Method to share responsibility knowledge of dependability cases

Takuya Saruwatari\textsuperscript{a,b,\*}, Takashi Hoshino\textsuperscript{b}, Shuichiro Yamamoto\textsuperscript{c}

\textsuperscript{a}Graduate School of Information Science Nagoya University, Furo-cho, Chikusa-ku, Nagoya, Aichi, Japan
\textsuperscript{b}Software Innovation Center NTT, NSS-II Bldg. 6F, 2-13-34 Konan, Minato-ku, Tokyo, Japan
\textsuperscript{c}Strategy Office Information and Communications Headquarters Nagoya University, Furo-cho, Chikusa-ku, Nagoya, Aichi, Japan

Abstract

Recently, critical incidents have occurred in complex IT systems. Thus, how to confirm the dependability of a system via dependability cases is becoming necessary. Information related to dependability is important knowledge that must be shared among stakeholders. However, in previous methods used to describe dependability cases, the relation between a dependability claim and responsibility cannot be clearly specified. Thus, since the cause investigation cannot be completed at the time of the incident, system knowledge could not fully be utilized. Hence, we propose a method to express a responsibility attribute for sharing information and attaining an agreement between stakeholders.

Keywords: dependability case; safety case; assurance case; responsibility; d*

1. Introduction

Recently, critical incidents have occurred in complex IT systems. The assurance of dependability in IT systems is becoming increasingly important and should be considered during system implementation. The information related to dependability is important knowledge that must be shared among stakeholders. Therefore, the use of dependability cases is attracting attention as a method of sharing between stakeholders and a confirmation of system dependability. However, in previous dependability cases that describe methods represented by goal structuring notation (GSN), the relation between the dependability claim and the responsibility cannot be clearly specified. Thus, since the cause investigation cannot be completed at the time of the incident, system knowledge could not fully be utilized. Hence, we propose a method to express a responsibility attribute for sharing information and attaining an agreement between stakeholders.

© 2013 The Authors. Published by Elsevier B.V. Open access under CC BY-NC-ND license.
Selection and peer-review under responsibility of KES International

* Corresponding author. Tel.: +81-3-6712-9927; fax: +81-3-6712-8372.
E-mail address: saruwatari_takuya_d5@lab.ntt.co.jp
of the incident, the system knowledge could not fully be utilized. Hence, we propose a method to express a responsibility attribute for sharing information to attain an agreement between stakeholders. Here the responsibility attribute specifies who takes responsibility of the dependability case. In this study, we propose a concept for incorporating a responsibility attribute into dependability cases. By incorporating responsibility attribute, accountability of the dependability case by humans can be achieved. For introducing responsibility attributes, we considered d* as the target method. The d* method is one of numerous methods for creating dependability cases. After considering the concept, we indicate a simple notation for it.

In Chapter 1, we describe the introduction. In Chapter 2, we describe the background of the research. In Chapter 3, we describe the dependability case. In Chapter 4, we describe the responsibility attribute incorporated into dependability cases. In Chapter 5, we discuss our findings, and in Chapter 6, we summarize our conclusions.

2. Research background

A dependability case is needed to satisfy the high requirement of critical systems. The notation of dependability cases and methods to create dependability cases have been proposed and researched. There are several methods for notating dependability cases: GSN [1], modular GSN [2-3], D-Case [4], and d* [5]. In [1], Kelly also proposed the following six procedures for creating GSNs: (1) identify goals to be supported; (2) define bases on which goals are stated; (3) identify a strategy to support goal; (4) define bases on which the strategy is stated; (5) elaborate the strategy (and therefore proceed to identify new goals—i.e., go back to (1); and (6) identify a basic solution. A method to create a dependability case by using deviation analysis has also been proposed [7]; further, a method to create a dependability case by using scenarios has also been proposed [2].

In requirement engineering, responsibility-related research is growing. In KAOS [8-9], a method to describe the responsibility relation between an agent and goal was proposed. Sommerville et al. defined responsibility as follows (and proposed an elicitation method of requirement using a responsibility model [10]): “A duty, held by some agent, to achieve, maintain or avoid some given state, subject to conformance with organizational, social and cultural norms.” Currently, research that explicitly combines responsibility and a dependability case is minimal. Therefore, in this study, we propose a concept for explicitly introducing a responsibility attribute for d*, which is the notation for dependability cases.

