TRANSLATING REQUIREMENTS SPECIFIED IN GOALS TO UML DIAGRAMS

by

FEI SUN

BBA, Shanghai Jiao Tong University, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE in

THE FACULTY OF GRADUATE STUDIES (Business Administration)

THE UNIVERSITY OF BRITISH COLUMBIA (Vancouver)

April, 2009 © Fei Sun, 2009
The concept of goal is an essential concept in requirements engineering (RE) field and has been widely adopted as an effective way to capture users' requirements. Goal-oriented requirements engineering receives substantial attention. On the other hand, information system must completely represent users' requirements in order to be considered as a success. Research has shown that one way to achieve this is to provide developers with system diagrams that fully capture business requirements.

However, few guidelines have been proposed in RE field to precisely map users' requirements (specified in goals) to systems analysis and design diagrams. To overcome this gap, we propose in this thesis a systematic and structured set of guidelines to translate users' requirements specified in goals to UML diagrams, which are considered as a formal modeling language used by practitioners. As the goal concept has been well accepted in requirements engineering (RE) to model business requirements, our methodology suggests a goal template approach to capture users' requirements in terms of goals. The goal templates approach and translation guidelines are illustrated using an auto dealership case study. The usefulness of the proposed translation guidelines is tested in a preliminary empirical study and the result is promising.
# TABLE OF CONTENTS

Abstract.................................................................................................................................. ii
Table of Contents..................................................................................................................... iii
List of Tables............................................................................................................................. v
List of Figures........................................................................................................................... vi
Acknowledgements.................................................................................................................. vii

Chapter 1: Introduction ............................................................................................................ 1
  1.1 Motivation.......................................................................................................................... 2
  1.2 Thesis Outline.................................................................................................................... 4

Chapter 2: Literature Review .................................................................................................. 6
  2.1 Goal Identification and Elicitation Techniques ............................................................... 6
    2.1.1 The KAOS Project........................................................................................................ 6
    2.1.2 Goal-Based Requirements Analysis Method (GBRAM)......................................... 10
    2.1.3 Goal Modeling Using Scenarios.............................................................................. 12
    2.1.4 Singh and Woo’s Goal Schema............................................................................. 16
  2.2 Existing Methods of Mapping Requirements to Systems Analysis and Design Diagrams .......................................................... 18
    2.2.1 Tropos Project............................................................................................................ 18
      2.2.1.1 The i* Modeling Framework.............................................................................. 19
      2.2.1.2 Tropos Project.................................................................................................. 23
    2.2.2 Class-Responsibility-Collaboration Card to UML Class Diagram......................... 27
      2.2.2.1 Elements of Class-Responsibility-Collaboration (CRC) Cards....................... 28
      2.2.2.2 Seven-Step Process to Create CRC Cards and Class Diagrams..................... 29
    2.2.3 Transform Service Responsibility Tables to UML Use Case and Class Diagrams .......................................................... 31
    2.2.4 Integrate Non-Functional Requirements into Systems Design............................... 33
      2.2.4.1 Integrating NFR’s into Class Diagrams.............................................................. 34
      2.2.4.2 Integrating NFR’s into Sequence and Collaboration Diagrams........................ 36
      2.2.4.3 Limitations........................................................................................................ 36
    2.2.5 Summary.................................................................................................................... 37
  2.3 Good Decomposition Model (GDM).................................................................................. 37
  2.4 Summary of Literature Review......................................................................................... 42
Chapter 3: Goal Template and UML Diagrams Translation Guidelines

3.1 Development of Goal Template ................................................. 47
3.2 UML Diagrams Translation Guidelines...................................... 56
  3.2.1 Translation Guideline for UML Use Case Diagram ................. 57
  3.2.2 Translation Guideline for UML Class Diagram ...................... 63
  3.2.3 Translation Guideline for UML Sequence Diagram ................. 73
  3.2.4 Translation Guideline for UML Collaboration Diagram .......... 79
  3.2.5 Translation Guideline for UML Activity Diagram .................. 84
  3.2.6 Translation Guideline for UML Statechart Diagram ............... 90
3.3 Summary ........................................................................... 94

Chapter 4: Empirical Study and Evaluation .................................. 97
4.1 Study Design .................................................................. 97
4.2 Analysis of Use Case Diagram ............................................. 98
4.3 Analysis of Class Diagram .................................................. 102
4.4 Analysis of Sequence Diagram ............................................ 104
4.5 Analysis of Collaboration Diagram ....................................... 106
4.6 Analysis of Activity Diagram ............................................... 108
4.7 Analysis of Statechart Diagram ........................................... 110
4.8 Summary ...................................................................... 112

Chapter 5: Conclusion and Future Research .................................. 115
5.1 Conclusion ................................................................... 115
5.2 Contributions ................................................................. 117
5.3 Limitations and Future Research ........................................... 118

References ........................................................................... 120

Appendix A: Goal Templates for Vision Quest Auto Dealership Case........... 125
Appendix B: Goal Analysis Form for Vision Quest Auto Dealership Case......... 130
<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Definition of Rolland’s Goal Parameters [9]</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>Template of the Attributes in a Schema [11]</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>Tropos Methodology Outline [3]</td>
<td>26</td>
</tr>
<tr>
<td>2.4</td>
<td>Three-Column Service Responsibility Table [13]</td>
<td>31</td>
</tr>
<tr>
<td>2.5</td>
<td>Criteria of Good Decomposition Model [17]</td>
<td>39</td>
</tr>
<tr>
<td>2.6</td>
<td>Applying GDM Criteria to UML</td>
<td>41</td>
</tr>
<tr>
<td>3.1</td>
<td>List of Elements Used in Goal Analysis Techniques</td>
<td>50</td>
</tr>
<tr>
<td>3.2</td>
<td>Elements of Goal Template</td>
<td>51</td>
</tr>
<tr>
<td>3.3</td>
<td>Goal Template Example – Template #1</td>
<td>55</td>
</tr>
<tr>
<td>3.4</td>
<td>Goal Template Example – Template #2</td>
<td>55</td>
</tr>
<tr>
<td>3.5</td>
<td>Goal Analysis Form</td>
<td>66</td>
</tr>
<tr>
<td>3.6</td>
<td>Goal Analysis Form (two examples)</td>
<td>67</td>
</tr>
<tr>
<td>3.7</td>
<td>Combined Description of Process Vehicle Purchase Use Case (sequence diagram)</td>
<td>76</td>
</tr>
<tr>
<td>3.8</td>
<td>Combined Description of Process Vehicle Purchase Use Case (collaboration diagram)</td>
<td>81</td>
</tr>
<tr>
<td>3.9</td>
<td>Combined Description of Process Vehicle Purchase Use Case (activity diagram)</td>
<td>86</td>
</tr>
<tr>
<td>3.10</td>
<td>Cross Reference of Goal Template Elements and UML Diagrams</td>
<td>95</td>
</tr>
<tr>
<td>3.11</td>
<td>Summary of UML Diagrams Translation Guidelines</td>
<td>96</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison of Diagrams by Guidelines and Diagrams by the Subject</td>
<td>114</td>
</tr>
<tr>
<td>App.1</td>
<td>Goal Templates Summary for Vision Quest Auto Dealership Case</td>
<td>129</td>
</tr>
<tr>
<td>App.2</td>
<td>Goal Analysis Form for Vision Quest Auto Dealership Case</td>
<td>132</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

Figure 2.1: KAOS Conceptual Meta-model (adapted from [6]) ........................................... 8
Figure 2.2: Example of Goal Schema in GBRAM (adapted from [8]) ............................... 12
Figure 2.3: Rolland’s Requirements Elicitation Process (adapted from [9]) .................... 13
Figure 2.4: Overview of Rolland’s Goal Discovery Process (adapted from [9]) .............. 14
Figure 2.5: Rolland’s Goal Structure (adapted from [9]) ................................................ 15
Figure 2.6: Combining Goal Elicitation and Specification (adapted from [11]) ............. 17
Figure 2.7: A Strategic Dependency Model (adapted from [4]) ...................................... 21
Figure 2.8: A Strategic Rationale Model (adapted from [2]) .......................................... 22
Figure 2.9: CRC Card Template (adapted from [12]) ...................................................... 28
Figure 2.10: Integrating NFRs into Class Diagrams (adapted from [14]) ...................... 35
Figure 3.1: E-R Diagram for Goal Template Elements.................................................... 53
Figure 3.2: Use Case Diagram for Vision Quest Auto Dealership Case ....................... 62
Figure 3.3: Class Diagram for Vision Quest Auto Dealership Case ............................. 72
Figure 3.4: Sequence Diagram for Vision Quest Auto Dealership Case ..................... 78
Figure 3.5: Collaboration Diagram for Vision Quest Auto Dealership Case ............... 83
Figure 3.6: Activity Diagram for Vision Quest Auto Dealership Case ...................... 89
Figure 3.7: Statechart Diagram for Vision Quest Auto Dealership Case .................... 93
Figure 4.1: Use Case Diagram Drawn by the Subject ..................................................... 100
Figure 4.2: Decomposition of Use Case Diagram Figure 3.2 ......................................... 101
Figure 4.3: Class Diagram Drawn by the Subject ......................................................... 103
Figure 4.4: Sequence Diagram Drawn by the Subject .................................................. 105
Figure 4.5: Collaboration Diagram Drawn by the Subject .......................................... 107
Figure 4.6: Activity Diagram Drawn by the Subject ................................................... 109
Figure 4.7: Statechart Diagram Drawn by the Subject .............................................. 111
ACKNOWLEDGEMENTS

First of all, I really appreciate the generous help from Dr. Carson Woo, my thesis supervisor. With his insightful comments and valuable feedbacks, Dr. Woo kindly directed me from topic selection, through literature review, and all the way to this final version of my thesis. It was a pleasure to work with him. This thesis wouldn’t have been possible without Dr. Woo’s guidance and support.

I also want to give my gratitude to Dr. Yair Wand and Dr. Andrew Burton-Jones for being my committee members. I’m very thankful for their time input and invaluable advice.

I’d like to thank Sase N. Singh for his precious discussions and suggestions over time. He gave me the initial idea of this thesis.

Thank Michael (Feihu) He and Lior Limonad for their participation in my empirical study and their comments.
CHAPTER 1: INTRODUCTION

Current research endeavors in requirements engineering field have paid substantial attention to the concept of goals and the use of goals to capture various objectives the current system or system-to-be is supposed to achieve. Goals have long been considered as essential components in the requirements engineering process [1].

According to [1], a goal is an objective the system under consideration should achieve and goal-oriented requirements engineering is concerned with the use of goals for eliciting, elaborating, structuring, specifying, analyzing, negotiation, documenting, and modifying requirements. Goal models should be built during the early phases of the requirements engineering process.

After building goal models in the early requirements engineering phase, system analysts and developers should continue developing a formal requirements document to precisely describe the future system. This phase of requirements engineering is commonly accepted as the late phase of the requirements engineering process. Moreover, in order to completely and adequately reflect the users’ requirements, which are captured using the concept of goals during the early phase requirements engineering, in the formal requirements document, attention should be given to this process and guidelines should be provided to system analysts and developers to guide them map stakeholders’ goals to formal requirements.

The research project presented in this paper is based on the above premise. It focuses on proposing structured guidelines to translate stakeholders’ goals to systems analysis and design diagrams. Given its wide acceptance and usage in the industry, Unified Modeling Language (UML) is adopted as a systems analysis modeling technique in this thesis. An auto dealership business case is used as an example to illustrate the
translating guidelines. In addition, a goal template is also developed to capture users' requirements in the form of goals.

1.1 Motivation

As discussed above, early phase requirements deal with stakeholders' intentions, which can be modeled as goals, and how they can be accomplished by analyzing organizational context. This phase of requirements engineering process receives increasing attention and is considered crucial [2]. The activities carried out during this stage of requirements engineering focus on how the intended system would meet organizational goals, why the system is needed, what alternatives might exist, what the implications of the alternatives are for various stakeholders, and how the stakeholder's intentions and concerns might be addressed [3]. Goal is an essential concept in this phase of requirements engineering process [1] and the output of early phase requirements engineering is an organizational model that includes relevant actors, their respective goals and their interdependencies [3]. In contrast, late requirements describe the requirements for the system-to-be. The objective of late phase requirements engineering is to produce a requirements document to pass on to developers, so that the resulting system would be adequately specified [2]. Furthermore, because business professionals lack the knowledge of formal systems analysis and design techniques, such as UML language, the usage of goal concept and organizational models to elicit users' requirements would promote user involvement in systems analysis and design activities.

It's considerably important to precisely and completely map early requirements to late requirements to ensure that information system reflects users' requirements of system capabilities and features. However, few studies in requirements engineering field have
been conducted in order to provide system analysts and developers with a complete and systematic approach to map early requirements to late requirements. Tropos [3] is one of these few research projects and it argues the importance of reducing the gap between information system and its operational environment. In order to achieve this, Tropos adopts i* organizational modeling framework as a foundation to model both early and late requirements [3]. Although the i* framework supports the modeling and reasoning about organizational environments and their information systems [4], it does not apply the concept of goals as a way to capture users’ requirements, which, however, is considered as an essential element, neither does it describe an explicit way of how to create i* Strategic Dependency model and i* Strategic Rationale model. Only partial steps and associated techniques are prescribed in the i* framework [5]. We will talk more about this in next chapter.

The approach proposed in this thesis suggests a set of comparatively systematic guidelines to translate users’ requirements captured as goals to systems analysis diagrams drawn in UML. These UML analysis diagrams could be further used by analysts to derive systems design and implementation diagrams. As a result of this, the design diagrams would be capable of reflecting system requirements more precisely and completely because the system analysis diagrams could more closely capture users’ requirements. In addition, although UML has long been adopted as a modeling language for designing computer software, the practice of using UML to model business processes is often challenged because UML does not explicitly provide mechanisms to express business semantics. The translation guidelines proposed in this thesis provide a way to model users’ requirements and their rationale of the requirements to UML diagrams. Therefore, the suggested translation guidelines can enable UML to be more suitable for business modeling. Without these guidelines, UML would not be able to enforce the inclusion of business context. We will illustrate in detail in Chapter 3 how UML can be used for business process modeling.

In order to capture the goals and other related information, such as actors and
stakeholders of a particular goal, a goal template is adopted. This goal template is developed based on literature review of current research in relevant fields and the objective of the thesis. Then guidelines are proposed to translate goal templates to six UML analysis and design diagrams, namely Use Case diagram, Class diagram, Sequence diagram, Collaboration diagram, Activity diagram, and Statechart diagram. Also, these translating guidelines and processes are illustrated using an auto dealership case. A small empirical study is conducted as well to validate the effectiveness of the proposed translation guidelines.

1.2 Thesis Outline

The next chapter of this thesis reviews research efforts that have been conducted in order to map users’ requirements or goal models to systems analysis or systems design models as well as their limitations. It also provides an overview of the studies that we based this work on in order to generate our goal template to capture stakeholders’ goals and relevant information. Examples of these studies are Goal Modeling Using Scenario [9], GBRAM (Goal-Based Requirements Analysis Method) [8] and KAOS project [6]. Chapter 2 also introduces theories and techniques that are used in this thesis to develop as well as refine our translation guidelines, including Good Decomposition Model (GDM) [16, 17], Rolland’s goal structure approach [9].

Chapter 3 is the heart and key contribution of this thesis. It introduces the goal template that is used to capture users’ system requirements modeled as stakeholders’ goals. It also discusses our structured guidelines to transform goal templates to UML diagrams. Chapter 3 illustrates the application of translation guidelines through an auto dealership case scenario. At the end of each section introducing guidelines for a particular UML diagram, we will provide the UML diagram that is developed from
goal templates using the translation approach proposed in the respective section. Unified Modeling Language (UML) is also briefly discussed and each of the six UML diagrams (Use Case diagram, Class diagram, Sequence diagram, Collaboration diagram, Activity diagram, and Statechart diagram) is introduced at the beginning of each section to provide readers with some basic knowledge of this systems analysis and design language.

Evaluation of our work is provided in Chapter 4 through an empirical experiment. The small empirical study is carried out to examine the usefulness and effectiveness of the methodology proposed in Chapter 3. Potential problems with our method are also discussed and possible solutions are suggested.

Chapter 5 concludes the thesis with a summary of its contributions and limitations and a discussion of future research work.
CHAPTER 2: LITERATURE REVIEW

In this chapter, we will first review some of the existing goal identification and elicitation techniques. These studies are the groundwork that we relied on to develop our goal template to capture users' requirements as goals.

Then, we will introduce some research studies that have been done aiming to map users' requirements or goals to system diagrams. These techniques are discussed as well as their limitations, which we argue our proposed methodology will solve.

In this chapter, we will also discuss the Good Decomposition Model (GDM) [16, 17], the ontologically-based theory that will be referred to so as to develop and refine our translation guidelines in Chapter 3.

2.1 Goal Identification and Elicitation Techniques

Studies presented in this section are techniques that are suggested by researchers to identify and elicit goals regarding a system. They are the ones we considered while developing our own goal template to reflect users’ system requirements.

2.1.1 The KAOS Project

Dardenne et. al. [6] suggested requirements analysis includes two subsequent
sub-steps: 1) requirements elicitation, where a preliminary architecture for the specification of the system and its environment is elaborated; and 2) formal specification, where the global model elaborated during requirements elicitation is refined using constructs of a formal language. KAOS project proposes an approach for requirements elicitation, where a model of the system and its environment is elaborated. This model contains concepts that are usually not found in the final formal specifications, such as goals to be achieved, agents and their responsibilities [6].

In their paper, Dardenne et. al. describe a conceptual meta-model (as shown in Figure 2.1) in terms of which requirements models are acquired. Requirements elicitation could be described as a model for acquiring requirements models. The requirements model developed during acquisition is maintained in a requirements database that is organized as components of the conceptual meta-model in Figure 2.1. The acquisition process therefore is to traverse the meta-model by acquiring corresponding instances. KAOS meta-model shown in Figure 2.1 consists of meta-concepts, meta-relations linking meta-concepts, meta-attributes characterizing meta-concepts and meta-relations, and meta-constraints of all the components. Acquired requirements model is thus composed of instances of meta-concepts, linked by instances of meta-relations and characterized by instances of meta-attributes. The meta-concepts are object, action, constraint, goal and scenario. Object can be further specialized to entity, relation, event and agent, while goal can be specialized to leafgoal.
KAOS project aims to support the elaboration of a requirements model that is guaranteed to satisfy the goals of the clients, and record how that satisfaction is realized. In addition to the conceptual meta-model for requirements model acquisition, KAOS also proposes a goal-directed acquisition strategy. The requirements acquisition strategy provides a systematic approach to acquire a requirements model from meta-model to specify users’ goals. The acquisition strategy can be explained as four steps below: (We only briefly introduce the acquisition strategy here, for further details about how to perform each step and any other information, please refer to [6].)

- **Step 1: Identify goals and agents**
- **Step 1.1: Build a goal structure for the particular application**

  Clients’ goals are progressively reduced in a goal-subgoal structure, which is an and/or graph. The leaf goals are primitive goals that operational constraints will be associated to in step 2 below.
• **Step 1.2: Identify agents**

Agents are identified as instances of agent meta-concept, but not necessarily completely described. Not all the relevant characteristics may be discovered at this step.

• **Step 2: Operationalize goals**

Operationalization of leaf goals in the goal structures is done by identifying constraints. These operationalized leaf goals are formal assertions that refer to objects and actions available to agents.

• **Step 3: Acquire object/action characterizations**

Complete characteristics are acquired for all objects and actions involved in specification. A characteristic is either an attribute (an instance of meta-attribute) or a relation (an instance of meta-relation). Instances of meta-concept, such as meta-event, meta-entity, meta-relation and meta-action, must be defined and linked to constraints that they contribute to ensure.

• **Step 4: Assign agents to actions**

Instances of the meta-relation performance must be identified to produce assignments of agents to actions.

The four-step acquisition strategy is considered as a directed way of acquiring application-specific instances of meta-concepts and meta-relations [6]. The end result of this acquisition process is a domain-specific requirement model representing requirements that satisfy goals.
2.1.2 Goal-Based Requirements Analysis Method (GBRAM)

Previous goal-based methods failed to provide strategies to identify goals, taking it for granted that the goals have already been documented [8]. The Goal-Based Requirements Analysis Method (GBRAM) is a systematic method used to identify, elaborate, refine and organize goals for software requirements specification [8]. GBRAM satisfies the need for procedural guidance for initially identifying and constructing goals. It also provides practical advice to practitioners in terms of goal identification heuristics to guide the process [8].

