D* framework creation procedure from collaboration diagram

Takuya Saruwatari* and Shuichiro Yamamoto
Nagoya University
Nagoya, Japan
saruwatari.takuya@e.mbox.nagoya-u.ac.jp, yamamotosui@icts.nagoya-u.ac.jp

Abstract

Recently, serious failures of complex IT systems are becoming social problems. Assurance cases are attracted as a technique to assure the dependability of critical systems. To describe an assurance case, d* framework is proposed. The d* framework is an extended assurance case notation based on the network of dependable actors. In this paper, d* framework creation procedure that creates the assurance case from the collaboration diagram is proposed. Moreover, the case study is performed to evaluate the proposed procedure.

Keywords: d* framework, assurance case, dependability, collaboration diagram

1 Introduction

Recently, serious failures of complex IT systems are becoming social problems. A failure of critical system raises a significant loss. Therefore, assurance of dependability of such critical systems is an important challenge. But, it is not an easy task. Especially, it is difficult to integrate different but related technologies without collaboration method for IT convergence. It is necessary to inter-connect many related systems to realize IT convergence. In such a situation, assurance cases are attracted as a technique to assure the dependability of critical systems. Assurance cases describe argument of dependability. GSN (Goal Structuring Notation) [18] is proposed as a graphical notation of assurance cases. D* framework is also proposed to extend assurance case [28] [26]. In this paper, d* framework creation procedure that creates assurance cases from collaboration diagrams is proposed and a case study is performed to evaluate the proposed procedure.

Section 2 describe related works. Section 3 defines the d* framework. In section 4, the d* framework creation procedure is proposed. In section 5, the case study is described to show the application process of d* framework. Discussions and conclusions are described in section 6 and section 7 respectively.

2 Related work

Recently, many researches related to the assurance case are performed. The assurance case is a document that describes the argument of system dependability. Such a document is needed for critical systems that require high assurance of dependability. Dependability is defined as an integrated concept including availability, reliability, safety, integrity, and maintainability [30] [8].

Safety case is proposed as the document that describes the argument of system safety [18]. Generally, safety case is described as a structured diagram. There are some notations for a safety case. GSN (Goal Structuring Notation) [18] [19] [3] [17], and CAE (Claim, Argument Evidence) [2] are proposed as graphical notation to describe a safety case. In GSN, the safety case is described by four types of node

IT CoNvergence PRActice (INPRA), volume: 2, number: 2, pp. 43-54
*Corresponding author: NTT Software Innovation Center, NSS2 Bldg., 6F, 2-13-34 Konan, Minato-ku, Tokyo 108-0075, Japan

and two types of relationship. Node types are goal, strategy, evidence, and context (there are assumption and justification as sub classes of context). Relationship types are "supported by" and "in context of". Recently, target of assurance case are extended to dependability [12]. An assurance case for dependability may be called as dependability case. There are several terms such as safety case, certification report, security case, or dependability case; we collectively call them assurance cases [27]. There are several definitions for assurance cases. One definition of them is given as follows [7]

A documented body of evidence that provides a convincing and valid argument that a system is adequately dependable for a given application in a given environment.

There are many researches related to assurance case. In [23], patterns of assurance case are researched. Some pattern examples are available in [1] [20]). There are also researches related to module of assurance case [9] [21] [13] [22] and formal check of assurance case [24]. By using module of assurance case, there are some merit as follows. That is, a large assurance case can be described and managed by module structure. In addition, reuse of modules is also possible. The d* framework that is used in this research uses module. In the d* framework, an actor is introduced to the assurance case and it is described by a module.

Creation process of assurance case is also researched. In [18] [19], six step to create assurance case (safety case) is proposed based on GSN. This step is shown as follows. (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. Other assurance case creation methods are also proposed. In [14], deviation analysis is used to create an assurance case. In [21], scenario based method is proposed. In terms of the development process for a system that it comprises multiple systems (i.e., a system of systems), a technique involving system analysis, goal elicitation, identification of candidate design alternatives, and resolution of conflicts has been proposed for the creation of assurance cases in a structured manner [15]. Meanwhile, methods for the decomposition of arguments in the assurance cases are categorized. [11]. Yamamoto evaluate the effectiveness of argument patterns of assurance case [29]. Graydon proposed assurance based development approach to constructing critical systems [16].

