Integrated Framework for Software Requirement Analysis

Andre Rusli, Osamu Shigo

Graduate School of Information Environment, Tokyo Denki University, Chiba, Japan
{andrerusli19@gmail.com, shigo@mail.dendai.ac.jp}

Abstract. Requirement elicitation is a very critical step in software development. In order to develop adequate software which answers to user's needs, it is essential to understand the real-world environment, the users themselves, goals, constraints, and risks and its possible solutions. Unable to describe correct requirements can lead to a massive software development failure. This paper aims to propose an integrated framework for requirement elicitation which combines the characteristics of goal-based requirement engineering methods, Problem Frame (PF), and Message Sequence Chart (MSC). The proposed framework uses *i** framework to describe the dependency relationships between actors, PF to analyze the constraints that exist in the real world, KAOS' to analyze obstacles, and MSC to show the dynamic behavior of the system.

Keywords: Requirement Analysis · Goal Models · i* · Problem Frame · KAOS · Message Sequence Chart

1 INTRODUCTION

Requirements engineering (RE) is an engineering activity that ties up the development activities with the real-world problems. It represents a series of engineering decisions that lead from recognition of a problem to be solved to a detailed specification of that problem [6]. Failing to describe the correct requirement can lead to a massive failure in the whole development. However, requirement engineering is not an easy task. It has to keep up with the ever-changing environment which caused it to be a continuous process in a software development. To keep up with changes, requirements must have a base condition, with details that can be changed flexibly without drifting apart from the main objective of the project.

There are several methods in analyzing requirements, but currently they differ from one to another. Each method has its own characteristics and fundamentals. In the requirement analysis community, goal-based modeling approaches have gained considerable attention. These approaches aim at capturing the rationale for the software system development [5]. Goal-based modeling approaches focus on the description of goals of the software development. However, goal-based modeling approaches like i* or KAOS lack describing the scenarios of the system. In requirement analysis stage, it is important to analyze the scenarios of the system that already exists and the one

which will be developed. By analyzing scenario, engineers can understand the sequences of the system and also constraints existing in the system.

This research aims to develop a framework that can analyze the actors and goals in a system, while analyzing scenarios that can explain which tasks go first and which go later along with the constraints exist in the environment. The framework combines i* approach to describe the relations between actors in the system along with their tasks and goals, with Message Sequence Chart (MSC) and Problem Frame method to explain the scenarios of the system, the process sequences and constraints in the system. Moreover, it also imports the obstacle analysis from KAOS to analyze risks in the software development and their possible solutions.

We understand that integration needs a good communication line between methods, and there will be intersection points in some methods that can be complicated and difficult to understand. To answer those problems, the next step of this research is to develop a support tool for the integrated framework.

2 EXISTING METHODS

2.1 Goal Based Requirement Engineering Methods

A goal is a prescriptive statement of intent that the system should satisfy through the cooperation of its agents [4]. The core of goal model consists of a refinement graph showing how higher-level goals are refined into lower-level ones and, conversely, how lower-level goals contribute to higher-level ones. Among many goal based methods, this research takes two popular methods into consideration, KAOS and i* framework.

KAOS is a goal-driven, model-based approach for elaborating a complete, adequate, consistent, and well-structured set of measurable software requirements and environment assumptions [2]. Obstacle analysis in KAOS is a goal-anchored form of risk analysis whereby exceptional conditions obstructing system goals are identified, assessed, and resolved to produce more complete requirements [4]. While *i** framework is an integration of goal based requirement engineering and agent based requirement engineering method. *I*'s* strategic dependency model is used to describe the dependency relationships among various actors in an organizational context [7].

2.2 Problem Frame

Problem Frame [3] (PF) considers that it is important to focus directly on a problem, not just going straight to the design of a solution. We need to recognize that the solution is located in the computer and its software, but the problem is in the world outside. A problem frame consists of *domains*, *interfaces* between them, and a *requirement* [5]. Domains describe entities in the real world, while interfaces connect domains and they contain shared phenomena. Each requirement constrains at least one domain. Such a constrained domain is the core of any problem description because it has to be controlled according to the requirements [5].

2.3 Message Sequence Chart

Message sequence chart describes the scenario in the system. It shows the flow of data and the actor responsible for sending, and receiving messages in the system. Message Sequence Charts (MSCs) are a technique to describe patterns of interaction between the components of interactive distributed systems by specific diagrams [1]. Processes are described in vertical lines and messages passed between actors (source and target) are written in a horizontal arrow with description above them. Unlike UML's sequence diagram, MSC does not specifically describe the sequences for the design-level which includes lifelines, methods, etc. It functions only to describe the scenarios and the orders of resources flowing in the system which are necessary in the requirement elicitation process.

