Requirements Specification and Modeling through SysML

Michel dos Santos Soares, Jos Vrancken

Abstract—Use Case diagrams are well-known for their use to specify and describe system requirements. From initial system requirements documents, use cases can be derived representing several scenarios. These scenarios can later be detailed in different ways, as for example, through informal descriptions. In this paper, system requirements are first specified using the SysML requirements diagram and later by use cases. The main goal is to fill the gap between documents written in natural language and use cases by modeling requirements in a graphical and tabular way, which can improve the requirements representation. Also, the relationship between requirements is enhanced. An example of a real time distributed system is given to illustrate the approach.

I. INTRODUCTION

REQUIREMENTS Engineering is the process by which the requirements for systems and software products are gathered, analyzed, documented, and managed throughout the development life cycle [1]. It is a fundamental phase of every system development, and a critical success factor. Basically, requirements are descriptions of how a product should perform when completed. There are several possible classifications, such as user requirements, which are high-level abstract requirements, and system requirements, meaning the detailed description of what the system should do [2]. This classification allows the representation of different views for specific stakeholders.

Requirements should be written at different levels of detail because several stakeholders use them for different purposes. User requirements are normally written in natural language or in structured language, with the help of informal diagrams. A recent on-line survey indicates that 95% of user requirements are written in natural or structured natural language [3]. The fundamental purpose of user requirements specification is to document the needs and constraints gathered in order to later develop a system based on those requirements. More specific systems requirements are also commonly written using natural language. The principal problem of this approach is the ambiguity of natural languages. To deal with this ambiguity, one of the most recommended solutions is to use formal or semi-formal specification languages [1]. These approaches can at least diminish the ambiguity.

There are several modeling techniques and languages for specifying and describing requirements. Mathematical notations, such as Z [4], have been applied in order to solve the ambiguity problem, but are of difficult application, due to the lack of professionals, high costs and limited scalability [5]. One good example of a semi-formal notation for requirements specification is the UML use case diagram. In [6], it is presented how use cases are applied with a scenario-based requirements approach in the development of safety critical systems. But two problems still arise: the natural language is still needed, and the diagrams are sometimes not user-friendly.

This paper is about applying the SysML requirements diagram to the specification of system requirements. SysML is a new systems modeling language that supports specification, analysis, design, verification and validation of a broad range of complex systems [7]. The language is an evolution of UML 2.0 [8] to be applied to systems that may include hardware, software, information, processes, personnel and facilities. SysML is supported by the OMG Systems Engineering Domain Special Interest Group and by INCOSE (International Council on Systems Engineering). It is expected that the language will become a de facto standard for Systems Engineering, just like UML is for Software Engineering.

SysML is based on UML and reuses some of its constructs. The basic difference is that SysML was built from scratch to support System Engineering, which means that some specific software oriented constructs, not necessary to systems modeling, were avoided. This may facilitate the communication between heterogeneous teams (for instance, mechanical, electrical and software engineers) that work together to design a system. According to [7], the language is effective in specifying requirements, structure, behavior, allocations of elements to models, and constraints on system properties to support engineering analysis.

In this paper, requirements are presented in a graphical and tabular form, and are modeled instead of just written in natural language. The SysML diagram specifies a defined semantics to be followed when relating requirements to each other. It is presented that the SysML requirements diagram can fill the gap between natural language based specifications, too ambiguous and informal, and UML use case diagrams. Another advantage is that there’s a defined semantics to associate SysML requirements diagram to other models created during system design. In addition, the SysML requirements diagram can facilitate the transformation of user requirements into system requirements and improve requirements traceability throughout the design life cycle.

The paper structure is given in the following manner. In
section II, some definitions and the problems of specifying user and systems requirements are discussed. In section III, the SysML requirements diagram is briefly explained, with several different ways to relate requirements in a model. In section IV, SysML requirements and use case diagrams are jointly applied to an example. The paper ends with a conclusion.

II. PROBLEMS WITH REQUIREMENTS SPECIFICATION

Basically, requirements are the services that a system must provide and the constraints that it must obey to solve some problems. The process of requirements engineering involves several critical activities, such as elicitation, specification, prioritization and management of requirements. The involvement of stakeholders in the requirements engineering process is widely considered as crucial to the quality of the final product. As a matter of fact, some studies based on a large number of projects states that the major causes of system failures are related to problems in requirements engineering [9]. Among these problems are incomplete requirements specifications, lack of user involvement and uncontrolled change in requirements.