2.1 D-Case Editor

D-Case Editor [25], [5] was developed as a tool for creating D-Case [10] in the DEOS project [6]. DEOS project is a Japanese national research project funded by Japan Science and Technology Agency. The D-Case Editor operates as a plug-in of Eclipse [4]. D-Case is an extended method of GSN. D-Case has a monitoring node that is sub class of evidence. Using monitoring node, real time monitoring at the time of system operation can be performed. D-Case can also use module notation of assurance case. D-Case Editor also can represent the d* framework. D-Case Editor is used in this research. A screen shot of D-Case editor is shown in Fig. 1.

3 D* framework

The d* framework is proposed as an assurance case notation [28] [26]. It extends the GSN. In d* framework, actor concept is introduced into assurance case. 5 node types (actor, goal, strategy, context, and evidence) are defined as elements of d* framework. And, 4 relationship types ("supported by", "in context of", "depend on", and "belong to") are defined between nodes. Actor node is a new element of assurance case. Previous notations such as GSN do not have it. Person, organization, system, subsystem,

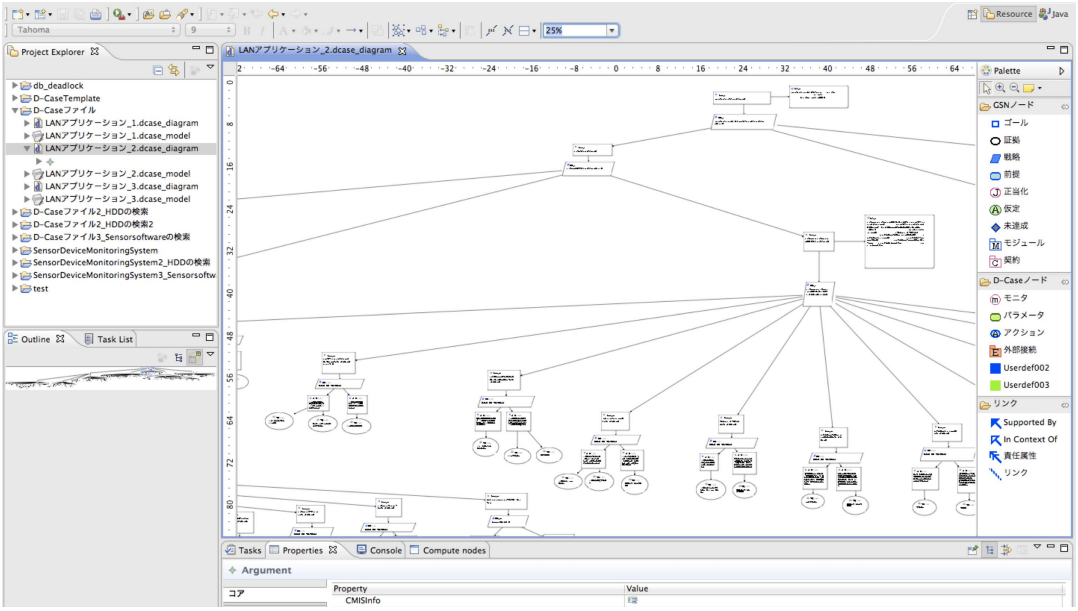

Figure 1: A screen shot of D-Case editor.

and component etc. can be defined as an actor. By introducing actor into the assurance case, the dependability argument in the actor and the dependability argument between actors can be separated. There are two types of d* framework. They are shown as below.

1. The whole d* framework that describes responsibility relationships that an actor depends on other actors. (Inter actor d* framework)

2. The individual d* framework that describes that each actor is dependable. (Intra actor d* framework)

3.1 Notation

The d* framework has a graphical notation that consist of 5 element types. These element types are described using defined characteristic shapes in d* framework. These shapes are shown in Fig. 2 (a). Each element type is described below:

- **Actor** is an element type that constitutes a system. It has dependability attributes, such as goals, strategies, evidences, and context.

- **Goal** is an element type that a system should satisfy. It may be decomposed into sub goals or sub strategies.

- **Strategy** describes argument strategy. That is it describes why element is decomposed into sub elements. Goal or strategy can be the subject of decomposition and may be decomposed into sub goals or sub strategies.

- **Evidence** is an element type. It assures 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, list of subject equipment may be considered as context of goal.

