Issues in Specifying Requirements for Adaptive Software Systems

Qian Peng
Acknowledgements

First of all, I would like to thank my families for their constant support. I also want to express my gratitude to my supervisors, Jesper Andersson and Mathias Hedenborg. They continuously and patiently proposed comments and hints that have made this thesis into what it is. I would also like to thank my opponent Noel Galindo pointing the insufficient parts and giving me suggestions.
Abstract:

This thesis emphasizes on surveying the state-of-the-art in software requirements specification with a focus on, autonomic, self-adapting software systems. Since various requirements are brought forward accord with environments, modeling requirements for adaptive software systems may be changed at run-time. Nowadays, Keep All Objectives Satisfied (KAOS) is an effective method to build goal model. Various manipulations, such as change, remove, active and de-active goals, appear new goals, could mediate conflicts among goals in adaptive software system. At specification time, specifications of event sequences to be monitored are generated from requirements specification.

Key words: requirements specification, self-adapting software systems, adaptive software systems, Keep All Objects Satisfied (KAOS), goal model
Figures

Figure 1.1 Research method ................................................................. 3
Figure 2.1 Intertwined activities ............................................................ 4
Figure 2.2 Types of requirements .......................................................... 5
Figure 2.3 Advantages of adaptive software system ............................... 6
Figure 2.4 Four models by KAOS ......................................................... 8
Figure 2.5 Procedure to build a requirement model ............................. 9
Figure 2.6 Properties of goal modeling ................................................. 11
Figure 2.7 Elevator goal patterns .......................................................... 12
Figure 2.8 Relations among goals ......................................................... 13
Figure 3.1 Capabilities of MediaToGo storage ..................................... 14
Figure 3.2 Generic goal model .............................................................. 15
Figure 3.3 Refinements of hard goal ..................................................... 16
Figure 3.4 Hard goal: Open files .......................................................... 16
Figure 3.5 Appear new goals ............................................................... 17
Figure 3.6 Opening any file—goal model .......................................... 18
Figure 3.7 Change goal—goal model .................................................. 18
Figure 3.8 Change goal—goal model .................................................. 19
Figure 3.9 Remove goal ................................................................. 20
Figure 3.10 Active / de-active goals—goal model ............................... 21
Figure 3.11 Soft goal model ............................................................... 22
Figure 3.12 Overlapping under two conditions ................................... 23
Figure 3.13 Integrated goal model ...................................................... 24
Figure 3.14 Process of deciding goal for goal model ........................... 25
Figure 5.1 Effects between goal model and implementation .............. 27
1 Introduction

This chapter gives a brief introduction about motivation and objectives of this thesis. We put forward the method how to drive the survey for specifying requirements of adaptive software systems. Moreover, we present the structure of this thesis.

1.1 Motivation

In the traditional software development, the first task is to confirm the specification. Then, other tasks could be deployed. If there some components degraded, the system maybe gets collapsed. Meanwhile, there is a challenge for complex environments.

First, there are increasingly users, not only professionals but also ordinary customers. Second, all sorts of systems intercross each other. For instance, a server connects variety of clients under variety of protocols. Third, the designer could not foresee all circumstances. The preferential specification could not direct the developing of software. Fourth, the ascending goals or requirements are different. From structured programming, object-oriented programming to the adaptive programming, adaptive software is being used increasingly frequently by various users. The goal of adaptive software is to adapt the changing of users needs without a need to rewrite a program. The software inclines to adaptive system. Therefore, understanding the requirements of an adaptive system is decisive to develop them correctly. Since the requirements are inconsistent and incomplete, it is imperative for developers reasoning about the requirements of a system’s adaptive behaviors.

Furthermore, Ross and Schoman represented in their seminal paper, “Requirements definition must say why a system is needed, based on current or foreseeable conditions, which may be internal operations or an external market. It must say what system features will serve and satisfy this context. And it must say how the system is to be constructed” [13]

Goals have long been considered to be leading component in the requirements engineering (RE) process. Traditional analysis and design method is partial to the functionality of the system and its interactions with users. On the contrary, goal-driven method asks why the functionalities are needed and how to implement them. It is ineluctability to face the conflicts among multiple viewpoints in RE concern. How can we find the way to detect conflicts among goals in goal model and get eventually model?

1.2 Research objectives

Why do we need to specify the requirements? Specifying the requirements for an adaptive system doesn’t mean that the finished-software just meets the original requirements. Actually, it indicates that under variable circumstances, the adaptive system make corresponding behaviors according to the environments. It includes reconfiguration, optimization and so on. When the goals for adaptive system require alteration, the system should be adjusted corresponding reflection. Goals are seen to have substantial promise in aiding the elicitation and elaboration of requirements. Modeling requirements for systems may change at run-time. For example, the KAOS methodology uses goal as the central concept in requirements acquisition. [1] Furthermore, goals are the refinements of requirements. The development of the system could be build conforms to the goal model.

Therefore, there are 4 questions about the goals in order to specify the requirements for adaptive systems should be considered,
I. Can new goals appear at runtime?

