Towards an Integrated Framework for Software Requirements Analysis and Its Support Tool

ANDRE RUSLI[†1] OSAMU SHIGO[†1]

Abstract: Requirements analysis 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, stakeholders, 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 requirements analysis which combines the characteristics of goal-based requirements engineering methods, Problem Frames (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. Furthermore, in order to assist engineers in using the framework, our research also emphasizes the importance of a support tool.

Keywords: Requirements Analysis, Goal Models, i* framework, Problem Frames, KAOS, Message Sequence Chart

1. Introduction

Requirements engineering is gaining more attention in recent years as it has become clear that extracting the correct requirements is vital to the success of a software project. Poor requirements have a negative effect on the estimation process; this then leads to schedule and cost underestimates, inadequate staffing and then staffing itself becomes a major risk factor [9]. 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 [8]. Those activities include requirements elicitation, analysis, specification, human machine interface design, and validation [1]. In this paper, we aim to propose an integrated framework that combines the advantages of several existing methods to assist engineers in analyzing requirements.

In the requirements engineering community, goal-based modeling approaches have gained considerable attention. In this sense, a goal is defined as a prescriptive statement of intent whose satisfaction requires the cooperation of agents forming the system [6]. In KAOS, one of Goal-Based Requirements Engineering (GBRE) methods, requirements are described in a goal hierarchy model. High-level abstract goals are identified and decomposed into low-level goals. A goal model is an AND/OR graph showing how goals contribute to each other. An AND-refinement requires all sub-goals to be satisfied for the parent goal to be satisfied. An OR-refinement captures alternative system options; the parent goal is satisfied provided one of the alternative sub-goals is satisfied [3]. Another popular method in GBRE is i* framework. I* framework is an improved goal model that inherited some of KAOS characteristics. This framework consists of two main modeling, Strategic Dependency Model and Strategic Rationale Model. I* focuses on describing the relationships between actors, tasks, goals, and resources in the environment where the software-to-be will be developed.

Other methods that we considered in our research are message sequence chart (MSC) and problem frames. In describing requirements, it is important to be able to visualize the behavior of the intended program. However, MSC in our research is not used to describe detailed behaviors of the software, such as method call, as it will be done in the next step of software development, not in requirements analysis process. MSC in this paper will only describe the flow of resources between actors in the system. On the other hand, problem frames, which main idea is to focus more on the real world than the computer world when analyzing requirements, helps our research in describing the real-world constraints that are limiting the system. Correctly describing constraints is vital as it will affect the software's behaviors and helps software designers to know what they should/should not build.

However, there are also some trade-offs. Using an integrated framework of several methods to analyze requirements can be complicated, especially if it is hand-written. There might some tasks or actors that are drawn in more than one diagram in the framework and also connections between diagrams that are hard to describe without the aid of a software tool. Our research also includes a development of support tool to assist and help software engineers in using the framework.

Section 2 will simply explain about the methods that are considered in this paper. Section 3.1 will say about the case study that is used in this paper to explain the framework and its support tool and the rest of section 3 will explain how the framework and support tool work.

2. Background

2.1 Goal Based Requirements Engineering Methods

A goal is a prescriptive statement of intent that the system should satisfy through the cooperation of its agents [6]. 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 pop-

†1 Graduate School of Information Environment, Tokyo Denki University

ular 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 [3]. 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 [6]. While i* frame-work is an integration of goal based requirements engineering and agent based requirements engineering method. I*'s strategic dependency model is used to describe the dependency relationships among various actors in an organizational context [11].

2.2 Problem Frames

Problem Frames [5] (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 out-side. A problem frame consists of domains, interfaces between them, and a requirement [7]. 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 [7].

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 [2].