GBRAM involves two activities: goal analysis and goal evolution. According to Anton [8], goal analysis is the process of exploring gathered documentation (ranging from information about the organization, such as enterprise goals, to system specific information, such as requirements) for the purpose of identifying, organizing and classifying goals, while goal evolution concerns the way goals change from the moment they are identified to the moment they are operationalized in a system specification.

During the process of goal analysis activity, goals can be identified from exploration of various process descriptions documents such as flow charts and entity relationship diagrams by searching for statements which seem to guide design decisions at various levels within a systems or organization. Analysts then proceed to analyze each statement by asking "What goal(s) does this statement exemplify?" and/or "What goal(s) does this statement block or obstruct?" However, these process descriptions are insufficient for achieving completeness. Thus, goals may also be extracted by searching for action words in stakeholder descriptions such as transcripts of interviews with stakeholders. For example, in a meeting scheduler, action words such as 'schedule' and 'reserve' give rise to goals such as Schedule Meeting and Reserve Room.
In addition to goals, the agents, stakeholders and constraints must also be identified. Agents are the entities that seek to achieve goals within an organization or system based on the implicit responsibility that they must assume for the achievement of certain goals. Responsible agents can be determined by asking the question what agents are ultimately responsible for the achievement or maintenance of a goal. Stakeholders are identified by determining which agents claim a stake in each goal achievement. Constraints are conditions that must be met for the achievement of a goal. GBRAM extracts constraints by searching for temporal connectives, such as during, before and after, and dependency relations.

Once the goals, agents and stakeholders are identified, the goals are then classified according to their target conditions and begin to evolve. Goal evolution is affected through goal elaboration and refinement. Identifying goal obstacles, analyzing scenarios and constraints, and operationalizing goals are techniques used for goal elaboration. Goals are refined by eliminating redundancies and reconciling synonymous goals. For example, the goals Meeting arranged and Meeting scheduled are synonymous and can be reconciled.

The operationalized goals, responsible agents, stakeholders, constraints and scenarios are ultimately consolidated into a set of goal schemas (as shown in Figure 2.2) that can be easily translated into a requirements specification. This set of goal schemas provides a textual representation of system requirements organized according to system goals. Figure 2.2 (adopted from [8]) provides an example of the goal schema in GBRAM.
<table>
<thead>
<tr>
<th>Goal:</th>
<th>Available course slots announced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Achievement</td>
</tr>
<tr>
<td>Description:</td>
<td>HRD must announce the courses and the number of slots available so that qualified persons can be identified, matched and notified.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action:</th>
<th>Announce course slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent:</td>
<td>HRD</td>
</tr>
<tr>
<td>Stakeholders:</td>
<td>HRD, employee, TSD</td>
</tr>
<tr>
<td>Obstacles:</td>
<td>1. No available slots; 2. Employee prefs not available.</td>
</tr>
<tr>
<td>Scenario:</td>
<td>1.1 All courses closed (reached max capacity). 1.2 Courses cancelled (no slots available).</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>1. Employee prefs ready; 2. IPs made available.</td>
</tr>
<tr>
<td>Postconditions:</td>
<td>Employee and course slot matched.</td>
</tr>
</tbody>
</table>

Figure 2.2: Example of Goal Schema in GBRAM (adapted from [8])

### 2.1.3 Goal Modeling Using Scenarios

Although goal modeling is an effective approach to identify requirements, there are many difficulties with goal driven approaches in practice [9]. According to the experience of Rolland et. al. [9], it is difficult for domain experts to fully understand the concept of goal and goal discovery is therefore rarely an easy task. Rolland et. al. used the word “fuzzy” to describe the nature of goal concept and since scenarios capture real requirements because a scenario is a possible behavior limited to a set of purposeful interactions taking place among several agents [10], they propose to couple goal modeling and scenario authoring to overcome these difficulties. While previous studies use scenarios to concretize goals, Rolland et. al. [9] use them to discover goals.

The requirements elicitation process in Rolland et. al.'s approach is to elicit requirements through the bi-directional goal-scenario coupling. This relationship between goals and scenarios can be understood as goal operationalization through
scenarios and scenario discovery from goals. It includes two phases (as shown in Figure 2.3): 1) scenario authoring and 2) goal discovery. A scenario can be developed as each individual goal is discovered. Once a scenario has been developed, it is explored to yield goals.

As we can see in Figure 2.3, goals are discovered from scenarios through the notion of Requirement Chunk Model. A requirement chunk (RC) is defined as a pair \(<G, Sc>\), where \(G\) is a goal and \(Sc\) is a scenario. Since a goal is intentional and a scenario is operational, a requirement chunk is a possible way of achieving the goal. RCs can be organized as a hierarchy either through composition (AND relationships) relationships and alternative (OR relationships) relationships or through refinement relationships. Figure 2.4 provides an overview of the goal discovery process. Requirements elicitation process starts from requirement chunk authoring through interleaved goal discovery and scenario authoring. Then these requirement chunks are hierarchized through either AND/OR relationships or refinement relationships.
The hierarchy structure of requirement chunks will not be discussed in further detail here in this thesis. However, as our translation guideline proposed in Chapter 3 relies heavily on Rolland's goal structure to analyze components of goals, the goal structure is illustrated in detail in the following session. Rolland's scenario structure will also be introduced briefly.

As discussed above, a requirement chunk is modeled as a pair $<G, Sc>$. It is an aggregation of goal and scenario classes, both of which are complex objects and have their own structures. Figure 2.5 depicts the structure of a goal. It shows that a goal is associated to a verb and one or more parameters. There are four types of parameters, namely target, direction, way, and beneficiary and three of the four parameters (target, direction, and way) have subtype parameters. For example, target parameter has two subtypes object and result, direction parameter has source and destination, way has means and manner. Table 2.1 summarizes the definitions of all the parameters in
Rolland’s goal structure approach. A goal is stated as a clause with a main verb and several parameters, where each parameter plays a different role with respect to the verb.

![Figure 2.5: Rolland’s Goal Structure (adapted from [9])](image)

<table>
<thead>
<tr>
<th>Parameter Type</th>
<th>Parameter Subtype</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>Object</td>
<td>Target designates entities affected by the goal. An object is supposed to exist before the goal is achieved.</td>
</tr>
<tr>
<td>Result</td>
<td>Target designates entities affected by the goal. Results can be two kinds, 1) entities which do not exist before the goal is achieved and 2) abstract entities which exist but are made concrete as a result of goal achievement.</td>
<td></td>
</tr>
<tr>
<td>Direction</td>
<td>Source</td>
<td>Source identifies the initial location of objects to be communicated.</td>
</tr>
<tr>
<td>Direction</td>
<td>Destination</td>
<td>Destination identifies the final location of objects to be communicated.</td>
</tr>
<tr>
<td>Way</td>
<td>Means</td>
<td>Means designate entities which act as instruments using which a goal is to be performed.</td>
</tr>
<tr>
<td>Way</td>
<td>Manner</td>
<td>Manner defines the way in which the goal is to be achieved. A manner, when complex, can be expressed as a goal.</td>
</tr>
<tr>
<td>Beneficiary</td>
<td>N/A</td>
<td>Beneficiary is the person (or group of persons) in favor of whom the goal is to be achieved.</td>
</tr>
</tbody>
</table>

Table 2.1: Definition of Rolland’s Goal Parameters (adapted from [9])
2.1.4 Singh and Woo’s Goal Schema

Requirements engineering field uses the concept of goals to align systems design with organizational context. Many frameworks have been proposed to conceptualize goals within an organization environment. However, most of these frameworks use different representations and are grounded on one of the four requirements engineering phases, namely requirements elicitation, requirements negotiation, requirements specification, and requirements validation [11]. Singh et. al. [11] proposed a goal-based modeling approach that incorporates the elicitation and specification aspects of the four RE activities. According to Singh et. al., the elicitation element is used to aid requirements acquisition in terms of goals. While given these requirements, specifications are derived to guide system design and implementation.

Figure 2.6 shows how the proposed model works. Strategy is the reflection of organization strategies in response to business context, environment and resources, which is represented as Business Reality. Operational is the operationalization of Strategy, such as organizational structure, descriptions of business processes and job descriptions. Structured Representation represents a specific model representation of the targeted organization. After a structured representation is developed for the domain, Goal Schemas are combined with Structured Representation to derive a requirements model for formal design purpose. Goal elicitation and specification of the attributes in goal schema are partially achieved through the elicitation process from Structured Representation to Goal Schemas. Attributes in the goal schema that are not derived during this process will then be specified by revisiting stakeholders, which is depicted as the link between Operational and Goal Schemas in Figure 2.6.
While the Structured Representation construct in Figure 2.6 captures the “what” aspects of the desired system, Goal Schemas consolidate the “why” features of the future system. As shown in Table 2.2, each goal schema consists of a set of attributes so that it can facilitate architectural decisions while at the same time be used to verify stakeholders’ needs [11].

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Goal</td>
<td>Name of goal that was identified in the OOEM</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the goal</td>
</tr>
<tr>
<td>Action</td>
<td>Action to achieve goal</td>
</tr>
<tr>
<td>Agent</td>
<td>Object that is responsible to achieve goal</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Objects that claim direct stakes in the goal</td>
</tr>
<tr>
<td>Constraints</td>
<td>Ways in which the goal can be blocked</td>
</tr>
<tr>
<td>Sub Goals</td>
<td>Other goals that leads to the achievement of the goal</td>
</tr>
<tr>
<td>Priority</td>
<td>Express the importance of the goal to the stakeholders</td>
</tr>
</tbody>
</table>

Table 2.2: Template of the Attributes in a Schema (adapted from [11])
In line with the proposed requirements elicitation and specification model (as shown in Figure 2.6), a structured methodology for developing goal schemas was developed. The method could be summarized as follows.

**Step 1: Developing Object Oriented Enterprise Model (OOEM) model**

**Step 2: Eliciting goals from OOEM**

**Step 3: Mapping of OOEM goals to schema attributes**

**Step 4: Filling missing attributes by mapping schema onto operational information**

Information concerning how to apply the above approach will not be discussed in further detail here. However, it’s noteworthy that the methodology proposed in this paper combines the “what” and the “why” features of requirements engineering. OOEM model is first derived to represent the “what” and then, the “why” aspect is identified through elicitation of goals from OOEM model. These goals are then mapped into a template representation in terms of proposed goal schema.

### 2.2 Existing Methods of Mapping Requirements to Systems Analysis and Design Diagrams

#### 2.2.1 Tropos Project

As discussed in Chapter One, Tropos project [3] is one of the major studies that aimed to decrease the gap between information system and its organizational environment. Tropos is a software development methodology that is built on the
concepts adopted from i* modeling framework. Tropos proposal has four phases, which are early requirements analysis, late requirements analysis, architectural design, and detailed design. Tropos claims to adopt concepts such as actors and dependencies in i* framework to model both early and late requirements because i* serves well as a foundation to describe organizational environment. It further proposes to use i* strategic dependency model and strategic rationale model to capture early requirements and then, represent information system as one or several actors and refine strategic dependency model and strategic rational model which results in late requirements analysis. The next session will introduce the i* framework briefly first and Tropos project will be discussed in detail then.

2.2.1.1 The i* Modeling Framework

Yu [2] suggested that the understanding and modeling of the rationales ("why" feature) of systems requirements could be accomplished by representing goals in an enterprise, which is considered as early phase of requirements engineering. It considers how the system-to-be would satisfy organizational goals, why the system is needed, what alternatives are for various stakeholders, and how the stakeholders' intentions might be addressed [2]. The early phase requires analyzing stakeholders' intentions and mapping the complex strategic relationships among actors in an enterprise. Most existing requirements engineering techniques focus on completeness, consistency, and verification of requirements. However, early phase RE activities have traditionally been done informally and without much tool support [2]. The i* framework supports this strategic reasoning in an organizational context. It captures the intentional relationships among organizational actors. According to Yu [4], organizations consist of actors and the behavior of actors is regulated by social relationships among them. Actors depend on each other for goals to be achieved, tasks to be performed, and resources to be provided [4]. The i* modeling framework consists of two types of
models, the Strategic Dependency (SD) model and the Strategic Rationale (SR) model.

**Strategic Dependency (SD) Model**

Strategic Dependency model is developed to describe the dependency relationships among various actors in an organization [2]. A SD model is a graph (as shown in Figure 2.7), in which each node represents an actor and each link between two actors indicates that one actor depends on the other to accomplish a certain goal [4]. The strategic dependency model focuses on capturing the external dependency relationships between any of the two actors. SD model distinguishes four types of dependency relationships, goal dependency, task dependency, resource dependency, and softgoal dependency [4]. According to Yu [4], in a goal dependency, the depender depends on dependee to achieve a certain state of events in the world. In a task dependency, the depender depends on the dependee to execute an activity. In a resource dependency, depender depends on dependee to gain a physical entity or information. Through resource dependency, the depender acquires the ability to use the entity as a resource. Last, in a softgoal dependency, depender depends on dependee to achieve some softgoal. (A softgoal is similar to a goal except that the criteria of success are not sharply defined [4].) SD model shows the intentional relationships among actors. By analyzing the four types of dependencies, one can obtain a better understanding of the rationales behind systems requirements [2].
Strategic Rationale (SR) Model

Strategic Rationale model is a graph (as shown in Figure 2.8), where nodes are goals, tasks, resources, and softgoals. These nodes could be linked by means-ends links or task-decomposition links [4]. While the Strategic Dependency model models external intentional relationships between two actors, the Strategic Rationale model provides a more detailed level of reasoning by modeling the internal intentional relationships within each actor. In a SR model, intentional elements (goals, tasks, resources, and softgoals) appear not only as external dependencies, but as internal elements connected by means-ends and task-decomposition relationships [2]. The means-ends links indicates the reason why an actor would carry a certain goal or softgoal, be involved in some task, and demand a resource. Task-decomposition links describe in a hierarchy format the intentional elements that are included in a business
process/routine. The SR model is used to describe actors' interests and concerns and how they might be addressed by various alternatives [2]. It enables the recognition of issues, alternatives identification, and settlement of issues [4].

The emphasis of the i* modeling framework is on understanding the rationales underlying system requirements, rather than focusing on the specification of what the system should do [2]. As a further step towards systems design, Tropos project adopts the i* framework as a basis to model early requirements and moves further to develop detailed design. This will be discussed in more detail below.
2.2.1.2 Tropos Project

Traditional system development methods (object-oriented, structured etc.) have been built on computer programming concepts, such as module, objects, data structures and interfaces. This has led to a semantic mismatch between information systems and their organizational environment, which is perceived as a collection of actors, responsibilities, objectives, tasks and resources [3]. In order to reduce this gap, Tropos, a software development methodology, adopts i* modeling framework and uses the concepts from i* framework, such as actor and dependency, to model early and late requirements, architectural and detailed design. Tropos methodology consists of four phases:

- **Early Requirements**

  Early requirements phase aims to understand stakeholders’ intentions by analyzing the organizational environment. The intentions of stakeholders are modeled as goals and eventually will produce both functional and non-functional requirements of the information system. The output of this phase is i* organizational models which include relevant actors, their respective goals and their inter-dependencies [3]. As discussed above, the concepts and modeling elements of the Strategic Dependency model in i* modeling framework are sufficient to produce a preliminary model of an organization context. Once the relevant stakeholders, their goals, and dependencies among them have been identified, Strategic Rationale models can be developed to further indicate how stakeholders’ goals may be achieved through dependencies with other actors by means-ends analysis.

- **Late Requirements**

  The objective of late phase requirements engineering activities is to produce a requirements document for system developers, in which the future system is adequately specified in terms of “what” the system should do [2]. Late requirements
phase describes relevant functions and qualities of the system-to-be within its operational environment. Tropos represents information system as one or more actors in the Strategic Dependency (SD) model. Thus, as a result of late requirements analysis, software system is included in SD model as one or more actors that participate in dependency relationships with other actors from system's organizational environment. As an actor in SD model, the software system contributes to the fulfillment of stakeholders' goals through dependency relationships.

- **Architectural Design**
  Architectural design is the phase where the system's global architecture is defined in terms of sub-systems, interconnected through data, control and other dependencies. Tropos has defined, based on organization management research, many different organizational architectural styles to guide the system architecture design. The organizational styles are identified at a meta-level as generic structures and they can be instantiated to derive specific application architecture [3]. The analysis during architectural design involves evaluating and selecting among alternative architectural styles against the criteria which the desired qualities identified earlier in late requirements phase.

- **Detailed Design**
  Detailed design phase defines in detail the behavior of each architectural component of the software system. Castro et. al. proposed to use agent communication languages like Agent Unified Modeling Language (AUML) to support detailed design phase. However, no detailed guidelines are provided in Tropos project.

The Tropos method is summarized in Table 2.3 as below.
<table>
<thead>
<tr>
<th>Phase</th>
<th>Phase Outputs</th>
<th>Detail Steps</th>
</tr>
</thead>
</table>
| Acquisition of Early Requirements | Strategic Dependency (SD) Model and Strategic Rationale (SR) Model | 1. Strategic Dependency model is developed to capture relevant actors, their respective goals and their interdependencies.  
2. Strategic Rationale model is developed to determine, through a means-ends analysis, how the goals can be fulfilled through the contributions of other actors.                                                                                           |
| Definition of Late Requirements  | Revised Strategic Dependency (SD) and Strategic Rationale (SR) Model | 3. In the original Strategic Dependency (SD) model, include an actor to represent the software system to be developed.  
4. Take this system actor and do a means-ends analysis to produce a new Strategic Rationale (SR) model.  
5. If necessary, decompose the system actor into several sub-actors and revise the SD and SR models.                                                                                                                                                                                                 |
| Architectural Design (Agents are introduced in this phase.) | Non-Functional Requirements (NFR) Diagram and revised SD and SR models | 6. Select an architectural style using the criteria that the desired qualities identified in previous phase. Produce a NFR diagram to represent the selection and design rationale.  
7. If required, introduce new system actors and dependencies, as well as the decomposition of existing actors and dependencies into sub-actors and sub-dependencies. Revise SD and SR models accordingly.  
8. Assigning actors to agents and roles/patterns to solve actors’ goals. |
<table>
<thead>
<tr>
<th>Phase</th>
<th>Phase Outputs</th>
<th>Detail Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10. Develop Sequence and Collaboration diagrams to capture inter-actor dynamics.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11. Develop Plan diagrams to capture both intra-actor and inter-actor dynamics.</td>
</tr>
<tr>
<td>Implementation</td>
<td>Beliefs-Desires-Intentions (BDI) agent architecture</td>
<td>12. From the detailed design generate Agents, Capabilities, Database Relations, Events and Plans in JACK programming platform.</td>
</tr>
</tbody>
</table>

Table 2.3: Tropos Methodology Outline (adapted from [3])

Tropos suggests the importance of aligning information system with its operational environment and proposes some steps to link early requirements analysis with late requirements analysis. However, we can see from Table 2.3 that these steps are not explicitly stated. For example, step 3 in Table 2.3 asks analysts to model software system as an actor in SD model. Nevertheless, questions like what dependency relationships with other organizational actors should the new system actor have and how to revise the SD model to represent these relationships are not discussed.

Furthermore, Tropos adopts i* framework, which is not widely used as a business modeling language, to model organizational environment. On the contrast, Unified Modeling Language (UML) is a popular business modeling language and widely accepted by both systems analysts and developers. Although UML diagrams are also derived from detailed design phase of Tropos method, these diagrams are more architecture and software design models rather than business modeling models. Many studies have been conducted regarding to business modeling using UML. The following sessions will discuss some of these methods as well as their limitations. As a solution to all these limitations, we will propose our methodology in Chapter 3.
2.2.2 Class-Responsibility-Collaboration Card to UML Class Diagram

During information systems analysis, systems analysts need to create conceptual models to represent how the business system will function. Since a conceptual model depicts the logical organization of system objects without representing software components or classes in an object-oriented programming language (although the conceptual model does contain classes, attributes, operations, and relationships among the classes and in some cases, uses similar notations as design model), analysts are able to focus on creating the model so as to reflect business requirements. Conceptual models are typically represented by Class-Responsibility-Collaboration (CRC) cards and class diagrams [12]. According to Dennis et. al. [12], as one kind of conceptual model, Class-Responsibility-Collaboration (CRC) cards are used to model responsibilities of classes and collaborations among them.

