Society for Science and Education
United Kingdom

TMLAI TRANSACTIONS ON
MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

Volume 6 No. 2
ISSN 2054-7390

# A Tool to Create Assurance Case through Models

**[1]Hiroyuki Utsunomiya, [2]Nobuhide Kobayashi, [1]Shuji Morisaki, [1]Shuichiro Yamamoto**
*[1]Nagoya University Graduate School of Information Science, Nagoya, Japan.;*
*[2]DENSO CREATE INC., Nagoya, Japan.;*
utsunomiya.hiroyuki@k.mbox.nagoya-u.ac.jp; nobuhide@dcinc.co.jp;
morisaki@i.nagoya-u.ac.jp; yamamotosui@icts.nagoya-u.ac.jp

## ABSTRACT

In this paper, an assurance case development tool is proposed to derive the argument decomposition structure from generic model definitions. The method solves O-DA issues for assuring business, application, and technology architecture of TOGAF. An example case study using the proposed tool is also shown for the system configuration model of the tool itself.

Discussions based on the case study showed the effectiveness and appropriateness of the proposed methods.

Future work includes the formalization of assurance case derivation process from ArchiMate, UML, and SysML models.

**Keywords:** dependability, architecture models, Enterprise Architecture, experimental tool evaluation, O-DA

## 1 Introduction

The Open Group Real Time & Embedded Systems Forum focuses on standards for high assurance, secure dependable and complete systems. The Open Group announced the publication of the Dependability through Assuredness™ Standard(O-DA) published by The Open Group Real-Time & Embedded Systems Forum[1]. At the heart of this O-DA(Open Dependability through Assuredness) standard, there is the concept of modeling dependencies, building assurance cases, and achieving agreement on accountability in the event of actual or potential failures. Dependability cases are necessary to assure dependable systems[2]. The DEOS process was proposed to manage dependability of complex systems by using dependability cases[3]-[5]. The dependability concept is able to define by quality properties[6].

Complex systems, especially where the boundaries of operation or ownership are unclear, are often subject to change: objectives change, new demands are made, regulations change, business partners are added, etc. So when the failure of the system can have a significant impact on lives, income or reputation, it is critical that a process is in place to identify these changes and to update the architecture by using the assurance cases and the agreements on accountability. It is also critical that a process is in place to detect anomalies or failures, to understand the causes, and to prevent them from impacting the system in the future.

**DOI:** 10.14738/tmlai.62.4428
**Publication Date:** 27th April, 2018
**URL:** http://dx.doi.org/10.14738/tmlai.62.4428

The O-DA standard outlines the criteria for mitigating risk associated with dependability of complex interoperable systems. It also outlines individual accountability. O-DA will benefit organizations relying on complex systems to avoid or mitigate the impact of failure of those systems. O-DA includes the DEOS process mentioned before. The Change Accommodation Cycle and the Failure Response Cycle that together provide a framework for these critical processes. O-DA brings together and builds on The Open Group vision of Boundaryless Information Flow. These concepts include O-DM(Open Dependency Modeling) and Risk Taxonomy of The Open Group Security Forum, and Enterprise Architecture(EA) models of The Open Group ArchiMate® Forum[7],[8]. However, the relationship between O-DA and ArchiMate concepts has not yet been clear. ArchiMate models include strategy, business, application, technology, and physical architecture as well as motivation of architecture. UML[27] only focusses to model software systems. SysML[28] extends UML by adding requirements and parametric diagrams for modeling systems engineering artifacts. Both UML and SysML are not able to model EA.

In this paper, an assurance case generation tool is proposed to argue the assuredness for these three kinds of architectures models. Section 2 describes related work on argument pattern approaches for assurance cases. Section 3 describes an assurance case creation tool which is proposed to generate the argument decomposition structure from various architecture models. In section 4, an example case study using the tool is presented. Discussions on the effectiveness of the tool are shown in section 5. Our conclusions are presented in section 6.

## 2 Related work

The safety case, the assurance case, and the dependability case are currently the focus of considerable attention for the purpose of providing assurance and confidence that systems are safe. Methods have thus been proposed for representing these using Goal Structuring Notation(GSN)[9]-[13]. GSN patterns were originally proposed by Kelly and McDermid[11]. In the absence of any clearly organized guidelines concerning the approach to be taken in decomposing claims using strategies and the decomposition sequence, engineers have often not known how to develop their arguments. It is against this backdrop that the aforementioned approaches to argument decomposition patterns —architecture, functional, attribute, infinite set, complete(set of risks and requirements), monotonic, and concretion—were identified by Bloomfield and Bishop[14]. When applying the architecture decomposition pattern, claims of the system are also satisfied for each constituent part of the system based on system architecture. Despotou and Kelly[15] proposed a modular approach to improving clarity of safety case arguments. Hauge and Stolen[16] described a pattern based safety case approach for the Nuclear Power control domain. Wardzinski[17] proposed safety assurance strategies for the autonomous domain. An experimental result of argument patterns was reported by Yamamoto and Matsuno[18]. Argument pattern catalogue was proposed based on the format of design patterns by Alexander, Kelly, Kurd and McDermid[19]. In their paper, Alexander and others showed a safe argument pattern based on failure mode analysis. Graydon and Kelly[20] observed that argument patterns capture a way to argue about interference management. Ruiz, Habli and Espinoza[21] proposed an assurance case reuse system using a case repository.