In d* framework, these elements are related each other. 4 relationship types are defined to relate these elements. These are described Fig. 2 (b) and are explained below:

- **"Supported by"** relationship describes that an upper element is supported by one or more lower elements. Goal and strategy can be upper elements and goal, strategy, and evidence can be lower elements.
- **"In context of"** relationship describes addition of context to goal and strategy.
- **"Depend on"** relationship describes that one actor depends on another actor.
- **"Belong to"** relationship describes that goal, strategy, context, and evidences belong to an actor.

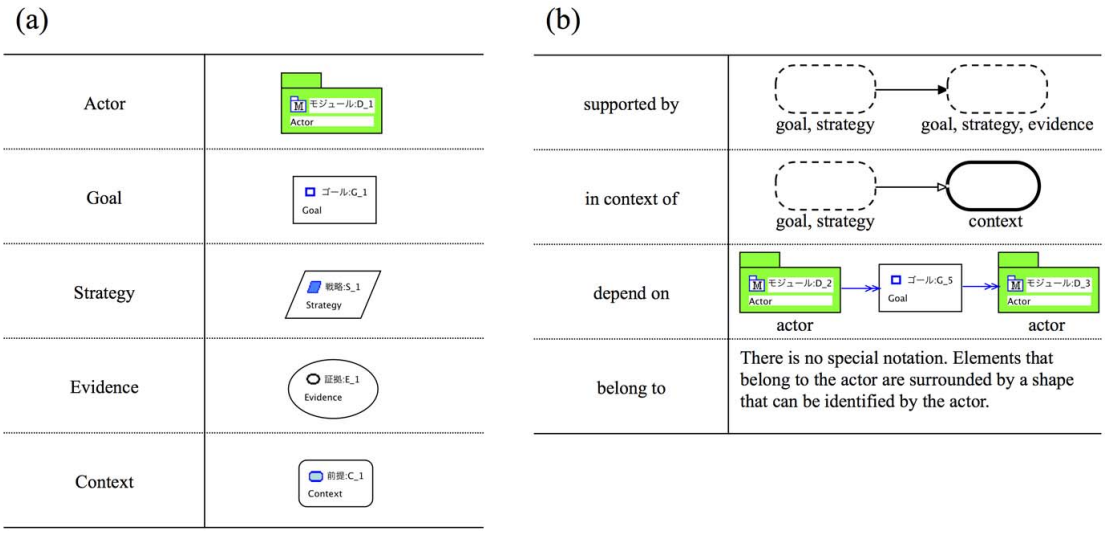

Figure 2: (a) elements of d* framework; (b) relationships of d* framework.

3.2 Meta model

In Fig. 3, Meta model of d* framework is shown. Element of d* framework, and their relationships are described in this model. In this Meta model, "argumentElement" is defined as a super class of "goal", "strategy", "evidence", and "context". "ArgumentElement" is used to describe argument of dependability in d* framework. "ArgumentElement" belongs to actor. Moreover, "argumentElement" is supported by other "argumentElement". And, "argumentElement" is in context of other "argumentElement". "Depend on" relationship is defined between "actors". "Actor" may be a depender and dependee. "ArgumentElement" also belongs to "depend on" relationship.

4 D* framework creation procedure

In this paper, a d* framework creation procedure is proposed. The d* framework is created from collaboration diagram in this procedure. It is consisted of 3 steps. They are shown below.

- Step 1 : Actor definition
  Actors of the target d* framework case are identified from objects in the corresponding collaboration diagram.

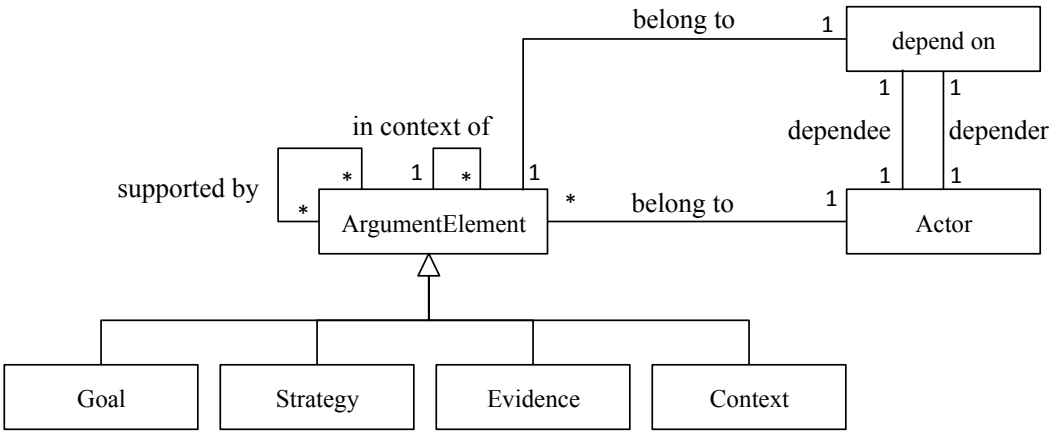

Figure 3: Meta model of d* framework.