3 INTEGRATED FRAMEWORK

3.1 Meeting Scheduler Problem

This paper uses a problem about meeting scheduler software in order to explain the integrated framework. The problem example used in this paper is taken from a version found in [7]. Meetings are typically scheduled as follows: an initiator informs participants about the need for a meeting and specifies a date range, then asks the participants to inform their availability. The scheduled meeting date should belong to the stated date range and to none of the exclusion sets; it should ideally belong to as many preference sets as possible [7].

3.2 Strategic Dependency Model

In order to elicit requirements, first, it is necessary to elicit who the actors involved in the system are because each actor has different goals, tasks, and resources. Actors can depend on other actors for some goals, resources, or tasks to be completed. In the meeting scheduler problem, initially there are two actors, meeting initiator and meeting participants. I*'s model shows how those two actors depend on each other. For example, meeting initiator depends on participants for goal "meeting is fully attended". Moreover, each actor's own goals and tasks decompositions are also described inside each actor in a hierarchical system.

By describing these relations in i* model, we can get the big idea of the environment, actors involved, and their relations. In addition, not only the outside relations of actors, the goal and task decomposition inside each actor can also be understood. These are considered essential in order to elicit software requirements.

3.3 Behavior Analysis

Although the i* model has already described the relations between actors, goals and tasks decompositions, and the idea of the system, it lacks in describing the dynamic

behavior of the system. By looking at i* diagram only, we cannot understand which task goes first and which goes later, nor we can know which resource is needed first before the other. Thus, this paper integrates the Message Sequence Chart (MSC) into the framework, to describe the scenarios exist in the system. However, unlike UML's sequence diagram, MSC supports the characteristics of i* model which separates the requirement analysis process and the software design process by not going too deep into the development processes. MSC shows the flow of the resources between actors and its orders which represent scenarios in the system, not computer methods nor classes. In this integrated framework, the actors in MSC are actors from i* model, and the messages in MSC are the resources from i*.

3.4 Inserting Machine Actor

After the system's environments are understood by doing the earlier steps, now it is necessary to describe how the environment will work if we add a new machine actor, which is the software that will be developed, the meeting scheduler system, into the i* model and reconnect the relations between actors, as shown in figure 1, except the requirement circle as it will be explained in the next step.

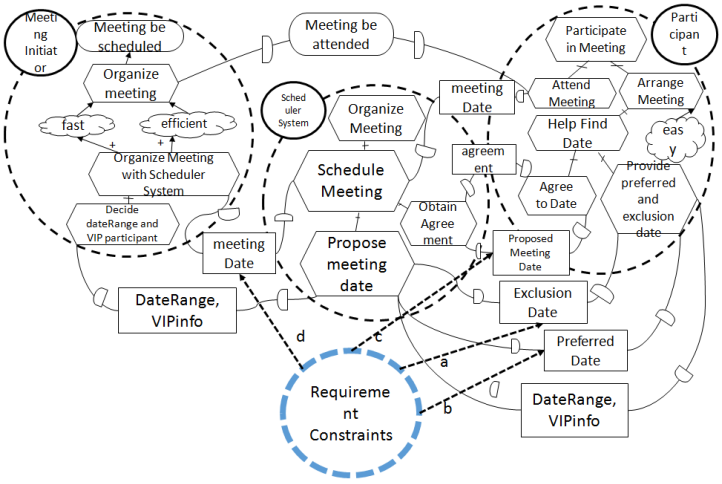

Fig. 1. Integrated Diagram with Machine Actor and Requirement Constraints

3.5 Defining Constraints

Problem Frame is used to describe the requirement constraints which exist in the real world to help software developers to realize and understands the outside circumstances. We already analyzed the relations between actors and their tasks and goals. The behavior of the system is also already drawn in the earlier steps. However, there are constraints in the environment that has not been elicited yet in the above steps. By analyzing the system from a problem frame view (figure 2), we can find that there are

constraints that is not elicited yet, so we need to find and add them into the integrated diagram.

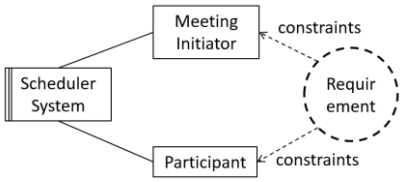

Fig. 2. Problem Frame View

The requirement constraints are written in a, b, c, and d as symbols and the descriptions are written separately to save space in figure 1. By analyzing the requirement constraints that exist in the system, we can enhance the behavior analysis by inserting constraints into the chart, as shown in figure 3.