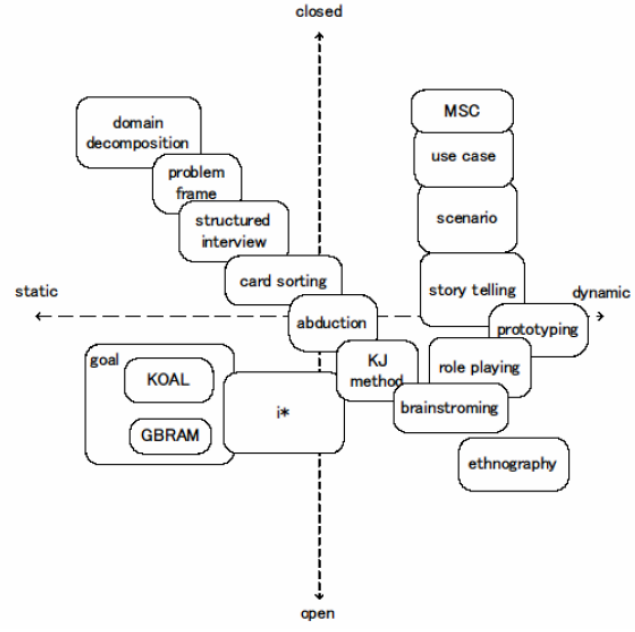


Fig. 1. RE Technology Map [11]

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, in our framework, 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 requirements analysis process.

These four methods have different characteristics, as shown in figure 1 which shows the different techniques and dimensional space of several requirements engineering methods in this world. A static and closed method means that requirements elicited from the domain by examining its static structure and the object space is relatively stable, known, and closed. While a dynamic and open method means that requirements are elicited from the domain focusing on their dynamic context and the object space is relatively unstable, unknown, changing, and open.

Problem frame is in the static and closed dimension, while KAOS and i* are in the static and open dimension and Message Sequence Chart which describe the dynamic sequence of a system is in the dynamic and closed dimension. Our framework aims to get the best of these methods by proposing a new framework which combines their advantages.

3. Research Question

The research that we are conducting are aimed to answer the following research questions :

1. **RQ1**. *What are the advantages of using the integrated framework to conduct a requirements analysis, compared to using each method separately?*

   In order to know the advantages of using the integrated framework in conducting requirements analysis, we implemented the framework on a case study, which is the Barbados Car Crash Management System (bCMS). The objective is to know whether the framework is usable and to find out what are the things that result better by using the framework in analyzing requirements.

2. **RQ2**. *Is a support tool needed to be able to use the proposed integrated framework effectively? What are the advantages of using the support tool?*

   We started to develop a support tool to assist requirements engineers as the users to use of the proposed framework. To understand whether the tool is truly needed and what the advantages are, we used the support tool to elicit the case study of bCMS and evaluate the tool in its early stage.

4. Integrated Framework

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 [10]. The simple flow of the proposed integrated model is shown in figure 2. We use i*'s strategic dependency diagram as the first diagram in the framework to describe the connection between actors that exist in the environment, including the machine software that will be developed. It is considered that recognizing actors in the early phase of requirements analysis is a huge boost for understanding the whole system, therefore, i*'s dependency diagram are de-

scribed first in our framework.

Furthermore, the actors in the diagram can be expanded, and then goals and tasks inside each actor will be described so at this point, the main diagram shows the dependency connection between actors and also inside each actor. Next, utilizing problem frames concept, constraints in the real world are added into the diagram with arrows pointing to tasks or goals that are constrained. So before the dependency diagram have goals or tasks there can be no constraint in the system.

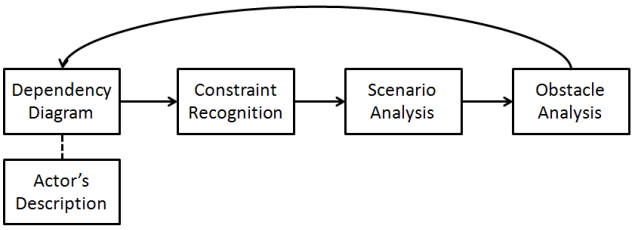


Fig. 2. Integrated Framework Model

System behaviors, resource flow and orders, will be described in the next step using Message Sequence Chart, importing actors and resources from the dependency diagram. Lastly, obstacle analysis is held based on the model from KAOS framework. Possible solutions for each obstacle are described, and solutions that should be implemented in the will-be-developed system will be added into the main diagram and constraints and scenario analysis are reconsidered in respect to the newly added task or goal.