3. Dependability case

Recently, safety cases, assurance cases, and dependability cases have been attracting significant attention [11-14]. They are prepared to ensure the safety of the given system. In particular, “dependability cases” are documents that are prepared to describe the dependability of a system. Dependability is defined as an integrated concept that encompasses availability, reliability, safety, integrity, and maintainability [15]. By creating a dependability case, the following can be realized:

- Arrangement of dependability information for ease of understanding
- Confirmation of system dependability
- Building consensus among stakeholders regarding system dependability
- Accountability for system behavior

In many cases, GSN is used to describe the dependability case. GSN was proposed as a graphical notation for safety cases [1]. After that, it was extended to be used as a notation for dependability cases [14]. Currently, there are other notations for dependability cases in addition to GSN. Modular GSN is a notation for modularized dependability cases [2]. Using modular GSN, you can create and arrange dependability cases by using modulus. The d* method incorporates the concept of actors into the dependability case [5].
3.1. The d* method

The d* method is a graphical notation for describing a dependability case; it was proposed to incorporate the actor concept into dependability cases. Here actors are defined as having dependability attributes, such as goals, strategies, solutions, and contexts. This idea is considered with reference to i* [16], a method for requirement engineering, and it was proposed by our group.

3.1.1. Notation for d*

The d* method has a graphical notation that consists of five element types that are described using defined characteristic shapes in dependability cases. These shapes are shown in Fig. 1(a). Each element type is described below:

- **Actor** is the element type that constitutes a system. It has dependability attributes, such as goals, strategies, solutions, and context.
- **Goal** is the element type that an actor should satisfy. It may be decomposed into subgoals or substrategies.
- **Strategy** explains why elements are decomposed into subelements. A goal or strategy can be the object of decomposition and may be decomposed into subgoals or substrategies.
- **Solution** shows evidence that the supporting goal is satisfied. It is shown in various ways (e.g., specifications, test reports, procedure manuals).
- **Context** is the external information that is required for goals and strategies. For example, you can consider a procedure list that is the goals target as context.

These elements are related to each other in a dependability case such that d* has four types of relationships. These are described in Fig. 1(b). The four types of relationships are explained below:

- **Support by** relationship shows that an upper element is supported by one or more lower elements. Goals and strategies can be upper elements, whereas goals, strategies, and solutions can be lower elements.

Fig. 1. (a) elements of d*; (b) relationships of d*

- The “supported by” relationship shows that an upper element is supported by one or more lower elements. Goals and strategies can be upper elements, whereas goals, strategies, and solutions can be lower elements.
• The “in context of” relationship shows the addition of a context to goals and strategies.
• The “depend on” relationship shows that one actor depends on another actor. When there is a “depend on” relationship between actors A and B, there is generally a “supported by” relationship between the elements of A and the elements of B.
• The “belong to” relationship shows that goals, strategies, contexts, and solutions belong to a given actor.

3.1.2. Creation procedure for d*

The creation procedure for d* was proposed in [6]. It consists of four procedures. In this method, by repeatedly using these procedures, the dependability case is gradually created. Figure 2 shows the four procedures, which are explained below:

![Fig. 2. procedure of the d* method](image)

• **Actor elicitation**: In this procedure, actors in the system are elicited. In the early stage of dependability case creation, they are elicited using the IT system configuration diagram and other documentation.
• **Dependability elicitation**: In this procedure, dependability information between actors is elicited as inter-dependability information. This relationship will become a “depend on” relationship.
• **Inter analysis**: In this procedure, the inter-dependability information of actors is analyzed by GSN. This information has been elicited in the dependability elicitation procedure. A subgoal should be elicited from the results of the analysis. Also, the following question is answered: which actor will the new goal depend on? If there is not an adequate actor, one must proceed with the actor elicitation procedure for eliciting an adequate actor.
• **Inner analysis**: In this procedure, the inner dependability of actors is analyzed by GSN. The “depend on” relationship is based on the “supported by” relationship. Therefore, for existing “depend on” relationships, “supported by” relationships must be added accordingly; however, there are situations in which “supported by” relationships have not been considered yet, although corresponding “depend on” relationships are allowed. Namely, it is allowed to consider “depend on” relationships before “supported by” relationships.
4. Introduction of a responsibility attribute in dependability cases

4.1. Responsibility attribute

In order for a dependability case to have meaning, it is necessary that each element of the dependability case actually be satisfied. That is, someone must take responsibility for each element of the dependability case. More specifically, humans or organizations must take responsibility for each element of the dependability case. The introduction of a responsibility attribute has been carried out by various methods. Lamsweerde proposed an agent model in a requirement engineering method [8-9], which includes the concept of responsibility. Sommerville et al. proposed a responsibility model for eliciting information requirements and socio-technical risk [10,17-18]. Feltus et al. proposed a responsibility model for verifying organizational structure and detecting policy problems [19]. Boness et al. proposed a responsibility model for appraising the intention of a development [20]. Strens et al. proposed a responsibility model for identifying and specifying requirements [21]. Responsibility attribute concepts are listed in Table 1.