In real life, the customers often offer new requirements, as an expert, we prefer the least energy to change the project and bring the least investment for the customer. During runtime, the goals divided into several sub-goals, in order to disposal the changeable requirements; we need to add new goals as a new module to meet the final goal. In the development of the computer, the initial goal is for computing. With the requirement for us, many new goals are introduced gradually, such as, office software, multiply media, network for communication and so on. When we are reading books through computer, we could also listening music concurrently. These two actions could run at the same time.

II. Can goals be changed at runtime?

The ultimate goal in a project should be realized step by step. However, the environment may be changed when the project is going along. For example, transmit the files between two cell-phones in a same place; the goal could be defined as “transmit the files between two cell-phones through Bluetooth”. However, when the cell-phones are in different places, the Bluetooth could not be applied. But we can transmit the files through SMS, thus, we should change the goal into “transmit the files between two cell-phones through SMS” at runtime. At the same time, if there is not the Bluetooth in the cell-phone, the files should be transmitted in a new way, the goal should be changed into another one. In adaptive systems, in order to realize the ultimate goal and meet the alterable environment, we should consider whether goals be changed at runtime.

III. Can goals be removed at runtime?

Some of applications are always limited by the outer environment. The adaptive system should be according to the changeable environment to alter the goals; some goals may be removed or replaced by new goals at runtime. Nowadays, the phone could be used with handwrite. The “press button” goal could be removed at runtime. When there are redundant goals or futile goals, the adaptive system could intelligently remove goals at runtime.

IV. Can goals be activated or de-activate at runtime?

In the modern development, we all prefer energy saving. The needless resource could be de-activating, when it is evoked by the system, it could be activated and be brought into application.

In these dynamic processes, it is inevitable to face the conflicts of the goals. We also focus on how to get the tradeoff among the goals. Moreover, we will put forward “How” questions to get the solution of specifying requirements around 4 questions.

1.3 Research method

This thesis puts forward the issues in specifying requirement for adaptive systems. Goals are central in some requirement engineering (RE) frameworks. Goals generally describe objectives which a system should achieve through cooperation of agent in the intended software. While reasoning about goal satisfaction in the RE process, we need to use condition properties. Under this condition, we present goal models and analyze models that may change at run-time for adaptive systems. According to 4 questions, we will detect and resolve the conflicts. Then, we will show ultimate goal model. At the end of this thesis, we will present some unsolved issues and concerns in this condition. Figure 1.1 shows the steps of research method:
1.4 Structure of the thesis

Chapter two represents an overview of software requirements engineering, adaptive software system, KAOS and goal model. Chapter three proposes a case to build goal model through KAOS. Through analyzing goals being appeared, changed, removed, activate, deactivate, we get the conflicts among goals but building goal model for adaptive system. In chapter four, it represents the evaluation for what we have done in this thesis and elaborating how to resolve the conflicts among goals. Finally, this thesis ends in Chapter five with conclusion and directions for feature research.
2 Backgrounds

This chapter presents the background from two aspects: adaptive software system and software requirements engineering. In the adaptive software system part, we will focus on what are the adaptive system and the advantage of adaptive system. In the second software requirements engineering part, we will present the definition of software requirements engineering. Then, we will introduce the goal modeling, and the KAOS methodology.

2.1 Software requirements engineering

Requirements engineering plays the role as the bridge between real-world needs of users and the capabilities, qualification of software system. It associates with the eliciting, analyzing, modeling, consolidating, specifying, evolving, evaluating and recording a set of specification for systems in order to satisfy their stakeholders.

2.1.1 Requirements Engineering

Requirements engineering present why a software system is needed and the fulfilled functions with constraints on how the software must be designed and implemented.

Van Lamsweerde [8] describes intertwined activities that are covered by requirements engineering:

![Intertwined activities](image)

Figure 2.1 Intertwined activities

Figure 2.1 represents a set of activities being featured, including domain analysis, elicitation, negotiation and agreement, specification, specification analysis, documentation, and evolution.

Domain analysis: The relevant stakeholders are explicit and interviewed. The environment for the system-to-be is studied. Through researching the problems of current system, the developers investigate the improvement. General objectives for the target system are identified. For example, when users play music via internet in the cell phone, the domain for this requirement is “play music”, but this phone is out of space, thus we should change the domain to “get enough space” first.

Elicitation: In order to meet the explicit objectives, alternative models for the target system are analyzed. Hypothetical interaction scenarios could help to find the assumptions and implicit requirements in the elicitation process. Alternative models assign different boundaries between the software-to-be and its environment.

Negotiation and agreement: Alternative requirements and assumptions are evaluated; with the gradually appeared conflicts and risks are analyzed; the best alternatives model could be specified.

Specification: with the progressively analysis of models, requirements and assumptions are formulated precisely step by step.
Specification analysis: Under a certain amount of resources, in order to assure the feasibility of the system, the specifications check the remained problems, such as incompleteness, inconsistency.

Documentation: it is the official statement for the system, which includes the definitions and specifications of the requirements.

Evolution: the requirements are modified regarding new objectives and changeable environments.

“Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families.” [2]

Zave’s definition emphasizes that the goals drives the development of software. The specifications may be violated via implementing or the requirements are changeable. Thus, it is necessary to apply goal modeling in a certain domain in response to the changeable environments.

2.1.2 Types of Requirements

Requirements are categorized in several ways. It could be roughly classified into three types, customer requirements, software requirements, and domain requirements. Figure 2.2 exhibits three types.