- Step 2 : Inter dependency argument definition
  Dependability arguments between actors in the target d* framework are defined as follows. The corresponding message relationship in the collaboration diagram is determined for using to define dependability arguments between actors. The following two types of arguments are defined.

  - Argument type1
    Messages in the message relationship shall correctly be transmitted between actors. For this type of argument, the claim that message is correctly transmitted is defined between actors.
  - Argument type2
    Message orderings between message relationships shall correctly be managed between actors. For this type of argument, the claim that message ordering is correctly managed between messages is defined internally for the corresponding actors.

- Step 3 : Intra actor dependability argument definition
  Dependability arguments of an actor in the d* framework case are defined as follows. First, the corresponding object in the source collaboration diagram is determined for the actor in consideration. The message relationships of the object are used to define intra dependability arguments of the target actor. The following two types of arguments are defined as the internal arguments of the target actor.

  - Argument type1
    The ordering of the transmitted messages to the object shall correctly be managed in the object. For this type of argument, the claim that message ordering is correctly managed between transmitted messages is defined internally for the corresponding actor.
  - Argument type2
    Messages parameters in the transmitted messages shall also correctly be managed in actors. For this type of argument, the claim that every message parameters are correctly managed is defined for the corresponding actors.

5 Case study

5.1 Collaboration diagram of the target system

The target system for case study is an "AP Download system". Using this system, user can select the application (AP) and download it from the system to his/her IC Card. The collaboration diagram of target system is shown in Fig. 4. In this diagram, 6 objects and 13 message relationships between them are represented. Each message may have parameters. In the figure, parameter is described in parentheses of the corresponding message. This is an input information to the proposed procedure in this case study.

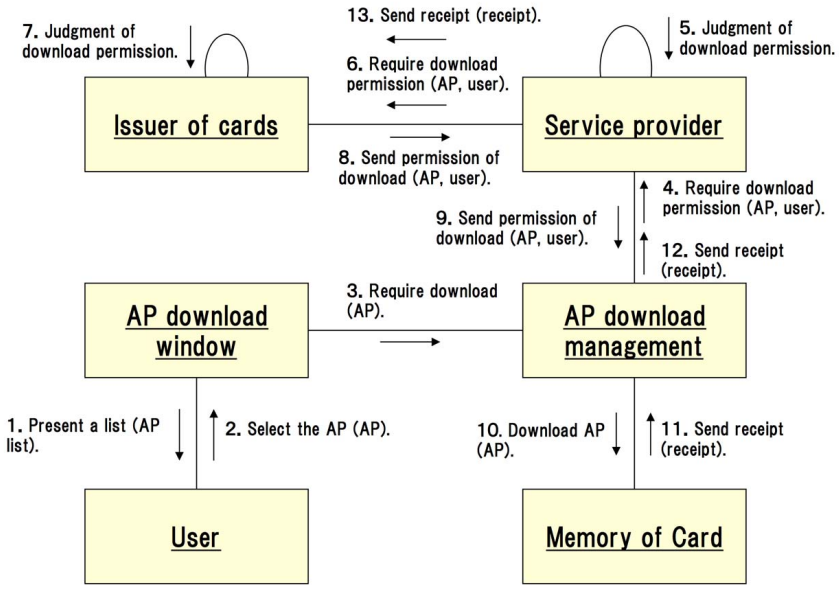

Figure 4: Collaboration diagram of AP download system.

5.2 Creation process of the d* framework

In the case study, the d* framework creation experiment is performed. The result of experiment for each step in proposed procedure is shown as below.

- Step 1
  In this step, 6 objects are defined as actor by using the collaboration diagram. The defined 6 actors are "Issuer of cards", "Service provider", "AP download window", "AP download management", "User", and "Memory of Card".