Dennis et. al. [12] proposed in their book a method to create UML class diagrams from CRC cards. They claimed that, because of the ease of role playing with CRC cards so as to identify complete classes in a system, it's preferred to create CRC cards first and then transfer the information from CRC cards into a class diagram. Furthermore, in order to build class diagrams from CRC cards, Dennis et. al. [12] extended CRC cards to include all relevant information associated with a class. In the following session, we first briefly introduce the structure and elements of a CRC card, then we will discuss the seven-step process that Dennis et. al. proposed to create CRC cards and transform CRC cards into a class diagram.
2.2.2.1 Elements of Class-Responsibility-Collaboration (CRC) cards

Figure 2.9 shows the elements in a CRC card template. These elements represent the classes, their responsibilities and collaborations and as a result, each CRC card captures and describes the essential elements of a class [12].

The template allows systems analysts to record the class name, ID, type, description, list of associated use cases, responsibilities, collaborators, attributes of the class, and three types of relationships with other classes. According to Dennis et. al. [12], together with ID number of the class, the list of associated use cases can be used to trace design decisions back to specific requirements, which is considered very important in later stages of information systems development process. The description element is a brief textual definition of the class. Responsibilities of a class are the operations that the class is supposed to accomplish in a business system. Attributes of a class represent the properties that describe the state information of the class. Relationships are modeled as three types, namely generalization relationship, aggregation relationship, and other associations.

<table>
<thead>
<tr>
<th>Class Name:</th>
<th>ID:</th>
<th>Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>Associated Use Cases:</td>
<td></td>
</tr>
<tr>
<td>Responsibilities:</td>
<td>Collaborators:</td>
<td></td>
</tr>
<tr>
<td>Attributes:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relationships:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generalization:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aggregation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Associations:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.9: CRC Card Template (adapted from [12])
2.2.2.2 Seven-Step Process to Create CRC Cards and Class Diagrams

UML Class diagram depicts the classes and the relationships among classes. It is a graphical description of the information contained on the CRC cards [12]. The proposed seven-step process to create UML class diagrams from CRC cards of the problem domain can be summarized as follows:

1. Create Class-Responsibility-Collaboration (CRC) cards by performing textual analysis on the use cases descriptions. Dennis et. al. claimed that most object identification begins with analyzing the use cases identified for the problem. They also proposed textual analysis guidelines for use case analysis. Objects are identified as a result of this step and a CRC card is created for each object identified.

2. Brainstorm additional candidate classes, attributes, operations, and relationships by using the common object lists. A common object list is a list of objects, such as physical or tangible things, incidents, roles, and interactions, which are common to the business domain of the system [12]. In addition to analyzing use cases in step 1, analysts are encouraged to uncover additional objects using common object list approach, modify relevant use cases and CRC cards, and create new CRC cards for newly-identified candidate objects if applicable.

3. Role-play each use case using the CRC cards to identify additional objects, attributes, operations, and relationships. This step aims to refine existing CRC cards and identify CRC card for any missing object. This process may be human-demanding because each CRC card should be assigned to one individual. As individuals act out their roles, additional objects, attributes, operations, or relationships will be identified.

4. Create the class diagram based on CRC cards. Dennis et. al. used the extended version of CRC cards to capture the same amount of information of a class as
in class diagrams, hence creating class diagrams based on CRC cards is a very straightforward process. Information contained in the CRC cards is simply transferred to the class diagrams. For instance, responsibilities in CRC cards are transferred to operations of the class in a class diagram, attributes of a CRC class are transferred to attributes of the class, and relationships, which have already been captured as three types in CRC cards, are drawn as generalization, aggregation, or association relationships in the class diagram.

5. Review the class diagram for missing and/or unnecessary classes, attributes, operations, and relationships.

6. Incorporate useful patterns by looking at the collection of patterns available and comparing the classes contained in the patterns with those in the evolving class diagram. This step also includes adding and deleting classes, attributes, operations, and/or relationships.

7. Review CRC cards and the class diagram. A walkthrough approach, in which an analyst presents the model to developers and users, is recommended.

The seven-step process presented above proposes a guideline to create CRC cards and then transform them into class diagram. However, the method used to create CRC cards at the first place stays weak. Although it explains how to transfer CRC cards to class diagram in step 4, CRC cards creation process (step 1 to 3) is based on textual analysis and brainstorming, which cannot guarantee to generate accurate and consistent representation of an organization or a system across different systems analysts. As class diagram is transferred directly from CRC cards, it’s considered to be important to create accurate and consist CRC cards. Moreover, Dennis et. al. did not provide clear guidelines on how to differentiate three types of relationships in class diagram.
2.2.3 Transform Service Responsibility Tables to UML Use Case and Class Diagrams

In order to make sure system requirements reflect user’s needs, it’s necessary for business professionals to participate in systems analysis and design process. However, many heavyweight systems analysis and design approaches such as Unified Modeling Language (UML) are inappropriate for business professionals [13]. Tan et. al. [13] proposed to use Service Responsibility Tables (SRTs) as a lightweight analysis tool for systems analysis by business professionals.

Table 2.4 shows us a template of a typical Service Responsibility Table (SRT). It identifies active or passive responsibilities of both service providers and service consumers during a service process [13]. For the third column in the SRT template, it could be any topic that might be important for analyzing the particular system, such as problems and issues of each service step, information used and generated at each step, and products or services produced. More examples of potential topics for the third column could be found in [13]. One of the advantages of Service Responsibility Table approach is that SRT focuses on activities and responsibilities rather than technology, which could help business professionals to understand business process and better specify their system requirements.

<table>
<thead>
<tr>
<th>Provider Activity or Responsibility</th>
<th>Customer Activity or Responsibility</th>
<th>Additional Column</th>
</tr>
</thead>
</table>

Table 2.4: Three-Column Service Responsibility Table (adapted from [13])
The paper also provided two sets of heuristics for transforming SRTs into UML use case diagram and class diagram, respectively, by IT professionals. With these heuristics, use case and class diagrams, which are used for design and implementation of information systems, can be developed from SRTs.

**Heuristic for Transforming SRT into Use Case Diagram**

1. Create a service responsibility table and fill out the activities and responsibilities of service providers and customers (the first two columns of Table 2.4).
2. Identify all service providers and customers in the SRT and model them as actors in the use case diagram.
3. Identify activities and responsibilities that associate with each provider and customer. Model these activities and responsibilities as use cases in the use case diagram.
4. Associate actors with corresponding use cases and identify *extend* or *include* relationships among use cases according to SRT.

**Heuristic for Transforming SRT into Class Diagram**

1. Identify information created and used at each step in the SRT that is used above to create use case diagram and write that into the third column.
2. Identify all actors mentioned in the first two columns of SRT and all information items mentioned in the third column.
3. Determine entities, attributes of each entity, and relationships between entities.
4. Identify additional entities, attributes and relationships that are not mentioned explicitly in the SRT.
5. Create class diagram and ensure the diagram reflect requirements expressed in the SRT.

Tan et. al. [13] suggested to differentiate between lightweight analysis by business professionals and heavyweight analysis by IT professionals in order to improve user’s involvement into information systems analysis and design process. They also
provided a lightweight analysis technique, Service Responsibility Tables approach, for business professionals. Furthermore, the proposed two sets of transforming heuristics could be seen as a linkage between lightweight analysis and heavyweight analysis. Nevertheless, SRTs are not able to capture stakeholders’ goals which are considered as a crucial element in requirements engineering fields. Another shortcoming of this research is that the heuristics discussed in the paper are not precise enough to produce consistent UML diagrams across different system analysts. For example, step 4 in use case diagram transforming heuristic does not show how to build extend or include relationships among use cases. Step 4 in class diagram transforming heuristic also fails to provide guidelines to exactly identify missing entities, attributes and relationships among entities.

2.2.4 Integrate Non-Functional Requirements into Systems Design

There are some other studies conducted to translate early requirements to UML diagrams. For example, Cysneiros et. al. [14] proposed an approach to integrate Non-Functional Requirements (NFRs) into UML class diagram, sequence diagram, and collaboration diagram. NFRs state constraints to the system as well as particular qualities that the system might have [14]. These non-functional requirements of software could be, for instance, cost, reliability, security, performance, usability, maintainability, usability, and accuracy. According to Cysneiros et. al. [14], satisfying stakeholders’ needs with regard to necessary functionality to be performed by software was considered an important accomplishment. However, as users demand higher and higher quality software, system analysts and programmers have been expected to deliver software that not only implements all the desired functional requirements but also non-functional requirements.
Cysneiros et. al. [14] proposed a systematic approach to integrate NFRs into three UML diagrams, namely class diagram, sequence diagram and collaboration diagram. The comprehensive approach is introduced below. But before that, we’d like to mention that the NFRs integration process starts after design diagrams have been created to reflect users’ functional requirements.

2.2.4.1 Integrating NFRs into Class Diagrams

Figure 2.10 below shows the approach to integrate NFRs into UML class diagrams. The bold arrows represent integrating steps and the number associated with each arrow suggests the sequence of the process.

The whole process of integrating NFRs into class diagram can be summarized as follows:

1. Pick up a class from the class diagram;
2. Search for the class name in NFR graphs;
3. Add necessary operations and attributes to the class;
   3.1. For each NFR graph in which the class name has been found, check if the existing operations that belong to this class already fulfill the non-functional needs expressed in the NFR graph. If not, add necessary operations to the class;
   3.2. For each NFR graph in which the class name has been found, check if the existing attributes that belong to this class already fulfill the non-functional needs expressed in the NFR graph. If not, add necessary attributes to the class;
   3.3. If the class is part of a generalization association, check if any superclass or
subclass of this class has operations or attributes that satisfy the needed operationalizations;

4. Analyze if new attributes need to be included in the class in order to implement the newly added operations;

5. Proceed to the next NFR graph that contains the class name picked up in Step 1;

6. Repeat step 1 to step 5 until there is no classes left in the class diagram to analyze.

According to Cysneiros et. al., once all the classes have been checked to search against NFR graphs, the revised class diagram will then reflect all the attributes and operations that are necessary to satisfy users' non-functional requirements [14].
2.2.4.2 Integrating NFRs into Sequence and Collaboration Diagrams

The process of integrating NFRs into sequence and collaboration diagrams can be carried out after NFRs have been integrated into class diagram. In other words, the NFRs integration into sequence and collaboration diagrams process is based on the revised class diagram that contains all necessary attributes and operations added due to NFRs satisfaction.

The process of integrating NFRs into UML sequence and collaboration diagrams can be summarized as follows:

1. Examine every class in the class diagram;
2. If there is any new operation included in this class because of NFRs satisfaction, search the sequence and collaboration diagrams and find where this class appears;
3. Check if the new operation implies any change in sequence or collaboration diagram;
4. Add necessary objects, messages or both to the diagram. If any pre or post condition is attached to the operation, specify it in a note attached to a message;
5. Repeat step 1 to step 4 until there is no classes left in the class diagram to analyze.

2.2.4.3 Limitations

Apparently, this approach assumes that there is already a set of UML design diagrams describing functional requirements. However, it's still not clear how to derive UML design diagrams that could precisely reflect functional requirements at the first place. As Cysneiros et. al. said in the paper, “we will not focus on any particular approach to build the conceptual models of the functional perspective” [14].
Use cases are often created first during systems analysis and design and used to construct class, sequence, and collaboration diagrams. Although they claimed in the paper that the conceptual models that will arise later will naturally reflect the NFRs if NFRs is integrated into use cases in the early stages of software development process, Cysneiros et. al. did not provide the details on how to integrate NFRs into use cases.

2.2.5 Summary

In summary, all of the works discussed above lack a structured and complete way of translating users' requirements to one or several systems analysis and design diagrams. This thesis proposes a structured way of doing the translation. Users' requirements are captured as stakeholders' goals and are specified using a goal template approach. A set of structured steps are also developed to map goals to six UML diagrams, including Use Case diagram, Class diagram, Sequence diagram, Collaboration diagram, Activity diagram, and Statechart diagram.

2.3 Good Decomposition Model (GDM)

Good decomposition model (GDM) criteria are used to refine UML diagrams developed according to our proposed guidelines. Therefore, GDM is reviewed in the following section.

Wand and Weber [16] defined decomposition as a set of subsystems of a system where a) every element in the composition of the system is included in at least one of the subsystems in the set, b) the (set) difference between the union of the environments of
the subsystems and the composition of the system equals the environment of the system, and c) each element in the structure of the system is included in at least one of the subsystems in the set [16]. To put it another simple way, a decomposition of a system is a set of subsystems of the system such that the composition of the system equals the union of the compositions of the subsystems in the set [17].

Weber [17] listed two motivations for the development of the decomposition construct. The first is that humans seem to find decompositions useful as a means of understanding the real world. The second motivation for the development of the decomposition construct is that it has proven useful as a basis for design [17].

A system analyst may have several different decompositions of a particular system or organization. Good Decomposition Model (GDM) aims to provide a set of criteria to evaluate the quality of alternative decompositions [16]. The purpose of GDM is to articulate certain structural characteristics that will impact the extent to which an information system script communicates the meaning of the real world phenomena it is intended to represent to users of the script. GDM specifies five criteria for a decomposition to be declared as a good one. The five criteria in GDM and the definition of each are summarized from [17] and shown in the following table (Table 2.5).
### Table 2.5: Criteria of Good Decomposition Model (adapted from [17])

<table>
<thead>
<tr>
<th>Condition</th>
<th>Definition</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimality</td>
<td>A decomposition is good only if for every subsystem at every level in the level structure of the system there are no redundant state variables describing the subsystem.</td>
<td>Each subsystem in a decomposition has no redundant state variables to describe it.</td>
</tr>
<tr>
<td>Determinism</td>
<td>For a given set of external (input) events at the system level, a decomposition is good only if for every subsystem at every level in the level structure of the system an event is either (a) an external event, or (b) a well-defined internal event.</td>
<td>All internal events in each subsystem of a decomposition are well-defined.</td>
</tr>
<tr>
<td>Losslessness</td>
<td>A decomposition is good only if every hereditary state variable and every emergent state variable in a system is preserved in the decomposition.</td>
<td>All hereditary and emergent state variables are preserved in a decomposition of a system.</td>
</tr>
<tr>
<td>Minimum Coupling</td>
<td>A decomposition has minimum coupling if the cardinality of the totality of input for each subsystem of the decomposition is less than or equal to the cardinality of the totality of input for each equivalent subsystem in the equivalent decomposition.</td>
<td>The total action of all environmental things on each subsystem of a decomposition is minimized.</td>
</tr>
<tr>
<td>Strong Cohesion</td>
<td>Two output variables are cohesive if they depend upon at least one common input. A set of outputs is maximally cohesive if the addition of any other output to the set does not extend the set of inputs on which the existing outputs depend and there is no other output which depends on any of the input set defined by the existing output set.</td>
<td>Each output variable in a subsystem depends upon at least one input variable on which another output variable in the subsystem depends.</td>
</tr>
</tbody>
</table>

Burton-Jones and Meso [18] operationalized five GDM criteria to UML diagrams (as shown in Table 2.6). They linked constructs in UML language to ontological constructs in GDM criteria, which is based on Bunge's ontology. Examples are provided as follows [18]:

39
• **Minimality** requires no redundant state variables in every level structure of the system or subsystem. Since attributes in UML represent state variables, a class in a UML class diagram should only contain attributes that are used by at least one method in the class diagram. Violations of minimality are evident in a class diagram when an attribute is shown that no method in the system uses and that the system does not need.

• **Determinism** means event in every level structure of the system is either a named external event or a well-defined internal event. As events are shown explicitly in UML statecharts, any event in a statechart therefore should lead to just one post-event state. If an event leads to multiple post-event states, a condition should be shown to indicate when the system would move to one event or the other.

• **Losslessness** states every hereditary and emergent state variable is preserved in the decomposition. In a “whole-part” relationship in a UML class diagram, the whole should show any important properties in the domain that emerge from its parts.

• **Loose coupling** requires cardinality of the totality of input for each subsystem of the decomposition is less than or equal to cardinality of the totality of input for each equivalent subsystem in the equivalent decomposition. To satisfy loose coupling, two classes in a UML sequence diagram should pass as few messages between them as possible.

• Last, to comply with **strong cohesion**, for every set of outputs, all output variables affected by input variables need to be contained in the same set, and the addition of any other output to the set does not extend the set of inputs on which the existing outputs depend. Thus, a use case in a UML use case diagram should address one function rather than many different functions.
<table>
<thead>
<tr>
<th>Condition</th>
<th>Relationship</th>
<th>UML Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimality</td>
<td>In UML, attributes represent state variables. Violations of minimality are evident in a class diagram when an attribute is shown that no method in the system uses and that the system does not need.</td>
<td>Class Diagram</td>
</tr>
<tr>
<td>Determinism</td>
<td>In UML, events are shown explicitly in statecharts and implicitly in activity and class diagram.</td>
<td>Statechart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activity Diagram</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class Diagram</td>
</tr>
<tr>
<td>Losslessness</td>
<td>In UML, properties are shown as attributes in class diagrams.</td>
<td>Class Diagram</td>
</tr>
<tr>
<td>Minimum Coupling</td>
<td>Coupling refers to interactions among subsystems (Weber, 1997). In UML, interactions are represented explicitly on sequence diagrams and implicitly on class diagrams.</td>
<td>Sequence Diagram</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class Diagram</td>
</tr>
<tr>
<td>Strong Cohesion</td>
<td>Chidamber and Kemerer (1994) operationalized cohesion as the similarity of methods in a class; the larger the number of similar methods, the more cohesive the class. In UML, similarity of methods can be shown in class diagrams and, at a higher level, in use case diagrams.</td>
<td>Class Diagram</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use Case Diagram</td>
</tr>
</tbody>
</table>

Table 2.6: Applying GDM Criteria to UML (adapted from [18])

Empirical studies [18] have shown UML diagrams that comply with better GDM criteria increase analysts’ actual understanding of a domain. The above table (Table 2.6) is thus used during our proposed translation process to refine UML diagrams developed through our translation guidelines to ensure they manifest good decomposition criteria. As a result of the compliance, these UML diagrams convey better users’ understanding of the problem domain.
2.4 Summary of Literature Review

This chapter summarizes the literature review work we have done, concerning goal identification and elicitation techniques and methods proposed for the development of system diagrams to reflect users' requirements.

In the next chapter, we will first introduce the goal template to gather information about stakeholders' goals and then, we will explain in detail the guidelines to guide the translation from goal templates to UML diagrams.
CHAPTER 3: GOAL TEMPLATE AND UML DIAGRAMS TRANSLATION GUIDELINES

This chapter introduces the goal template we used to capture stakeholders’ goals and the guidelines to transform the set of goal templates into several UML diagrams. Examples are provided within each section to demonstrate our goal template as well as translation guidelines. These examples are based on an auto dealership business case scenario. Thus, we will provide the text description of the business case first.

**Vision Quest Auto Dealership**¹

Lee Atwell is the president/owner of Vision Quest Auto dealership. He employs a vice president, a business manager, a sales manager, a service manager, a parts manager, seven sales associates, ten service technicians. The vision of the Auto dealership is to establish a firm ground in the province of Manitoba and eventually across Canada and finally into the United States. Currently Vision Quest Auto controls an estimated 8% of vehicle sales in Manitoba. Lee Atwell is hoping to increase the sales to 12% sales within the next 5 years.

The primary activities of Vision Quest Auto include ordering new vehicles for inventory; pricing used vehicles; deciding which vehicles to sell to wholesalers; servicing vehicles; selling vehicle parts. To have an understanding of the operations, an overview of the typical day-to-day business transactions (selling a new/used vehicle and parts) is presented in the section below.