Figure 3 shows the simplified meta-model of the proposed integrated framework. We are aware that the following meta-model is not a complete model that represents details and relationships in the system, which will be our aim in the future as it holds further challenges. The following model simply shows the components exist in the framework. Dependency diagram contains five components which are constraint, resource, actor, goal, and task. The same resource and actor are also belonged to MSC to describe the sequences, which also have Control that is divided into two type (IF and LOOP). MSC's Control is designed depending on the condition get from Constraint in Dependency Diagram. Moreover, the Actor's Hierarchy diagram have three components inside it which are actor, goal, and task, taken from the Dependency Diagram. Task from the Dependency Diagram could have Risk and Solution, which will be described in the obstacle analysis.

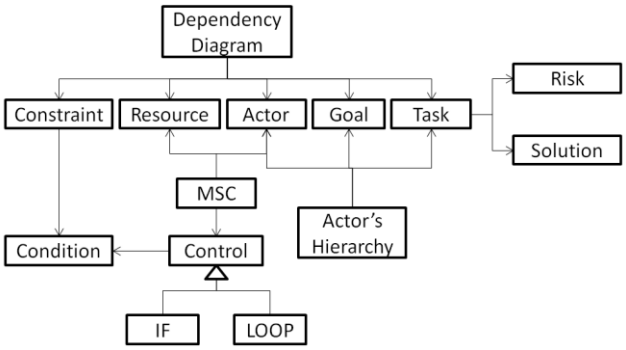


Fig. 3. Simplified Meta-Model of Integrated Framework

4.1 Case Study

This paper uses the Barbados Car Crash Management System (bCMS) [3] as an application example to implement the use of our framework. The proposed system is intended to coordinate the communication between a fire station coordinator (FSC) and a police station coordinator (PSC) to handle crises in a timely manner. However, not all of the modeling steps are discussed in our paper because of the lack of space, the complete discussion is available in [4] Implementation of our framework on other case study is written in [10].

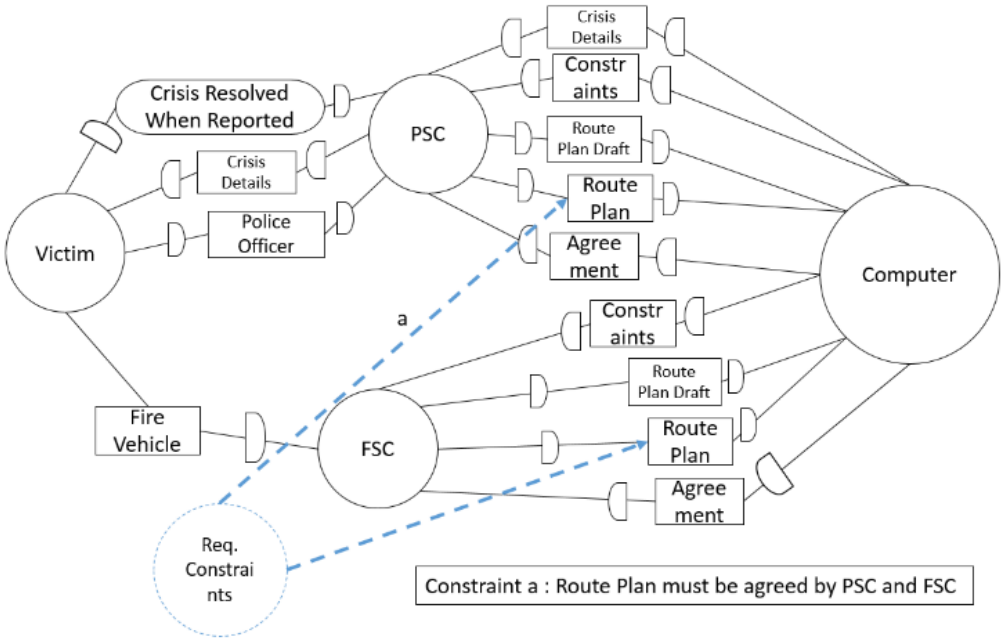


Fig. 4. Dependency Diagram

4.2 Implementation to bCMS

First of all, we describe the dependency diagram of the system with all of its actors as shown in figure 4 below. We recognized 4 actors exist in the system, Victim, PSC, FSC, and the computer system which will be developed in the project. The diagram in figure 4 is focused on describing the dependency relationships between those four actors. Moreover, the direction of half-circles on lines that connect actors shows the connection of who depends on who, this concept is taken from i* framework's concept, as also the diagram.