I. Customer requirements; from the customer viewpoint, they give the description of the objectives of the system that include the functions, environment, constraints, and the qualities of the system. The statement is in natural language with diagrams about the service that the system provides. As the general requirements for the system, this is summarized and represented as the form of use cases by developers.

II. Software Requirements; it includes functional requirements and non-functional requirements. It is a translation and precise description of customer requirements.
   a) Functional requirements; it define the functionalities of the system. It comprises necessary tasks and activities which must be fulfilled in system. For example, what inputs the system could accept; what outputs the system could produce; the system should perform computations and so on.
   b) Non-functional requirements; it mainly views the qualities of the system, such as availability, performance, security, reliability, usability. The non-function requirements should be more significant than the functional requirements. The system is useless if it doesn’t satisfy the non-functional requirements.

III. Domain requirements; this requirements decide the running of the system. It is from the application domain of the system and describes the system characteristics that reflect the domain. It may be some new functional requirements as the interfaces or the limitation of requirements. For instance, in the library system, there is a standard of
database. Regarding the rules, copyright documents should be deleted immediately after printing online, or these documents should be printed locally on the system server.

2.2 Adaptive software systems

Adaptive software system is built to respond to multivariate requirements of the users. The prominent trait of adaptive software system is self-adapting. It allows several possible operations being performed by a given user.

2.2.1 Definition of adaptive software systems

We all get that robots have some intelligent functions through film, series or report. It could play actions which are response to environments’ changing. The robots exactly adopt adaptive system to adapt accordingly. An adaptive system is a set of interacting agents. It is launched by the competition or cooperation among agents. The processes are dynamic, allowing the response to changes, acting in parallel or pipeline. The computer is a set of the simple self-adapting process. For example, the computer has the power management, if your laptop connected with power, the computer shift to charging the battery and using the power. If you cut off the power of the laptop, the computer inspires battery immediately and decreases the brightness of your screen.

An adaptive system has the capability of self-adapting according to changing environments. The self-adapting concludes self-configure at runtime, self-optimize for system, self-modify to update or shift the actives. Adaptive system is contrary to the static system. Static systems are not equipped for self-modification and only are able to function within a small range of environmental change. Static systems are unable to adjust to novel environmental scenarios. [3] In most desk-top, the power management is a static system. It only contains the on and off functions. However, in laptop, the power management is an adaptive system; it has the storage for power. And the screen could be adjusted by the power.

2.2.2 Advantage of adaptive software system

When we develop a software system, we may be confused by the question, what are the advantages for adaptive software system? Figure 2.3 shows the advantages of adaptive software system.

<table>
<thead>
<tr>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-adapting</td>
</tr>
<tr>
<td>Harmonizes different types of requirements</td>
</tr>
<tr>
<td>Meet much more requirements</td>
</tr>
<tr>
<td>Competitive in the market</td>
</tr>
</tbody>
</table>

Figure 2.3 Advantages of adaptive software system

First, self-adapting; comparing to the static systems, the foremost is the self-adapting trait for adaptive system. It could get the feedback and perform reaction. It happens to
meet the changeable environment. Human organizations, nature ecosystems are nature adaptive system. In artificial systems, for instance, robots employ control systems to use the feedback to recognize and adapt its environment. It simplifies the manipulations and brings the convenience to the users.

Second, it harmonizes different types of requirements. According to above types of requirements, the system should be adaptive. The standard for the domain could be various. Thus, a system satisfies different standard in different environments. In order to meet the functional requirements, the non-functional requirements should be self-adjusted, and vice versa. For instance, when we connect with internet, in order to achieve the safe environment from non-functional requirements, we should provide the function—firewall to defend computer out of attack.

Third, it could meet much more requirements. In the software developing, it is inevitable face the changeable requirements and environment. The stakeholders would modify their requirements again and again to meet their final accesses.

Fourth, adaptive system could be more competitive in the market. In our daily life, the electrical productions are apt to the intelligence, in other words, they are all comprising self-adapting functions. For the sake of conforming to the market, it is incline to develop adaptive system.

The above views reveal the advantages of the adaptive software. Regarding inconsistent requirements, adaptive system is much more suitable for changeable requirements.

In order to get the system perfect and emulative, it is prone to research and develop the adaptive system. However, there are also limitation and rules for adaptive system. For instance, owning to the large scale changes to an environment which induce the falling of performance, the system should eliminate some adaptive functions to achieve the performance. It is impossible that an adaptive system could possess global adaptability. Generally speaking, adaptive system is more suitable for changeable environment.

2.3 KAOS

KAOS stands for Knowledge Acquisition in automated Specification or Keep All Objects Satisfied. KAOS is a methodology for requirements engineering approach analysis techniques and deriving requirements documents from models.

