 12 describes a customer making a request for a book at the bookstore, in the case that sales person needs to help to deal with the request process. After the request is matched with the database of the book bookstore, the data of requested book is obtained, and sales person use the data with the store data of the book to fill the order (Fill order process), as a result of producing the data of published book order. Then the sales person needs to assure that customer confirms the book order, and that will result in the data of confirmed book order (and after that the data will be delivered to dispatch department). With this simplified example, we use the predicate to represent the process as:

\[
\text{process([Inputs], [Outputs], Process\_Name).}
\]
Then we instantiate the processes in the *Figure 12* as follows:

```
process([customer,salesperson], [requested_book], request_book).
```

```
process([request_book,salesperson,book_sale_data],
        [published_book_order], fill_order).
```

```
process([request_book,salesperson,book_sale_data],
        [confirmed_book_order], confirm_order).
```

Now, let’s turn to what will happened when the agent of a sales person is not present in the whole book request-order process. Because a sales person no longer assists the request process, customer should carry on the tasks of each step alone before finishing the whole process. In this simplified case, the model transformed is easy to be visualized for us: all links from sales person are substituted by the links from customer. We try to formally represent the mechanism to execute the transformation.

The first model M1 is planned and changed to the second model M2. In this transformation, the entity E1 (sales person) in the model M1 should be cancelled and its links should also be replaced by the entity E2 (customer), which is:

```
transform(replace_role(E1, E2), M1, M2) ←
contains_entity(M1, E1) ∧
contains_entity(M2, E2) ∧
replace_process_role(E1, E2, M1, M2).
```

In the replacement, we need to figure out the behaviours of transformation to include all possible situations, if both sale person and customer relate to the process, then after transformation, we keep the link from the customer, if not we need to create the link to process for customer.
The first condition is that the processes in a model are linked to both entities E1 and E2. We design a mechanism to substitute E1 entity with E2 entity recursively over the processes. Firstly, we remove the link (Inputs) from the model, remove E1 from the link, and then put the link back to the model. We represent a design as follows:

\[
\text{replace\_process\_role}(E1, E2, M1, M2) \leftarrow \\
\quad \text{process}(\text{Inputs}, \text{Outputs}, \text{Process\_Name}) \land \\
\quad E1 \in \text{Inputs} \land \\
\quad E2 \in \text{Outputs} \land \\
\quad \text{remove}(\text{process}(\text{Inputs}, \text{Outputs}, \text{Process\_Name}), M1, M3) \land \\
\quad \text{remove}(E1, \text{Inputs}, \text{Inputs1}) \land \\
\quad \text{append}([\text{process}\text{(Inputs1, Outputs, Name)}], M3, M4) \land \\
\quad \text{replace\_process\_role}(E1, E2, M4, M2).
\]

The second condition is that the processes is linked to E1 entity but not E2. Therefore, after checking the link related to E1 but not E2, we remove the link (Inputs) from the model, remove E1 from the link, append E2 to the link, and then put the link back to the model.

\[
\text{replace\_process\_role}(E1, E2, M1, M2) \leftarrow \\
\quad \text{process}(\text{Inputs}, \text{Outputs}, \text{Process\_Name}) \in M1 \land \\
\quad E1 \in \text{Inputs} \land \\
\quad \neg E2 \in \text{Inputs} \land \\
\quad \text{remove}(\text{process}(\text{Inputs}, \text{Outputs}, \text{Process\_Name}), M1, M3) \land \\
\quad \text{remove}(E1, \text{Inputs}, \text{Inputs1}) \land \\
\quad \text{append}([E2, \text{Inputs1}, \text{Inputs2}) \land \\
\quad \text{append}([\text{process}(\text{Inputs2}, \text{Outputs}, \text{Process\_Name})], M3, M4) \land \\
\quad \text{remove\_process\_role}(E1, E2, M4, M2).
\]
The third rule states if there is no E1 in the model, the goal is achieved and the execution of transformation is finished.

\[
\text{replace_process_role}(E1,E2,M,M) \leftarrow \\
\neg (\text{process}(\text{Inputs}, \text{Outputs}, \text{Process\_Name}) \in M \land E1 \in \text{Inputs}).
\]

In condition one and two, all processes in the model M1 will be checked exhaustively.

