Visualized Assurance Approach for Enterprise Architecture

Qiang Zhi*, Zhengshu Zhou, and Shuichiro Yamamoto

Graduate School of Informatics, Nagoya University, Nagoya, Aichi, CO 4648601 Japan

Abstract

In software engineering, to ensure reliability between systems, describing both system architecture and assurance arguments between system elements is considered necessary. There are proposals for system architecture assurance, but use of these traditional methods often requires development of different diagrams using different editors. Because the visual sense of the traditional methods is inadequate, errors readily occur when manipulating different diagrams. Therefore, it is essential that the assurance of dependability between components and systems is visualized and easy to understand. In this paper, an integrated approach to describe the relationship between system actors and system architecture is proposed, and this approach is clarified using an enterprise architecture modeling language. A case study is carried out and comparison to the traditional approach d* framework is explained. The comparison results show that the proposed approach is more suitable for ensuring dependability in system architecture.

Index Terms: Assurance Case, Dependability Assurance, Enterprise Architecture, Modeling Language, System Architecture

I. INTRODUCTION

System modeling is indispensable in software development [1]. Unified Modeling Language (UML) [2] is a modeling language for object-oriented analysis and design, and Systems Modeling Language (SysML) [3] encompasses the entire system, including software and hardware. However, these traditional modeling languages do not directly implement dependability assurance between components and systems.

Interdependency management is important for developing contemporary systems that comprise acquired components and communications. To assure the dependability of a system A, assuring the internal dependability of A, interdependence of A, and internal dependability of all the systems with which A is interdependent will be necessary. Although Yu [4] showed that the network of intentions among the actors can be represented using the i*framework, the problem of how to treat the dependability of systems has not been solved.

Some other methods can be combined with system models to achieve dependability assurance. When a system architecture contains assurance cases, d*framework [5] can be used to assure system dependability. An assurance case is a goal tree that shows a top claim based on decomposition into sub-claims, with evidence to show the validity of the bottom claims. The general form of an assurance case is a goal tree including a Top Claim, Sub Claims, and Bottom Claims. The validity of the bottom claims should be proven by evidence.

In the d*framework, an actor node is used to relate the assurance case. The premise of development using the d*framework is the existence of a collaboration diagram; therefore, the scope of the d*framework is limited when assuring dependability between software components. Moreover, in terms of visualization, the performance of traditional methods is not outstanding. A newly unified and visualized
method that describes not only the system architecture but also assurance cases is necessary.

In this paper is proposed a composite dependability assurance approach to describe dependability arguments among actors including business actors, application actors, and technology actors. This is completed using ArchiMate [6], which is a modeling language for development of enterprise architecture models. ArchiMate provides a clear method for visualizing construction and business processes: operation, organizational structure, information flow, application service, and technology infrastructure. The remainder of this paper is organized as follows. The related work is described in Section II. A visualized assurance approach is explained in Section III. In Section IV, the proposed approach is evaluated using a case study. In section V, the difference between the proposed approach and d*framework is explained. Conclusions and directions of future work are also provided in the last section.

II. RELATED WORK

The Open Group Real-time & Embedded Systems Forum [7] proposed an approach, termed Dependability through Assuredness, with the aim to reduce the risk of malfunctioning systems and security vulnerabilities for customers and end-users. The core of this approach is the O-DA standard that focuses on the dependency of architectural modeling, development of security cases, and product solutions that are based on the architectural methodology, standards and certification programs.

To visualize the dependability of systems, the notion of Assurance Case was proposed to model dependency for system architecture. In addition, Goal Structuring Notation (GSN) [8], is used to visualize methods and thoughts for system requirements. Dependability Case (D-case) [9,10], a derivative of GSN, is a technology and a tool used to describe dependability. In addition, when there is a need to describe the relationship between models, safety case construction [11] can be used. However, the i*framework is suitable for describing the relationship between actors but is not suitable for describing system architecture dependability.

To assure system architecture dependability, the d*framework [5,12-14] was proposed to model dependability between system architecture. In the d*framework, the interaction parameters in system architecture can be represented by D (X, Y) in the d*framework. In detail, X and Y are related systems and D (X, Y) is the inter-dependability condition to assure that Y meets the dependability requirements of X. Such a relationship is termed an “open depend-on relationship.” If an actor is undefined, the undefined actor is described by “...”. The relationships in system architecture can be graphically represented in the d*framework, but the visual sense of the d*framework is inadequate.