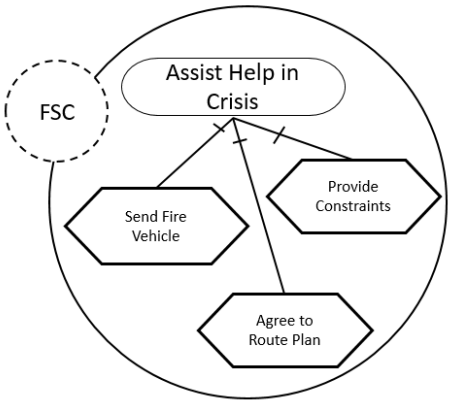


Fig. 5. Actor's Hierarchy

In figure 4, we can see that, for example, PSC depends on Victim on providing the Crisis Detail and Computer depends on FSC for the "agreement" resource which computer cannot provide by itself without FSC. These kinds of dependencies are vital as requirements to know how the whole system works and who are involved in the system. Requirement constraints is drawn as a dotted blue circle, and it constraints the re-source "Route Plan" in such that a Route Plan must be agreed by both PSC and FSC to be valid. These constraints will later affect the making of message sequence chart for describing the scenario in the system.

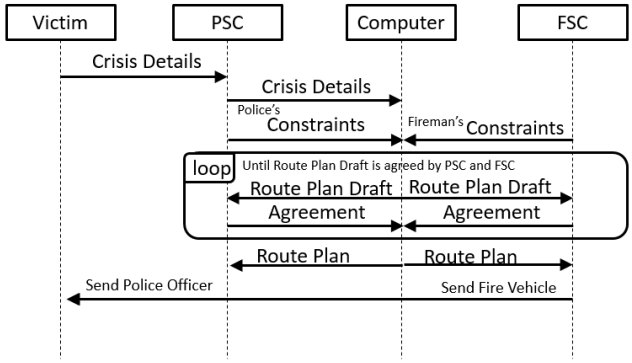


Fig. 6. Message Sequence Chart

The next step is to analyze and describe the goal and task hierarchy inside each actor in the previous diagram. Figure 5 shows one of the examples of actor FSC's inside hierarchy. FSC's main goal is to assist help in crisis, and to achieve that, there are several tasks that need to be done, provide constraints, agree to route plan suggested by the computer system, and finally send fire vehicle to the victim's location.

As it has already stated in the early pages of this paper that dependency diagram cannot explain the timely order of how resources flow in the system, thus we make use of message sequence chart to explain the scenarios of the system. Figure 6 shows the chart, after also taking constraints described early into consideration. First, victim sends crisis details to PSC then PSC forwards it to the computer. After that, both PSC and FSC are asked to provide their constraints to computer system, then it mocks up a route plan draft and send it to both PSC and FSC. As explained in the requirement constraints in the dependency diagram, both parties (PSC and FSC) need to agree on the draft suggested by the computer in order for it to become a working route plan and the crisis resolving can start, or else the computer have to reconsider the draft and propose another one until it is approved. After route plan is agreed, PSC and FSC will then be able to operate and send the necessary helps for victim according to the agreed route plan.

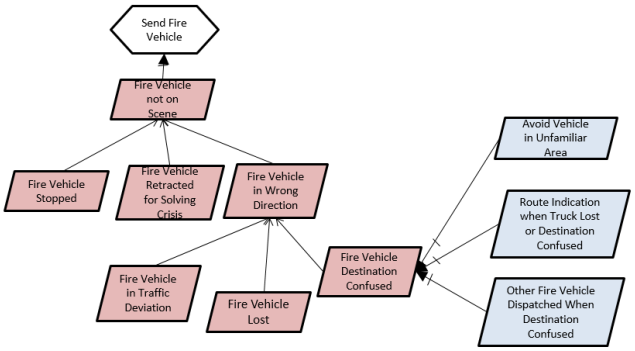


Fig. 7. Obstacle Analysis Diagram

The next step of our framework is to analyze obstacle that might exists in the sys-tem and possible solutions for each of those obstacles/risks. Figure 7 will take an example of a task that is considered risky from FSC's circle, the task of "Send Fire Vehicle". Task "Send Fire Vehicle" has the risk of fire vehicle is not on scene when it is needed. The cause of this risk can be several things, so we decomposed it into three possible causes. Among those three, we tried to focus on the cause of when fire vehicle get in wrong direction and decomposed it again into several risks. Finally, we analyze the possible solution for each risk. As shown in figure 7, an example of possible solutions for the risk "Fire Vehicle Destination Confused".