---

¹ We would like here to give our acknowledgement to Sase N. Singh for granting us the permission to use the case in this study. The case was adapted from [29] and revised by Sase N. Singh.
Selling a Vehicle

The selling process begins when a customer enters the showroom. The sales associate’s job is to promote a sale, which includes answering the customer’s questions and arranging a test drive. Once the customer decides to purchase a vehicle, the sales associate completes a sales order form and informs the customer of a routine cleaning and inspection for the vehicle. The customer is then asked to return on an agreed date to complete the sales transaction. The sales associate then requests the service manager to provide a cleaning schedule for the vehicle. After the vehicle is cleaned, the service manager requests the sales associate to perform a final inspection before the customer arrives. When the customer returns, he pays the business manager. Payment may be made through cash, cheque or a financial institution. If a payment is to be made through a financial institution, the business manager contacts the financial institution for confirmation. After the financial details are completed, the business manager then tries to sell the customer a maintenance contract, which the customer may accept or decline. Finally, the customer is escorted back to the sales associate who delivers the documentation and the keys to the vehicle. Used vehicle sales follow much the same process, but there is some negotiation on prices.

Selling Vehicle Parts

When a customer purchases parts, the parts manager completes an invoice with the total cost and instructs the customer to pay the business manager. After paying, the customer returns with an invoice marked paid and collects the parts from the parts manager.

Management and Executive Transactions

This section describes other activities performed by the managers to ensure that Vision Quest Auto is maintaining customer’s loyalty, a high
quality service, and increasing sales to 12% over the next 5 years.

Sales Manager: The sales manager strives to have his sales team meet weekly and monthly vehicle sales quotas. He is also responsible for tracking deals (vehicle sales) and customers follow up; monitoring the daily newspaper advertisements for competitive prices; and, adjusting vehicles prices. Every month, the nearby wholesaler makes a request for purchasing vehicles. The wholesaler sends in an order to the sales manager who then determines whether he is capable of fulfilling the request.

Business Manager: The business manager is responsible for calculating the commissions for the sales associates. The sales associates work for a commission, and are paid between 5-10% commissions for each vehicle sold. Sales associates who do not meet their commission target after 12 months are fired.

Parts Manager: The parts manager is responsible for maintaining appropriate inventory levels while ensuring periodic parts turnover. He also adjusts the stock to curtail accumulation of unused parts. The reorder point varies (due to the season, the age the part etc.) throughout the year. When reordering, the parts manager prepares an invoice and sends it to the business manager, who then prepares an ordering invoice. The parts manager is then responsible for faxing the invoice to the manufacturer, tracking the order's progress, receiving the part, notifying the service manager when the part arrives, and submitting the bill to the business manager who then pays the supplier.

Service Manager: Vision Quest Auto also runs a servicing department and the service manager is responsible for managing the workflow in that department, and preparing a weekly schedule for the service technicians. Vision Quest Auto's policy is to keep roughly between 5-15% of the
service technicians’ time available to deal with drop-ins and emergencies. This percentage is not easy to determine since one of the goals of the servicing department is to return customer’s vehicles in the shortest possible time.

**Vice President:** The vice president (VP) job is to ensure that Vision Quest Auto is working progressively towards achieving the long term targets, for example — achieving a 12% increase in customer sales over the next 5 years. Some of the tasks performed for the realization of these goals are:

- **Forecasting Sales** — the VP continuously analyzes the competitors, the growth of the vehicle industry and the value of services it is offering to its customers. He performs quarterly analysis and if the forecasts do not match with the projected forecast, he suggests changes to the short term strategies, for e.g. reducing the profit margin on some vehicles.

- **Bundling of Services** — the VP analyzes the models of vehicles visiting the service department and compares these models with the models that Vision Quest Auto is carrying in its showroom. Vehicle models that visit the service shop most often are deemed “problematic” models. If the profit margins for these “problematic” models are too small, then the VP would suggest to the sales manager to discontinue selling these vehicles. If the profit margin is large, the VP will propose bundling services to customers who purchase the “problematic” models, for e.g. he may suggest offering the customer a bundle which includes free tire alignment and free oil change for one year.

- **Keeping Loyal Customers** — the VP analyzes the frequency of customer visits, whether it is to buy a new part or buy a new vehicle. He also analyzes the demographics of the customers such as age, income, type of vehicle purchased. Based on these data, he will suggest to the sales manager to recognize the customer loyalty by either offering the customer a gift; or a discount on his/her next purchase.
3.1 Development of Goal Template

There are four tasks that have been identified in Requirements Engineering (RE) process, namely requirements elicitation, requirements negotiation, requirements specification, and requirements validation [5]. Kavakli et. al [5] gave the definition of these four activities. According to their definition, requirements elicitation is about identifying stakeholders' requirements and organizational constraints relating to the system under development and understanding the current organizational situation which the system aims to improve. The purpose of requirements negotiation is to reach an agreement on conflicting requirements of the system among stakeholders in the organization. Requirements specification task involves the activity of mapping system requirements and stakeholders' goals onto a requirements model. Last, requirements validation ensures that requirements derived from requirements specification phase comply with stakeholders' goals that have been identified in requirements elicitation.

Given that our research concentrates on translating users' requirements to UML analysis diagrams and the goal template is used mainly to capture system requirements in the form of stakeholders' goals, thus, our goal template is derived by considering the elements and attributes that are mostly used by existing goal analysis techniques in requirements elicitation and specification. In other words, attributes in the goal template that we developed should be commonly used by widely-accepted goal modeling frameworks.

Kavakli et. al [5] reviewed current goal analysis research and categorized existing goal-oriented analysis approaches into the four different RE activities listed above. Goal-based Workflow [19] and i* framework [2] have been considered as requirements elicitation approaches to describe organizational behavior, while GBRAM [8], KAOS [6], and Goal-scenario coupling framework [9] have been
categorized as specification techniques [5]. In the next paragraph, we will discuss these goal-oriented techniques. For further information regarding to these techniques, please refer to literature reviews in chapter two of this thesis.

The organizational framework in Goal-based workflow is a tuple \([G, A, R]\) where \(G\) is a set of goals, \(A\) is a set of actors, and \(R\) is a set of resources. Attributes used in \(i^*\) framework to describe dependency among actors in organizational environment are actors, goals, tasks, and resources. GBRAM applies a goal schema to capture goals and the goal schema includes goals, type, description, action, agents, stakeholders, scenarios, obstacles, preconditions, postconditions, and subgoals. KAOS elaborates requirements by adopting a conceptual meta-model consisted of objects such as entity, relation, event, agent, action, constraint, and goal. Rolland et. al. proposed to use the requirement chunk \(<\text{Goal}, \text{Scenario}>\) to discover goals from scenarios by coupling goal modeling and scenario authoring. Furthermore, by analyzing the disadvantage of above goal modeling frameworks, Singh et. al. [11] proposed an incorporated goal schema to elicit and specify requirements. Attributes used in their schema are primary goal, description, action, agent, stakeholders, constraints, sub goals, and priority.

In addition to taking into consideration the commonly-used attributes in these requirements elicitation and specification techniques, we also based our selection of attributes in the goal template that we used to capture stakeholders’ goals on the criteria that the attributes when combined together should be adequate to capture system requirements in terms of stakeholders’ goals and they should be able to provide appropriate information as input to later phase requirements engineering, for example, developing UML design diagrams.

In Table 3.1, we listed all the elements used by goal analysis techniques for requirements elicitation and specification purposes. The summary is done in a table format where rows are the elements and columns are goal-oriented techniques. We put a check mark in the corresponding cell if certain goal analysis technique uses a
particular element. The last column of the table shows the elements we have incorporated in our goal template because those are the information needed for design. People can feel free to add other elements into the goal template, just the newly-add are not needed for this phase.

Table 3.2 depicts in a tabular format the goal template with seven elements derived based on former discussion. Template ID is simply a sequential number starting from one for ease of tracking. There are a number of goals associated with the system under development and each goal template captures information about one goal. Thus, the final product of goal analysis is a set of goal templates with each one capturing one goal. Action is job description of an actor summarized in one phrase. Action is executed by the organizational actor to satisfy a goal. Action element is included in the template because it contains information about services of an actor, which could be mapped to, for example, operations of a class in design diagrams. Agent is the organizational actor who directly executes the action and realized the goal. Goal is satisfied by the agent through executing the action. Goal is an essential element in the template and it captures users’ requirements. Description of a goal is a decomposition of the action. It consists of detailed steps to carry out the job by agent. Description element indicates the activity flow of an action, which is needed to develop design diagrams such as sequence diagram and activity diagram. It also discloses resources needed and created during execution of the action. Scenario lists alternative ways of realizing the goal or executing action. Last, the stakeholders element includes all actors in the organization that directly contribute to or are influenced by the realization of the goal. This element is necessary as it shows relationships among organizational actors.
Table 3.1: List of Elements Used in Goal Analysis Techniques

<table>
<thead>
<tr>
<th>Elements used in Goal Analysis Techniques</th>
<th>Goal-based Workflow \cite{19}</th>
<th>i* Framework \cite{2}</th>
<th>GBRAM \cite{8}</th>
<th>KAOS \cite{6}</th>
<th>Goal Scenario Coupling \cite{9}</th>
<th>Singh et. al. Goal Schema \cite{14}</th>
<th>Elements Adopted in Our Proposed Goal Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sub-goal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agent</td>
<td>✓</td>
<td>✓ (as Actor)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>✓ (as Task)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Goal Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scenario a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraint</td>
<td>✓ (as Obstacle)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre/post condition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scenario b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(named as Description)</td>
</tr>
<tr>
<td>Entity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

1 The scenario element has been used in two goal analysis methods, GBRAM (Goal-Based Requirements Analysis) \cite{8} and Rolland et. al. Goal-Scenario Coupling approach \cite{9}, however, with different meanings. Definitions of the two scenario elements are as follows:

Scenario a: Scenarios are behavioral descriptions of a system arising from restricted situations. They are useful to evaluate design alternatives \cite{8}.

Scenario b: A scenario is composed of one or several actions, the combination of which describes a unique path leading from initial to final states of agents. The combination of scenarios describes the behavior of a complex system \cite{9}.

---

50
Moreover, it is worth mentioning that we require the *goal* element to be written in the form of Rolland et. al.’s goal structure [9] because Rolland et. al.’s goal structure facilitates analyzing components of goals. As we already discussed in last chapter, a goal is associated to a verb and to one or more parameters (see Figure 2.6) [9]. According to Rolland et. al., it is expressed as a clause with a main verb and several parameters (target, direction, way, and beneficiary), where each parameter plays a different role with respect to the verb [9]. (Please refer to Table 2.1 for definitions of four types of goal parameters and subtypes.) This way, we will be able to identify as much information as possible from analyzing each goal element. For instance, direction parameter indicates the relationships among actors in the organization. Target parameter implies objects used and generated by actors satisfying goals. We will further explain this when developing translation guidelines in the next section of the thesis.

After discussing why the seven elements in Table 3.2 are chosen for the goal template, we would like to further explain the reason that some of the attributes used in existing

<table>
<thead>
<tr>
<th><strong>Template Element</strong></th>
<th><strong>Description of Element</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Sequential number of the goal template.</td>
</tr>
<tr>
<td>Action</td>
<td>Service name or job responsibility that is executed by an actor in the organization to satisfy a goal.</td>
</tr>
<tr>
<td>Agent</td>
<td>Actor in the organization that claims direct responsibility of realizing the goal.</td>
</tr>
<tr>
<td>Goal</td>
<td>Name and content of the goal that is realized by the agent through executing the action.</td>
</tr>
<tr>
<td>Description</td>
<td>Decomposition of the action element, list detailed steps of executing the action.</td>
</tr>
<tr>
<td>Scenario</td>
<td>Alternative ways of carrying out the action or realizing the goal.</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Actors in an organization that directly interact with the actor (the agent listed above in the same goal template) during the realization of the goal.</td>
</tr>
</tbody>
</table>

Table 3.2: Elements of Goal Template
goal analysis techniques are not included in our goal template. *Sub-goal* is defined as other goals that lead to the achievement of the goal [14]. As we require the attribute *goal* to be composed in the form of Rolland et. al.’s goal structure [9], information about sub-goal could be acquired from goal attribute. The same reason applied to the *resource* attribute, which could be derived by analyzing the *goal* in Rolland et. al.’s structure [9]. By using Rolland et. al’s approach, some other attributes such as *entity*, *relation* and *event* could also be indentified from goal element. We will illustrate this in detail in class diagram translation guideline step 3. *Scenario*, which is incorporated in goal template, can be considered as instances of goal *constraint* [8]. Therefore, we didn’t include constraint as a template element. Because we also assume that the goal templates are accurate and they are verified by business professionals before provided to systems analysts as an input to translation process, goal obstacles and alternative ways to fulfill the goal (the scenario element in the template) should already be properly identified. It is thus not necessary to include constraint in goal template. *Precondition* and *postcondition* attributes are useful for describing activity sequencing of a business process. However, elaboration of these two attributes is based on a good understanding of business process sequence in the first place. We thus suggest obtaining the sequence information by visiting organizational actors directly rather than including precondition and postcondition attributes in the goal template.

The Entity-Relationship Diagram in Figure 3.1 below clearly shows the relationships among six goal template elements, namely Action, Agent, Goal, Description, Scenario, and Stakeholders. The arrow in each relationship suggests the direction in which that relationship should be read.
In order to decide what are the needed goal templates and how many templates are needed for a certain business, we refer to an approach developed by Singh et. al. [14]. Since this topic is beyond our research in this thesis, we will just briefly describe the process. First, the Object Oriented Enterprise Model (OOEM) diagram [22, 23, 25, 26, and 27] is developed. Then, goals are elicited from OOEM diagram using Wang’s method [28] and mapped to goal template elements. The methodology proposed in this thesis to draw UML diagrams assumes that all the goal templates are given. These goal templates could be generated in a way similar to the approach introduced above. Next, we will provide two goal template examples derived using Singh et. al.’s approach [14]. With a complete set of goal templates elicited, we will be able to proceed to generate UML diagrams by consulting proposed translation guidelines.

Table 3.3 and table 3.4 are two examples of finished goal template based on auto dealership business case described at the beginning of this chapter. (Please refer to
"Appendix A: Goal Templates for Auto Dealership Case" for a complete set of goal templates for the case.) Goal template #1 captures information related to a goal of business manager. The action, process customer payment for vehicle, is one of the job responsibilities of agent business manager. If agent business manager is modeled as an object class in design diagrams, "process customer payment for vehicle" is then a service of business manager and will be converted to an operation of the class. (Another responsibility of business manager could be process payment via financial institution in template #2.) Action "process customer payment for vehicle" is executed by business manager to realize the goal "Process payment by cash or cheque accurately in a professional and friendly manner to ensure customer satisfaction, promote maintenance contract sales to provide better after sales service, and return customer a payment receipt". It is noticeable that this goal element is composed in a way to conform to Rolland et. al.'s goal structure [9]. The goal contains three clauses and each one begins with a verb (such as process, promote, and return) and is followed by several parameters. For example, the target parameters are "payment by cash or cheque", "maintenance contract sales" and "payment receipt". The phrases "accurately in a professional and friendly manner to ensure customer satisfaction" and "to provide better after sales service" are way parameters, they are the ways in which the goal is achieved. Customer obviously is the beneficiary because he/she is the person benefits from achievement of the goal. Action "process customer payment for vehicle" is decomposed into three detailed steps to be completed by business manager. When a customer requests to make a payment either in cash or cheque, business manager first accepts the payment, provides the customer with a receipt of the transaction, and last, promote maintenance contract to customer. This activity flow of "process customer payment for vehicle" is depicted in description element. Resources used and created during execution of "process customer payment for vehicle" can also be conceived from analysis of description element. For example, resources used while business manager processes customer payment are cash payment and cheque payment and resources created are payment receipt and maintenance contract. Actors who directly contribute to or benefit from business manager's
"process customer payment for vehicle" action are business manager and customer. This relationship between business manager and customer should be reflected in later design diagrams. Regarding to the scenario element, another way that customer could make a payment is to pay through a financial institution. As a result, in addition to processing cash and cheque payments, business manager alternatively should be able to process payments via banks. It is an alternative way to contribute to the strategic goal of increasing sales and customer satisfaction. The action "process payment via financial institution" is captured in goal template #2.

<table>
<thead>
<tr>
<th>ID</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action</strong></td>
<td>Process customer payment for vehicle</td>
</tr>
<tr>
<td><strong>Agent</strong></td>
<td>Business Manager</td>
</tr>
<tr>
<td><strong>Goal</strong></td>
<td>Process payment by cash or cheque accurately in a professional and friendly manner to ensure customer satisfaction; promote maintenance contract sales to provide better after sales service; return customer a payment receipt</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>When the customer requests to pay, business manager 1. accepts the payment by cash or cheque 2. returns customer a payment receipt 3. promotes maintenance contract sales</td>
</tr>
<tr>
<td><strong>Scenario</strong></td>
<td>Customer requests to pay through a financial institution.</td>
</tr>
<tr>
<td><strong>Stakeholders</strong></td>
<td>Business Manager, Customer</td>
</tr>
</tbody>
</table>

Table 3.3: Goal Template Example – Template #1

<table>
<thead>
<tr>
<th>ID</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action</strong></td>
<td>Process payment via financial institution</td>
</tr>
<tr>
<td><strong>Agent</strong></td>
<td>Business Manager</td>
</tr>
<tr>
<td><strong>Goal</strong></td>
<td>Process and confirm payment through financial institution accurately in a professional and friendly manner to ensure payment authenticity and customer satisfaction and return customer a payment receipt</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>When the customer requests to pay through a financial institution, business manager 1. confirms payment with the financial institution 2. returns customer a payment receipt</td>
</tr>
<tr>
<td><strong>Scenario</strong></td>
<td>Customer makes payment by cash or cheque.</td>
</tr>
<tr>
<td><strong>Stakeholders</strong></td>
<td>Business Manager, Customer, Financial Subsidiary</td>
</tr>
</tbody>
</table>

Table 3.4: Goal Template Example – Template #2
3.2 UML Diagrams Translation Guidelines

Unified Modeling Language (UML) is a widely adopted industry standard language for information systems analysis and design. However, as discussed earlier, since it fails to provide mechanisms for describing business processes and surroundings, UML is not easily understandable for business professionals. As a consequence of this, it is not adequate for early phase requirements engineering, which concentrates on analyzing stakeholders' goals and requirements and usually requires active participation of business professionals. Our proposed UML diagrams translation guidelines aim to solve this problem. Complying with the translation guidelines, system analysts could develop systems analysis diagrams with the help of goal template approach. Because goal templates are able to capture users' goals and other business context, translation guidelines enable the derived UML diagrams capable of reflecting business environment.

In this section, we present our structured guidelines for translating stakeholders' requirements captured using our goal template to six UML design diagrams, namely use case diagram, class diagram, sequence diagram, collaboration diagram, activity diagram and state chart diagram. Guidelines are developed based on the following steps: 1) list elements used in each UML diagram; 2) analyze goal template to identify elements in the goal template that contain necessary information to develop UML diagram elements; and finally, 3) formalize the analysis and translation process in step 2 to develop translation guidelines and draw UML design diagrams.
3.2.1 Translation Guideline for UML Use Case Diagram

_use_case_diagram_introduction_

A use case diagram captures interactions among organizational players. It helps system analysts understand users' functional requirements [20]. Elements used in a use case diagram to show interactions in the organization are _actors_, _use cases_, and _use case relationships_. According to Fowler et. al.'s definition of actors, an actor is a role that a user plays with respect to the system. In other words, an actor is a role that a user has in relationship to the system [21]. Actors carry out use cases. A use case is one use of the system by an actor [21]. One actor may perform many use cases while one use case may have a couple actors using it. With regard to use case relationships, in addition to the relationships among use cases and actors, there are three types of relationships among use cases. An _include_ relationship between two use cases indicates that one use case can use the other one [21]. The include relationship converges similar behavior across more than one use cases [20]. An _extend_ relationship shows that one use case, which is called base use case, is extended with additional behavior by extending use case [21]. The extension can be considered as an optional or a special functionality added to the base use case. _Generalization_ among use cases is used to imply that one use case is specialized to another use case that inherits and adds additional functionalities to it [21]. According to Fowler et. al. [20], generalization relationship is similar to extend relationship but with less rigorous rules to it. They both describe variations on normal behavior with extend relationship explicitly specifying the situation when to extend the base use case.