- Step 2
  In this step, an inter actor d* framework is created and 15 dependability goals are defined between actors. For example, in the collaboration diagram, "Issuer of cards" send message ("Send permission of download (AP, user)") to "Service provider". Therefore, it is considered that "Service provider" depend on "Issuer of card". The goal is ("Permission of download is sent correctly"). Created d* framework is shown in Fig. 5.

- Step 3
  In this step, intra actor d* framework are created and dependability argument of each actor is

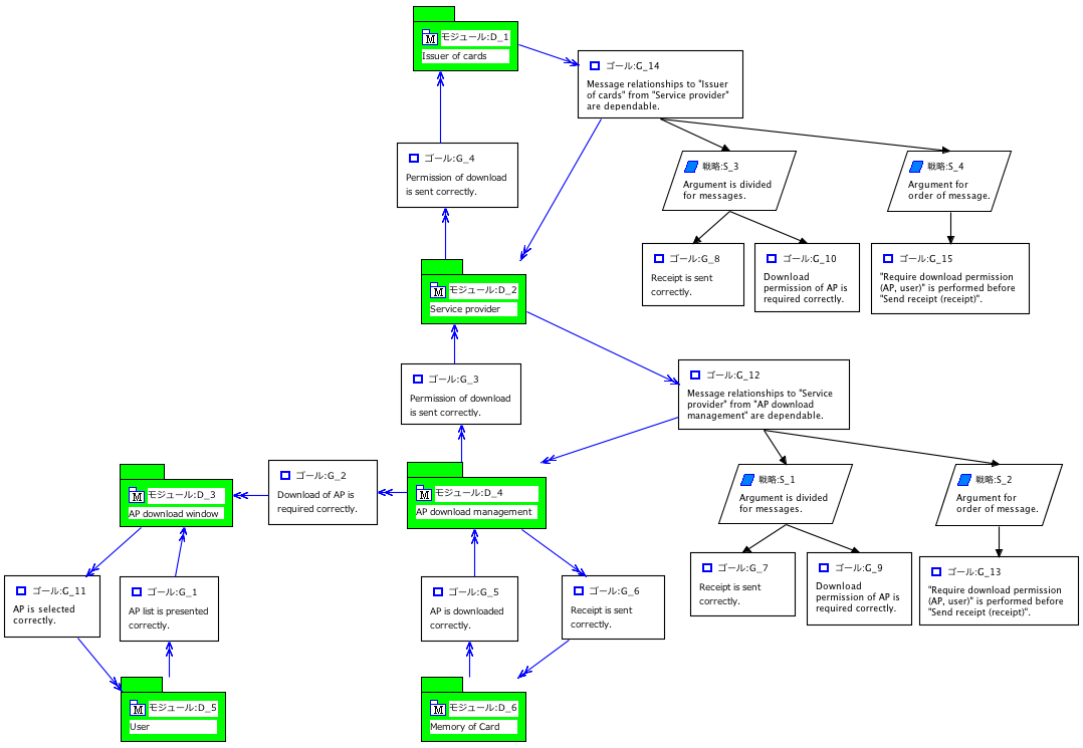

Figure 5: Inter actor d* framework that is created in Step 2.

described in them. 6 d* frameworks are created. Here, 2 d* frameworks are shown as example. The d* framework of “Issuer of cards” is shown in Fig. 6. The d* framework of “Service provider” is shown in Fig. 7. In this step, dependability goals are described in each d* frameworks. For example in the collaboration diagram, “Issuer of cards” send message (“Send permission of download (AP, user)”) to “Service provider”. Therefore, the goal (“Parameters of “Issuer of card” is dependability managed”) is described in the d* framework of “Issuer of card” (Fig. 6). This goal is decomposed to two goals through the strategy. These two goals are ”Information of AP that is permitted to download is managed” and ”Information of user who is permitted to download is managed”.

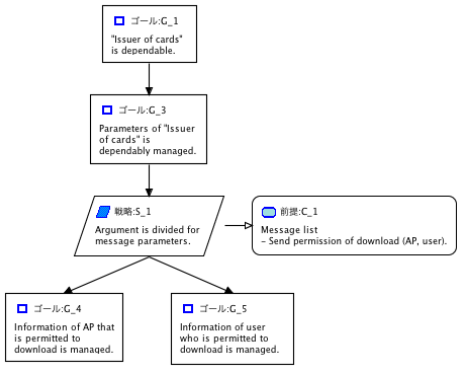

Figure 6: Intra actor d* framework of “Issuer of cards”. This is created in Step 3.