Among those possible solution, engineers need to decide which solution can be done by the software and which cannot be. Solutions that can be done by the will-be-developed software will then be inserted to upgrade the earlier dependency diagram. Moreover, because dependency diagram is changed the message sequence chart also needs to be reconsidered for changes.

5. Support Tool

Combining several methods into one integrated framework has its own trade-offs. The connections between diagrams caused some objects to be drawn more than one time, the actors

in dependency diagram and MSC, for example. Integrating methods has the possibility of redundancy of work for the engineer. Furthermore, using the proposed framework, there will be several diagrams to be drawn, as a result the difficulty and complexity to hand-write all of them accurately are also increased. Requirements analysis is a process done by both engineers and stakeholders exist in the system, thus the complexity of the diagrams can make people who are not used to requirements engineering to have difficulty in understanding what the diagrams mean.

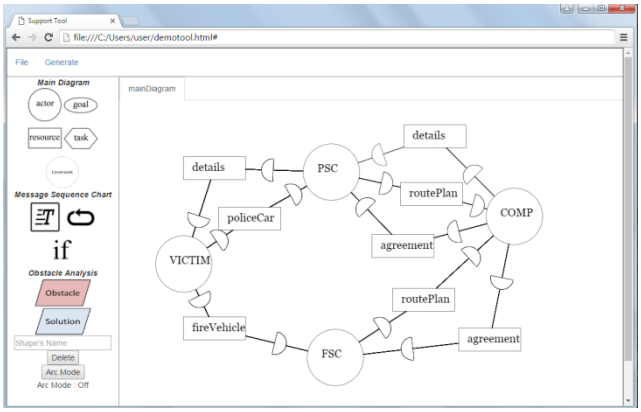


Fig. 8. Dependency Diagram Editor

In order to overcome those disadvantages, our research also develops the support tool to make it easier for engineers to use our framework and to help user to get better understanding of the diagrams. Our support tool offers help to draw diagrams with computer on a web application, so engineers do not have to hand-write all of the diagrams which means that it can reduce work.

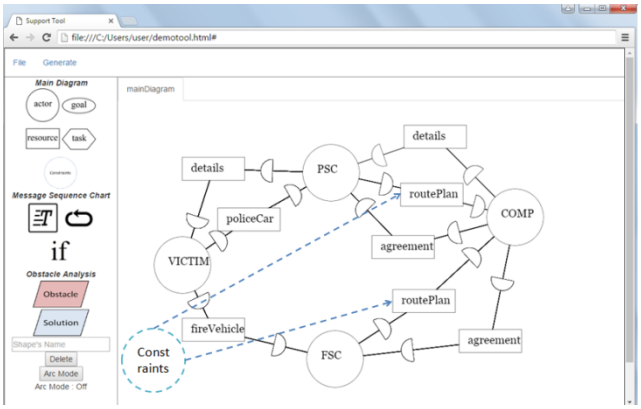
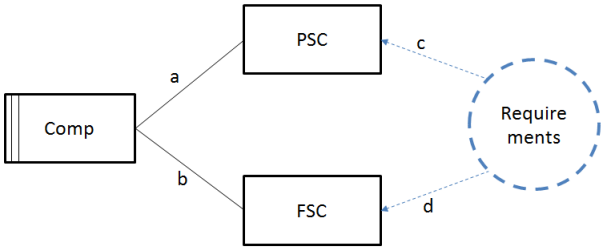


Fig. 9. Dependency Diagram with Req. Constraints

Figure 8 shows the main interface of our support tool to draw the dependency diagram. User can select the tools on the left side of the program to draw shapes, and shapes will be drawn in the right hand side of the screen. Tools on the left panel are divided into three categories. The first category is the main diagram editor tools, in this part, there are five buttons, actor, goal, resource, task, and constraint button. Constraint button is the button to draw requirement constraint after the dependency diagram is created. After shapes are drawn, user can adjust the position of the shapes and connect lines between them by clicking and dragging left-mouse click. Lines drawn from actors are straight lines with half-circle indicating the dependency direction, while lines drawn from requirement constraints are blue-dotted lines with arrows pointing to the tasks that are constrained, as shown in figure 9.