Table 1. Comparison of proposed responsibility attributes

<table>
<thead>
<tr>
<th>Proposer</th>
<th>Purpose</th>
<th>Main Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lamsweerde</td>
<td>Show the distribution of responsibilities within the system and define system scope and configuration</td>
<td>Goal, Agent, Operation, Object</td>
</tr>
<tr>
<td>Sommerville</td>
<td>Explore the structure and dependability of socio-technical systems</td>
<td>Responsibility, Resource, Agent</td>
</tr>
<tr>
<td>Feltus</td>
<td>Verify the organizational structure and detect policy problems</td>
<td>Organization, Responsibility, User (Agent), Task</td>
</tr>
<tr>
<td>Boness</td>
<td>Appraise the intention of a development</td>
<td>Goal, Actor (Agent)</td>
</tr>
<tr>
<td>Strens</td>
<td>Identify and specify requirements</td>
<td>Responsibility, Resource, Agent</td>
</tr>
</tbody>
</table>

Although there is a marginal difference between these concepts, five proposals resemble each other. We discovered that these concepts have a basic pattern of responsibility relationships. That is, there are relationships in which an agent takes responsibility for an object. There are two main concepts that are included in a responsibility relationship: (1) an agent and (2) a responsibility object. A responsibility object was expressed as a responsibility or a goal element in the models. Therefore, when we consider the incorporation of a responsibility concept into d*, we felt that two key elements (responsibility and agent) had to be considered. Some of the proposals could use agent as a non-human system. However, we define an agent as a human or an organization of humans, because non-human systems cannot take responsibility in a human society.

4.2. Responsibility attribute in d*

In this study, we consider the introduction of responsibility attributes into d*. The d* method is one of numerous methods for describing dependability cases. In d*, actor elements are explicitly considered. Here actors are elements of a system and become a dependability subject. A human, an organization, a mechanical system, an IT system, etc. can become an actor in a dependability case of d*. When considering responsibility attributes in d*, it is necessary to consider that non-human objects cannot take responsibilities. More
specifically, mechanical and IT systems cannot take responsibility themselves. Therefore, the introduction of the agent concept is necessary to consider responsibility in d*. We added agent elements and the “responsible for” relationship into d*, which are defined as follows:

- An agent takes responsibility for elements of a dependability case in d*. That is, an agent must have accountability for the contents of elements. A human or an organization consisting of humans can become an agent for actor responsibilities. Further, it is acceptable for the same object to become an actor and an agent simultaneously in d*.

- The “responsible for” relationship shows that elements other than an agent take responsibility held by an agent.

With an actor and other elements, the cardinality of the relationship “responded by” differs. Actors and agents have a many-to-many relationship, as shown in Fig. 3(a). More specifically, it is possible that one agent supports multiple actors, or multiple agents support one actor. When viewed from the perspective of the actor, this relationship can be considered by the following two cases:

- **Case (1):** One agent supports the actor. In this case, the agent must take responsibility of all elements of the actor.

- **Case (2):** Multiple agents support the actor. In this case, each agent must support each of the elements of the actor.

Relationships between other elements (i.e., goals, strategies, solutions and contexts) and agents will be many-to-one. The responsibility of an element must be supported by only one agent. On the other hand, an agent may support multiple elements. This relationship is shown in Fig. 3(b).

![Fig. 3. cardinality between (a) actor and agent; and (b) other elements and agent](image)

### 4.3. Notation of the responsibility attribute in d*

We adopted a new simple graphical notation for the responsibility attribute in d*. As an alternative, it was possible to extend the format of statements that explain the elements in a dependability case. The d* method originally had graphical notation for dependability cases. We felt it natural that the same graphical notation be adopted. We decided to use a shape enclosed by a dashed line to describe a responsibility attribute. Examples using this new notation are shown in Figs. 4 and 5 and are described below.

A portion of a dependability case using this new notation is shown in Fig. 4. In the figure, some actor’s responsibilities are supported by a single agent. For example, “Cage” is such an actor. This corresponds to case (1) above. More specifically, the dependability elements of “Cage” are supported by a “development section.” On the other hand, some actor’s responsibilities are supported by multiple agents. For example, “Rope” is such an actor. This corresponds to case (2). More specifically, the dependability elements of “Rope” are supported by “maintenance section” and “wire company.” In this case, two shapes are overlapped at the actor in the graph.