ArchiMate is a visual business analysis model language (a type of Architecture Description Language: ADL) that is able to integrate multiple architectures. It belongs to the describes object, behavior, subject, product, organization, process, information, data, application, and technology from three layers: business, application, and technology layers. In addition, the goal model can also be described by the motivation elements of ArchiMate. Lankhorst [15] and Wierda [16] introduced the application of ArchiMate for enterprise architecture. A non-functional requirements (NFR) framework [17] can also be modeled as the function of motivation elements in ArchiMate. The elements in the business and technology layers cannot be represented by UML or SysML. SysML has the capability of requirements diagram description, but different types of elements are indistinguishable in SysML. However, the elements in the business layer, application layer, and technology layer can be related by the motivation elements in ArchiMate.

Some proposals have been made to assure the dependability of systems by using ArchiMate, such as security risk management [18], information security risk assessment [19], and mobile security [20]; then, these approaches were improved [21] and summarized [22]. The transformation from traditional methods for dependability assurance to ArchiMate was proposed by Yamamoto [23]. Intra-model security assurance [24] was proposed based on this transformation and integrated system architecture and assurance case.

III. VISUALIZED ASSURANCE APPROACH

In this section, a general structure of visualization for enterprise architecture is described in Fig. 1 and Fig. 2 shows a model that can integrate system architecture and security cases. The mapping between system architecture and actors in the business layer is also clarified in Fig. 2. This model enables improvement of the efficiency of dependability assurance and of the visual sense. This is because the cognition and operation gaps caused by manipulating different diagrams, such as assurance case and system architecture diagrams, are reduced by using an integrated diagram and a unified modeling language. Compared to traditional methods, this approach can also provide visual convenience for people in different fields. In the following, an example is provided to describe the internal behavioral relationship between actor and assurance case.

In this section, the meta-model for assurance case and business layer actors of the system architecture is proposed. In previous research [24, 25], the security assurance methods for the application and technology layers were only clarified; no proposal was made for the relationship between the actors

https://doi.org/10.6109/jicce.2019.17.2.117 118
of the business layer. For visualization of the dependability in the business layer, herein is presented an interdependence relation method that is explained in Section IV using ArchiMate. Regarding the assurance cases in the application and technical layers, the previous research methods remain applicable.

A. Model of the Relationship for the Business Layer and Assurance Case

Fig. 2 shows the meta-model that integrates the system architecture and assurance case. The business layer of the system architecture is represented by orange in the diagram. In this meta-model, the business layer consists of two actors, which show an inter-dependency relationship. In the assurance case, there are four elements: property, risk, countermeasure, and evidence. The evidence will be realized by actor Y in the business layer; then, the countermeasure will be proved by the evidence. Next, the risk that threatens the property will be resolved by the countermeasure. Lastly, the property satisfies actor X in the business layer. A case study is used in Section IV to analyze this model.

B. Using ArchiMate to Define the Mapping Relationship between the Assurance Case and Actor

Table 1 shows the mapping relationships for the model in Fig. 2.

In the previous study [24], the internal relationship of assurance case in ArchiMate was proposed, this paper proposed a method for combining the assurance case with the actors in business layer as shown in Fig. 3. In this model, the interrelationship between business actor and assurance case is described by the influence, association and realization relationship. Actor Y realizes the requirement and influence the countermeasure, the property associates Actor X.

<table>
<thead>
<tr>
<th>Meta-model</th>
<th>ArchiMate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Driver</td>
</tr>
<tr>
<td>Risk</td>
<td>Assessment</td>
</tr>
<tr>
<td>Countermeasure</td>
<td>Goal</td>
</tr>
<tr>
<td>Evidence</td>
<td>Requirements</td>
</tr>
<tr>
<td>Actor</td>
<td>Business Actor</td>
</tr>
</tbody>
</table>

Fig. 1. General structure of visualization for enterprise architecture.

Fig. 2. Model of relationship for business layer and assurance case.