The second category consists of the tools used for drawings in the message sequence chart. MSC can be generated from the dependency diagram, but even after generated and re-arranged, there are occasions where user needs to add new messages or resources. Furthermore, after analyzing requirement constraints, there are possibilities that loops or if statements are needed. Using the loop symbol and if button, user can add them into the MSC. The tools in the third category, obstacle analysis, are used to draw risks and their possible solutions. User then can click and drag the risks or solutions to draw arrows connecting them.



a: C! {routePlan}
PSC! {details, agreement}
b: C! {routePlan}
FSC! {agreement}
c: PSC! {agree, disagree}
d: FSC! {agree, disagree}

Fig. 10. Problem Frames View

As already said earlier in section 3, requirement constraints in this framework is taken from problem frames' idea that focuses on the real world. Besides that, the resources in the dependency diagram can also be compared to the domain concept in problem frames, and the computer actor, which is the software that is being developed, can be treated as machine domain in problem frames' concept. Our tool also enable user to see the relationship between the actors, seen as domain in the PF's concept, that are involved in a constrained task or resource. Figure 10 shows an example of a problem frames' view of the constrained resource, "routePlan", between the PSC, FSC, and COMP actors, drawn in a simple problem diagram.

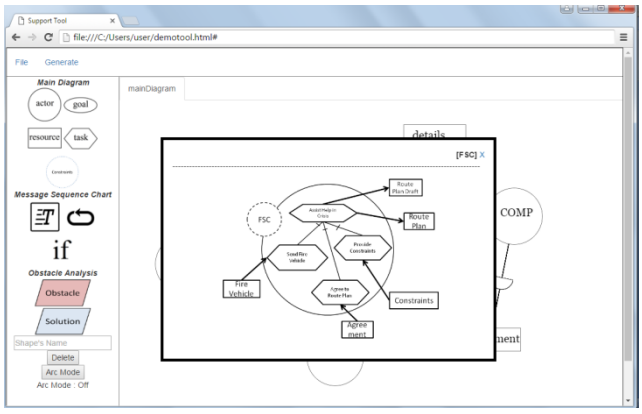


Fig. 11**.** Actor's Hierarchy Editor

Moreover, actors on the diagram can be enlarged by double clicking them and user will be able to edit or view the goal and task hierarchy inside each actor, as shown in figure 11. The resources drawn outside the actor's circle are representing the resources that it depends on other actors or that are depended on them by other actors. Tasks inside and outside each actor can be marked as risky by right-clicking them for further analysis in the later process of the framework.

The proposed support tool will also enable diagram auto-generation from dependency diagram into MSC. This is possible because actors and resources presented in the dependency diagram can be reused in MSC to explain their behavior. First, the tasks and actors from dependency diagram are imported just as they are ordered in the dependency diagram, as shown in figure 12.

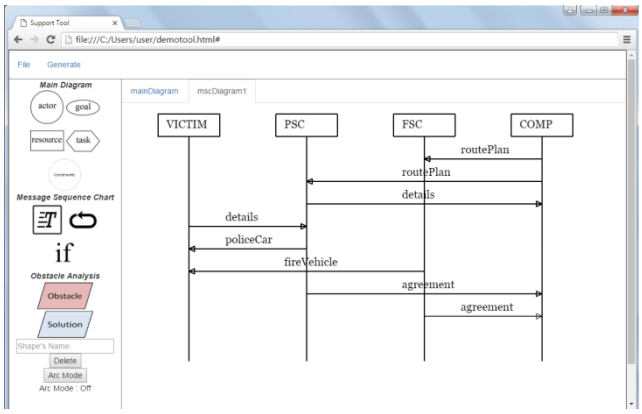


Fig. 12. Auto-Generated MSC

After they are all generated, user can re-arrange the timely order of the tasks by clicking and dragging the resources drawn on the diagram, so the ideal scenario of the system can be understood, which is critical in analyzing requirements. The re-arranged MSC is shown in figure 13. Furthermore, using the tools in the left panel, user can also add loops or if statements according to the constraints exist in the environment as stated by the earlier diagram.