One of the possible classifications of requirements is to divide it into two categories: user and system requirements. User requirements are normally written in natural language, with the help of informal diagrams and tables. The specification is based on elicitation of the requirements involving the stakeholders by applying several techniques, such as interviews and questionnaires. At least two difficulties arise in the description of user requirements through natural language. The first is that natural language is ambiguous, leading to misunderstandings between stakeholders and developers. One approach to deal with this problem is to formalize the documents and specify the requirements in more detail. This normally leads to the second problem: too much detail in the early stages of system development may constrain the freedom of the system developer, inhibiting more creative and effective solutions.

There's no simple solution to the lack of clarity when specifying user requirements in a natural language. One option is to write user requirements in a very high level manner, leaving the details to be specified in the system requirements document. The challenge is that from user requirements with few details, it's difficult to start specifying use cases. Another possibility is to write user requirements in a structured language, to facilitate system requirements specification.

Several different approaches are applied to describe system specifications. Recently, with UML as a standard for software engineering, the use case diagram has been widely applied to specify system requirements. Use cases are visual, which is good to system analysts and the users to better comprehend the system. Also, a use case can be detailed, and its several scenarios described in natural language, pseudo-code or other UML diagrams.

Although use case diagrams are applied to involve non-technical stakeholders in the requirements engineering process, this is not always easy, and there are some disadvantages in their use. In large, complex systems, the number of components and scenarios can be very large and there is a learning curve for users to understand use cases. Even thought a use case is a simple diagram, it has specific syntax and semantics. For instance, the difference between include and extend arrows and when to use each construct can be confusing, and the concept of inheritance, which is not simple for non-technical persons to understand. Another common problem is that some analysts represent too many details in the diagram, which makes the model difficult to read. Also, there seems to be a gap between the natural language texts and the graphical language of use cases, used at the system requirements level.

The SysML requirements diagram helps in better organizing requirements, and also shows explicitly the various kinds of relationships between different requirements. Another advantage of using this diagram is to standardize the requirements specification through a defined semantics.

One important quality factor in systems design is to know what happens to a requirement during system modeling and specification. This activity is known as requirements traceability. A definition of requirements traceability is given in [10] as: “the ability to describe and follow the life of a requirement, in both a forward and backward direction, i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases”. Basically, requirements traceability helps in identifying the sources, destinations and links between requirements and models created during system development.

Identifying and maintaining traces between requirements is considered one of the most important activities during requirements engineering [11]. The requirements tracing is very useful, for example, to identify how requirements are affected by changes, what is the purpose of a requirement and to prioritize requirements. Traceability also provides a possibility to ensure that all requirements are fulfilled by the system and sub-system components As a matter of fact, important decisions on requirements and the correspondent models are better justified [12]. One way to manage the requirements traceability in SysML is within requirements tables.

III. REQUIREMENTS SPECIFICATION WITH SYSML

SysML provides a specific diagram for requirements specification, which is an advantage over UML. As a direct consequence, SysML allows the representation of requirements as model elements, which mean that requirements are part of the system architecture. The SysML requirements constructs are intended to provide a bridge between traditional requirements management tools and the other SysML models. When combined with UML for software design, the requirements constructs can also fill the
gap between the user requirements specification, normally written in natural language, and the use case diagrams, used as an initial specification of system requirements.

SysML requirements diagram can be used to standardize the requirements documents. There's a specific pattern to be used, which limits the wide freedom that allows so many different documents styles for requirements specifications. The basic graphical node to construct requirements diagram is shown in Fig 1.

![Fig 1. Basic node for SysML requirements diagrams](image)

### A. Requirements Relationship

It is well-known by software engineering researchers and practitioners that requirements are related to each other and these interactions affect software development in various ways. The survey [13] looks at the evolution of supporting concepts and their related literature, presents an issues-based framework for reviewing processes and products, and applies the framework in a review of requirements interaction state-of-the-art.

The SysML requirements diagram allows several ways to represent requirements relationships. These include relationships for defining requirements hierarchy, deriving requirements, satisfying requirements, verifying requirements and refining requirements [7]. The relationships can improve the specification of systems, as they can be used to model requirements. The relationships: hierarchy, derive, satisfy, verify, refine and trace are briefly explained below.