Then we can formulate the goal, in the case, which is the requirement about a substitution of the agent of E1 with E2, to form a new model M2.

\[
\text{?- transform}\left(\text{replace\_role}(\text{Role1}, \text{Role2}), M1, M2)\right).
\]

*Figure 13* A book-order conceptual model without assistance from sales person

As a result of the transformation of the business model, the new model will be represented as follows and visualized as *figure 13* (This Prolog program is defined in the Appendix):
process( [customer], [requested_book], request),
process( [request_book, customer, book_sale_data],
    [published_book_order], fill_order),
process( [request_book, customer, book_sale_data],
    [confirmed_book_order], confirm_order).

4.2 Example 2: The automation transformation of multi-agent replacement

Figure 14(a) price-quote to a contractor

14(b) price-quote to multiple contractors

Figure 14(a) describes a customer quoting a price for a piece of goods from a contractor. As a company plans to introduce different contractors to customer, it needs a new model in which the customer can do business with different kind of contracts rather than only one, requiring it to reconstruct its business model as figure 14(b). We need to represent the as-is model and formalise a mechanism to do automatic transformation from the as-is model (a) to the to-be model (b). We present the first model as:

process( [InputC, InputP], [Outputs], Process_Name)

34
We instantiate it as:

\[
\text{process( [customer, contractor], [quoted\_price], quote)}
\]

In the new model, an agent will interact with multiple agents, so a contractor will be substituted with multiple agents — contractors. We make a generalization about a possible number of agents InputP as a list [Input1, Input2,..,Inputn], so a request for replacing a Input with a list [Input1,Input2,..,Inputn] can be represented as:

\[
|\text{?- transform( replace\_multi\_role([ Input,[ Input1,Input2,..,Inputn]], M1, M2).}|
\]

Then we can formalize the mechanism of automatic transformation as:

\[
\text{transform(replace\_multi\_role(E1, [P1,..,Pn]), M1, M2)←}\]
\[
\text{contains\_entity( M1, E1) ∧}
\]
\[
\text{contains\_entity( M2, [P1,..,Pn]) ∧}
\]
\[
\text{replace\_process\_multi\_role( E1, [P1,..,Pn], M1, M2).}
\]

\[
\text{replace\_process\_multi\_role(E1, [P1,..,Pn], M1, M2)←}\]
\[
\text{process( Inputs, Outputs, Process\_Name) ∧}
\]
\[
E1\in\text{Inputs} ∧
\]
\[
\text{remove( process( Inputs, Outputs, Process\_Name), M1, M3) ∧}
\]
\[
\text{remove( E1, Inputs, Inputs1) ∧}
\]
\[
\text{append( [P1,..,Pn], Input1, Input2) ∧}
\]
\[
\text{append( [process( Inputs2, Outputs, Process\_Name)], M3, M4) ∧}
\]
\[
\text{replace\_process\_multi\_role( E1, [P1..Pn], M4, M2).}
\]

After we execute this mechanism, the new model will be represented as:

\[
\text{process([customer,contractor1,contractor2,..,contractorN],}
\]
\[
[quoted\_price], quote).
\]

35
Thus for designing an inference system, it will be very helpful to make a generalization and classification of the patterns about the transformation of business model and design its mechanism for automation.

4.3 Support for Reasoning Ability in the Transformation of Business models

4.3.1 Type of transformation

Although the example above represents only a transformation of a business model related to agents, we also need to survey another two aspects of the transformation: process and datastore. When dealing with the transformation related to process, we need to check and update the link with the agents and datastores to maintain or update the functions. Furthermore, when we deal with transformations that combine different types of entities, such as cancelling a process and rearranging the role to the appropriate process, the situation becomes more complex.