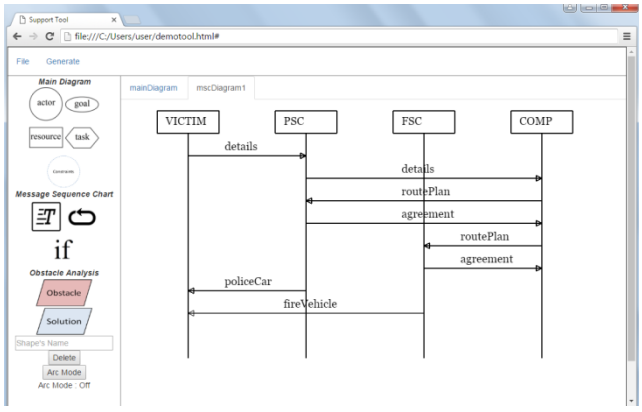


Fig. 13. Re-arranged MSC

Furthermore, by clicking "obstacle diagram" from the "generate" menu bar located on the top of the display, tasks that were marked as risky task in the dependency diagram can be further analyzed by importing them to the obstacle analysis. In this part, only task's name is imported with the polygonal shapes around it. Figure 14 shows an example of an obstacle diagram, the risky task is "send fire vehicle", which was already shown also in figure 11, as one of the tasks of FSC. From here, user needs to analyze the risks that might happen in the future and add them using the obstacle button in red color from the left panel into the canvas. Risks are then decomposed into smaller and more clearly defined risks, and then possible solutions are added too. Solutions can be added using the solution button colored in blue from the left panel.

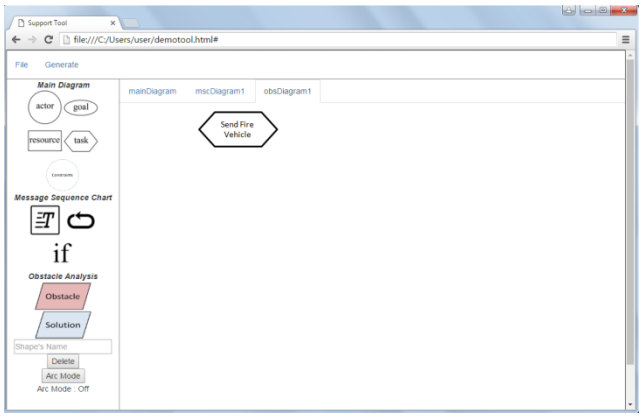


Fig. 14. Risky Actor Imported from Dependency Diagram

Figure 15 shows the risk analysis for the particular task. It is considered that there is a risk of fire vehicle is not on scene when it is needed. Then this risk is decomposed into several smaller risks that can be the cause of it. Here, it is decomposed into three main causes, which are, fire vehicle stopped, fire vehicle retracted for solving crisis, and fire vehicle in wrong direction. Particularly for the problem of fire vehicle in wrong direction, we can further analyze the reasons of why it is in the wrong direction. One of the reasons is that the fire vehicle is confused of the destination. Then, possible solutions are considered and drawn on the diagram.

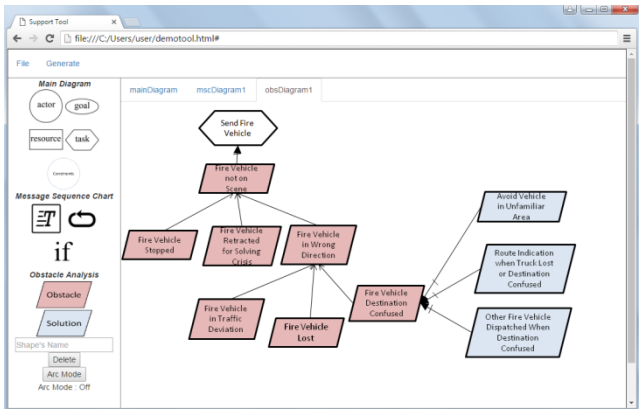


Fig. 15. Obstacle Diagram Editor

To avoid fire vehicles from being confused of their destination, one of the ways is to avoid them from being in an unfamiliar area. Other way considered in this study is to give route indication when trucks lost their way or confused and to dis-

patch other fire vehicle to the scene. After these risks are analyzed, to mitigate them, solutions that are considered can be done by the will-be-developed software can be added into the dependency diagram as new tasks or resources. By adding new tasks or resources into the diagram, actor's hierarchy and MSC also might need changes, so it is important to recheck them too for consistency.