Another example that describes inner of actor is shown in Fig. 5. Elements of actor are divided and associated with an appropriate agent that has a responsibility for it. All elements of actor must be supported by
the appropriate agent. If there is an element that is not supported by any agent, its dependability case is not complete, and the model would be thought of as being at the middle phase of development.

5. Discussions

5.1. Use of responsibility attributes

One of the purposes of a dependability case is system accountability. To ensure satisfactory accountability for the dependability case, the responsibility of elements of the dependability case should be shared among stakeholders. In previous dependability cases, sharing data was limited to goals, strategies, contexts, and solutions. As shown in this study, a responsibility attribute can be shared among stakeholders by introducing responsibility attributes into dependability cases. Therefore, we extended the dependability case concept to incorporate a responsibility attribute for knowledge sharing among stakeholders.

5.2. Organizations and dependability cases

Generally, agents belong to an organization that has a layered structure. In such an organizational structure, responsibility and authority are transferred from upper to lower layers. A dependability case also has a layered
structure. In dependability cases, the layered structure is modeled via “supported by” relationships between upper and lower goals. That is, an upper goal is supported by one or more lower goals. Relationships of two elements in the layered structure are divided into two types—i.e., hierarchical order or no hierarchical order. When two agents of an organization have a hierarchical order, the corresponding elements of the dependability case also have a hierarchical order.

In general, it is better if two hierarchical orders are combined. For example, consider a case in which agent A takes responsibility for goal A, agent B takes responsibility for goal B, and goal B is supported by goal A. We feel that it is better if agents A and B are the same or if agent A is at a lower layer than agent B. If the hierarchical order is different between the organizational structure and dependability case, a responsibility may be present in the structure of the dependability case, but not in the organizational structure. In such a case, additional action may be required. Checking dependability cases against an organizational structure can identify unsuitable transfers of authority.

Fig. 5. dependability case example that shows inner dependability of actors (Rope).
5.3. Actor and agent concepts

The concept of an agent resembles the concept of an actor. More specifically, these may indicate the same object, but we stress that their concepts must be clearly distinguished. When we think of a responsibility attribute, the concepts of agent and actor are different. An actor can become an agent when an actor is a human or an organization of humans, since a human can take responsibility. Conversely, when an actor is not a human or an organization of humans, the actor cannot become an agent, because the actor cannot take responsibility. For example, IT or physical systems can become actors in a dependability case, but they cannot take responsibility. Therefore, they cannot become agents in a dependability case. Instead, a human that can take responsibility must be an agent. Also, while an actor is a human, the responsibility of an actor might be supported by other humans. In such a case, actors and agents are different humans. In general, we felt that in human society, humans must take responsibility. We must consider supporting responsibility by humans in dependability cases. Only humans or organizations of humans can satisfy accountability in dependability cases.

5.4. Notation of responsibility of d*

In this study, we adopted a graphical notation for describing a responsibility attribute in dependability cases by using d*. Other notations can also be considered for describing responsibility attributes in dependability cases. For example, the methods below can also be considered:
- Extend a statement format of an element to show responsibility
- Use an external table to indicate the relationship between element and responsibility

In the future, it is necessary to evaluate a suitable method while considering the purpose of the dependability case.

5.5. Extending the creation procedure for responsibility attributes

In this study, we created a dependability case example that introduced a responsibility attribute by using an existing dependability case. We were not creating the dependability case example that has responsibility attributes from the beginning. We appended the responsibility attributes to dependability case that did not have responsibility attributes. Therefore, we did not consider the procedure for introducing responsibility attributes into a dependability case. Nonetheless, we can conclude the following: (1) it is necessary to prepare the procedure of eliciting an agent; and (2) it is better to add an agent to a dependability case after dependability analysis (i.e., inter analysis and inner analysis). After creating a dependability case, the procedure that checks by comparing the organizational and dependability case-layered structures may be also needed.

6. Conclusion

In this study, we proposed a concept for incorporating responsibility attributes into a dependability case described by d*. We also proposed a simple notation for it. In previous dependability cases, sharing data of attributes was limited to goals, strategies, contexts, and solutions. We added the responsibility attribute. Therefore, the whereabouts of accountability in a dependability case can be clearly shown. In introducing the responsibility attribute, we investigated previous research that relates to responsibility models. We found that responsibility attributes consist of two type elements—i.e., an agent and a responsibility object. We used the agent concept in dependability cases to introduce responsibility attributes.

In the future, we plan to improve the concept and corresponding notation through experiments in describing dependability cases. Considering procedures for creating dependability cases that have responsibility attributes will also be future studies.
References