Even though we temporarily ignore the effects of the entity constraints, being important in reality of business modelling, the transformation of a business models mixing with different types of entity is still beyond our recent discussion. But let’s make an assumption about possible types of transformation as figure 15 and briefly discuss it.
From our definition of components of the business model, if we assume a transformation only happens on each kind of entity, there will be three types of transformation: agent transformation, process transformation and datastore transformation. Moreover, we can classify the sub-types of transformation in terms of agent entity; such as cancel an agent and substitute with another agent, combine two agents into one agent, reassign a part of job held by an agent to another agent, and so on. Equally, we can also consider building the sub-types in terms of process entity; such as remove a process, combine two processes, add a new process between two processes, etc; and define the sub-types in terms of datastore, such as substitute a datastore with a datastore, remove a datastore, divide the job of a datastore to a new datastore, etc. Furthermore, it is also possible to make a generalization and refer to this as a subtype of transformation behaviour from the aspect of the behaviour acting on these entities, such as moving out, moving in, combining, separating, etc. By these classifications, we will represent the patterns of transformation and approach an infrastructure for designing an automation system of the transformation of business models.
4.3.2 Parameterising the reasoning mechanism

By observing the formulation of the transformation on the examples above, we can tell that there exist similar structures in the representation. This may enable us to structure the reasoning representation and build up an inference system using parameterisable components. In our examples, it is easy to combine the two representations into one: we just use replace_process_role instead of replace_multi_role, and take E1 as a multiple input list. But if we need to give further condition to the representation, we may state one of possible representation as:

\[
\text{transform(replace\_role(E1, E2), M1, M2, Condition) } \leftarrow \\
\text{contains\_entity(M1, E1) } \land \\
\text{contains\_entity(M2, E2) } \land \\
\text{replace\_process\_role(E1, E2, M1, M2, Condition).}
\]

At the next step, using Condition as an option selector, we can initiate some functions or disable some functions that are defined in predicates calculus. Thus, after classifying different types and subtypes of transformation, we expect to build the reasoning framework on a transformation in terms of the structure of parameterisable components. We can refine the formulation of “replace_process_role” and “Condition” and make our reasoning mechanism parameterised into component structure. For example, we may generalize different types replacement and represent it as Action_Name, and in the mean while input Condition to constraint on the behaviour of replacement in the type:

\[
\text{transform(Action\_Name(E1, E2), M1, M2, Condition)}
\]

Thus the requests of two examples above, with no condition, can be represented respectively as:

38
transform( replace_role( customer,salesperson),M1,M2,[ ]) 

transform(replace_multi_role(contractor, 
    [contractor1,contractor2,..,contractorN],[],M1,M2) 

Having explored more examples from the domain, we may abstract characteristics that are common and structuralize them to contribute a framework for building parameterisable components of the reasoning mechanism.
Chapter 5

Reasoning about Transformation of Business Models (A More Realistic Example)

Following our discussion of the formalization and automation of transformation of business models, we now give a more realistic example of representing a transformation of business models and consider its possible logic structure and theory. When the business changes its processes or practices, this will be reflected in the composition, interaction and function of the entities. What kind of knowledge is needed by an engineer to define the problem and to track the process of change, and what kind of rules will be used to deal with the problem. In this chapter, we will take a scenario with simplified components, properties and structure of entities to formulate the skeleton of a conceptual model for track the problem of transformation of business models in more detail.