2.3.1 Four models by KAOS

Through analyzing with KAOS, there are 4 models: the goal, responsibility, object, and operation models. Goal drives the development of the system. However, beside goal, agent, a person or an automated component, plays an important role in a system. Agents are responsible for achieving requirements. Objects could be a set of entities, agents and associations. Object model links the application domain and provides constraints on the operational system. Furthermore, operation model contains all behaviors of the system. Figure 2.4 exhibits four models by KAOS.
The goal model represented the requirements. It is a set of interrelated goal diagrams that are blended them together to deal with a problem. Through asking how a goal can or should be implemented, the goal could be broken down into a set of sub-goals. The model also could reflect the relation among sub-goals. The sub-goals could be independence or associate with each other. For example, John wants to borrow a book from the library. The goal is “borrow book from library”. In one way, he could book his requirement via internet. Then, the librarian posts the available book to his address. In another way, he could find the book in the library and borrow it through machine. Above two ways reveal that the two sub-goals: “book from internet” and “find and borrow in the library” are independence. No matter what way John will take, he could borrow the book. This thesis will concentrate on the goal model.

Responsibility model is a set of derived responsibility diagrams. An agent is response for a diagram. It is derived from goal model. According to goal model, it would be assigned an agent. In the “borrow book” example, “John” is response for the goal: “book from internet”.

The object model describes the concepts of the application domain and constrains the operational system. The objects could be 3 types: entities, agents and associations. Entities describe and convert the state of the object but not carrying out operations. And the agents could perform operations. Associations are dependent objects and could not perform operations.

Operation model presents all behaviors which are handled by agents. The operations depend on the requirements of stakeholders. Operations manipulate objects that are connected to operations as “input” and “output”.

2.3.2 Key ideas underlying KAOS

Development teams adopt modeling techniques for specifying solutions. KAOS is a method to directly build model to analyze the requirements. Through KAOS, we could process following ideas:

1. Build a requirements model
2. Justify your requirements by linking them to higher-level goals
3. Build a model of the whole system, not just the software part of it
4. Build a responsibility model

Figure 2.4 Four models by KAOS
| 5. | Build a consistent and complete glossary of all the problem-related terms you use to write the requirements |
| 6. | Describe how the agents need to behave in order to satisfy the requirements they are responsible for |
| 7. | Base the requirements document on the requirements model |
| 8. | Validate your requirements by first reviewing the model |
| 9. | Use a defensive approach to the building of a requirements model |

**Figure 2.5 Procedure to build a requirement model**

KAOS could describe the problem, then analysis to provide a systematic way for researching and structuring requirements, and clarify the stakeholders’ responsibilities for efficient communication. Each goal is refined to a set of sub-goals in the model, we should investigate why we introduced new goal in the model. However, there may be some goals are conflicting. Then, we should consider can goals could be changed, removed, active and de-active at runtime. For example, in order to improve the performance of the system, we should remove some goals to decrease the usability. Dealing with conflicts and achieving the balance in the adaptive system need to build a more complete requirements document.

The goal model should consider all requirements and assumptions that even refer part of environment interacting with the system. The goal model is from the system-to-be to relate the high-level and low-level system.

Beside goals, agents are response for achieving the requirements. The goals are always assign to several agents. When there is a single agent response for the goal, the goal refinement is no longer necessary.

KAOS brings an easy way to understand the requirements. The KAOS model reveals a remarkable style to communicate about the requirements and help to discover the conflicts among goals. After analyzing the requirements, we could follow different strategies to address conflicts.

### 2.3.3 Why use KAOS

Nowadays, the efficient and prominent way to build goal model is KAOS. KAOS can be used for any type of information system. It has been using it in different industries such as mechanics, telecommunication, and health care. Furthermore, the KAOS is approach public administrations. [5] There are several reasons for why we use KAOS for building goal model.

First, KAOS provides a methodical and logical way to structure requirements. KAOS puts forward graphical way to tackle a problem. Moreover, it classifies the requirements into 4 models regarding goals, agents, objects and operation. For instance, goal model represents the desired properties of system.

Second, KAOS allows defining concepts relevant to the problem description. The initially description of requirements is from customer viewpoint, but developers could confirm the problem to define and operate concepts by KAOS. Goal model is the desired properties of system.

Third, KAOS could clarify the responsibilities of all stakeholders. Each requirement is related with an agent who is responsible for. Responsibility model is used to illustrated responsibility of an agent.

Finally, KAOS provides a clear hierarchy for involvers and stakeholders could easily and efficiently communicate about the requirements.
Regarding above reasons, we build goal model by KAOS. It improves the problem analysis process by a systematic approach. Moreover, it also proposes a graph to make it readable and understandable.

2.4 Goal model

Goal modeling represents stakeholder intentions and their refinements using formally defined relationships. Traditional analysis and design methods focused on functions of the system. However, goal-driven requirements engineering emphasize the requirements should be launched by why the functions are need and how these could be implemented, but not focus on what needs to be implemented.

2.4.1 Definition of goal

At the beginning of building goal model, we should figure out what a goal is and where goals come from. The definitions found in the literature

"A goal is a nonoperational objective to be achieved by the composite system. Nonoperational means that the objective is not formulated in terms of objects and actions available to some agent in the system; in other words, a goal as it is formulated cannot be established through appropriate state transitions under control of one of the agents." [6]

"Goals are targets for achievement which provide a framework for the desired system. Goals are high level objectives of the business, organization, or system. They express the rationale for proposed systems and guide decisions at various levels within the enterprise. Corporate profits maximized are an example of a high-level enterprise goal. The two primary types of goals discussed in this thesis are achievement and maintenance goals." [19]

A goal is an objective for the system which includes software and environment. The formulation of goal indicates the intended behaviors which should be achieved. A goal could be temporary for current system or the system-to-be. The system-to-be goals often involve both software and its environment.