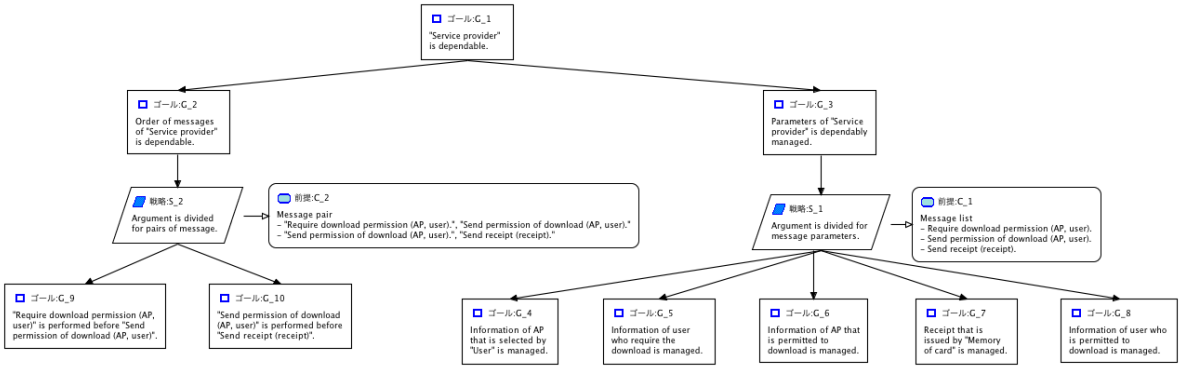

Figure 7: Intra actor d* framework of “Service provider”. This is created in Step 3.

5.3 Result of the case study

Numbers of elements of collaboration diagram that is used to create d* framework are shown in Table. 1. And, numbers of elements of created d* framework are shown in Table. 2.

Table 1: Number of elements of collaboration diagram.

Object	Message relationship
6	13

Table 2: Number of elements of created d* framework.

D* framework	Actor	Goal	Strategy	Context
Inter actor d* framework	6	15	4	0
Issuer of cards	-	4	1	1
Service provider	-	10	2	2
AP download window	-	6	2	2
AP download management	-	9	2	2
User	-	3	1	1
Memory of Card	-	3	1	1

6 Discussions

6.1 Inter relationship between collaboration diagram and d* framework

When creating d* framework, it is convenient if there is information that can be used. In this research, it is confirmed that collaboration diagram is useful to create a d* framework. Relationships are as follows. 1) Actors of a d* framework can be defined by using objects of a collaboration diagram. 2) The dependability arguments can be considered by using message relationships of a collaboration diagram. In the case study, 6 actors are defined in the d* framework from 6 objects in the collaboration diagram directly

(Step 1). Moreover, 15 dependability goals between actors are defined by using message relationships in the collaboration diagram (Step 2) and 35 dependability goals of actors are defined by using message relationships in the collaboration diagram (Step 3). In this definition, 13 message relationships in collaboration diagram are used. Thus, a collaboration diagram is useful, when d* framework is described.

6.2 Goal Definition in d* framework

In the proposed procedure, goals of d* framework are defined by message relationships of collaboration diagrams. There are 3 types of goal definitions. These types are derived by "message", "message ordering" and "message parameters" of the message relationships of the collaboration diagrams. These goal types are discussed below.

6.2.1 Message assuredness

Message relationships are defined between objects in collaboration diagrams. The message relationship indicates the existence of the message flow between objects. As for the d* framework argument, it should be assured that messages on the message relationship are sent correctly between objects. Therefore, goals to assure the correctness of messages shall be defined in d* frameworks based on collaboration diagrams.

In the case study, these goals are defined between actors. These actors are extracted from objects of collaboration diagrams. For example, goal ("AP list is presented correctly") is defined between "User" and "AP download window" (Fig. 5). This goal is extracted from the message relationship ("2. Select the AP") of the example collaboration diagram.