5.1 An example - book order from an Internet bookstore

Now, we look at part of our processes in more detail in figure 16. We make the entire process a stream, which is chained from each process to another process. Beginning with the “start” operator, the “receive customer enquiry details” process receives the customer’s request with the assistance from sales agent, then the process flows to the next process “search stock for customer order”, in which stock control agent will check book datastore to find the required book data, and subsequently the next process goes to the “provide stock search result” process providing the information to customer through sales agent, and continues to the next, etc., until the process arrives to the last “dispatch customer order” process and is finished at the “end” operator.
A main stream, denoted with a chain of processes, forms the most important aspect of this business model: the structure of the processes becomes the main skeleton, which organizes the other entities, agent and datastore. In this model, agent and datastore are linked to the process to satisfy the requirement of the process, without changing the timing constrain between agent, datastore and process. After defining the attribute of the process, we track the procedure in which agent or datastore related to the process. For example, the stock control agent uses the book datastore to complete the process ”search stock for customer order”, in which the book datastore reacts to an inquiry from the stock control agent. Taking another example, in the process “produce order dispatch info + receive customer confirmation”, after a sales agent receive customer confirmation, the datastore “completed order data” will be updated. As the examples show there exists a more subtle relationship between the datastore and process entity that needs to be captured, classified and formalized in our mechanism for a transformation of business models.

Notice that for simplicity we do not consider in our model the question of what happens if the sequence of processes is interrupted. This would have been possible by adding more elements to the model, branching from the main skeleton but this would complicate the example.

Figure 16 describes our book order model before engineering into the Internet Business. When we cancel the sales agent, all links from the sales agent disappear. If we do not wish to cancel the functions, one agent or some of the agents will serve the functions to different processes that originally are served by the agent cancelled now. For instance, in this case, the function in the process “produce order dispatch info” will need to be updated by alternative means: either we use the existent agent or create a new agent, such as adding the function to the datastore “completed order data” or adding a datastore “produced order dispatch info”.

41
Figure 16 A book-order conceptual model
5.2 Attributes of the entities in the business model

Following the description above, our transformation design mostly relates to the specification of the attribute in the entity. Before surveying more cases, we assume a possible specification about attribute on the discussion above. There are three kinds of entities, process, agent and datastore, which we will discuss below. How the attributes can be properly defined is based on the specification of problems, and it will need more experiments to verify, validate and refine the specification of attributes in our system. Initially, we loosely define the attributes in these entities and survey their effect related our design.

5.2.1 Process specification:

Because a process follows another process, a process must recognize its predecessors and successors. Moreover a process will need to recognize when to initiate and in what condition. Sometimes a process does not necessarily need to be executed, unless it is a triggered from an agent or a process. Assuming an agent with appropriate right can execute a process, we need to assure that this process has the property for checking the authority of an agent to act on it. Besides, we need to build the slot for action condition, which help us define the condition for process to interact with agent. For example, we put “if $>1,000 then need to get authority of agent > 7” as the action condition of the process “receive_customer_order_request”.

We describe the process specification as follows with an example:

```
process_name: search_stock_for_customer_order
precondition: exits(receive_customer_enquiry)
postcondition: exists(search_stock_request)
preceeding_process: receive_customer_enquiry_detail
 succeeding_process: provide_stock_search_result
```
Six categories of variables we assume in the property of a process above are name, condition (pre/post), sequence (preceding/succeeding), tasks and data assigned (agent/datastore), trigger, action (action/action condition) and criticality of authority.

5.2.2 Agent specification

Each agent, machine or actor, has its responsibility and capability. When we build a new relation between an agent and a process, that means this agent must has the ability and authority to participate in the process, and after we build up this new relation, this agent will expand its responsibilities (the jobs).

We describe the property of the agent as:

    agent_name: stock_control_agent
    responsibility: initiate the book_datastore to begin a search
    capability: search_stock
    requirements: a general computer knowledge, database search skill
    criticality of authority (1-10): 7

We represent these variables above to capture the possible behaviour of agent. For example in case of replacing an agent with the other agent, we need to build sort of variables to check what kind of requirements an agent need to replace the other agent and whether an agent has capability to replace the other agent, such as capability variable and requirement
variable above. An agent may be responsible for several tasks interacted with several processes. Thus when a replacement occurs and all tasks are transferred to another agent or the other agents, we need to check and assure each task is transfer properly and legally.