_Actor_ element in the goal template is a summary of an actor's job and it is performed by the actor. Action contains information about services of an actor in relation to the system. Therefore, each action element could be mapped to one use case and all use cases combined suggest different aspects of the usages of system. _Agent_ is the actor
who directly executes the action and *stakeholders* include all actors that directly contribute to or benefit from the action. Since an actor in use case diagram is a role that a user has with respect to the usage of system, agent and stakeholders elements in our goal template convey relative information to derive actors associated with use cases in UML use case diagram. *Description* of a goal decomposes the action into detailed steps to carry out an action. It indicates activity flow of an action, which shows similar actions in terms of similar activities. Given that actions are converted to use cases, further analysis of description should identify "include" relationships between use cases, where two or more use cases include similar activities and thus, use a common use case. Finally, goal template element *scenario* lists alternative ways of executing a use case. In effect, this gives us hints to derive extend and generalization relationships among use cases, both of which captures variant ways of realizing a use case.

*Use Case Diagram Translation Guideline*

Next, we will explain our translation guidelines in further detail using the auto-dealer case as an example.

1. **Identify use cases.**

Map goal template element *action* in each goal template to one use case. As we have already discussed, action element is a service of an actor and it is one aspect of usage of the system under development by the actor. The result of this step is the same number of use cases as the number of goal templates. For instance, our auto dealership business case contains 19 goal templates to capture users' goals and requirements. As a result, there should be 19 use cases, such as "Process customer payment for vehicle" from template 1, "Process payment via financial institution" from template 2 and "Process sales commission update" from template 4, after this step. Some of these use cases need to be combined together because some of them are transaction-focused operational level services while use case
diagram usually represents an external view of the system [20]. However, analysts by now do not need to worry about this. It’s a concern of step 3 and we will solve the problem then.

2. **Identify actors associated with each use case.**

   For each goal template, organizational actors listed in the element *stakeholders* are actors associated with the action. Organizational actors are associated with the action in a goal template in terms of direct contribution or benefit, which is considered as a kind of relationship that actors have with the system. Actions have been mapped to corresponding use cases in step 1. Therefore, associations can be established between use case actors and use cases. For example, stakeholders in goal template 1 are customer and business manager and they are mapped to actors in use case diagram. Furthermore, template 1 action is translated to a use case “Process customer payment for vehicle” and therefore, “Customer” and “Business Manager” are actors associated with “Process customer payment for vehicle” use case.

3. **Aggregate use cases.**

   By analyzing the element *description* in goal template, find similar actions that could be combined together into a single action. According to Wand et. al. [22], actions are similar if they have similarity in internal transformations (similar processes) or similarity in state information (several steps in a continuous business process). Identify these similar actions and aggregate corresponding use cases into one. In the auto dealership case, by looking at the description elements in goal template 8, 9, 10 and 17, we can see that they are four continual steps for processing vehicle purchase. Then, the four use cases, “Outline procedures for purchasing vehicle”, “Process purchasing request”, “Inspect vehicle for final time” and “Process scheduling”, are incorporated into one single use case “Process vehicle purchase”. Use cases “Process customer payment for vehicle”, “Process payment via financial institution” and “Process parts payment” are
similar business processes and are combined into one use case “Process customer payment”.

4. **Identify <<include>> relationships.**

Apply strong cohesion criteria in Good Decomposition Model [16, 17] to use case diagram. Strong cohesion criteria says the larger the number of similar methods, the more cohesive the class [18]. Using <<include>> relationship shows similar use cases that have similarity in internal transformations (similar processes) [22]. For instance, use cases “Process vehicle purchase”, ”Process parts purchase”, and ”Process used-vehicle purchase” are similar in terms of payment process. Thus, by including a use case “Process customer payment” into these three use cases indicates that the three use cases, “Process vehicle purchase”, ”Process vehicle purchase”, and ”Process used-vehicle purchase”, contain a similar action “Process customer payment”.

5. **Identify <<extend>> relationships.**

In UML use case diagram, the <<extend>> relationship is used to describe variation on normal behavior [20]. Moreover, in our goal template, element *scenario* indicates alternative ways of accomplishing an action. Therefore, <<extend>> relationship can be built based on the analysis of information in the *scenario* element. In our case, customer makes payments in cash or cheque to business manager, which is captured in template 1. Alternatively, customer could make a payment through a financial institution, which is indicated in the scenario of template 1. And the task of business manager accepting customer payments made through banks is further illustrated in detail in template 2. As shown in the description elements of template 1 and template 2, the step of business manager processing payments made via a bank is similar to processing cash and cheque payments, except that in former case, business manager needs to confirm with the financial institution to verify the payments. In Figure 3.2, “Process customer payment” is extended by use case “Confirm with financial institution”.
Summary

Figure 3.2 is the use case diagram developed for the auto dealership case according to the guidelines described above. The five steps allow use case diagram to capture the essence of a business. Based on goal template element action and element description respectively, step 1 and step 3 provide in use case diagram information about business activities occurring in the organization. Step 2 derives organizational actors that are related to each business activity from template element stakeholders. The three steps combined together show in the diagram business activities and their relationships with relevant organizational actors. Step 4 and step 5 enable the use case diagram to explicitly represent relationships among certain business activities, where step 4 applies Good Decomposition Model and step 5 relies on goal template element scenario.
Figure 3.2: Use Case Diagram for Vision Quest Auto Dealership Case
3.2.2 Translation Guideline for UML Class Diagram

Class Diagram Introduction

UML Class diagram describes objects in a system, attributes and operations of an object and static relationships among them [20]. Systems are built from objects, which can be physical or abstract, and objects are described by their attributes and relationships to other objects [21]. Objects also have behavior, which is described with operations attached to them. A class is a set of objects with the same characteristics [21]. Classes are modeled with names, attributes, and operations, services that the class provides. Operations are the processes that a class carries out and are used to indicate the primary responsibilities of that class, summarized using a couple words [20]. For example, business managers need to process customer payments. A business manager class could, thus, represent one or several business managers (objects) in Vision Quest case, use “Business Manager” as its name, possess “Employee ID” attribute and have “process vehicle purchase payment” and “confirm payment with banks” operations.

Classes are related to each other by relationships and there are two types of relationships, association and generalization [20]. Associations are given names and are represented in class diagram with lines drawn between two classes. Each association has two association ends with each connected with one of the classes participating in the association relationship [20]. Each association end also has multiplicity, which implies the number of objects that can participate in the particular relationship [20]. Generalization is a special kind of relationship between classes. Two classes that are related with a generalization relationship have differences but also many similarities [20]. According to Fowler et. al. [20], the similarities can be put in super-type class, with subtype class showing the differences. In other words, subclass,
in addition to its own attributes, operations and relationships with other classes, also
inherits the attributes and operations from superclass. In a generalization relationship,
all instances of subclass are instances of superclass. Take Vision Quest dealership case
as an example, whether customers pay by cash or pay through banks, business
manager should return customers a receipt as a proof of payment transaction.
Therefore, two different types of receipts are required, where the receipt for bank
payment needs to include bank name or bank ID, but two kinds of receipts share some
similar attributes, as both are records of payment transactions, and are both subclasses
of the superclass, payment receipt, with common attributes, such as customer ID and
payment amount.

Organizational actors listed in agent and stakeholders elements of our goal template
are part of an organization. They perform tasks to provide services to other parties in
the organization, which results in relationships with each other. These actors could be
modeled as classes in a class diagram. Since operations are responsibilities of a class
and template element action is a summary of job description of an actor who is
converted to a class, services in the action element could be modeled as operations of
the class that the corresponding agent in the same goal template is mapped to. In
Rolland et. al.'s goal structure [9], a goal is associated to a verb and one or several of
the four parameters, namely target, direction, way, and beneficiary. For instance,
target parameter is an indication of objects used and generated by actors during
performing responsibilities. Therefore, in addition to template element description
which discloses resources needed and created during the action, in-depth analysis of
goal element in the goal template will also discover relative information to derive
attributes in a class and additional classes. Furthermore, direction parameter identifies
initial and final objects of a communication process. It indicates relationships among
actors in the organization. Way and beneficiary parameters will also be helpful to
build association and generalization relationships among classes.
Class Diagram Translation Guideline

1. Identify classes.

Map actors listed in the elements *agent* and *stakeholders* of our goal template to classes in a class diagram. These actors constitute partial system. Actors in the agent element provide services by performing the job given in the template element action, which facilitates to establish relationships with people in template element stakeholders because stakeholders claim direct contribution to or benefit from agent performing the action. As seen in Figure 3.3, Vision Quest actors given in agent and stakeholders elements from all 19 goal templates, such as Business Manager, Sales Associate, and Customer, are all converted to UML classes with respective class names.

2. Identify operations in each class.

For each *agent* element in the goal template, element *action* is translated to an operation in the same class that the agent is mapped to in previous step 1. Action could be considered as an operation of a class because the corresponding agent is responsible for performance of the action to satisfy a goal, which is equivalent to the definition of an operation of a class (an operation is defined as a process carried out by a class to fulfill a responsibility of that class [20]). Goal template #1 (see Table 3.3), for instance, indicates that agent “business manager” is in charge of the action “processing customer payment for vehicle”. As a result, Figure 3.3 shows that “process vehicle purchase payment ()” is modeled as an operation in “business manager” class. “Confirm payment with financial subsidiary ()” is identified as another operation in “business manager” class as indicated in goal template #2 (see Table 3.4) that business manager is responsible for the process of payment made through a financial institution.

3. Identify attributes, additional classes and associations among classes.

Take each goal template and do the following step 3.1 through step 3.4 to derive
any additional classes that missed in step 1 as well as attributes of classes and association relationships between every two classes. These steps first require that the template element *goal* is written in the form of Rolland et al.'s goal structure [9].

### 3.1. Analyze each primary goal using Rolland et al.'s goal structure approach.

Identify the four parameters of each *goal* element and summarize them in the Goal Analysis Form (Table 3.5). According to Rolland et al. [9], a goal is associated to a verb and to one or more parameters (see Figure 2.6). It is expressed as a clause with a main verb and several parameters (target, direction, way and beneficiary), where each parameter plays a different role with respect to the verb. Table 2.1 summarizes the definition of four types of goal parameters and subtypes. Goals written in such structure facilitate to get the largest possible amount of information from analyzing the components of each goal element as proved in later steps 3.3 and 3.4. Table 3.6 gives two examples of goal analysis result. A complete list of goal analysis form is provided in Appendix B: Goal Analysis Form for Vision Quest Auto Dealership Case.

<table>
<thead>
<tr>
<th>Goal ID</th>
<th>Target</th>
<th>Direction</th>
<th>Way</th>
<th>Beneficiary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object</td>
<td>Result</td>
<td>Source</td>
<td>Dest.</td>
</tr>
</tbody>
</table>

Table 3.5: Goal Analysis Form
3.2. Categorize the Target column in the Goal Analysis Form.

Based on the analysis of goal structure in step 3.1 and the template element description in the goal template, categorize the Target parameter in Table 3.5, both Object and Result subtypes, into Category 1 if 1) it is a static object and it's manipulated by active objects; (As opposite to an active object, which is defined as a representative or a service provider, a static object is used to represent an inactive thing in information systems [23].) or 2) it is an event that needs to be recorded so as that stakeholders could retrieve state information about the event later on. Otherwise, if the Target parameters do not fall into any of the above two classifications, classify them into Category 2. Object parameter “cash payment” in Table 3.6 is grouped in Category 1 because customer payment is an event and Vision Quest needs to track this information to report vehicle sales and forecast its future sales. Same reason also applies to “payment via banks” object. On the contrast, the Result
parameter "receipt" is sorted in Category 2 even though it could be considered as a static object; it is not manipulated by any active object. In fact, receipt is a part of a successful payment process and it is the "payment" that is directly manipulated by business manager not "receipt". A successful "payment via banks" process involves "payment confirmation" and business manager directly operates "payment via banks" service, thus "payment confirmation" is also categorized into Category 2. As for "customer satisfaction", Vision Quest stores this information to ensure its long-term strategy but customer satisfaction is not an event, therefore it's been put in Category 2.

3.3. Identify attributes and additional classes.

Target parameter implies objects used or generated by actors satisfying goals. All the Target parameters that were already categorized in Category 1 are modeled as classes in the class diagram in addition to those identified in step 1 and Targets in Category 2 become attributes in related classes. The reason for this will be provided. Wand et. al. [24] claimed that, in ontology, an event is a change of state of a thing and should not be modeled as an entity or object. However, they further stated that, from a database design viewpoint, an event can be represented as a record and could be modeled as an entity in an ER diagram or an object in an object-oriented diagram [24]. Static objects provide a way to model the state of things that users require to keep [23]. Static objects indicate what information needs to be stored in the system. The state of a static object can only be changed by active objects. In Figure 3.3, "cash payment" in Category 1 is transferred to a class called "Purchase Payment (Cash/cheque)". "Receipt" in Category 2 is modeled as an attribute in the "Payment Record" class.

The classes that are derived as a result of step 1 and step 3.3 fall into three different types. A class in the class diagram is either an organizational actor, or a static object, or a record of an event.
3.4. Identify associations between classes.

There are two complementary steps to identify associations among classes. First, the Direction and Way parameters in Goal Analysis Form (Table 3.5) indicate the actors that communicate with each other in each goal achievement process. These communication relationships can be used to derive associations among organizational actors and they are modeled as association relationships between respective classes in the class diagram. While Direction parameter implies the initial and final classes of an association relationship, Way tells us how the initial and final classes are associated. Second, according to Wand et. al. [23], static objects are directly manipulated by active objects. As a result, in addition to the associations built using the above approach, associations are also developed between classes derived from Category 1 in step 3.3 and classes that are responsible for changing the states of these static objects. In the Goal Analysis Form for Template #1, the Direction parameter says that business manager and customer communicate with each other in the payment process. Therefore, there should be an association relationship between the class “Business Manager” and “Customer” class. Way parameter further discloses that these two classes are connected in a way that business manager processes customer’s cash or cheque payment for vehicle purchase and returns a payment receipt to the customer. Because “Purchase Payment (Cash/cheque)” is mapped to a class in step 3.3 and it is a static class that needs to be operated by business manager, the “Business Manager” class is associated to “Customer” class by “Purchase Payment (Cash/cheque)” class as indicated in Figure 3.3.

In addition, we consider it is necessary to clarify that there should not be any direct association between two actor type classes. (As discussed earlier, classes are classified into three types: actor, static object, record of an event.) The above two complementary sub-steps in step 3.4 are able to ensure that
actor type classes are linked via either static object type classes or event record
type classes. In other words, the Direction attribute in Table 3.5 (Goal Analysis
Form) implies actor type classes that interact with each other in a certain
business activity. Way attribute in Table 3.5 can be used further to derive the
static object class or event record class that links these two actor type classes.
Actor classes are associated by static object classes because it is the actor
classes' responsibility to manipulate the states of static classes. Actor classes
are related by event record classes because events are triggered by actors.

3.5. **Go through the Goal Analysis Form to ensure complete associations.**

Repeat step 3.4 to associate all classes identified in step 1 and step 3.3.

4. **Examine the evolving class diagram to build generalization relationships.**

As the translation process goes on, realize generalization relationship and add a
superclass for the classes that a) are modeled to represent records of similar events
OR b) share common attributes with different ones to differentiate themselves. As
indicated in Figure 3.3, “Invoice Record” in the generalization relationship is a
superclass of four other classes (“Purchase Payment (Cash/cheque)”, “Purchase
Payment (via Bank)”, “Parts Payment” and “Reorder Payment”) which record
similar payment events and share some common attributes, such as payment
amount and receipt ID.

5. **Refine the diagram by revisiting stakeholders.**

By revisiting and confirming with stakeholders in the organization, delete
redundant attributes that no operations use; remove unnecessary associations
between any two classes; ensure to include emergent attributes. Possible questions
that could be asked are a) information and resources used in the process of action
carry-out; b) all the other stakeholders communicated with while completing the
action; c) information and resources generated as a result of action realization; d)
products or services produced as a result of action realization.
Multiplicity of an association will not be derived directly from goal template using the guidelines we proposed here. However, this is considered to be acceptable [13] because the class diagram developed in this thesis serves primarily as a conceptual model. If necessary, information about multiplicity of a certain relationship could be gathered by consulting with respective stakeholders who participate in the relative relationship.

Summary

Figure 3.3 is the class diagram completed by complying with above class diagram translation guideline. Step 1 and step 3.3 together results in the differentiation of objects in a business. Three types of class (actor type, static object type, and event record type) are represented using different legends in the class diagram. Step 1 maps goal template elements agent and stakeholders to actor type classes, while step 3.3 identifies static object type and event record classes by analyzing template element goal in step 3.1 and step 3.2. Step 2 shows business activities, which are essential to support various business processes, of each organizational actor by translating template element action to related actor type classes. Using Goal Analysis Form (see Table 3.5), step 3.4 identifies interactions of objects in a business and ensures that actor type classes are associated to each other through a static object or an event record class, which clearly indicates the actor relationships in a business context and how actors interact in a certain business process. In summary, translation guideline increases class diagram’s ability to reflect the core business of an organization.
Figure 3.3: Class Diagram for Vision Quest Auto Dealership Case
3.2.3 Translation Guideline for UML Sequence Diagram

Sequence Diagram Introduction

Interaction diagram models the way a group of objects collaborate with each other in a certain business process. This section discusses one kind of interaction diagrams, sequence diagrams, and how to develop sequence diagrams for a business process from goal templates. The other type of interaction diagrams, collaboration diagrams, will be discussed in the next section. Generally speaking, one interaction diagram captures the behavior of objects in a single use case [20]. It depicts a number of objects and messages that are communicated by these objects to each other within the use case [20].

A sequence diagram visually displays the sequence in which system or organizational objects communicate with each other to complete a business process. Objects in a sequence diagram are drawn as rectangular, with each at the top of a vertical object's lifeline. The lifelines show when objects are created and destroyed during the interaction [21]. Each passed message is represented as an arrow, beginning at the communication initiator and ending at the message recipient, between the lifelines of two interacting objects [20]. Typically, the sequence and order of messages, in which these messages are passed between objects, is clearly depicted from top to bottom. An activation box is included and drawn overlapping the lifeline to show when an object is active during the interaction process [20].

Each sequence diagram depicts the behavior of actors and sequence of messages within only one use case. Once decided which use case to represent and related goal templates, all the stakeholders attributes in the associated goal templates convey the organizational actors that participate in the use case, and in result can be transferred to
objects in the sequence diagram. Moreover, as *description* attribute decomposes an actor’s job in a business process into detailed activity flows, in-depth analysis of description attributes in all the goal templates related to a certain use case can be used to derive information about the messages and sequence of these messages passed between participating objects in a business process.

*Sequence Diagram Translation Guideline*

Sequence diagrams, showing interactions and sequence of interactions among organizational actors, could be drawn for primary use cases in a system. One sequence diagram corresponds to one use case that is derived in section 3.2.1. The first thing to develop a sequence diagram from goal templates is to choose a use case of users' interest and proceed to drawing the sequence diagram so as to depict actors' interactions within that particular use case.

1. **Identify related goal templates.**

   This first step in deriving a sequence diagram from goal templates is to choose a certain use case from the use case diagram, in which users are interested in knowing further detailed interactions. Trace back to identify all the goal templates that are related to that use case, including templates relating to any use case that is in a *include* or *extend* relationship with the chosen use case. As an example, customer’s purchase of vehicle is a major business activity and profit source for Vision Quest. It’s of great importance to show the organization and its management detailed interactions among objects in the “process vehicle purchase” use case. Once we have reached a decision on which use case to depict in detail, goal templates that are related to “process vehicle purchase” use case are identified. These templates are template #1, 2, 8, 9, 10 and 17.

2. **Aggregate description elements.**

   With all related goal templates derived from previous step, combine all their
*description* elements into one sequential description based on the sequence of business process described in the description elements. Additional knowledge about a certain business process sequence is needed when combining the description elements. This knowledge could be obtained by revisiting related organizational actors and relevant business process documents. Number the steps in the combined description, starting from one (e.g. 1, 2...). The first two columns in Table 3.7 present the final result for “process vehicle purchase” use case in our auto dealership case.