In large, complex systems, it is common to have a hierarchy of requirements, and their organization into various levels helps in dealing with system complexity. For instance, high-level business requirements may be gradually decomposed into more detailed software requirements, forming a hierarchy. SysML allows splitting complex requirements into more simple ones, as a hierarchy of requirements related to each other. The advantage is that the complexity of systems is treated from the early beginning of development, by decomposing complex requirements into several smaller ones.

The concept of hierarchy also permits the reuse of requirements. In this case, a common requirement can be shared by other requirements. The hierarchy is built based on master and slave requirements. The slave is a requirement whose text property is a read-only copy of the text property of a master requirement. The master/slave relationship is indicated by the use of the copy keyword.

The derive relationship relates a derived requirement to its source requirement. During the requirements engineering process, new requirements are created from previous ones. Normally, the derived requirement is under a source requirement in the hierarchy. The relationship allows, for instance, a connection between high-level (user oriented) and low-level (system oriented) requirements. This facilitates to relate the dependency of user requirements mapped into systems requirements. In a requirements diagram, the derive relationship is represented by the keyword deriveReqt.

The satisfy requirement describes how a model satisfies one or more requirements. It represents a dependency relationship between a requirement and a model element, such as other SysML diagrams, that represents that requirement. The keyword satisfy represents this relationship. One example is to associate a requirement to a SysML Block diagram.

A test case refers to the method of verifying a requirement. The verify relationship defines how a test case can verify a requirement. This includes standard verification methods for inspection, analysis, demonstration or test. For example, given a requirement, the steps necessary for its verification can be summarized by a state-machine. The keyword verify represents this relationship.

The refine relationship describes how a model element (or set of elements) can be used to later refine a requirement. For example, how a use case diagram can represent a text requirement. The relationship is represented in the diagram by the keyword refine.

The trace relationship provides a general purpose relationship between a requirement and any other model element. Its semantics has no real constraints and is not well-defined as the other relationships. For instance, a generic trace dependency can be used to emphasize that a pair of requirements are related in two different ways [14].

### B. Requirements Table

SysML allows the representation of requirements, their properties and relationships in a tabular format. Several information items can be represented in the table. For example, the identification of the requirement, the name of the requirement and its description. Also, the type of requirements relationship can be shown using a tabular matrix. This allows an agile way to identify, prioritize and improve requirements traceability.

### IV. Example

A real time distributed system is presented as example. The system is a Road Traffic Management implemented around the city of Alkmaar, in the Netherlands [15]. First, two fragments of the systems requirements are given in natural language in two paragraphs. Later, they are modeled using SysML requirements diagram, and finally a use case diagram is presented for the example.

#### A. System Requirements

Paragraph 1: In this project there are two types of actuators: the TLS (Traffic Light System) and the DRIP (Dynamic...
Route Information Panels). The TLS reacts based on information sent by the links, which represent the links in the road network bounded by two discontinuities (ramps, junctions, etc.). They are used to reduce and increase the intensities of traffic flows. The DRIPs are used for rerouting and informing drivers on the current traffic state of routes downstream of the DRIP. In case of congestion, drivers will know the extent of the congestion that they can expect, which adds to the driver's comfort level. More importantly, route information enables drivers to make a different route choice. Drivers are thus diverted away from the congested road with positive effects on its congestion. In this case, the TLS green times are controlled in order to optimize the traffic in the road. Links and nodes (connections between links) play the most important role for traffic control. Their cooperation is responsible for controlling the TLS green times. The default settings for a TLS will be determined by signal schemes that are designed off-line based on relevant traffic statistics. As a matter of fact, the use of real-time data combined with historical statistics enables the system to be more predictive, which enhances the efficiency of the traffic.

Paragraph 2: To make the system responsive to the traffic state, links will adjust TLS green times. Links have three possible sources of information that they use to determine their traffic state: velocity-intensity measure points, the induction loops at TLS sites, and a traffic simulation model called MaDAM. The induction loops and measure points are sensors that collect real-time traffic information from the environment, such as the velocity. The information gathered by the sensors is published every minute (publish-subscribe architecture). MaDAM acquires the information from the TLS loops and velocity-intensity measure points and determines what the traffic state is on links that have no sensors of their own. In addition, MaDAM predicts what the traffic states will be for the links in the next 30 minutes in blocks of 5 minutes.