5.2.3 Datastore specification

In a datastore, whether it supplies data to process or it is updated data by process, because an agent acts on it through a process, it must have its responsibility and level of authority. We describe the property of the datastore as:

```
datastore_name: book_datastore
responsibility: result_book_store_data,
             input_book_store_data
capability: result_book_store_data
criticality of authority (1-10): 7
```

We use the responsibility variable to represent the functions datastore provides to processes, and the capability variable to represent its potential functions. We use these variable to consider constrains of a transformation of models in terms of datastores. For example if the capability variable of a datastore includes the values in the capability of the other datastore, the previous one has higher priority to substitute the later. When we use the previous one to replace the later, the values of the responsibility variable of the later will be incorporated in that's of the previous one.
5.3 Use of properties during transformation

5.3.1 Constraints imposed by properties of processes

Here we give an example of constraints imposed by a property of a process. After we request to replace salesperson with customer, the transformation mechanism must examine the authority_value of each process in which this replacement occurs before it executes each replacement. We represent this constraint checking as:

\[
\text{replace\_process\_role}(E1, E2, M1, M2) \leftarrow \\
\text{process}(\text{Inputs, Outputs, Process\_Name}) \land \\
E1 \in \text{Inputs} \land \\
E2 \in \text{Outputs} \land \\
\text{check\_authority}(E1, E2, M1) \land \\
\text{remove}(\text{process}(\text{Inputs, Outputs, Process\_Name}), M1, M3) \land \\
\text{remove}(E1, \text{Inputs, Inputs1}) \land \\
\text{append}([\text{process}(\text{Inputs1, Outputs, Name}], M3, M4) \land \\
\text{replace\_process\_role}(E1, E2, M4, M2).
\]

(The condition one: both E1 and E2 link to the same process)

\[
\text{check\_authority}(E1, E2, M1) \leftarrow \\
\text{abstract}(\text{process(\text{process\_name,.. process\_authority}, M1)} \\
\text{agent(\text{agent\_name, .. agent\_authority})} \land \\
\text{process(\text{process\_name, .. process\_authority})} \land \\
\text{agent\_authority} \geq \text{process\_authority}.
\]

Before we replace the agent with another, firstly we execute \text{check\_authority}. If it passes, the program will perform the replacement in this process, otherwise we would like the program send us a report about failure and execute the other check, replacement or action.

46
5.3.2 Constraints imposed by entities

Figure 17 A partial book-order conceptual model

Figure 18 A partial book-order conceptual model without the process “provide stock search result to customer”

In figure 17, if we want to take the “provide stock search result to customer” process out of the whole book order process, we need to consider what changes this entails to the remaining functions.
Here we give an example to illustrate and track the effect of removing a process. Because the “provide stock search result to customer” process is removed, if we want to keep the functions of it, we will need to move them to the other process/processes (such as figure 18). Then we will encounter the problems:

1. Which process/processes is best used to replace the functions?

2. What kind of the constraints from entities will affect the replacement?

In this case, because of the sequence of processes, we would choose the nearest processes to substitute the functions of the “provide stock search result to customer” process. The previous “search stock for customer order” process and the subsequent “receive customer order request” process correspond with our first requirement. However, before beginning a procedure of the function replacement, we need to check the authority of the process among the dismissed process, the previous process and the subsequent process. Because an agent with lower authority could not access a process with higher authority, normally a function will be incorporated in the process that agent could access to execute the function. As a result, the functions of process will be reassigned to the process with the equal or lower authority. In our example, we assume three processes have a same level authority so that either previous process or subsequent process will be possible candidates to be reassigned the functions.