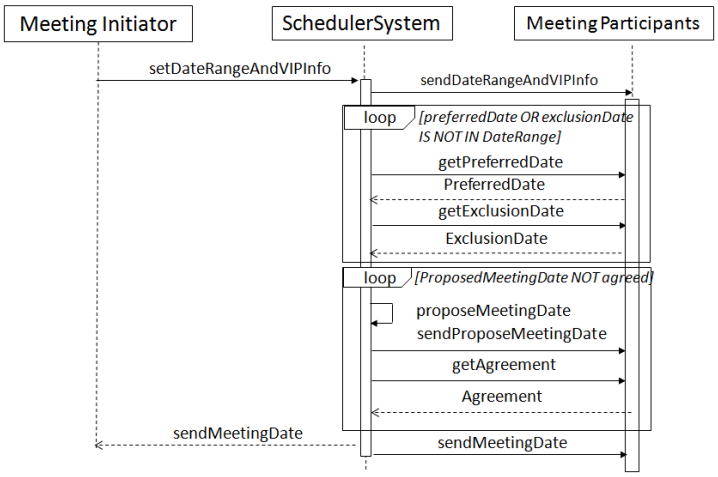

Fig. 3. Message Sequence Chart

3.6 Obstacle Analysis

Then after the whole diagram is done, using KAOS method [2], we draw an obstacle analysis diagram to analyze the obstacles that exist in the development process, and try to mitigate them. Obstacle diagrams are drawn in a hierarchical system along with its possible solutions. In the meeting problem, participants' task "attend meeting" has risks "date forgotten". The possible solutions for that risk are "send reminder email" and "remind participant manually". With the software, task "send reminder email" can be added into the system to prevent "date forgotten". Then after the obstacles are analyzed and preventions are considered, we re-arrange the main diagram with consideration from the obstacle analysis done in the last step.

4 CONCLUSION AND FUTURE WORK

This research proposes a new integrated framework in software requirement analysis, taking advantages from existing methods like *i**, Message Sequence Chart, KAOS, and Problem Frame, which are currently used separately even though they all used for the same purpose which is to elicit requirements. By combining the advantages of those methods, this research is expected to be able to cover each methods' disadvantages, and by doing so, requirements can be described suitably and clearly. This paper explains the early result of our research, which is the integrated framework. We also show the applicability of our framework by implementing it on the meeting scheduler problem.

In order to integrate several methods into one framework, there are some setbacks in our framework. The framework consists of several methods in its implementation which can cause the diagrams to be complicated and difficult to understand. Moreover, there are some intersection points between each approach in the framework, for example, actors in the i* model and MSC, or tasks in the i* model and KAOS' obstacle analysis, which draw the possibility of inconsistency and redundancy if they are manually drawn. Thus, for future work, our research aims to develop a support tool to assist user in implementing the integrated framework that can auto-generate diagrams from known variables in order to mitigate redundant drawings, consistency checks, and to provide simple views of each diagram if the users feel the need to check diagrams individually, not as a whole which can be complicated. By using this support tool, it would be easier for users to analyze requirement while adapting to the fast-changing environments which software are built in.

5 REFERENCES

1. Broy, M. : The Essence of Message Sequence Chart. In : Proceedings of the International Symposium on Multimedia Software Engineering, pp. 42-47. IEEE (2000).
2. Cailliau, A., et al. : Modeling Car Crash Management with KAOS. In : Proceedings of the 3rd International Comparing Requirements Modeling Approaches (CMA@RE), pp. 19-24. IEEE (2013).
3. Jackson, M. : Problem Frames : Analysing and Structuring Software Development Problems. Pearson Education (2001).
4. Lamsweerde, A.v., Requirement Engineering : From System Goals to UML Models to Software Specifications, John Wiley & Sons (2009).
5. Mohammadi, N.G., et al. : A Framework for Combining Problem Frame and Goal Models to Support Context Analysis during Requirement Engineering. In : Availability, Reliability, and Security in Information Systems and HCI, vol. 8127 of Lecture Notes in Computer Science, pp. 272-288. Springer (2013).
6. Nuseibeh, B. and Easterbrook, S. : Requirements engineering: a roadmap. In : Proceedings of Conference on the Future of Software Engineering, pp. 35–46. ACM (2000).
7. Yu, E. : Towards Modelling and Reasoning Support for Early-Phase Requirement Engineering. In : Proceedings of the Third IEEE International Symposium on Requirement Engineering, pp. 226-235. (1997).