B. SysML Requirements Diagram
Each system requirement paragraph is specified by a single SysML requirements diagram respectively in Fig 2 and Fig 3.

C. SysML Requirements Table
The SysML requirements table is a good manner to improve the traceability between requirements. In this paper, two types of tables are shown: one for decomposition and one to represent interdependency between requirements.

The hierarchy tables allow the representation of decomposition of a complex requirement. Tables 1 and 2 are examples of a hierarchy table for Fig 2. The relationship tables allow representing the interdependency, identifying its type and what is the referenced requirement. Tables 3 and 4 are examples of interdependency representation for Fig 2.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Decomposition of Sysreq1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Name</td>
</tr>
<tr>
<td>DReq1</td>
<td>Driver</td>
</tr>
<tr>
<td>DReq2</td>
<td>Drip</td>
</tr>
<tr>
<td>Areq1</td>
<td>Actuators</td>
</tr>
<tr>
<td>TLSReq</td>
<td>TLS green times</td>
</tr>
<tr>
<td>TLSReq1</td>
<td>TLS reaction</td>
</tr>
<tr>
<td>TLSReq2</td>
<td>TLS default settings</td>
</tr>
<tr>
<td>TLSReq3</td>
<td>TLS controlling</td>
</tr>
</tbody>
</table>

Fig. 2 SysML Requirements Diagram for paragraph 1

Fig. 3 SysML requirements diagram for paragraph 2
### Table 2
**Decomposition of a Requirement (TLSReq)**

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSReq1</td>
<td>TLS reaction</td>
</tr>
<tr>
<td>TLSReq2</td>
<td>TLS default settings</td>
</tr>
<tr>
<td>TLSReq3</td>
<td>TLS controlling</td>
</tr>
</tbody>
</table>

### Table 3
**Requirements Tree (Driver)**

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Relation</th>
<th>Id_Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DReq1</td>
<td>Driver</td>
<td>deriveReqt</td>
<td>DReq2</td>
</tr>
</tbody>
</table>

### Table 4
**Requirements Tree (Actuators)**

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Relation</th>
<th>Id_Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AReq1</td>
<td>Actuators</td>
<td>deriveReqt</td>
<td>DReq2</td>
</tr>
<tr>
<td>AReq2</td>
<td>Actuators</td>
<td>deriveReqt</td>
<td>TLSReq</td>
</tr>
</tbody>
</table>

D. **Use Case Diagram**

The SysML use case diagram is derived without modifications from UML. The purpose is to describe the usage of a system by its actors in order to achieve a goal. The use case can also be viewed as functionality and/or capabilities that are accomplished through the interaction between the subject and its actors.

The SysML refine relationship can be used to relate requirements and use cases. For example, the use case “Control Green Times” can be associated by the refine relationship to the requirement “TLS green times” of Fig 2, which means that the use case can be represented by the requirement and specific sub-requirements. This is one way to specify a use case. Fig 4 shows this example.

The use case diagram of Fig 5 represents the actuators and sensors as actors of the system. They collaborate to realize the use cases. After modeling the requirements, knowing better the decomposition and relationship, the construction of the use case diagram is facilitated.

![Fig 4. Use case associated to a requirement](image)

Other requirements relationships are also useful and can be applied during all system modeling and design phases. One example is which use cases (or other models, for instance, a sequence or block diagram) are intended to satisfy a requirement. This can be represented by the satisfy relationship.

![Fig 5. Use case diagram for the system requirements of paragraphs 1 and 2.](image)

V. **Conclusion**

It is essential to have properly structured and controlled requirements specifications that are understandable and consistent. To achieve this important success factor, in this paper, the SysML requirements diagram is presented, jointly with the SysML requirements table, for system requirements specification and modeling. It is shown that modeling requirements through diagrams can be useful to explicitly represent the various ways that requirements can be related to each other. Using a specific diagram for requirements is an advantage over UML, and improves standardization. Also, requirements tables are useful to present decomposition in a tabular form and improve traceability, which is an important quality factor when building systems.

The SysML requirements diagram is applied to an example of a real time distributed system. Finally, from the requirements models it is easier to construct a use case that represents the system requirements, and relate the requirements to the use cases. With SysML requirements diagram, requirements can be modeled from the early beginning of the system life cycle, being part of the system architecture.

For future research, further relationships between SysML requirements diagram, use cases and other models are going to be investigated, and also the traceability aspects of the requirements table.
REFERENCES