After authority checking, we will check what functions the “provide stock search result to customer” process has and where them came from respectively. We need to find out which agents execute the tasks in the process and which datastore input data into process or retrieve data from the process.
To do this, we use the following form of expression to represent the relevant information:

\[ \text{agent\_tasks(Agent, responsibility(Tasks), Process)}. \]

This denotes that it is a responsibility of a given agent to perform the given tasks within a given process. Then we can design a mechanism to abstract the agent entities linked to the process deleted.

After finding out the agent responsible for the tasks in the process deleted, we need to figure out whether the agent links to the previous process or the subsequent process (because the rule stated above: the functions of a process is replaced in the way that the process is close to the process deleted). In this example, the “provide stock search result to customer” process, providing the result to the agent customer, also links to the subsequent “receive customer order request” process. Thus, the task in this process deleted could be moved to the subsequent process.

We now give two instances of agent\_task corresponding to the link in \textit{figure 17} from the customer entity to the “provide stock search result to customer” process and to the link from the customer entity to the “receive customer order request” process:

\[ \text{agent\_task( customer,} \]
\[ \text{ responsibility([provide\_stock\_search\_data,} \]
\[ \text{ receive\_stock\_search\_result]),} \]
\[ \text{ provide\_stock\_search\_result\_to\_customer).} \]

\[ \text{agent\_task( customer, responsibility(place\_order\_data),} \]
\[ \text{ receive\_customer\_order\_request).} \]
We then know that if “provide stock search result to customer” is removed, we must transfer the first set of responsibilities above to some other location.

After the “provide stock search result to customer” process is deleted, the “receive customer order request” process will absorb the new task from the previous process and its properties will be changed to:

process_name: receive_customer_order_request
precondition: exits(retrieve_stock_search_result)
postcondition: exists(receive_customer_order_request)
preceeding_process: search_stock_for_customer_order
succeeding_process: receive_customer_credit_relevancy
tasks_assign: provide_stock_search_data,

receive_stock_search_result, place_order_data
data_assign: provide_book_sales_data
trigger: none

criticality of authority (1-10): 7

The properties of the “search_stock_for_customer_order” process, prior to the deleted process, will also be changed as:

process_name: search_stock_for_customer_order
....
succeeding_process: receive_customer_order_request
....
5.4 Summary

From the illustration above, we could summarise the possible procedure about a transformation of business models:

- Checking the property of a process
- Comparing the authority level among the processes
- Finding the process with appropriate agent or datastore to reassign the functions from the substituted process
- Reassigning the tasks of the agent or the data of the datastore to the candidate process in their properties.
- Reassigning the tasks or the data from the dismissed process to the candidate process
- Changing the property of the processes in the sequence after a process is dismissed

Before becoming involved in a more complicated situation, we need to build a simplified mechanism with the ability described above, to execute the transformation and test it.

In order to design this sort of mechanism, we would need to examine the following points more deeply:

- To represent the event, activity, role, agent and procedure, etc., as function types
- To relate activities, events, roles, agents and procedures into processes, and construct their structures
- To verify the requirements of attributes in these entities according to the process pattern. Maybe we can abstract process types to constrain the specification of attribute.
• To define the rules to make a transformation

• To capture exceptions

For dealing with business models constructed from different modelling methods, do we not only need to abstract a higher level viewpoint than other modellers, but also build our system to have a capability to coordinate different requirements in our system to achieve a transformation goal. The problems involved will easily expand beyond the scope of our control. For pragmatic reasons, we will constrain our research to the specification of problems in the basis of prototype of system behaviours, and construct formal models incrementally in a way that conforms to a more general practice in a given domain, following the path of our argument above.
Chapter 6

Research to Date and Future Targets

6.1 Research to Date

Our research requires a balance between the technical constraints of model construction and the requirements of business modelling. On one hand, we have explored several relevant techniques for transforming business models with basic notation and semantics on selected business scenarios. Based on this initial notation, we plan to use a more sophisticated and rich notation to describe standard business scenario, which enables our models to be concise, germane, and easier to trace. On the other hand, because the problems involved in transforming of business models are very complex, ideally, this transformation mechanism would need to be robust for application in real scenarios.