6.2.2 Message ordering assuredness

In the collaboration diagram, the message ordering between message relationships is defined. Therefore, goals to assure the message ordering are defined for the d* frameworks based on collaboration diagrams. The goals for message ordering shall be defined as the inter-actor goal and intra-actor goal. The inter-actor goal is defined between two actors. These actors exchange the corresponding message. The intra-actor goal is define in each actors that are inter related by the corresponding message. The order of messages is assured between the two actors is described as for the inter-actor goal. The order of messages for the specific actor is assured as for the intra-actor goal.

In the case study, these two types of goals are defined by using objects of the collaboration diagram. For example, goal (""Require download permission (AP, user)" is performed before "Send receipt (receipt)"") is defined between "Issuer of card" and "Service provider " (Fig. 5). This is an example of inter-actor goal. The goal (""Require download permission (AP, user)" is performed before "Send permission of download (AP, user)"") is defined in "Service provider" (Fig. 7). This is an example of intra-actor goal.

6.2.3 Message parameter assuredness

Messages can have parameters in collaboration diagrams. D* frameworks for collaboration diagrams should have argument to assure that message parameters are managed correctly. Therefore, goals to assure the correct management of message parameters should be defined in d* frameworks.

In the case study, these goals are defined in actors. Parameters are input-output data of an actor. Therefore, these parameters should be managed correctly by a corresponding actor. For example, goal ("Information of AP that is permitted to download is managed.") is defined in "Issuer of card" (Fig. 6).

6.3 Limitation of experiments

Scale of case study was small. And, one researcher created the d* framework in the case study. Real developers did not create it. Therefore, the case study situation is different from concrete situation. Moreover, validity of the d* framework is not clear.

7 Conclusion

Assurance cases are attracted as a technique to assure the dependability of critical systems. The d* framework is proposed to describe assurance cases. In this paper, the d* framework creation procedure is proposed. The proposed procedure consists of 3 steps and a collaboration diagram is used to create d* framework. Moreover, the case study is performed to evaluate the proposed procedure. The d* framework of "AP download system" was created from the collaboration diagram in the case study. Experimental case study shows that d* framework can be easily developed by collaboration diagrams.

8 Acknowledgments

This research is partially supported by JSPS Research Project Number:24220001.

References

[1] Technical report.

[2] Adelard. `http://www.adelard.com/web/hnav/ASCE/`.

[3] ©2013 origin consulting on behalf of the gsn working group contributors: Gsn standard. `http://www.goalstructuringnotation.info/`.

[4] ©2013 the eclipse foundation. all rights reserved: Featured eclipse project. `http://www.eclipse.org/`.

[5] D-case editor a typed assurance case editor. `http://www.dependable-os.net/tech/D-CaseEditor/`.

[6] Deos project. `http://www.crest-os.jst.go.jp/`.

[7] T. S. Ankrum and A. H. Kromholz. Structured assurance cases: Three common standards. In *Proc. of the 9th IEEE International Symposium on High-Assurance Systems Engineering (HASE'05), Heidelberg, Germany*, pages 99–108. IEEE, October 2005.

[8] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, January-March 2004.

[9] I. Bate and T. Kelly. Architectural considerations in the certification of modular systems. In *Proc. of the 21st International Conference on Computer Safety, Reliability and Security (SAFECOMP'02), Catania, Italy, LNCS*, volume 2434, pages 321–333. Springer-Verlag, September 2002.

[10] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison Wesley Publishing Company, 1987.

[11] R. Bloomfield and P. Bishop. Safety case composition using contracts - refinements based on feedback from an industrial case study. In *Proc. of the 18th Safety-Critical Systems Symposium (SCSS'10), Bristol, UK, LNCS*, pages 51–67. Springer-Verlag, February 2010.

[12] G. Despotou and T. Kelly. Extending the safety case concept to address dependability. In *Proc. of the 22nd INTERNATIONAL SYSTEM SAFETY CONFERENCE (ISSC'04)*, pages 645–654, 2004.

[13] J. Fenn, R. Hawkins, P. Williams, and T. Kelly. Safety case composition using contracts - refinements based on feedback from an industrial case study. In *Proc. of the 15th Safety-critical Systems Symposium (SSS'07), Bristol, UK, LNCS*, pages 133–146. Springer-Verlag, February 2007.

[14] T. K. G. Despotou. Structured assurance cases: Three common standards. In *Proc. of the 1st Institution of Engineering and Technology International Conference on System Safety (IETIC'06), London, UK*, pages 29–38, June 2006.