Goal is classified into function goal and non-function goal. Function goal refers to the functions of the system. Non-function goal indicates qualities of system, such as performance, security, usability and so on.

Sometimes goals are explicitly stated by stakeholders. However, after analyzing current system, a list of goals would be identified. Thus, many goals are elaborated by asking HOW and WHY questions associated with available goals.

2.4.2 Properties of goal model

“The subject of RE is inherently broad, interdisciplinary, and open-ended. It concerns translation from informal observations of the real world to mathematical specification languages. For these reasons, it can seem chaotic in comparison to other areas in which computer scientists do research.” [2]

Although there are lots of methods to research the RE, it is helpful to apply the goals into this area. The goal model is the hierarchy of goals and relates the high-level goals to the low-level goals. There are some properties of goal modeling for reasons about goals in RE.
Figure 2.6 Properties of goal modeling

Figure 2.6 exhibits the properties, including guiding requirements elaboration, relating requirements to organizational and business context, clarifying requirements, and dealing with conflicts, driving design, traceability of rationales.

Guide requirements elaboration

Goals are helpful for eliciting and elaboration of requirements. Around the root goal, putting forward to “why” and “how” questions step and step. The requirements may be implied in the process of elaborating a goal hierarchy. Moreover, stakeholders may be aware of the potential alternatives for meeting their substantive goals.

Relating requirements to organizational and business context

Nowadays, from the business viewpoints, the relationship between systems and their environments are being expressed in terms of goal-based relationships. Modeling techniques need to support "why" and "how" types of reasoning analysis.

Clarifying requirements

Requirements are often complex and unclear when first elicited from stakeholders. Goal modeling allows the requirements to be refined and clarified though hierarchy in many levels. Each level addresses the demands of the next level. It widely used in the non-functional requirement, such as flexibility, performance, maintainability and security.

Dealing with conflicts

Due to various viewpoints inducing the overlap and confliction, it is inevitable that goals conflict with each other. The foremost is to find the tradeoff among the conflicts. Goals provide a useful way of dealing with conflict because the meeting of one goal may interfere with the meeting of others. Different conceptions of meeting a goal – satisfaction [6] vs. satisfying [7] has led to different ways of handling conflict. [4]

Driving design

Goals have been used as an important mechanism for connecting requirements to design. The Composite Systems Design approach used goals to construct and later prune the design space. The NFR framework treats non-functional requirements as goals to guide the design process [7].

Traceability of rationales

Goal modeling provides rationale relationship. We can track down through the traceability links.

Goal modeling is a hierarchy of goals. It relates the high level goals to low level system requirements. Goal modeling represent variability in the way high level
stakeholder objectives can be satisfied by the system with changeable environment. It could capture the variability in certain condition.

2.4.3 Patterns of Requirements in Goal Model

Goals comprise functional or hard goals and non-functional or soft goals.

I. Hard goals are described as some behaviors, such as data manipulation, technical and processing. What’s more, hard goals drive the application architecture of a system.

II. Soft goals are used to judge the operation of a system from qualities, such as availability, maintainability, reliability, safety, performance and so on. And soft goals drive the technical architecture of a system.

Take the elevator control system for an example. The hard goals fulfill receiving calling, starting elevator, stopping elevator functions; the soft goals involve safe system, usable system, and efficient system. Figure 2.7 shows the elevator system to illustrate the patterns of requirements in goal model. [5]

Hard goals refer to the system’s functions, either satisfied or denied the function. Soft goals describe desired system qualities, such as the reliability, performance and security and so on.

When the two kinds of goals interlace each other, there should be conflicts among them; the method that we could deal with conflicts is getting the tradeoff through supporting the positive quality and improving the negative quality.

2.4.4 The use of symbols in the Goal model

Van Lamsweerde gives the definition for the goal that is an objective the composite system should meet. Goals must be eventually AND/OR refined into requisites assignable to individual agents. [11] Generally speaking, a goal corresponds to an objective the system should achieve through cooperation of agents in the software-to-be and in the environment. [8]
Two basic symbols are used in goal modeling via KAOS technique. The elements of a KAOS goal model are presented in following figure.

The rectangle represents the goal which presents desired behaviors to meet the cooperation of agents.

Condition property presents objects’ state. It may be a domain invariant or a hypothesis.

The symbol presents the link among goals.

These symbols are linked to each other with annotated lines representing semantic relationships. These relations include:

AND: In order to achieve the goal, all of sub-goals should be satisfied.

OR: in order to achieve the goal, at least one of sub-goal should be satisfied.

Figure 2.8 Relations among goals
3 Case study

This chapter we will represent case to build the goal model for adaptive system. In this adaptive system,

I. Can new goals appear at runtime?
II. Can goals be changed at runtime?
III. Can goals be removed at runtime?
IV. Can goals be activated or de-activate at runtime?

We emphasis the hard goal models in this case. Through the goal models, we bring forward the resolution for each question and try to find the tradeoff in response to variable goals. We also simply analyze the soft goal model. Furthermore, we study what we need to do in implementation.