It is not our intention to build another generic modelling language. Instead, our formal models are used to describe precisely key features of transformational problems in the business-modelling domain. We will choose the retail aspect of networking business as a specific domain to survey the problems of business model transformation.

We now return to the list of issues given in the introductory chapter, and summarise how these will be addressed in future work.

1. What are the characteristics of business networking and business models?

In business networking, new processes are created to achieve new configured goals, and are adjusted into the internal operation of the business under constraints from the existent business model. From the perspective of modelling, the main concern is to extend its capability beyond the existent operation model to satisfy requirements from operating with new processes.
2. What are the problems of business transformation modelling?

It is important to identify the scope of the problem domain and analyse activities involved in new processes occurring through the interaction between internal and external features of the organization. We must think of how new processes operate in the business, but with the requirements and constraints from the networked operation. The basic issue for a transformation modelling is to analyse what these requirements and constraints are, how they happen, which entities are responsible for them, what states of the system may occur because of them, and what formal methods could be used to capture and represent these factors.

3. What are the appropriate methods (and/or methodologies) for transforming business models? How will they be used?

Two key areas are targeted: the business modelling methods which capture standard and common business scenarios, and formal logic methods which represent those business models. We need to further investigate and identify those methods that are appropriate in representing requirements and constraints in a business model. In particular, we select those methods that have the ability to identity entity, relationship, interactions between business entities, business structure, and activities (or business processes) that create, modify and make use of them. It is important to identify and specify a declarative language to represent the above business problem in a concise and unambiguous format. A declarative language of formal representation of the model description provides an appropriate basis for exploring automatic mechanisms and rules of business model transformation in a specific domain.

4. Why choose logic methods to handle issues such as automatic model transformation?

Logic helps us to clarify issues, trace changes, and implement business models. The experiment given in chapter 4 is a good example, although the scenario was relatively simple. Not only does logic help us to refine our conceptual model by improving our ability in isolating and observing the structural relationships, but it also provides a
convenient basis for investigating automatic reasoning mechanisms in a domain.

5. Is it feasible to automat the transformation of business models?

As described previously in this proposal, we found that automation of the transformation of business model is possible. However, if we consider making automation useful to the extent for exploring the problems in real applications, we need to specify a declarative language of model description to be flexible to capture transformation requirements and to be consistent to represent the elements, their operation and rules involved in a specific domain.

With different business scripts of transformation problems in enterprise, in the future we intend to specify detailed problems in the retail aspect domain of networking business. In this domain, to identify the requirements and specification of transformation modelling, to build and validate our conceptual models of them with a declarative language, and to survey automatic mechanisms that are applied for them will help us fill in more concrete contents to answer the questions proposed above.

### 6.2 Future Targets

- Collect and analyse business scenarios of business transformation in the retail aspect of networking business
- Survey and select appropriate modelling methods (e.g. Function and Process Model, IDEF3)
- Build simple business models based on the selected modelling methods
- Investigate and select appropriate formal language (e.g. PIF, PSL)
- Specify a domain-specific formal language that is suitable for business model transformation
- Collect and represent business model transformation rules
- Survey automatic mechanisms for business model transformation
• Illustrate the use of transformation rules based on the above formal language

Following we present the timetable with the planned activities:

6.3 Likely Outcome

The proposed work should result in a detailed study of formal specifications of business model transformation and in the survey of development of an
approach to automatic mechanisms of transformation, based on the ideas described in chapter 5 and 6.

In order to demonstrate the viability of the proposed approach, a prototype of declarative language of model description should be explored and possible strategies to survey automatic mechanism for business model transformation should be specified.

The expected contributions of this research are as follows:

- This work aims to identify possible methods, modelling languages and procedures to support an approach of business model transformation in a specific domain. As a result, the requirements and specifications, and business modelling languages, qualified for representing business model transformation in a specific domain, should be identified.