Hawkins, Habli, Kolovos, Paige and Kelly proposed a Model-Based Assurance Case development approach by weaving reference information models and GSN argument patterns[22]. They used a script language to define precise weaving procedures. These approaches assume specific adaptation mechanisms to generate assurance cases for reusing GSN patterns.

Although Yamamoto and others proposed the method to create assurance cases based on ArchiMate models[23],[24], the tool to automate the method was not mentioned. This paper proposed a tool based on the method.

## 3 Assurance case creation tool

### 3.1 Overview

The configuration of the tool named as UC2CT(Unified Context to Claim Tool) is shown in Fig.1. The tool reads architecture model, quality property, and risk measure definitions written in XML. UC2CT can decompose claims by using these three types of definitions. The generated assurance cases are represented in the SACM(Structured Assurance Case Metamodel) v1.0 XMI schema definition.

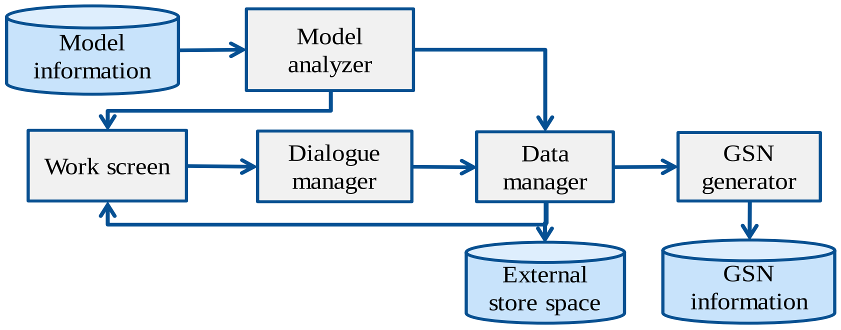
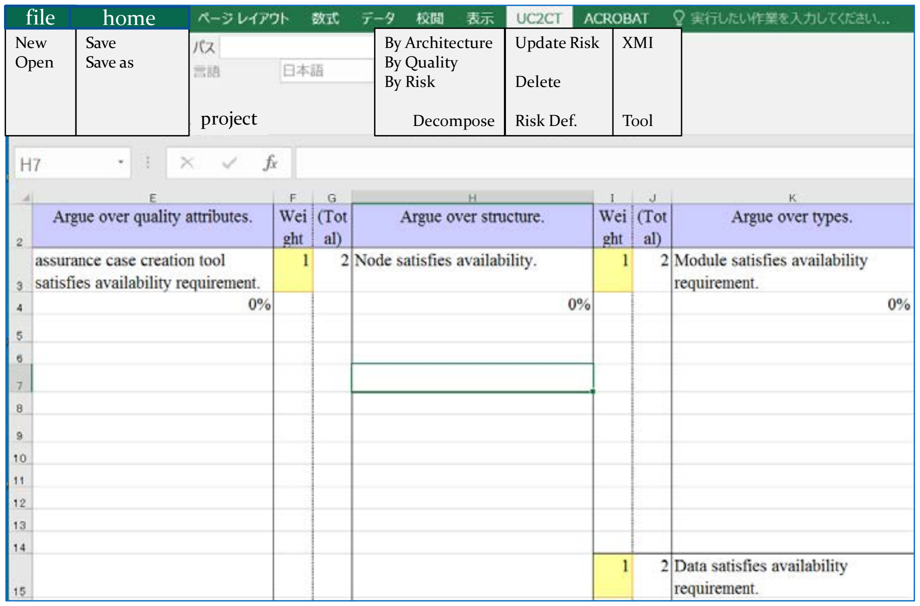


**Figure.1 Configuration of the assurance case tool.**

The generated XMI information is used to develop graphical structures of the assurance cases by using assurance case editors which support the XMI import facility. The tool was implemented by using Excel. These three xml definitions are used as input to decompose a top goal claim in table format. UC2CT is developed by extending the Microsoft Excel. The example screen is shown in Fig.2.



| | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|
| 2 | Argue over quality attributes. | Weight | (Total) | Argue over structure. | Weight | (Total) | Argue over types. |
| 3 | assurance case creation tool satisfies availability requirement. | 1 | 2 | Node satisfies availability. | 1 | 2 | Module satisfies availability requirement. |
| 4 | 0% | | | 0% | | | 0% |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |
| 9 | | | | | | | |
| 10 | | | | | | | |
| 11 | | | | | | | |
| 12 | | | | | | | |
| 13 | | | | | | | |
| 14 | | | | | | | |
| 15 | | | | | 1 | 2 | Data satisfies availability requirement. |

**Figure.2 Display example of the assurance case tool.**

The extended new menu commands are, “New,” “Open,” “Decompose,” “Risk Definition,” and “Tool.” “Decompose” menu consists of “By Architecture,” “By Quality,” “By Risk” sub menus. “Risk Definition” menu consists of “Update” and “Delete” risk sub menus. Tool menu command provides XMI export function to generate the assurance case information developed on UC2CT. As shown in the table, there are weight and total columns in the tool table. These are attributes of nodes and relationships of assurance cases proposed in [25]. The attributes are used to reduce numbers of assurance case nodes and conflicts among quality claims, such as safety and security.