Fig. 3. Model of relationship for business layer and assurance case in ArchiMate.
C. Definition of the Mapping between the Business Actor and Composite Dependability Goals

To describe the dependability goals in the d*framework [5], a method for mapping the depend-on relationship between actors is proposed. For a goal, actor X depends on another actor, Y. The goal realized by actor Y will influence Actor X. In this study, the depend-on relationship is expressed by the association and realization relationships in ArchiMate. Fig. 4 shows an example of the depend-on relationship. In this figure, a student is associated with the goal expressed as “The lecture is interesting,” and the teacher realizes this goal.

D. Combination of Depend-on Relationship and Assurance Case

To clarify the depend-on relationship between actors, the composite dependability goal will be decomposed into an assurance case such as that shown in Fig. 3. In the case presented here, a student wants to have a dependable cram school (college preparation school) but is worried that the teaching level of the teacher is not sufficiently high. He needs a qualified teacher. In this case, that the cram school is dependable will be the property; the risk is insufficient teaching level; the countermeasure is verifying the teacher's teaching level; and lastly, an adequate teacher qualification will be the evidence.

According to Table 1, property, risk, countermeasure, and evidence will be represented by the drivers, assessment, goal, and requirements in ArchiMate. Fig. 5 shows this case in ArchiMate.

IV. CASE STUDY

A. Digital Signature Process

To explain this proposal, a case study is highlighted in the following. The target system is the digital signature process and the dependability among the sender, receiver, and key generator is analyzed. Although this process is typically more complicated, the system structure is simplified here to explain accurately the approach of this paper. Fig. 6 shows the architecture of the digital signature process using ArchiMate.

In this diagram, there are 3 actors, 6 business processes, 2 application elements, and 4 technology elements. Actors include sender, receiver, and key generator. These business processes include requests for key generation, signature generation, key pair generation, decryption signature, hash calculation, and comparison. In the application layer, there is a sender interface and a receiver interface. In the technology layer, there is a sender device, receiver device, and key generator device.

The digital signature process works as follows:
1. Sender requests key generation from key generator.
2. Key generator generates and sends key pair to sender.
3. Sender sends public key to receiver.
4. Sender encrypts hash value of data; sends to receiver.
5. Receiver decrypts encrypted data with public key; makes comparison.

B. Depend-on Relationship in a Digital Signature Process

For this digital signature process, all of the depend-on relationships among the actors are explained as follows:
1. The sender depends on “Generated key is reliable” for the key generator.
2. The receiver depends on “The signature is dependable” for the sender.

In Section III, the model of a relationship for the business layer and assurance case in ArchiMate is introduced. Fig. 7 describes the composite dependability by using the proposed approach based on Fig. 6. Because the depend-on relationship between actors in the business layer is only highlighted, Fig. 7 does not show the elements of Risk, Countermeasure, and Evidence for each assurance case. When the details of an assurance case are required, all elements should be treated similar to the example in Fig. 5. This is done by using the method proposed in Section III, and the elements in the application and technology layers are used as evidence because of the previous proposal [24].

C. Assurance Case in the Application and Technology Layers

Fig. 7 only defines the depend-on relationships; if the cor-
responding evidence for dependability is necessary, according to the model in Section III, the evidence for these depend-on relationships can be treated as follows. In Fig. 8, taking “Signature is dependable” as an example, the realization relationship between Evidence and Goal is clarified.

In the receiver interface, there are 3 functions: signature decryption, hash calculation and hash comparison. If the result of the hash comparison is consistent, the goal of “Signature is dependable” can be achieved.

In a complex system architecture, there may be many actors. To rapidly clarify dependencies among complex relationships, it is necessary to highlight the depend-on relationships among actors in an acceptable manner. Fig. 9 shows the depend-on relationships between business actors based on Fig. 7. For the assurance case in Fig. 8, the evidence for the assurance case is the hash comparison function, but the relationship between the hash comparison function and the receiver actor in the business layer is a serving relation; therefore, it can be said that the dependability of a signature is realized by the receiver actor, as shown in Fig. 9.

V. DISCUSSION

The method for description of the interdependency relationship between business actors was previously illustrated.
The d*framework can also be used to assure an interdependency relationship. In the following section, the d*framework and the proposed approach are compared in detail, and the effectiveness and limitations of the new proposal are discussed.