- This work should provide formal methods used in a declarative language of modelling description to deal with problems occurred in a specific domain. Domain-specific methods of validation, verification and reusable architectures should be also addressed in this work.

- To our knowledge, many companies have adapted and coordinated new processes nowadays. Automation is an important issue in business model transformation where new specifications resulting from new requirements and adjusted into existent model have been brought forward. In order to identify the properties of automatic business model transformation, this work should explore possible reasoning mechanisms and rules applied with different requirements and constraints in our specific domain.
Appendix

Business model transformation

E1 Entity1
E2 Entity2

Process is composed of Entities and Datastores
process(Inputs, Outputs, Name)

Model is composed of many processes
M1 Model1
M2 Model2

Replace E1 with E2
replace_role(E1,E2)

Check E1 and E2 in M1
1. The condition one: both E1 and E2 exist in the same process.
2. The condition two: E1 exists in the process and E2 not.
3. The condition three: E1 does not exist in the process

transform(replace_role(E1,E2),M1,M2)←contains_entity(M1,E1) Λ
contains_entity(M2,E2) Λ
replace_process_role(E1,E2,M1,M2).

replace_process_role(E1,E2,M1,M2)←process(Inputs,Outputs,Name) ∈ M1 Λ
E1 ∈ Inputs Λ
E2 ∈ Inputs Λ
remove(process(Inputs,Outputs,Name),M1,M3) Λ
remove(E1,Inputs,Inputs1) Λ
append([(process(Inputs1,Outputs,Name)],M3,M4) Λ
replace_process_role(E1,E2,M4,M2).

replace_process_role(E1,E2,M1,M2)←process(Inputs,Outputs,Name)?M1 Λ
E1 ∈ Inputs Λ
E2 ∈ Inputs Λ
remove(process(Inputs,Outputs,Name),M1,M3) Λ
remove(E1,Inputs,Inputs1) Λ
append([E2],Inputs1,Inputs2) Λ
append(process(Inputs2,Outputs,Name),M3,M4) Λ
replace_process_role(E1,E2,M4,M2).

replace_process_role(E1,E2,M,M) ← ¬ (process(Inputs,Outputs,Name) ∈ M Λ
E1 ∈ Inputs).
Sicstus Prolog Program

:-use_module(library(lists)).

transform(replace_role(E1, E2), M1, M2) :-
  contains_entity(M1, E1),
  contains_entity(M2, E2),
  replace_process_role(E1, E2, M1, M2).

contains_entity(M,E):-
  member([process(Inputs, Outputs, Name), M], Inputs).

replace_process_role(E1,E2,M1,M2):-
  member([process(Inputs, Outputs, Name), M1], Inputs),
  member(E1, Inputs),
  member(E2, Inputs),
  delete(M1, process(Inputs, Outputs, Name), M3),
  delete(Inputs, E1, Inputs1),
  append([process(Inputs1, Outputs, Name)], M3, M4),
  replace_process_role(E1, E2, M4, M2).

replace_process_role(E1,E2,M1,M2):-
  member([process(Inputs, Outputs, Name), M1], Inputs),
  member(E1, Inputs),
  delete(M1, process(Inputs, Outputs, Name), M3),
  delete(Inputs, E1, Inputs1),
  append([E2, Inputs1, Inputs2],
        [process(Inputs2, Outputs, Name)], M3, M4),
  replace_process_role(E1, E2, M4, M2).

replace_process_role(E1,_,E2,M,):-
  \+ (member([process(Inputs, Outputs, Name), M], Inputs)),
  member(E1, Inputs).

?- transform(replace_role(salesperson, customer),
  [process([customer, salesperson], [book_request], request),
   process([book_request, salesperson, book_sale_data], satisfied_book_order, order_filling),
   process([customer, salesperson, satisfied_book_order], [confirmed_book_order], confirmation)], M).
Bibliography