3.1 Case description

The media application is targeted for mobile devices. Every user has access to several secure media storages. A MediaToGo-storage can be described as a combination of the following characteristics; local or remote, private or shared, and free or pay-service. For example, combine in to “A local, shared storage”, “A remote, pay-service storage”, or “A remote, shared, free storage”. The user organizes its storages in the MediaToGo application.

Figure 3.1 exhibits the capabilities of MediaToGo storage.

![Figure 3.1 Capabilities of MediaToGo storage](image)

The MediaToGo architecture should be a flexible plug-in architecture, where different communication protocols, media viewers and access-control systems are (de)-installed on demand. The installation and activation of a plug-in is controlled by an autonomic element, which is a rule driven engine that reacts to events and guarantees that an adequate combination of viewer-protocol-access control is available when a user opens a media-file or selects a MediaToGo-storage for browsing. The MediaToGo-application is targeted for mobile devices, which means that the set of
available storages, suitable and communication protocols will change over time. The application must self-adapt to such change also. In this thesis, we just consider connecting internet as the changeable condition.

### 3.2 Building goal model

The goal of the system is to build a system that could automate to identify the file’s format and load the program to open it. However, due to changeable environment, the goals should get modified but still satisfy the requirements.

According to above description of the case, and we focus on analyzing about changeable internet. The following table represents the goal, hard goals and soft goals, and then we will get the generic goal model.

<table>
<thead>
<tr>
<th>Goal</th>
<th>The media application: MediaToGo –storage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hard goals</strong></td>
<td>1. Open any file</td>
</tr>
<tr>
<td></td>
<td>2. Download the viewer</td>
</tr>
<tr>
<td></td>
<td>3. Connect the internet</td>
</tr>
<tr>
<td><strong>Soft goals</strong></td>
<td>1. Safe system</td>
</tr>
<tr>
<td></td>
<td>2. Usable system</td>
</tr>
<tr>
<td></td>
<td>3. Efficient system</td>
</tr>
</tbody>
</table>

![Generic goal model](image)

As above figure showing, this system is distinguished to 2 parts, hard goals (function goals) and soft goals (non-function goals). From the hard goals, we should fulfill these functions: “open a file”, “download viewers”, “connect internet or not”.

The soft goals we should assure the system to be safe, usable, efficient system.

#### 3.2.1 Hard goals

In our expectation, we would like to get the storage that is always able to open any kind of files. Considered the question: how can or should be a goal implemented? The goal “open any file” is refined as two sub-goals: “access file” and “open a viewer”.

15
The files are distinguished into local or remote files if the storage connected with internet. Furthermore, open a viewer could consider as two goals “find viewer” and “launch viewer”. When the viewer has installed, user could use it directly. In another way, the user should download viewer rest with the “connect internet” environment. The following figure shows the elementary of the hard goals.

3.2.1.1 Can new goals appear at runtime?

Goals are appeared by asking how to implement the ultimate goal. At the runtime, new goal appears step by step.

As above mentioned, the environment is switched between the connected internet and out of internet. The new goal: “space checked” appears to confirm the usability. When the space is enough, we combine the “connection” and “enough space” so that the storage could open any file. Otherwise, the goal is still “open supported file”.

Figure 3.3 Refinements of hard goal

Figure 3.4 Hard goal: Open files
However, according to the requirement or changeable conditions, new goal appears with more complex control system and manipulation for the storage. Moreover, new goals bring strictly conditions and conflicts with other goals. In reality, we will avoid appear much more new goals but still an effective goal model.

### 3.2.1.2 Can goals be changed at runtime?

The ultimate goal for the storage is “open any file”. However, the goal is not available under all conditions. Due to the constraints of environment, the storage could connect the internet or not. In this case, there are two condition properties: “internet connected” and “out of internet”. The goal model should be changed or not accordingly.

Following shows the goal model for ultimate goal.
Condition hypotheses 1: the MediaToGo is out of connecting with internet, but it could use the local stuffs.

When the storage is out of internet, no adaptive viewers could apply for some files. In this situation, the storage could open local files and read the files under the supported viewer. Therefore, the goal "Open any file" should be changed into "open supported file".

Condition hypotheses 2: the MediaToGo connects internet, and it could use the remote stuff.

Although the storage could download viewers via internet, the space of the storage is another issue. The space could be used for downloading files and viewers. Therefore, new goal—"space checked" should be appeared at runtime owing to the capability of
the storage. There are two conditions for space checked, first, accessing and downloading remote files; second, downloading viewers. On one hand, there is not enough space for downloading files or viewers, the goal “Open any file” should also be changed into “open supported file”. On the other hand, the space is big enough, the goal should be “open any file”.

3.2.1.3 Can goals be removed at runtime?

Under the internet connection, the goal model could implement the ultimate goal. MediaToGo could “open any file”. However, as soon as the storage is out of internet, some capabilities could not be fulfilled. Thus, the series of goals via internet should be removed. For instance, “download the viewer” and its sub-goals should be removed due to disconnection of internet. What is more, some goals are also removed due to the space. Following is the comparison of storage opening supported file under connection of internet. One is for enough space of file and viewer. Another is for enough space of file but lack of space of viewer. Thus, the goal model removes the “download view” goal.
Although we propose to avoid appearing new goal, we consider removing goal if necessary. All users prefer getting a fully functioning storage. Removing goal is a way to diminish the negative effect of the storage based on performing ultimate goal. But removing goal tends to limit the function of the storage.