[15] T. K. G. Despotou. Design and development of dependability case architecture during system development. In *Proc. of the 25th International System Safety Conference (ISSC'07), Baltimore, USA*, pages 1–1, August 2007.

[16] P. J. Graydon, J. C. Knight, and E. A. Strunk. Assurance based development of critical systems. In *Proc. of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), Edinburgh, Scotland*, pages 347–357. IEEE, June 2007.

[17] D. Jackson, M. Thomas, and L. I. Millett. *Software for Dependable Systems: Sufficient Evidence?* National Academies Press, 2007.

[18] T. Kelly. *Arguing Safety - A Systematic Approach to Managing Safety Cases*. PhD thesis, University of York, September 1998.

[19] T. Kelly. A six-step method for developing arguments in the goal structuring notation (gsn). Technical report, York Software Engineering, UK, 1998.

[20] T. Kelly. *The Safety of Software–Constructing and Assuring Arguments*. PhD thesis, University of York, September 2003.

[21] T. Kelly. Using software architecture techniques to support the modular certification of safety-critical systems. In *Proc. of the 11th Australian workshop on Safety critical systems and software (SCS'03)*, pages 53–65, 2006.

[22] T. Kelly. Modular certification to support open systems dependability. In *Proc. of the 2nd International Workshop on Open Systems Dependability (WOSD'12)*, 2012.

[23] T. Kelly and R. Weaver. The goal structuring notation–a safety argument notation. In *Proc. of the the dependable systems and networks 2004 workshop on assurance cases (DSN'04)*, 2004.

[24] Y. Matsuno and K. Taguchi. Parameterised argument structure for gsn patterns. In *Proc. of the 11th International Conference on Quality Software (QSIC'11), Madrid, Spain*, pages 96–101. IEEE, July 2011.

[25] Y. Matsuno, H. Takamura, and Y. Ishikawa. A dependability case editor with pattern library. In *Proc. of the 12th IEEE International Symposium on High-Assurance Systems Engineering (HASE'10), San Jose, California*, pages 170–171. IEEE, November 2010.

[26] T. Saruwatari and S. Yamamoto. Definition and application of an assurance case development method (d*). *Journal of SpringerPlus*, 2(1):1–8, May 2013.

[27] S. Yamamoto and Y. Matsuno. Workshop on assurance cases: best practices, possible obstacles, and future opportunities 1 ,. In *Proc. of the International Conference on Dependable Systems and Networks (DSN'04), Florence, Italy*, pages 841–841. IEEE, July 2004.

[28] S. Yamamoto and Y. Matsuno. d*framework: Inter-dependency model for dependability. In *Proc. of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'12), Boston, USA*, pages 1–1. IEEE, June 2012.

[29] S. Yamamoto and Y. Matsuno. An evaluation of argument patterns to reduce pitfalls of applying assurance case. In *Proc. of the 1st International Workshop on Assurance Cases for Software-intensive Systems (ASSURE'13), San Francisco, USA*, pages 12–17. IEEE, May 2013.

[30] C. V. Zhou, C. Leckie, and S. Karunasekera. Collaborative detection of fast flux phishing domains. *Journal of Networks*, 4(1):75–84, February 2009.

Author Biography

Takuya Saruwatari received the B.S. and M.S. degrees in Electrical Engineering from university of Tsukuba in 1994 and 1996, respectively. He is with NTTDATA from 1996. From 2011, he is a doctoral course student in Graduate School of Information Science, Nagoya University.

Shuichiro Yamamoto received the B.S. degree from Nagoya Institute of Technology and the M.S. and Dr.Eng. degrees from Nagoya University in 1977, 1979, and 2000, respectively. He joined Nippon Telegraph and Telephone Public Corporation (now NTT) in 1979 and engaged in the development of programming languages, CASE tools, network-based smart card environments, and distributed application development platforms.His research interests include distributed information systems, requirements engineering, ubiquitous computing, knowledge creation, dependability engineering, and Enterprise Architecture. He moved to NTT DATA in 2002. He became the first Fellow of NTT DATA Research and Development Headquarters in 2007. He moved to Nagoya University as a professor in 2009.He is currently the leader of working group for the requirements evolution based system development method that is a research project of the Software Engineering Center of IPA (Information technology Promotion Agency, Japan). He is the chairman of AEA (Association of Enterprise Architects) Japan. He is the chairperson of SIG Knowledge Sharing Network of The Japanese Society for Artificial Intelligence.