A. Comparison of ArchiMate and d*framework

In Table 2, the comparisons between the proposed method and the d*framework include actor, dependability claim, relationship, and system configuration. For the actor, only the module node can be used in the d*framework, but in ArchiMate, all of the following (Business Actor, Component, Interface, Device, and Artifact), can be used as an actor. For example, the hash comparison function is regarded as an actor in Fig. 8 and the business actor is regarded as an actor in Fig. 9.

For the dependability claim, the goal represents the dependability claim in d*framework. However, in ArchiMate, the dependability claim can be represented as a requirement, goal, assessment, or driver, as shown in Fig. 3.

For the relationship, d*framework can only use the depend-on relationship to represent the relationship between actors. However, in ArchiMate, the relationships of Realize, Association, Influence, and Serving can be used to represent relationships between actors in a more delicate way.

For the system configuration (because the d*framework does not have the capability to describe system architecture), collaboration diagrams are indispensable. However, because ArchiMate can be used for modeling system architecture, system architecture and dependability relationships can be well visualized.

In addition, in the appendix, an example of a medical system is provided for comparing the number of architecture nodes and the number of relationships, and then explained in the ArchiMate and d*framework, respectively. The results are summarized in Table 3. For this case, the assurance case nodes number in the proposed method is 16 and the number of nodes in the d*framework is 21. The number of assurance case relationships in the proposed method is 17, while it is 20 in d*framework.

In summary, for the description in Actor, Dependability Claim, Relationship, and System Configuration, ArchiMate is superior to d*framework, and ArchiMate is more suitable in the terms of assuring composite dependability. However, the number of relationships and nodes in ArchiMate is less than that of the D-case according to the example in the appendix. This result shows that, to some extent, the proposed method is more concise than the d*framework.

![Fig. 8. An assurance case for the digital signature process in ArchiMate.](image)

![Fig. 9. Depend-on relationships between business actors.](image)

Table 2. Comparison between ArchiMate and d*framework

<table>
<thead>
<tr>
<th>Objects</th>
<th>ArchiMate</th>
<th>d*framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Business Actor, Component, Interface, Device, Artifact</td>
<td>Module</td>
</tr>
<tr>
<td>Dependability Claim</td>
<td>Requirement, Goal, Assessment, Driver</td>
<td>Goal</td>
</tr>
<tr>
<td>Relationship</td>
<td>Realize, Association, Influence, Serving</td>
<td>Depend-on</td>
</tr>
<tr>
<td>System Configuration</td>
<td>Enterprise Architecture</td>
<td>Collaboration diagram</td>
</tr>
</tbody>
</table>
Table 3. Number of nodes and relationships in ArchiMate and d*framework

<table>
<thead>
<tr>
<th></th>
<th>Number of nodes</th>
<th>Number of relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArchiMate</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>D* framework</td>
<td>21</td>
<td>20</td>
</tr>
</tbody>
</table>

**B. Effectiveness**

As previously mentioned, the ArchiMate diagram is effective for dependability assurance. The effectiveness is summarized as follows.

In Section III and Section IV, assurance case methods based on system architecture diagrams were clarified. When compared to the d*framework, ArchiMate is more effective for element diversification. The assurance case can be concretized by relating the elements in the application or technology layer. The assurance case can also be simply abstracted to represent the relationship between the actors in the business layer.

Because actors can be directly defined with system architecture in ArchiMate, the relationships between actors and the relationships between actors and system components are further clarified. In addition, the arguments of the assurance case can be easily defined using the motivational elements and the relations in ArchiMate.

Finally, from the visualization perspective, using ArchiMate to describe dependability relationships has advantages over the traditional methods.

**C. Limitations**

Although the digital signature process was well evaluated in this study, it is necessary to evaluate some other systems to confirm the effectiveness of this proposal. Moreover, because the dependability assurance in the business layer is a depend-on relationship and is different from the description of the application or technology layers, it may be necessary to separately evaluate the method for the business layer.

**VI. CONCLUSIONS**

In this study, an approach was proposed to assure dependability in the business layer. The assurance case in the application and technology layers was also clarified. First, a model of the relationship for the business layer and assurance case was proposed; then, the mapping relationships between the assurance case and actor were defined using ArchiMate based on this model. The presentation of the dependability between actors was also clarified.