Actually, removing goal interlace with de-active goal. If it is not necessary, the part could be de-activating according to following analysis.

3.2.1.4 Can goals be activate or de-activate at runtime?

Generally speaking, in the relation of the goal model, the “OR” relation of goals could be active or de-activated at runtime. According to changeable condition, the goal should be active or de-activated at runtime.

As we have emphasis the condition properties, when the condition changed, it could induce the goals to be changed. Moreover, there can be some goals activated or
de-activate. When network is internet connected, the sub-goal “space checked”, “remote file” and “download viewer via remote” could be activate. On the contrary, the network is out of internet, the aforementioned sub-goals could be de-activated.

Figure 3.10 Active / de-active goals—goal model

Activate or de-activate goal is a significant method to keep capabilities of the storage. At the same time, it is wasteful of resources. Now that some parts, may be changed by other essential goals or never be active, these parts could be removed. In consideration of further researcher or an emergency measure, these parts could be built in extra model in application.

3.2.2 Soft goals

From the viewpoint of soft goals, it is difficult to define which aspect is more important. Software always contributes positively to several quality attributes but some are affected negatively. What we intend to do is how can minimize the negative contribution and maintain the positive attributes.

As a matter of fact, in this storage, there are some conflicts among those qualities of soft goals. For instance, if it is required to get high performance storage; we could
OR

Safety system

Efficient system

Usable system

Cheap system

High performance

Low performance

Figure 3.11 Soft goal model

I. Can new goals appear at runtime?
Yes, if the storage is running at the internet condition, the new goal—“safe system” should appear to assure the safe under internet. Therefore, the storage could be safe and efficient.

II. Can goals be changed at runtime?
Yes, the storage is running at the local system with high performance. However, the storage is required to connect with internet; the system would occupy several resources due to safe and cheap system. Therefore, the initial goals—“high performance” without internet should be changed into “low performance” with internet at runtime.

III. Can goals be removed at runtime?
Yes, if the storage is prone to high-level for using, the “cheap system” should be removed. Although under the connection of internet, the “Low performance” should be removed to satisfy high-level. When the storage switches from “connection” to “disconnection”, the safe system could also be removed.

IV. Can goals be activated or de-activate at runtime?
The same as we mentioned above, the goal could be activated and de-activate at runtime accordingly. The “OR” of relation just reflects the feasibility in the goal model. The “High performance” and “Low performance” could be activated and de-activate base on variable environment.

From the soft goal model, we can get how many aspects that we should pay attention to. Then, in the adaptive system, the system should use different soft goal model to promote the configuration of the device. Therefore, the storage must improve cost and consume other qualities. Moreover, the storage could connect internet, users should face the problem of safe when users access network. The efficient, usable, and safe and other soft goals should be considered depending on the variable condition. MediaToGo architecture is lean to specific aspects of quality depending on requirements.
answer the request. The self-adapting of the system could be build referring to the goal model.

3.3 Discussion and related work

Consider the tradeoff among goals and the influence of manipulation at runtime; we get the hints for building ultimate goal model.

The above example explained the process of building goal model with KAOS. In order to adapt the changeable conditions and minimize the conflicts among goals, we introduce 4 manipulations: appearing, changing, and removing, active / de-active with goals relevant to goal model at runtime.

However, each manipulation has its drawback. Changing goal could not bring the ultimate goal; appearing new goal could bring complex control system; removing goal could limit the functions of storage’ active/ de-active could waste resources. Each manipulation could only resolve partially problems at runtime. How can we get the tradeoff of goals and integrate goal model to implement goal model?

We summarize above analysis and discuss the case from two goal models. On the one hand, condition hypotheses 1: the MediaToGo storage is out of internet. On the other hand, condition hypotheses 2: the MediaToGo storage connects internet.

In the implementation, the “internet connected” properties could be de-active when the storage is out of internet. At runtime without internet, the goal is supposed “open supported file”. Moreover, the series of goals related to internet should be de-active.

In these two goal models, new goal appear by processing of storage, ultimate goal can be changed at runtime if the condition changed. In these two conditions, some parts are overlapped. Figure 3.12 exhibits the goal overlapping under the two conditions.

![Diagram](image)

Figure 3.12 Overlapping under two conditions

The upper text is the pre-active for running the storage. At runtime, the two red parts are the same no matter what condition it is. When the internet connected, we represent
one soft goal—“safe system” into the model. Figure 3.13 shows the goal model for the storage.

In above figure, in addition to the condition could be changed, the two goals with an arrow connected could also be changed at runtime. However, when the goal is “open supported file”, the red part should be deactivated. Instead of active the red part, the storage should connect internet. If and only if the space is checked enough, the goal could be changed into “open any file”.

In implementation, first of all, we should decide the goal and construct the goal model. The goal model is constructed by ask how to deploy implementation, but the goal should be confirmed with condition satisfied. Since the goal is changed by the condition, we should recognize the condition. At the same time, the sub-goal “space checked” also affects the goal model. The goal model for “open any file” is satisfied by “internet connected” and “space enough”, or else the goal should be “open supported file”. Figure 3.14 shows the process of deciding the goal for goal model.
Second, the figure 3.11 shows the red part is overlapping. It could be sorted out the same module and economize resource in the development.