<table>
<thead>
<tr>
<th>#</th>
<th>Description of Step</th>
<th>Sender</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A customer enters the showroom</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>Customer ask sales associate questions</td>
<td>Customer</td>
<td>Sales associate</td>
</tr>
<tr>
<td>3</td>
<td>Customer request sales associate to arrange a test drive</td>
<td>Customer</td>
<td>Sales associate</td>
</tr>
<tr>
<td>4</td>
<td>Sales associate outlines purchase procedure</td>
<td>Sales associate</td>
<td>Customer</td>
</tr>
<tr>
<td>5</td>
<td>Customer decides to make a purchase</td>
<td>Customer</td>
<td>Sales associate</td>
</tr>
<tr>
<td>6</td>
<td>Sales associate completes a sales form</td>
<td>Sales associate</td>
<td>Customer</td>
</tr>
<tr>
<td>7</td>
<td>Sales associate arranges a new vehicle cleaning schedule with service manager</td>
<td>Sales associate</td>
<td>Service manager</td>
</tr>
<tr>
<td>8</td>
<td>Service manager provides a cleaning schedule</td>
<td>Service manager</td>
<td>Sales associate</td>
</tr>
<tr>
<td>9</td>
<td>Service manager ensures technicians clean the vehicle before the scheduled date</td>
<td>Service manager</td>
<td>Service manager</td>
</tr>
<tr>
<td>10</td>
<td>Service manager requests sales associate to perform a final inspection before delivering the vehicle to customer</td>
<td>Service manager</td>
<td>Sales associate</td>
</tr>
<tr>
<td>11</td>
<td>Customer returns</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>12</td>
<td>Sales associate asks customer to pay business manager</td>
<td>Sales associate</td>
<td>Customer</td>
</tr>
<tr>
<td>13</td>
<td>If customer requests to pay by cash or cheque, business manager accepts the payment by cash or cheque</td>
<td>Customer</td>
<td>Business manager</td>
</tr>
<tr>
<td>14</td>
<td>If customer requests to pay through a financial institution, business manager confirms payment with the financial institution</td>
<td>Customer</td>
<td>Business manager</td>
</tr>
<tr>
<td>15</td>
<td>Business manager returns customer a payment receipt</td>
<td>Business manager</td>
<td>Customer</td>
</tr>
</tbody>
</table>

75
<table>
<thead>
<tr>
<th>#</th>
<th>Description of Step</th>
<th>Sender</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Business manager promotes maintenance contract sales</td>
<td>Business manager</td>
<td>Customer</td>
</tr>
<tr>
<td>17</td>
<td>Customer returns receipt to sales associate</td>
<td>Customer</td>
<td>Sales associate</td>
</tr>
<tr>
<td>18</td>
<td>Sales associate confirms payment receipt</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>19</td>
<td>Sales associate delivers keys and related documents to customer</td>
<td>Sales associate</td>
<td>Customer</td>
</tr>
</tbody>
</table>

Table 3.7: Combined Description of Process Vehicle Purchase Use Case (sequence diagram)

3. Identify participating objects.
   
   Element **stakeholders** in the goal templates identified in step 1 become objects in the sequence diagram. Actors in **stakeholders** element are actors who directly participate in task execution. They interact with each other to complete a business process and communicate by sending messages to one another. Objects identified for "process vehicle purchase" use case are customer, sales associate, service manager, business manager and financial institution.

4. Generate messages and identify sender and recipient of each message.
   
   Basically, each numbered step in the combined description, such as the one in Table 3.7, corresponds to one message that starts from message sender and ends at message recipient. Object that initiates each request is considered as message sender and object that handles the request is message recipient. It is important to mention that message senders and recipients must be among the participating objects derived in step 3. The last two columns in Table 3.7 summarize the sender and recipient of each message.

5. Represent alternatives.
   
   Messages in a sequence diagram can have parameters and guards [21]. Use condition expression to represent alternative ways of finishing a task within a use
case. These alternatives are indicated by \textit{scenario} element in the goal template. For example, a customer can make payment by cash/cheque (Goal Template #1) or via a financial institution (Goal Template #2). Therefore, use the condition expression "\textit{payment: = financial institution ()}" to show the alternative.

\textit{Summary}

Figure 3.4 shows the sequence diagram for "process vehicle purchase" use case developed using our sequence diagram translation guideline. Sequence diagram developed according to guideline provides a clear view of the business process, which shows both participating actors and communications relationships among these actors. Step 3 identifies objects participating in a certain business process by analyzing goal template element stakeholders. Based on template element description, step 2 and step 4 reinforce the sequence diagram to clearly reflect actor communication relationships in a business process and requests and messages sent among them during carrying out of business activities.
Figure 3.4: Sequence Diagram for Vision Quest Auto Dealership Case
3.2.4 Translation Guideline for UML Collaboration Diagram

Collaboration Diagram Introduction

Collaboration diagram is the second type of interaction diagram. Similar to sequence diagrams, collaboration diagrams also represent how objects interact and collaborate within a certain use case. While sequence diagrams focus on the sequences of interactions, collaboration diagrams place an emphasis on depicting the iterations and collaborations among objects [21]. According to Eriksson et. al. [21], collaboration diagrams are more powerful than sequence diagrams in terms of representing more complex interactions and showing the relationships between the collaborating objects.

Same as in a sequence diagram, the messages passed between objects are represented as arrows. However, the sequence of sending messages in a collaboration diagram is indicated by numbering the messages. UML adopts decimal numbering method [20], beginning with 1, followed by 1.1, 1.1.1, 1.2, 1.2.1, 1.3, 1.3.1 and so on. Although this makes it more difficult to see the overall order of messages as in a sequence diagram, numbering the messages more easily shows the interactions among objects and how the objects are connected [20]. In other words, the numbering of messages in a collaboration diagram clearly indicates which messages must be fulfilled before which other messages can be sent [21]. For example, message 1.1.1 must be sent and accomplished before message 1.2.

Typically, a collaboration diagram of a use case contains same amount of information as a sequence diagram of that use case. Collaboration diagrams can be derived from sequence diagrams. However, a major difference between the two kinds of interaction diagrams is that, instead of implying the sequence of messages from top to bottom in a sequence diagram, sequence in a collaboration diagram is indicated by numbering
the messages.

*Collaboration Diagram Translation Guideline*

1. **Re-number the description steps.**
   
   Use decimal numbering scheme to re-number the steps in combined description of a use case, like the one shown in Table 3.7. The first column in Table 3.8 shows the numbering of messages in the collaboration diagram for “process vehicle purchase” use case. The second column in Table 3.8, description of each step, is the same as in Table 3.7 for corresponding sequence diagram of “process vehicle purchase” use case.

2. **Identify interacting objects.**
   
   Objects are the same as those in the sequence diagram. As indicated in Figure 3.5, objects identified for the collaboration diagram of “process vehicle purchase” use case are customer, sales associate, service manager, business manager and financial institution, same as those in the corresponding sequence diagram (Figure 3.4).

3. **Generate messages and identify sender and recipient of each message.**
   
   Each re-numbered step is mapped to a message passed between any two objects. And the associated sender and recipient of each message are the same ones for the equivalent message in a sequence diagram. For a particular message in Table 3.8, the message sender and message recipient are identical to the sender and recipient of equivalent message in Table 3.7.
<table>
<thead>
<tr>
<th>#</th>
<th>Description of Step</th>
<th>Sender</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Customer ask sales associate questions</td>
<td>Customer</td>
<td>Sales associate</td>
</tr>
<tr>
<td>1.1</td>
<td>Customer request sales associate to arrange a test drive</td>
<td>Customer</td>
<td>Sales associate</td>
</tr>
<tr>
<td>1.1.1</td>
<td>Sales associate outlines purchase procedure</td>
<td>Sales associate</td>
<td>Customer</td>
</tr>
<tr>
<td>1.2</td>
<td>Customer decides to make a purchase</td>
<td>Customer</td>
<td>Sales associate</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Sales associate completes a sales form</td>
<td>Sales associate</td>
<td>Customer</td>
</tr>
<tr>
<td>1.2.1.1</td>
<td>Sales associate arranges a new vehicle cleaning schedule with service manager</td>
<td>Sales associate</td>
<td>Service manager</td>
</tr>
<tr>
<td>1.2.1.2</td>
<td>Service manager provides a cleaning schedule</td>
<td>Service manager</td>
<td>Sales associate</td>
</tr>
<tr>
<td>1.2.1.3</td>
<td>Service manager ensures technicians clean the vehicle before the scheduled date</td>
<td>Service manager</td>
<td>Service manager</td>
</tr>
<tr>
<td>1.2.1.4</td>
<td>Service manager requests sales associate to perform a final inspection before delivering the vehicle to customer</td>
<td>Service manager</td>
<td>Sales associate</td>
</tr>
<tr>
<td>1.3</td>
<td>When customer returns, sales associate asks customer to pay business manager</td>
<td>Sales associate</td>
<td>Customer</td>
</tr>
<tr>
<td>1.3.1</td>
<td>If customer requests to pay by cash or cheque, business manager accepts the payment by cash or cheque</td>
<td>Customer</td>
<td>Business manager</td>
</tr>
<tr>
<td>1.3.1.1</td>
<td>If customer requests to pay through a financial institution, business manager confirms payment with the financial institution</td>
<td>Customer</td>
<td>Business manager</td>
</tr>
<tr>
<td>1.3.2</td>
<td>Business manager returns customer a payment receipt</td>
<td>Business manager</td>
<td>Customer</td>
</tr>
<tr>
<td>1.3.2.1</td>
<td>Business manager promotes maintenance contract sales</td>
<td>Business manager</td>
<td>Customer</td>
</tr>
<tr>
<td>1.4</td>
<td>Customer returns receipt to sales associate</td>
<td>Customer</td>
<td>Sales associate</td>
</tr>
<tr>
<td>1.4.1</td>
<td>Sales associate confirms payment receipt, delivers keys and related documents to customer</td>
<td>Sales associate</td>
<td>Customer</td>
</tr>
</tbody>
</table>

Table 3.8: Combined Description of Process Vehicle Purchase Use Case (collaboration diagram)
After identifying the messages communicated within the use case "process vehicle purchase", the sequence of messages passed among objects and corresponded message sender and recipient, the collaboration diagram (Figure 3.5) for that use case can be developed.

Summary

Step 2 identifies participating objects in a business process from sequence diagram, in which participating objects are derived from goal template element stakeholders. Step 1 and step 3, with the information from template element description, substantiate the interaction relationships among participating actors in the business process.
Figure 3.5: Collaboration Diagram for Vision Quest Auto Dealership Case
3.2.5 Translation Guideline for UML Activity Diagram

Activity Diagram Introduction

Activity diagrams are used to describe workflows and business processes. The major element in an activity diagram is the activity. An activity is a state in which some activity is performed [21], either a business process or a method of a class. An activity diagram represents sequencing flow of activities and it's able to reflect both conditional and parallel behavior [20]. According to Fowler et. al., conditional behavior could be depicted by branches and merges [20]. A branch has one incoming transition and multiple mutual exclusive outgoing transitions. Mutual exclusive means only one outgoing transition can be taken. On the other hand, a merge has several input transitions and one output and it implies the end of a conditional behavior which is started by a branch. Regarding parallel behavior, it can be delineated by forks and joins [20]. A fork has one incoming transition and several simultaneous outgoing transitions that occur in parallel. A join has one single outgoing transition and it occurs only after all incoming transitions have been completed. A join must be used in pair with a fork to join together the threads started by that fork [20]. A swimlane groups activities with respect to their responsibility [21]. It's a way to label each activity with the responsible class or actor [20]. Swimlanes explicitly indicate where or in which part of an organization the activities are performed.

Activity is a core building block of an activity diagram. To decide the activities and the control flow of these activities in a business process, goal template element action and description provide some clue. Especially because the description element is a decomposition of an action, it indicates the activities needed to accomplish a certain action and the sequential steps of performing the action. Analysis of description element and further consulting with system users would enable to discover parallel
behaviors that are independent to one another and could occur in parallel. For conditional behaviors in a workflow, template element *scenario* implies alternative ways of accomplishing an action and can be relied on to derive branch and merge bunch. Swimlane is the delegate of responsible actor of an activity and goal template element *agent* lists the actor who directly executes the activity. It includes relevant information to draw swimlane.

*Activity Diagram Translation Guideline*

1. **Identify related goal templates.**
   
   Before developing an activity diagram, we need to decide which business process or workflow to reflect. The business process chosen to be further depicted should be of great value to the organization and its business. As use case diagram is an overview of an organization’s business, analysts and developers could adopt use case diagram as a start point and identify a use case from the diagram that interests users. After agreeing on a use case to investigate in detail, identify all the goal templates that relate to the target use case. With “process vehicle purchase” use case, goal templates that are identified to be relevant are template #1, 2, 8, 9, 10 and 17.

2. **Aggregate description elements.**
   
   Similar to step 2 in sequence diagram guideline, aggregate *description* elements in all related goal templates into one sequential description (for instance, Table 3.9 for “process vehicle purchase” use case). The difference with sequence diagram guideline is that there is no need to identify message senders and recipients.

   In order to aggregate description elements, analysts' knowledge regarding the sequencing of activities within a business process is required and the knowledge could be acquired through consulting relevant business process participants in the organization.
<table>
<thead>
<tr>
<th>#</th>
<th>Description of Step</th>
<th>Swimlane</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A customer enters the showroom</td>
<td>Customer</td>
</tr>
<tr>
<td>2</td>
<td>Customer asks sales associate questions</td>
<td>Sales associate</td>
</tr>
<tr>
<td>3</td>
<td>Customer requests sales associate to arrange a test drive</td>
<td>Sales associate</td>
</tr>
<tr>
<td>4</td>
<td>Sales associate outlines purchase procedure</td>
<td>Sales associate</td>
</tr>
<tr>
<td>5</td>
<td>Customer decides to make a purchase</td>
<td>Customer</td>
</tr>
<tr>
<td>6</td>
<td>Sales associate completes a sales form</td>
<td>Sales associate</td>
</tr>
<tr>
<td>7</td>
<td>Sales associate arranges a new vehicle cleaning schedule with service manager</td>
<td>Sales associate</td>
</tr>
<tr>
<td>8</td>
<td>Service manager provides a cleaning schedule</td>
<td>Service manager</td>
</tr>
<tr>
<td>9</td>
<td>Service manager ensures technicians clean the vehicle before the scheduled date</td>
<td>Service manager</td>
</tr>
<tr>
<td>10</td>
<td>Service manager requests sales associate to perform a final inspection before delivering the vehicle to customer</td>
<td>Service manager, Sales associate</td>
</tr>
<tr>
<td>11</td>
<td>Sales associate asks customer to pay business manager</td>
<td>Sales associate</td>
</tr>
<tr>
<td>12</td>
<td>If customer requests to pay by cash or cheque, business manager accepts the payment by cash or cheque</td>
<td>Customer, Business manager</td>
</tr>
<tr>
<td>13</td>
<td>If customer requests to pay through a financial institution, business manager confirms payment with the financial institution</td>
<td>Customer, Business manager</td>
</tr>
<tr>
<td>14</td>
<td>Business manager returns customer a payment receipt</td>
<td>Business manager</td>
</tr>
<tr>
<td>15</td>
<td>Business manager promotes maintenance contract sales</td>
<td>Business manager</td>
</tr>
<tr>
<td>16</td>
<td>Customer returns receipt to sales associate</td>
<td>Customer</td>
</tr>
<tr>
<td>17</td>
<td>Sales associate confirms payment receipt</td>
<td>Sales associate</td>
</tr>
<tr>
<td>18</td>
<td>Sales associate delivers keys and related documents to customer</td>
<td>Sales associate</td>
</tr>
</tbody>
</table>

Table 3.9: Combined Description of Process Vehicle Purchase Use Case (activity diagram)
3. **Identify activities.**

Each numbered step in the combined description derived from step 2 is modeled as an activity. And the sequential number of the steps in the description suggests the direction of control flow. If activities could be executed in parallel, use *Fork* and *Join* to parallel them. Activities are considered parallel if they 1) are independent to one another, meaning the execution of one activity does not rely on the accomplishment of another activity AND 2) can occur at the same time point and are initiated by the same actor. As an illustration, please refer to Figure 3.6 the finished activity diagram for “process vehicle purchase” use case. After customer makes a decision to buy a car, sales associate is required to fill in a sales form with customer and arrange a cleaning schedule with service manager. These two activities are parallel activities because they happen at the same time point (after customer deciding to purchase), they are responsibilities of the same actor (sales associate), and they are independent (completing sales form does not depend on the complete of arranging cleaning schedule and arranging cleaning does not rely on the accomplishment of completing sales form with customer).

4. **Identify swimlanes in activity diagram.**

*Agent* element in the goal templates identified in step 1 contain actors who claim direct responsibility of corresponding action as well as all the decomposed tasks listed in respective description element. Actors indicated in agent elements are translated to swimlanes in the activity diagram, which are responsible for their own activities. Add a third column to Table 3.9 and record swimlane actors for each activity.

5. **Represent alternatives.**

Use *Branch* and *Merge* combination to represent alternatives, which are indicated by *scenario* element in goal template. For instance, a customer can make a payment by cash/cheque or he/she can pay through a financial institution. As shown in Figure 3.6, use branch/merge to represent this. One guard for the branch
is [cash/cheque] and the other guard is [else]. Guard [else] means the customer pays via a bank and as a result, business manager should follow up to confirm the banking payment.

Summary

The activity diagram in Figure 3.6 is composed using activity diagram translation guideline. Step 2 and step 3 together models all activities in a certain business process from goal template element description. Step 3 also shows in activity diagram parallel activities that could be completed simultaneously. To identify responsible actors participating in a business process, step 4 maps template element agent to respective swimlanes. Step 5 uses template element scenario to represent alternative ways of finishing an activity in activity diagram. Therefore, translation guideline enables resulting activity diagram to increase the understanding of the business activity flow and facilitate communication about the business process.
Figure 3.6: Activity Diagram for Vision Quest Auto Dealership Case
3.2.6 Translation Guideline for UML Statechart Diagram

Statechart Diagram Introduction

The statechart diagram is a technique to describe what states an object can have and how different events affect those states [21]. Eriksson et. al. [21] stated that a statechart diagram indicates how an object reacts to different events and how the object changes its states as a result of the occurrence of events. A state transition usually has an event attached to it and the state transition will be performed when the event occurs [21]. Statechart diagrams are usually drawn for a single class, which has explicitly identifiable states and complex behavior, to show the behavior of a single object over its life cycle [20].

A statechart diagram reflects change of states of a class over its life long time. Relevant goal templates need to be recognized so that they altogether could capture different states of the class over its life cycle. The action elements in these goal templates are services of actors, which result in state changes of a particular object, and they are the events that trigger state transitions. States information could be summarized from the description element because it indicates how an organizational actor performs certain tasks to complete an action, which could affect an object’s states.

Statechart Diagram Translation Guideline

1. Decide on a class and identify relevant goal templates.
   The first step to come up with a statechart diagram is to pick a class of organization’s interest, which the organization care about and want to be aware of its detailed state transitions during business processes. Then, identify all relevant
goal templates so that altogether they could capture the different states of a class over its life cycle. Information in these goal templates contains all possible states of the interested class and all transition events that lead to its state changes. For instance, vehicle sale is a major business activity for Vision Quest. It is of great interest to depict what different states of a vehicle sales order could have and how the state would change over time in response to different events. Once the "vehicle sales" class have been picked, identify all the relevant goal templates that contain actions dealing with the "vehicle sales" class. These goal templates are template #1, 2, 8, 9, 10 and 17. They are the templates that relate to "vehicle sales" class and some of actions in these templates might lead to the changes of the state of that class.

2. Identify all the different states of a class over its life cycle.

Go through description elements of the goal templates highlighted in step 1, identify every significant state of the class after every action in the goal template is completed, and eliminate the goal templates that will not affect the class’s states. In our auto dealership case, the state of "vehicle sales" class is pre-order after customer enquires and test-drives in template 8, "vehicle sales" state transits to create once the customer decides to purchase the vehicle in template 9, it's ready for final inspection after service manager schedules a clean-up in template 17, ready for pick-up after sales associate inspects the vehicle as in template 10, paid after template 1 and 2, and finally, "vehicle sales" is completed once sales associate delivers keys to the vehicle and documents to the customer in template 9, which completes a sales process.

3. Identify the transition event that leads to the change of each state discovered in step 2.

The transition event that leads to each state change identified in step 2 could be extracted from the action element of each goal template, which is corresponded to the particular state. Since goal templates that result in state changes of a class have
been identified in step 2, take the action and agent elements from these templates and map them into the particular transition event format [21]: agent [action].