By using the support tool, it makes it easier for the user to do consistency check, because some of the variables are auto-generated so it will be consistent, and other variables can be easily check because it is just one or two click away to change view from one diagram and the other.

Diagrams can be saved, and user can also load existing diagram into the software. Auto-generation can be done by clicking the "generate" panel on top of the screen, and there will be options whether user wants to generate MSC or obstacle diagram.

6. Conclusion and Future Work

This research proposes a new integrated framework in software requirements analysis, taking advantages from existing methods which are i* framework, Message Sequence Chart, KAOS, and Problem Frames, 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 case study of Barbados Car Crash Management System.

In order to answer the research question RQ1, by using the integrated framework, the dependency relations between actors in the system are made clear, and the constraints in the real world that are limiting resource(s) are also described clearly by implementing the problem frame concept which is absent in the goal based requirements engineering methods. Furthermore, the hierarchies inside each actor were described, goals and tasks. In addition, the orders of messages in the system, which we could not know by describing the dependency diagram only, can be clearly described using the message sequence chart along with the constraints constraining resource(s) that can be expressed using if and/or loop statement. Lastly, obstacle analysis is also conducted in this framework to mitigate future risks and help analyst to provide possible solutions that might or might not be implemented in the system.

However, integrating several methods into one framework holds some setbacks. 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, tasks in the dependency diagram and obstacle diagram, which draw the possibility of inconsistency and redundancy if they are manually drawn. To mitigate those setbacks in using our framework, our research also aims to develop a support tool to assist user in implementing the integrated framework.

This support tool can auto-generate diagrams from known variables in order to mitigate redundant drawings, consistency checks, and to provide easier control to display each diagram in its complexity, which answer research question RQ2. Moreover, by using this support tool, it would also be easier for users to analyze requirements while adapting to the fast-changing environments which software are built in.

The future of our work will include the completion of the support tool, mainly because the current tool is still in its prototype version. After the support tool is completed, it needs to be tested and evaluated by having users to actually use the tool to analyze requirements and get their critics and comments to better improve the build of support tool software. Moreover, it is also interesting to have real industrial people to implement our framework into their work and get inputs from them, so that our framework can be evaluated in the actual world, not only from case study.

Reference

[1] Bray, I. K. : An Introduction To Requirement Engineering. Addison Wesley (2002).
[2] Broy, M. : The Essence of Message Sequence Chart. In : Proceedings of the International Symposium on Multimedia Software Engineering, pp. 42-47. IEEE (2000).
[3] 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).
[4] Cailliau, A., et al. : Modeling Car Crash Management with KAOS, UCL (2013), kaos.info.ucl.ac.be/bcms.html.
[5] Jackson, M. : Problem Frames : Analysing and Structuring Software Development Problems. Pearson Education (2001).
[6] Lamsweerde, A.v., Requirement Engineering : From System Goals to UML Models to Soft-ware Specifications, John Wiley & Sons (2009).
[7] 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).
[8] Nuseibeh, B. and Easterbrook, S. : Requirements engineering: a roadmap. In : Proceedings of Conference on the Future of Software Engineering, pp. 35–46. ACM (2000).
[9] Ogheneovo, E. E. : Software Dysfunction: Why Do Software Fail? In : Journal of Computer and Communications, 2, 25-35 (2014).
[10] Rusli, A., Shigo, O. : Integrated Framework for Software Requirement Analysis. In : REFSQ Workshop (2016). http://ceur-ws.org/Vol-1564/paper11.pdf.
[11] Tsumaki, T. and Tamai. T. : A Framework for Matching Requirement Engineering Techniques to Project Characteristics and Situation Changes. In : Proceedings of SREP'05, Paris, France. (2005).
[12] Yu, E. : Towards Modelling and Reasoning Support for Early-Phase Requirement Engineer-ing. In : Proceedings of the Third IEEE International Symposium on Requirement Engineer-ing, pp. 226-235. (1997).