A case study of a digital signature process was carried out to explain this approach, and the study showed that the composite dependability for the business, application, and technology layers could be well treated using ArchiMate. Finally, a comparison between ArchiMate and d*framework was conducted by which the effectiveness and superiority of the proposed method were proven by analyzing the actors, dependability claim, relationship, and system configuration. Moreover, in the appendix, a comparison between the proposed method and d*framework regarding the number of nodes and relationships was conducted from a data perspective. The proposed method is more concise; however, whether the proposed method is easier to understand than the d*framework will be determined in future work.

The significance of the proposed method in terms of the enterprise structure, is that it can directly assure the system architecture. However, because the d*framework uses UML, it cannot directly assure models of enterprise architecture.

In future work, it is necessary to apply the proposed method to additional systems to confirm its effectiveness. Because the quality of the newly developed models has not been evaluated, it is also necessary to evaluate their quality. It is also important to determine if others in different fields can understand the dependability of the system through the proposed approach; thus, a survey regarding the versatility of the proposed method will also be undertaken.

**REFERENCES**


Qiang Zhi
Is a doctoral student in the Graduate School of Informatics at Nagoya University, Japan. Previously, he was a software engineer in Japan. He received his master's degree in information science from Hiroshima University in 2016. His research interests mainly include software engineering and information security.

Zhengshu Zhou
Is a master student of the Graduate School of Informatics at Nagoya University, Japan. Previously, he was a software engineer in Japan. He received a bachelor’s degree in software engineering from Tianjin University, Tianjin, China, in 2013. His research interests include software engineering, information security, and system architecture dependability assurance.

Shuichiro Yamamoto
Received a D.Eng. degree from Nagoya University in 2000. He is a professor at the Graduate School of Informatics at Nagoya University. He joined Nippon Telegraph and Telephone Public Corporation (now NTT) in 1979 and was engaged in the development of CASE tools, network-based smart card environments, and distributed application development platforms. He moved to NTT DATA in 2002. He became the first Fellow of the NTT DATA Research and Development Headquarters in 2007. He moved to Nagoya University as a professor in 2009.

Qiang Zhi
Is a doctoral student in the Graduate School of Informatics at Nagoya University, Japan. Previously, he was a software engineer in Japan. He received his master's degree in information science from Hiroshima University in 2016. His research interests mainly include software engineering and information security.

Zhengshu Zhou
Is a master student of the Graduate School of Informatics at Nagoya University, Japan. Previously, he was a software engineer in Japan. He received a bachelor’s degree in software engineering from Tianjin University, Tianjin, China, in 2013. His research interests include software engineering, information security, and system architecture dependability assurance.

Shuichiro Yamamoto
Received a D.Eng. degree from Nagoya University in 2000. He is a professor at the Graduate School of Informatics at Nagoya University. He joined Nippon Telegraph and Telephone Public Corporation (now NTT) in 1979 and was engaged in the development of CASE tools, network-based smart card environments, and distributed application development platforms. He moved to NTT DATA in 2002. He became the first Fellow of the NTT DATA Research and Development Headquarters in 2007. He moved to Nagoya University as a professor in 2009.
APPENDIX

This appendix provides a comparison of the proposed approach and d*framework. Fig. A.1 shows the assurance case and the corresponding system architecture in the proposed approach. This a healthcare system for blood pressure.

Fig. A.1. Assurance case of a healthcare system in ArchiMate.
measurement. The user can measure blood pressure by using this device and the measured data can be sent to a cloud server via mobile phone. The data are received and stored by the cloud server. Here, only data confidentiality is considered for the assurance case of the target system architecture.

In Fig. A.1, the left side is system architecture, and the right side (purple) is the assurance case for the system architecture. It shows that the proposed approach can directly assure the system architecture.

Fig. A.2 shows the framework, in which the definition of the d*framework is as follows.
• Actor is the node type for a system. Actor can have goals, strategies, evidence, and contexts.
• Goal is achieved by Actor. It is represented by node type,
and it can be decomposed into sub-goals or sub-strategies.

- A strategy clarifies how to achieve Goal. A strategy can also be decomposed into sub-goals and sub-strategies.
- Evidence is the node type supporting Goal, such as test reports, specifications, and procedure manuals.
- Context is the external information regarding Goals and Strategies.

Fig. A.3 shows an example of an assurance case in the d*framework. D-Case Editor [26] was developed as a tool for creating D-Case in the DEOS project, and the example shown below was created by the D-Case Editor.