Figure 3.7 is developed complying with the above three steps. To illustrate, Customer [Enquire & Test Drive] event moves a “vehicle sales” to pre-order state. Once the customer decides to make a vehicle purchase (Customer [Purchase]), a “vehicle sales” is created. After the customers pays business manager (Customer [Pay]), the sales state changes to paid. Last, because a customer could call off an order (Customer [Cancel]) at any time before making a payment, a canceled state is also included in Figure 3.7.

Summary

The statechart diagram derived is able to reflect all potential states of an object in a business cycle because step 2 indicates this information through analyzing goal template element description. Business activity that results in object state change is extracted from template element action in step 3 and responsible participating actor is also specified in step 3.
Figure 3.7: Statechart Diagram for Vision Quest Auto Dealership Case
3.3 Summary

In this chapter of the thesis, we first introduced the goal template used to capture user's requirements in the form of goals. We also presented our proposed sets of systematic translation guidelines used to generate UML diagrams from goal templates. These guidelines are also demonstrated in-detail through a car dealership business case.

In Table 3.10 below, we show the relationships between goal template elements and UML diagrams, where the columns are the goal template elements while the rows are UML diagrams. In each cell of the table, we briefly summarize what element we take out from the goal template and how we use it to draw a particular UML diagram. Furthermore, in the last column of the table, we clarified, in addition to the knowledge of goal templates, design knowledge needed in order to complete the diagram. Take sequence diagram as an example, knowledge of activity sequencing in a business process is necessary to define the sequence of messages that are sent between actors in sequence diagram translation guidelines step 2.

Table 3.11 below further summarizes the proposed goal-based translation guidelines to develop UML diagrams based on users' requirements, grouped by each of the six UML diagrams.
<table>
<thead>
<tr>
<th>UML Diagrams</th>
<th>Goal Template Elements</th>
<th>Design Knowledge Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Diagram</td>
<td>Mapped to use cases</td>
<td>Used to aggregate use case</td>
</tr>
<tr>
<td>Class Diagram</td>
<td>Mapped to Agent class’s operations</td>
<td>Analyzed in Table 3.5 to identify classes, attributes, associations</td>
</tr>
<tr>
<td>Sequence Diagram</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collaboration Diagram</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity Diagram</td>
<td>Mapped to swimlanes</td>
<td>Aggregated to derive activities</td>
</tr>
<tr>
<td>Statechart Diagram</td>
<td>Mapped to transition events</td>
<td>Derive significant states of a class</td>
</tr>
</tbody>
</table>

Table 3.10: Cross Reference of Goal Template Elements and UML Diagrams
<table>
<thead>
<tr>
<th>UML Diagram</th>
<th>Goal Template Elements Used</th>
<th>Brief Description of Translation Steps in Our Proposed Translation Guidelines</th>
</tr>
</thead>
</table>
| Use Case Diagram | Action Stakeholders Description Scenario | 1. Identify use cases from action elements.  
2. Identify actors associated with each use case.  
3. Aggregate use cases.  
4. Identify <<include>> relationships.  
5. Identify <<extend>> relationships. |
| Class Diagram | Agent Stakeholders Action Goal Description | 1. Identify classes from agent and stakeholders.  
2. Identify operations in each class.  
3. Identify attributes, additional classes and associations among classes.  
3.1. Analyze each primary goal using Rolland et al.’s goal structure approach.  
3.2. Categorize the Target column in the Goal Analysis Form.  
3.3. Identify attributes and additional classes.  
3.4. Identify associations between classes.  
3.5. Go through the Goal Analysis Form to ensure complete associations.  
4. Examine the evolving class diagram to build generalization relationships.  
5. Refine the diagram by revisiting stakeholders. |
| Sequence Diagram | Description Stakeholders Scenario | 1. Identify related goal templates.  
2. Aggregate description elements.  
3. Identify participating objects.  
4. Generate messages and identify sender and recipient of each message.  
5. Represent alternatives. |
| Collaboration Diagram | Same as sequence diagram. | 1. Re-number the description steps developed in sequence diagram translation.  
2. Identify interacting objects.  
3. Generate messages and identify sender and recipient of each message. |
| Activity Diagram | Description Agent Scenario | 1. Identify related goal templates.  
2. Aggregate description elements.  
3. Identify activities.  
4. Identify swimlanes in activity diagram.  
5. Represent alternatives. |
| Statechart Diagram | Description Action | 1. Decide a class and identify relevant templates.  
2. Identify all the different states of a class over its life cycle.  
3. Identify the transition event that leads to the change of each state discovered in step 2. |

Table 3.11: Summary of UML Diagrams Translation Guidelines
CHAPTER 4: EMPIRICAL STUDY AND EVALUATION

A small scale empirical study has been carried out to verify the effectiveness of our translation approach proposed in the last chapter. One subject was asked to draw six UML diagrams, namely a use case diagram (see Figure 4.1 below), a class diagram (see Figure 4.3 below), a sequence diagram (see Figure 4.4 below), a collaboration diagram (see Figure 4.5 below), an activity diagram (see Figure 4.6 below) and a statechart diagram (see Figure 4.7 below), based on the Vision Quest auto-dealership case narrative presented at the beginning of previous chapter.

We aim to use the empirical study to compare UML diagrams drawn by the subject with the ones developed according to our translation guidelines and assess the adequacy of the proposed six sets of translation guidelines.

4.1 Study Design

Our subject is a graduate student from MIS division of a large North American university who has been chosen because he possesses a fair knowledge of UML modeling technique. As the first step of the empirical study, the subject was asked to read carefully the Vision Quest auto dealerships case as an introduction of the company and its businesses. The subject was told that he could keep the case description to refer to while drawing UML diagrams later if necessary. Then, the subject was asked to generate a set of UML diagrams for the case, containing one use case diagram, one class diagram, one sequence diagram, one collaboration diagram, one activity diagram.
and one statechart diagram.

One expert, who knows UML technique well and has much UML modeling experience, was asked to rate and compare each individual UML diagram drawn by our experiment subject with the respective diagram developed using our proposed translation guidelines. The expert’s independent judgment was used to validate the effectiveness of the translation guidelines.

The rest of this chapter analyzes separately the six UML diagrams that were generated by the subject.

4.2 Analysis of Use Case Diagram

The use case diagram drawn by the experiment subject is depicted in Figure 4.1. It contains 11 actors and 16 use cases, while there are only 7 actors and 11 use cases in the diagram developed using our use case diagram translation guideline (please refer to Figure 3.2 from Chapter 3). However, a close analysis conveys that use case translation guideline step 2 ensures Figure 3.2 to capture more precise relationships between use cases and actors because it derives actor-use case relationships by mapping template element stakeholders to actors in use case diagram and stakeholders are directly related to a certain use case. For example, VP works together with service manager and sales manager to realize long-term goals, which has not been reflected in Figure 4.1. In the use case diagram in Figure 4.1, VP works independently to forecast sales, bundle services, and keep loyal customers.

Furthermore, although Figure 4.1 has more use cases, some of them are actually combined together into one use case in Figure 3.2 by complying with use case
guideline step 3. Given the use case diagram is usually used to provide high level instead of lower level detail information about the organization and the future system, Figure 3.2 is considered better in this case. The expert evaluation confirms our analysis: "Figure 3.2 is better because the abstraction level of each activity (use case) is higher and the description (name of each use case) is more informative."

Moreover, since Figure 3.2 is more abstract than Figure 4.1, a comparative analysis between Figure 4.1 and the decomposition of Figure 3.2 is desirable to verify whether the decomposed diagram contains at least the same amount of information as Figure 4.1. Decomposition of the use case diagram derived from goal template is shown in Figure 4.2, which contains 18 use cases and 10 actors. As shown in Figure 4.2, use cases with shade are the ones that are not shown in Figure 4.1. Having complied with guideline step 1, each of these shaded-background use cases can be traced back to a respective action element in a particular goal template, which proves that our goal based translation approach is a more adequate technique to derive complete use case diagram.

As we can see in Figure 4.1, four use cases with shaded background color are the ones that are not captured in Figure 4.2. Closer analysis indicates that, among these use cases, three of the four use cases, "Forecasting Sales", "Bundling of Services" and "Keeping Loyal Customer", are combined into one single use case "Process long term objectives" in Figure 4.2. The last use case "Services" and the associated actor "Service Technician" are missing in Figure 4.2. However, it is not mentioned in the case description that service technicians have a direct service contact with customers while Figure 4.1 suggests there is.
Figure 4.1: Use Case Diagram Drawn by the Subject
Figure 4.2: Decomposition of Use Case Diagram Figure 3.2
4.3 Analysis of Class Diagram

The class diagram (please refer to Figure 3.3) developed using our goal based translation approach contains 23 classes compared to 15 classes in the class diagram drawn by the experiment subject (see Figure 4.3 below), 26 attributes compared to 17 in Figure 4.3, 19 operations compared to 9 in Figure 4.3, and obviously more associations compared to only 17 associations in Figure 4.3. This has indicated that because class diagram translation guideline step 2 and step 3 make sure to fully utilize three template elements goal, action and description, Figure 3.3 is able to capture more classes, class attributes and operations. Guideline step 3 also ensures that the class diagram models more complete and more complex associations between classes. Thus due to guideline step 2 and 3, Figure 3.3 represents much more information than Figure 4.3 and it more precisely captures users’ requirements. After comparing the two class diagrams, our expert rated Figure 3.3 as more adequate because “it provides more details, especially about interactions (associations) between classes”.

Further comparative analysis between Figure 3.3 and Figure 4.3 shows that the diagram developed by the subject (Figure 4.3) omits some classes and operations that are considered crucial to the system. For example, parts manager is responsible for maintaining parts inventory and sales manager is supposed to adjust vehicle price to remain competitive in market, both of which are important roles to realize organization’s long-term strategy, yet neither is modeled as a class in Figure 4.3. Moreover, one responsibility of business manager is to update sales commission for sales associates, which is not represented as an operation of the business manager class in Figure 4.3 though it shows an association between business manager and commission accumulated class.
Figure 4.3: Class Diagram Drawn by the Subject
4.4 Analysis of Sequence Diagram

Figure 4.4, the sequence diagram reflecting "process vehicle purchase" use case, is drawn by the experiment subject. Compared with the sequence diagram generated using our guidelines (please refer to Figure 3.4), it contains the same objects as Figure 3.4 and close amount of messages, 16 messages in Figure 4.4 versus 15 messages in Figure 3.4. The sequencing of messages could in general be considered alike, with both diagrams capturing the organizational behavior of "process vehicle purchase". However, our subject missed some necessary messages and failed to show them in the diagram. For instance, after confirming a customer's payment, the business manager is supposed to promote a maintenance contract sale. Therefore, there should be a message initiated by business manager and sent to customer, which is missing in Figure 4.4. On the other hand, sequence diagram guideline step 2 and 4 require goal template element description is mapped properly to corresponding messages, which ensures that all necessary messages are identified in Figure 3.4.

Our expert raised a good point concerning consistency with the case narrative when rating two sequence diagrams. In other words, the sequence diagram developed according to translation guidelines (Figure 3.4) provides some additional details that are not explicitly stated in the case. For example, as indicated in Figure 3.4, after arranging a cleaning schedule with sales associate, a service manager initiates a message to him/herself to schedule technicians to clean the purchased car. Although the task of assigning technicians is not clearly listed in selling a vehicle process in the case (it can be inferred from the narrative though), this detail information is provided in a corresponding goal template (template No. 17), which from another point of view could imply the importance of utilizing a goal template approach in terms of capturing more complete users' requirements.
Figure 4.4: Sequence Diagram Drawn by the Subject
4.5 Analysis of Collaboration Diagram

For use case "process vehicle purchase", our subject developed the collaboration diagram as in Figure 4.5. Given that sequence and collaboration diagram only differ in the way of representing sequences of interactions, comparative analysis of two collaboration diagrams (Figure 3.5 and Figure 4.5) conveys similar result with the comparison of two sequence diagrams. Figure 4.5 contains the same five objects as Figure 3.5 and two less messages than Figure 3.5. Again, our subject ignored some necessary messages, such as message 1.3.2 return receipt and 1.3.2.1 promote maintenance contract in Figure 3.5, and failed to show them in collaboration diagram Figure 4.5. Collaboration diagram guideline step 3, by corresponding goal template element description to messages, guarantees that Figure 3.5 captures these additional yet necessary messages missed by the subject.

Collaboration diagram easily shows the interactions among objects. We can see from Figure 4.5 that our subject suggested an interaction between business manager and sales associate, that is after confirming customer’s payment, business manager informs sales associate to deliver the car. However, this is neither indicated by the case, nor is it implied in any goal template.
Figure 4.5: Collaboration Diagram Drawn by the Subject
4.6 Analysis of Activity Diagram

The activity diagram depicted in Figure 4.6 on the next page is drawn by our subject for “process vehicle purchase” workflow. It contains 18 activities, 2 less compared to Figure 3.6 developed by activity diagram translation guideline. Moreover, Figure 3.6 employs three pairs of fork/join pairs to represent parallel behaviors, two more pairs than Figure 4.6. This results from activity diagram translation guideline step 3, where parallel behaviors are implied through the analysis of description element in goal template. For example, while sales associate replies customer’s enquiries about vehicles, he/she could in the meanwhile arrange a test drive of customer’s interested model and explain to customer the purchase procedure. Figure 3.6 also uses one more branch or branch/merge combination to reflect conditional behavior by analyzing template element scenario in activity diagram guideline step 5. After inquiring, a customer has an option not to make a purchase transaction, which is not permitted as in the case in Figure 4.6. In other words, purchase is mandatory after enquire and test-drive according to the process delineated in Figure 4.6.

However, our expert gave preference of “allocation of tasks (activities) to swimlanes” to Figure 4.6. Further examination of the two diagrams implies that this is mainly due to the inclusion of Financial Institution swimlane and activity “confirm payment” of that swimlane in Figure 4.6. Our argument is that we derive swimlanes from agent elements in goal templates and actors in agent elements are the ones who claim direct responsibility of corresponding actions. Thus, in Figure 3.6, we allocate “confirm payment” activity to business manager swimlane because business manager is the main internal responsible actor for “confirm payment” activity. Financial institution and its “confirm payment” activity are rather external to the organization or system. We also notice that Figure 4.6 uses same name for several different activities, including activities belonging to different swimlanes but with same activity name.
Figure 4.6: Activity Diagram Drawn by the Subject
4.7 Analysis of Statechart Diagram

Our subject identified in Figure 4.7 only 5 states for “vehicle sales” class and 5 transition events that lead to these different states, while, in Figure 3.7, we have elicited 7 states for “vehicle sales” class and 10 transition events by applying statechart diagram translation guideline. For instance, after a customer test drives a vehicle and before he/she decides to purchase, the vehicle sales changes state, which is not reflected in Figure 4.7. Another missing state is that if the customer decides not to proceed to make a purchase, he/she could cancel it.

This comparison analysis suggests that by complying with the proposed guideline, the statechart diagram would include more valid states of a class over its life time and more detailed information regarding events that would cause state transitions. Statechart diagram translation guideline step 2 is able to recognize all states of a class over its life by extracting information from goal template element description, while guideline step 3 maps template element action to events in the statechart diagram so as to derive more complete transition events. Further, expert evaluation comments “Figure 3.7 describes the states of the order (vehicle sales) better”, while some states in Figure 4.7 “shows the states of the car/vehicle” instead of vehicle sales.
Figure 4.7: Statechart Diagram Drawn by the Subject
4.8 Summary

This chapter has described a preliminary empirical study. The study has been conducted with one subject and we have testified the effectiveness of our proposed set of translation guidelines for mapping goal templates to six UML diagrams. No significant statistical conclusion can be derived due to such a small sample size. However, the results of this empirical study could qualitatively demonstrate that by applying our proposed translation guidelines, we would be able to generate more complete UML diagrams that contains more information and captures user's requirements more precisely. Analysis and comparison of diagrams drawn by the study subject with ones derived by applying translation guidelines shows that our approach of mapping requirements to UML diagrams is more adequate.

Table 4.1 in the next page summarizes the comparison of differences between UML diagrams developed according to translation guidelines and diagrams drawn by the experiment subject. The rows in that table are six UML diagrams and columns are goal template elements. Each cell of the table shows what the major differences are and traces back to specific steps of translation guidelines to explain the reason why the differences occurred. If the subject diagram is missing any important information, each cell also indicates the specific translation step that is able to indentify the missing information from a certain goal template element.
<table>
<thead>
<tr>
<th>Action</th>
<th>Agent</th>
<th>Goal</th>
<th>Description</th>
<th>Stakeholders</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case</td>
<td>Step 1 and 3 enable diagram</td>
<td></td>
<td>Step 3 enables diagram by guidelines to represent more proper level of</td>
<td>Step 2 enables diagram by guidelines to more accurately identify actor-use</td>
<td>Step 5 enables diagram by guidelines to identify &lt;&lt;extend&gt;&gt; relationships that are missed by subject.</td>
</tr>
<tr>
<td>Diagram</td>
<td>by guidelines to capture more</td>
<td></td>
<td>information.</td>
<td>case relationships.</td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>enable diagram by guidelines</td>
<td></td>
<td>Step 3 enables diagram by guidelines to have more classes and attributes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagram</td>
<td>to include more class</td>
<td></td>
<td>and reflect more complex associations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UML Diagrams</td>
<td>operations.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence</td>
<td>Step 2 and 4 enable diagram</td>
<td></td>
<td>Step 2 and 4 enable diagram by guidelines to show all necessary messages,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagram</td>
<td>by guidelines to show</td>
<td></td>
<td>some of which are missed by subject.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collaboration</td>
<td>all necessary messages, some</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagram</td>
<td>of which are missed by subject.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity Diagram</td>
<td>Statechart Diagram</td>
<td>Goal Template Elements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------</td>
<td>------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Description</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Step 3 enables diagram</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>by guidelines to capture</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>more parallel behaviors</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>using fork/join pairs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Stakeholders</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Step 5 enables diagram</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>by guidelines to show</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>more conditional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>behaviors using</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>branch/merge pairs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Goal</strong></td>
<td><strong>Scenario</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of Diagrams by Guidelines and Diagrams by the Subject
5.1 Conclusion

Goal is an essential concept in requirements engineering. Goal-oriented requirements engineering has received much attention. It concerns the use of goals for eliciting, elaborating, specifying, analyzing, negotiation, and modifying requirements [1]. However, only a small number of studies have been conducted in requirements engineering (RE) field aiming to ensure that goals are adequately captured in the future system and the application of these proposed approaches and frameworks can be rather subjective. A primary motivation of this thesis is to propose a complete and systematic methodology to develop systems analysis and design diagrams based on business requirements specified in goals. Developed in such a fashion, the information system would be able to adequately reflect users’ requirements.

In order to model requirements, we first introduced in Chapter 2 some requirements identification methods in RE field. Since the use of the goal concept has long been widely adopted in RE as a satisfactory way to capture requirements and it has gained popularity in recent years, we have placed major efforts on studying several well-accepted goal based identification and elicitation techniques, such as KAOS project [6], Goal-Based Requirements Analysis Method (GBRAM) [8], Goal Modeling Using Scenario [9], and Singh et. al.’s Goal Schema approach [11]. Based on the review of these techniques, we developed a goal template approach in Chapter 3 to capture stakeholders’ goals and some other relevant information. In Chapter 2, we also reviewed four methodologies of mapping requirements to systems analysis and design diagrams as well as their limitations. These four techniques include Tropos...
project [3], Dennis et. al.'s approach of creating UML class diagram from CRC (Class-Responsibility-Collaboration) cards [12], Tan et. al.'s heuristic for transforming Service Responsibility Tables into use case and class diagrams [13], and Cysneiros et. al.'s method of integrating Non-Functional Requirements into systems design [14].