Third, what above mentioned are all around the goal model, but with the method of KAOS we should consider other 3 models: responsibility, object, and operation models, which I have introduced in the chapter 2.2.2.1. The details about building 3 models could refer to KAOS tutorial. [5]
4 Evaluation

Our goal for this thesis is to build the goal model and analyze the adaptive goal model to variable environment with 4 questions,

a) Can new goals appear at runtime?

b) Can goals be changed at runtime?

c) Can goals be removed at runtime?

d) Can goals be activated or de-activate at runtime?

First, we give the background knowledge to bring forward the requirement engineering, adaptive system, the technique—KAOS and goal model. Second, we represent the goal model through case study. From the case study, and analyzing above 4 questions, we get the following results and experience.

The goal model is build by asking how can or should be a goal implemented. Hard goals and soft goals are prone to interlace in a system. The goals drive the development of system. Hard goals are from the function aspects to fulfill the behaviors of the system. Nevertheless, the soft goals evaluate the system from qualities. In the adaptive system, the goal model could be considered being individually changed, appeared, removed, active and de-active at runtime according to variable environments.

The distinct goal model is build to meet the requirements according to changeable environments. In real systems, it is inevitable existing conflicts among goals. If the system achieves a state but be conflicted with another goal. It is not possible to meet both goals simultaneously. As we have showed in the case study, for instance, the performance may conflict with safe system and cheap system. Moreover, different users also put forward different requirements to produce conflicts. Therefore, how to solve the conflicts among goals and get the tradeoff among them is important in the software life cycle.

In the process of analyzing, we get that if goals are in face of conflicts, one of goal could not be satisfied. That is, one of sub-goals could not be meet. There are three way could solve partially problems.

First, appear a new goal to prevent the conflicts. For example, if the storage runs at local and high performance. When it is switched into remote, the configuration could not guarantee the safe and high performance; there should appear new goal “safe system” and “low performance” to continue the performance.

Second, change the goal, the system could record the conflicts and restart. When the system is out of connection of internet, the storage should only open the supported files in local. Thus, the goal should be changed from “open all files” into “open supported files”.

Third, de-active or remove the goal, if it is necessary to remove goals but still satisfy the requirements, else preserve the goal but de-active the goal. The choice for removed goal depends on the tendency of requirement.

Dealing with conflicts but preventing goals from being achieved demand you to build a more robust system at the first shot. Minimize the negative effects and get the tradeoff among those goals.

In implementation, integrating analysis around 4 questions and solving the conflicts, we could assure the goal model for the storage. Besides goal model, KAOS also give weight to other three models in implementation.
5 Conclusion and future work

In this chapter we will get conclusion. Then, we will represent the future work.

5.1 Conclusion

In this thesis, we want to learn how can specify the requirements for an adaptive system and build the goal model via KAOS method. We deploy goal model from hard goals (functional) and soft goals (nonfunctional) ways. Surrounding whether goals could be changed, removed, and active/ de-active and new goal appear or not, specify distinct goal model for the adaptive system.

In the beginning of this thesis, we start with the introduction adaptive system, requirements engineering. Second, we simply introduce KAOS as the method building goal model. Then, we construct goal models around 4 questions to detect the conflicts in case study. We simply describe how to implement the goal model and introduce the problems that should be studied a step further.

As we exemplify above, we generate the goal model according to changeable environment. Through the goal model, we analyze hard goals and soft goals respectively. At runtime, new goals appear by processing for implementation; the goals could be changed according to its conditions. Due to the requirement, remove a goal if necessary, else de-active a goal to deal with emergency or study more. However, it is unavoidable to face the conflicts. Whether goals being changed, removed and active / de-active and appearing or not resolve the conflicts. Whatever kind of conflicts, the goal model should be build base on the minimizing the negative effects to remodel.

It is not difficult to individually consider to directly changing, removing, appearing, and active, de-active goal in goal model. And the functions of the system are also simply altered separately. However, the non-functional requirements affect the functional requirements, and vice versa.

![Figure 5.1 Effects between goal model and implementation](image)

In real implementation, the developer should consider both functional and non-functional requirements of the system. The implementation is carried out according to goal model. Sometimes, the conflicts could be resolved in the step of modifying the goal model. Similarly, new conflicts may be appeared in goal model at the same time. Once the goal model is arbitrarily changed, it will waste former components and bring loss from the financial resources. Therefore, it could be easy to modify goal model and functions separately, but it is not easy to modify the goal model related to implementation.
5.2 Future work

In the current approach, we introduce the KAOS as the methodology and around appearing, changing, and de-active / active, and removing goals to build the goal model. Instead of correctly requirements specification in goal model, for future work, it would be worthwhile to investigate the tests targeting adaptation strategy embodied in the runtime system. Furthermore, we focus analyzing the hard goals in case study, but what kind of influence is affected combined with soft goals.

There is a long way to build goal model effectively for adaptive system. Although we design the model for above case, how to implement the case, and how to achieve the effective adaptive system, are still issues which should be researched more.
Reference

[1] Eric Yu and John Mylopoulos, *Why Goal-Oriented Requirements Engineering*, University of Toronto