Given the wide acceptance of UML modeling language in industry, we’ve decided to base our translation guidelines on UML. Nevertheless, since UML is not easily understandable for business professionals, it is thus not adequate for analyzing stakeholders’ goals and business requirements. In Chapter 3 of this thesis, we presented our proposed UML diagrams translation guidelines for translating stakeholders’ requirements (captured using our goal template approach) to six UML diagrams, namely use case diagram, class diagram, sequence diagram, collaboration diagram, activity diagram and statechart diagram. During the development of guidelines, we first identified the building-blocks used in each UML diagram. Then, by analyzing goal template, we highlighted elements in the goal template that contain relevant information to develop UML diagram elements identified earlier. Finally, we formalized the translation process and derived diagram translation guidelines. We used an imaginative car-dealer case to illustrate the process and draw UML diagrams for the case according to guidelines.

We did a preliminary test by having a subject draw several UML diagrams for the car-dealership case. For each type of the six UML diagrams, the subject was asked to provide a diagram based on his/her understanding of the business described in the case. We then compared these diagrams against the ones we developed by applying translation guidelines. An expert was also invited to rate the two sets of diagrams. Results tend to be in favor of guidelines-generated diagrams, which indicates the usefulness and adequacy of the proposed UML diagrams translation guidelines.
5.2 Contributions

A major contribution of this thesis is that we have developed a systematic and complete set of translation guidelines to derive six UML diagrams (use case diagram, class diagram, sequence diagram, collaboration diagram, activity diagram and statechart diagram) from users' requirements captured as goals. Prior works concerning modeling requirements in analysis and design diagrams are either incomplete, only addressed one or two UML diagrams, or relatively subjective and judgment-intense, unable to guarantee development of consistent diagrams across different system analysts.

By conforming to the proposed translation guidelines, we are able to generate UML diagrams that are capable of reflecting stakeholders' goals and business requirements. These precisely drawn UML diagrams are crucial to build an information system and ensure that the future system captures users' requirements, thus increase the possibility of success of the IT project.

Second, our methodology successfully adopted UML, which is often criticized of its lack of capability to reflect organization's business environment, as a technique to conduct business modeling. The proposed methodology comprises a goal template approach to capture uses' goals and six systematic sets of guidelines to map goal templates to six UML diagrams. Chapter 3 has illustrated how UML could be used for business process modeling: users' requirements and the rationale of the requirements are elicited through the goal template approach and then mapped to UML diagrams according to translation guidelines. Given that the UML diagrams developed by translation guidelines are able to better capture business requirements, these guidelines could enable UML to better describe business context.

In addition, we also suggested adopting the goal template approach to elicit users'
requirements and goals. The goal template is created by incorporating commonly used attributes in goal identification and elicitation techniques and attributes that are useful for design. This approach also provides flexibility and independency of goal elicitation process, no matter what technique is employed to identify and elicit goals, as long as necessary goal templates are filled out, we are ready to go. UML diagrams could then be generated based on goal templates.

5.3 Limitations and Future Research

The research presented in this thesis is the first attempt to provide complete guidelines to guide the development of UML diagrams that conform to business requirement. It certainly has limitations and requires further research. In the following, we will discuss some limitations of our work and suggest possible solutions and future research directions.

First of all, the empirical evaluation presented in the thesis is on a very small scale. Only one subject participated in the experiment, thus the empirical test is rather preliminary and limited as a result. The result of this test could only demonstrate the usefulness of our translation guidelines on a qualitative basis and no statistical conclusions could be drawn due to the small sample size. A further empirical study could be conducted with a much larger sample size to examine the effectiveness of translation guidelines. The effectiveness of translation guidelines could also be tested by using a control group. While the control group will be trained to use goal template approach, experiment group will be provided with knowledge of both goal template approach and translation guidelines and asked to develop UML diagrams. Then, two groups' diagrams would be compared through expert evaluation. Further interviews could be conducted as well with experiment subjects with regard to the test of
guideline clearness and understandability.

Second, another limitation with the empirical study is that only one expert was asked to evaluate UML diagrams. In the future, more experts should be hired as raters to grade diagrams. Another possible future research direction would be to provide experts with a rating schema as a basis for evaluation and comparison. Moreover, feedbacks from experts' evaluations could be used as a basis to refine translation guidelines so as to ensure high quality diagrams developed from guidelines.

Third, non-functional requirements are not handled in due diligence. We proposed the use of goal template approach to model users' requirements. However, the goal template puts more emphasis on functional requirements in a manner that an agent executes an action to realize a goal. Issues with non-functional requirements are not addressed properly. Future research should find a way to improve the goal template structure and enable it to capture non-functional requirements. A possible solution we could now think of would be to add an additional element to the goal template that is specific to non-functional requirements. For each goal of a certain agent, we identify whether there are any non-functional requirement affiliated to it. Of course, as a result of the reform to the template, translation guidelines should be adjusted accordingly as well to incorporate non-functional requirements in UML diagrams.

Last but not least, there are no goal evolution and refinement processes in our goal template approach. Goals are constantly changing over time, while from developers' point of view, it is desirable that goals and requirements stay constant [8]. Goal refinement techniques are thus helpful to keep refined goals as stable as possible. Goals can be refined through elimination of redundancy and reconciling synonymous goals [8]. Future research may be needed to enhance our goal template method to identify and elicit goals.
References


28. Wang, W., An OOEM-Based Goal Modeling, M.Sc. thesis, Faculty of

### Appendix A: Goal Templates for Vision Quest Auto Dealership Case

<table>
<thead>
<tr>
<th>ID</th>
<th>Action</th>
<th>Agent</th>
<th>Goal</th>
<th>Description</th>
<th>Scenario</th>
<th>Stakeholders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process customer payment for vehicle</td>
<td>Business Manager</td>
<td>Process payment by cash or cheque accurately in a professional and friendly manner to ensure customer satisfaction; promote maintenance contract sales to provide better after sales service; return customer a payment receipt</td>
<td>When a customer requests to pay, business manager 1. accepts the payment by cash or cheque 2. returns customer a payment receipt 3. promotes maintenance contract sales</td>
<td>Customer requests to pay through a financial institution.</td>
<td>Business Manager, Customer</td>
</tr>
<tr>
<td>2</td>
<td>Process payment via financial institution</td>
<td>Business Manager</td>
<td>Process and confirm payment through financial institution accurately in a professional and friendly manner to ensure payment authenticity and customer satisfaction and return customer a payment receipt</td>
<td>When a customer requests to pay through a financial institution, business manager 1. confirms payment with the financial institution 2. returns customer a payment receipt 3. promotes maintenance contract sales</td>
<td>Customer makes payment by cash or cheque.</td>
<td>Business Manager, Customer, Financial Subsidiary</td>
</tr>
<tr>
<td>3</td>
<td>Process parts payment</td>
<td>Business Manager</td>
<td>Process vehicle parts purchase payment accurately in a professional and friendly manner to ensure high quality service and customer satisfaction, mark invoice as paid</td>
<td>When a customer requests to pay, 1. collect payment amount indicated on invoice prepared by parts manager 2. mark invoice as paid</td>
<td>N/A</td>
<td>Business Manager, Customer</td>
</tr>
<tr>
<td>ID</td>
<td>Action</td>
<td>Agent</td>
<td>Goal</td>
<td>Description</td>
<td>Scenario</td>
<td>Stakeholders</td>
</tr>
<tr>
<td>----</td>
<td>--------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------</td>
<td>---------------------------------</td>
</tr>
</tbody>
</table>
| 4  | Process sales commission update | Business Manager | Update sales commission target timely and accurately to motivate Sales Associates and ensure to meet sales quotas; monitor Sales Associates performance and fire bad-fit associates | 1. calculate and update sales commission  
2. monitor sales associates meet sales target/quota  
3. fire bad performers | N/A      | Business Manager, Sales Associate                                   |
| 5  | Prepare and process reorder payment | Business Manager | Facilitate parts reordering to ensure inventory availability by preparing invoice and paying suppliers in a timely and accurate manner | 1. prepare reorder invoice based on info from parts manager  
2. accept the reorder bill from parts manager  
3. pay supplier based on the bill | N/A      | Business Manager, Parts Manager                                     |
| 6  | Process parts purchase request | Parts Manager   | Provide high quality services to ensure customer satisfaction by completing part invoice accurately and delivering vehicle parts to customers after collecting invoice receipt in a timely and accurate manner | 1. complete parts purchase invoice  
2. instruct customer to pay  
3. collect payment receipt  
4. deliver parts to customer | N/A      | Parts Manager, Customer                                              |
| 7  | Process parts reorder and payment invoice | Parts Manager | Maintain appropriate vehicle parts availability and parts turnover and reduce inventory cost by reordering from suppliers in a timely and accurate manner | 1. monitor inventory level and reorder parts at reorder point  
When reordering, parts manager  
1. ask business manager to prepare reorder invoice  
2. invoice suppliers; track order  
3. receive parts and notify service manager  
4. send the reorder bill to business manager | N/A      | Parts Manager, Business Manager, Supplier                           |
<table>
<thead>
<tr>
<th>ID</th>
<th>Action</th>
<th>Agent</th>
<th>Goal</th>
<th>Description</th>
<th>Scenario</th>
<th>Stakeholders</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Outline procedures for purchasing vehicle</td>
<td>Sales Associate</td>
<td>Promote vehicle sales by answering questions about car, model info and price and arranging test drive in a professional, friendly and knowledgeable manner; help customers make purchase decision by providing as much information as possible</td>
<td>When a customer enters the showroom, sales associate 1. answers customer’s questions 2. arranges a test drive 3. outlines purchase procedure</td>
<td>Customer decides not to purchase.</td>
<td>Sales Associate, Customer</td>
</tr>
<tr>
<td>9</td>
<td>Process purchasing request</td>
<td>Sales Associate</td>
<td>Provide high quality services to ensure customer satisfaction by completing purchase sales form in a timely and accurate manner, scheduling with Service Manager a vehicle cleaning-up before the customer returns, and delivering dealer’s accessories and warranties related documents and the keys to vehicle after payment receipt confirmed</td>
<td>Once a customer decides to make a purchase, sales associate 1. completes a sales form 2. arranges a cleaning schedule with service manager When the customer returns, 1. asks customer to pay business manager 2. confirms payment receipt 3. delivers keys and related documents to customer</td>
<td>Customer decides not to purchase before paying business manager.</td>
<td>Sales Associate, Customer, Service Manager</td>
</tr>
<tr>
<td>10</td>
<td>Inspect vehicle for final time</td>
<td>Sales Associate</td>
<td>Ensure no problem will pop up and leave customer a good image by inspecting vehicle before the customer arrives</td>
<td>perform a final inspection before delivering the vehicle to customer</td>
<td>N/A</td>
<td>Sales Associate, Customer</td>
</tr>
<tr>
<td>11</td>
<td>Process used-vehicle price</td>
<td>Sales Associate</td>
<td>Promote used-vehicle sales in a professional, knowledgeable and friendly manner and offer customer an acceptable price by negotiating with Sales Manager</td>
<td>1. negotiate with sales manager 2. offer customer the negotiated price 3. complete sales if customer accepts the price</td>
<td>Customer decides not to purchase.</td>
<td>Sales Associate, Customer, Sales Manager</td>
</tr>
<tr>
<td>ID</td>
<td>Action</td>
<td>Agent</td>
<td>Goal</td>
<td>Description</td>
<td>Scenario</td>
<td>Stakeholders</td>
</tr>
<tr>
<td>----</td>
<td>---------------------------------------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>12</td>
<td>Process negotiation</td>
<td>Sales Associate</td>
<td>Negotiate with sales manager to offer customer an attractive used car price, promote used vehicle sales and ensure customer satisfaction with price and services provided</td>
<td>When a customer asks for a discount on used vehicle, negotiate with sales manager.</td>
<td>Customer doesn’t ask for a bargain.</td>
<td>Sales, Associate, Customer, Sales Manager</td>
</tr>
</tbody>
</table>
| 13 | Process negotiation price              | Sales Manager | Offer customer a reasonable price with acceptable profit margin for used vehicle to increase used car sales | When sales associate asks for a discount price on used vehicle,  
1. provide a discount to sales associate, but with enough profit margin | Reject used vehicle sales with low profit margin. | Sales Manager, Sales Associate |
| 14 | Generate potential customers contact list | Sales Manager | Promote new business by recognizing potential new customers and providing contact list to Sales Associate | 1. Identify potential customers and generate a contact list  
2. provide sales associate with the contact list | N/A | Sales Manager, Sales Associate |
| 15 | Set competitive prices on vehicles     | Sales Manager | Increase sales by monitoring external auto-dealer market and adjusting vehicle prices to timely and accurately reflect competitive prices, under VP’s supervision | Under VP’s supervision,  
1. monitor auto-dealer market for competitive vehicle prices  
2. adjust vehicle price accordingly | N/A | Sales Manager, VP |
| 16 | Process wholesale prices for vehicles  | Sales Manager | Promote sales by fulfilling wholesalers’ vehicle purchase request and recommending additional vehicles | When wholesaler sends in purchase request,  
1. fulfill wholesaler’s vehicle purchase order  
2. recommend additional models to the wholesaler | Sales manager is not capable of fulfilling purchase request. | Sales Manager, Wholesaler |
<table>
<thead>
<tr>
<th>ID</th>
<th>Action</th>
<th>Agent</th>
<th>Goal</th>
<th>Description</th>
<th>Scenario</th>
<th>Stakeholders</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Process cleaning schedule</td>
<td>Service Manager</td>
<td>Provide the earliest possible clean-up schedule for new vehicle sales and ensure high quality cleaning service by service technicians</td>
<td>When sales associate requests a new vehicle clean-up, service manager 1. provides a cleaning schedule 2. ensures technicians clean the vehicle before the scheduled date 3. ask sales associate to inspect</td>
<td>N/A</td>
<td>Service Manager, Sales Associate</td>
</tr>
<tr>
<td>18</td>
<td>Plan technicians work schedule</td>
<td>Service Manager</td>
<td>Prepare weekly service technicians work schedule to achieve optimal servicing and complete customer satisfaction (servicing customer’s vehicle ASAP and enough technicians available for drop-ins and emergencies)</td>
<td>1. manage the workflow in the service department 2. plan technician’s work schedule every week to ensure enough technicians available for drop-ins</td>
<td>N/A</td>
<td>Service Manager, VP</td>
</tr>
<tr>
<td>19</td>
<td>Process long term objectives: increasing sales by 12% in next 5 years</td>
<td>VP</td>
<td>Increase sales and market share and benefit stakeholders by coordinating among Vision Quest’s departments and monitoring progression of long term objective through forecasting sales, bundling services to vehicle models, and rewarding loyal customers</td>
<td>Coordinate among all departments and monitor long-term objective progress through: 1. forecast sales against projection quarterly 2. terminate problematic models or bundle services to large profit margin problematic models 3. recognize loyal customers</td>
<td>N/A</td>
<td>VP, All Managers</td>
</tr>
</tbody>
</table>

Table App.1: Goal Templates Summary for Vision Quest Auto Dealership Case
# Appendix B: Goal Analysis Form for Vision Quest Auto Dealership Case

<table>
<thead>
<tr>
<th>Goal ID</th>
<th>Object</th>
<th>Target</th>
<th>Source</th>
<th>Destination</th>
<th>Means</th>
<th>Manner</th>
<th>Beneficiary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cash payment</td>
<td>Receipt</td>
<td>Business Manager</td>
<td>Customer</td>
<td>N/A</td>
<td>Process cash or cheque payment, return receipt, promote maintenance contract sales</td>
<td>Customer</td>
</tr>
<tr>
<td>2</td>
<td>Payment via banks</td>
<td>Receipt, Payment confirmation</td>
<td>Business Manager</td>
<td>Customer</td>
<td>Payment authenticity</td>
<td>Confirm payment with banks, return customer a receipt</td>
<td>Customer, Vision Quest</td>
</tr>
<tr>
<td>3</td>
<td>Parts payment</td>
<td>Receipt</td>
<td>Business manager</td>
<td>Customer</td>
<td>N/A</td>
<td>Process parts payment accurately, mark invoice as paid</td>
<td>Customer</td>
</tr>
<tr>
<td>4</td>
<td>Sales commission, Sale quotas</td>
<td>Sales Associates performance</td>
<td>Business manager</td>
<td>Sales Associate, Sales commission, Sale quotas</td>
<td>Update sales commission, monitor performance, fire bad-fit sales associates</td>
<td>Sales Associate, Vision Quest</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>Reorder payment</td>
<td>Business manager</td>
<td>Supplier</td>
<td>N/A</td>
<td>Prepare invoice and pay supplier to facilitate parts reordering</td>
<td>Parts manager, Supplier</td>
</tr>
<tr>
<td>6</td>
<td>Vehicle parts, Parts payment invoice</td>
<td>Parts payment receipt</td>
<td>Parts manager</td>
<td>Customer</td>
<td>N/A</td>
<td>Complete part invoice and deliver parts to customer after collecting receipt</td>
<td>Customer</td>
</tr>
<tr>
<td>Goal ID</td>
<td>Target</td>
<td>Direction</td>
<td>Way</td>
<td>Beneficiary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
<td>-----</td>
<td>-------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Parts availability, Parts turnover, Inventory cost</td>
<td>Parts manager</td>
<td>Supplier</td>
<td>N/A</td>
<td>Maintain inventory level and turnover, reduce cost, reorder from supplier</td>
<td>Vision Quest</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Vehicle info, Price</td>
<td>Test drive, Purchase decision, Vehicle sales</td>
<td>Sales associate</td>
<td>Customer</td>
<td>Test drive, vehicle info</td>
<td>Customer</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Docs and keys</td>
<td>Purchase sales form, Vehicle clean-up</td>
<td>Sales associate, Service manager</td>
<td>Purchase sales form</td>
<td>Complete purchase sales form, schedule clean-up, deliver docs and keys after payment</td>
<td>Customer</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>N/A</td>
<td>Final inspection</td>
<td>Sales associate</td>
<td>N/A</td>
<td>Inspect vehicle before customer returns</td>
<td>Customer, Vision Quest</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>N/A</td>
<td>Used-vehicle sales</td>
<td>Sales associate</td>
<td>Customer</td>
<td>Promote used vehicle sales, offer customer an acceptable price by negotiating with sales manager</td>
<td>Customer</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
<td>Attractive price</td>
<td>Sales associate</td>
<td>Sales manager</td>
<td>Negotiate with sales manager</td>
<td>Customer</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>N/A</td>
<td>Reasonable price, Profit margin</td>
<td>Sales manager</td>
<td>N/A</td>
<td>Offer customer a reasonable price with acceptable profit margin</td>
<td>Customer</td>
<td></td>
</tr>
<tr>
<td>Goal ID</td>
<td>Object</td>
<td>Target</td>
<td>Direction</td>
<td>Source</td>
<td>Destination</td>
<td>Means</td>
<td>Manner</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
<td>--------</td>
<td>-------------</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>14</td>
<td>N/A</td>
<td>New business, Potential new customer, Contact list</td>
<td>Sales manager</td>
<td>Sales associate</td>
<td>Contact list</td>
<td>Recognize new customers and provide contact list to sales associates</td>
<td>Vision Quest</td>
</tr>
<tr>
<td>15</td>
<td>Auto-dealer market, Vehicle price, Competitive prices</td>
<td>N/A</td>
<td>Sales manager</td>
<td>VP</td>
<td>N/A</td>
<td>Monitor external market, adjust vehicle prices to reflect competitive prices</td>
<td>Vision Quest</td>
</tr>
<tr>
<td>16</td>
<td>Wholesalers’ purchase request</td>
<td>Wholesaler sales, additional vehicles</td>
<td>Sales manager</td>
<td>Wholesalers</td>
<td>N/A</td>
<td>Fulfill wholesalers’ purchase request and recommending vehicles</td>
<td>Vision Quest</td>
</tr>
<tr>
<td>17</td>
<td>N/A</td>
<td>Clean-up schedule</td>
<td>Service manager</td>
<td>N/A</td>
<td>Provide earliest clean up schedule and ensure high quality cleaning</td>
<td>Customer</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>N/A</td>
<td>Technician work schedule</td>
<td>Service manager</td>
<td>N/A</td>
<td>Prepare weekly technicians work schedule</td>
<td>Customer</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Long-term objectives</td>
<td>Sales, Market share, stakeholders benefit</td>
<td>VP</td>
<td>All departments</td>
<td>N/A</td>
<td>Increase sales and benefit stakeholders by coordinating among departments and monitoring long-term objectives</td>
<td>Vision Quest</td>
</tr>
</tbody>
</table>

Table App.2: Goal Analysis Form for Vision Quest Auto Dealership Case