## 3.2 Model definitions

In general, every model is defined by using nodes and their relationships. Therefor models can be defined by the following XML template.

```
-<modelDefinition>
  -<model name=“ModelName”>
    -<types>
      -<nodes>Node Name Definition Part </nodes>
      -<relations>Relation Name Definition Part </relations>
    </types>
    -<instances>
      -<nodes>Node instance definition Part </nodes>
      -<relations>Relation instance definition Part</relations>
    </instances>
  </model>
</modelDefinition>
```

Node Name Definition Part includes a list of the following statement.

```
  <node> Name Of Node </node>
```

Relation Name Definition Part includes a list of the following statement.

```
  <relation> Name Of Relation </relation>
```

Node instance definition Part includes a list of the following statement.

```
  <node id=“Id” type=“NameOfNode”>
    NodeInstanceName
  </node>
```

Relation instance definition Part includes a list of the following statement.

```
  <relation id=“Id” type=“NameOfRelation” target=“Id” source=“Id” />
```

The quality properties and risk measures are also defined by using XML in the same way. The XML notation can universally be applied to describe any models that has nodes and relationships among nodes.

## 3.3 Pattern of created assurance case

The tool is based on the structure of assurance case proposed in [22] as shown in the following table.

**Table 1. Assurance case pattern.**

| Hierarchy | Description |
|---|---|
| Root goal | The root goal states that the model shall satisfy dependability principles |
| Node and relationships | Root goal is decomposed by nodes and relationship of the model |
| Types of nodes and relationships | Second level goals are decomposed by the types of nodes and relationships of the model |
| Instances of nodes and relationships | Third level goals are decomposed by instances of nodes and relationships of the model |
| Risk mitigation for instance risks | Fourth level goals are decomposed by risks for the corresponding instances |
| Evidence | Evidence supports to mitigate all the risks |

The first level sub-goal claims state that concept elements and relationships of the model satisfy dependability principles. The second level sub-goal claim states that category of elements and their relationships among the model satisfy dependability principles. The third level goals are decomposed by instances of concepts and relationships of the models. The fourth level goals are decomposed by risks for the corresponding instances and are supported by the evidence to mitigate risks. Therefore, the fifth level of the assurance case consists of evidences for the fourth level goals.

## 4 Case Study

The example study was conducted to evaluate the effectiveness of the proposed assurance case creation tool for assuring the dependability of the tool itself.

## 4.1 Target system

The target system of the case study is the assurance case creation tool proposed in this paper. The model of the tool was defined described below in the form of the model definition in the previous section. In Fig.1, there are module and data. Therefore, node types in the definition are Module and Data. Module-Module and Module-Data are two types of relationships.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<modelDefinition>
  <model name="assurance case creation tool">
    <types>
      <nodes>
        <node>Module</node>
        <node>Data</node>
```

```xml
        </nodes>
        <relations>
            <relation>Module_Module</relation>
            <relation>Module_Data</relation>
        </relations>
    </types>
    <instances>
        <nodes>
            <node id="in-001" type="Module">Model Analyzer</node>
            <node id="in-002" type="Module">Dialogue manager</node>
            <node id="in-003" type="Module">GSN generator</node>
            <node id="in-004" type="Module">Data manager</node>
            <node id="in-005" type="Module">Work screen</node>
            <node id="in-006" type="Data">Model information</node>
            <node id="in-007" type="Data">External store space</node>
            <node id="in-008" type="Data">GSN information</node>
        </nodes>
        <relations>
            <relation id="ir-011" type="Module_Module" source="in-001" target="in-004" />
            <relation id="ir-012" type="Module_Module" source="in-001" target="in-005" />
            <relation id="ir-013" type="Module_Module" source="in-002" target="in-004" />
            <relation id="ir-014" type="Module_Module" source="in-004" target="in-005" />
            <relation id="ir-015" type="Module_Module" source="in-004" target="in-003" />
            <relation id="ir-016" type="Module_Module" source="in-005" target="in-002" />
            <relation id="ir-017" type="Module_Data" source="in-006" target="in-001" />
            <relation id="ir-018" type="Module_Data" source="in-007" target="in-004" />
            <relation id="ir-019" type="Module_Data" source="in-004" target="in-007" />
            <relation id="ir-020" type="Module_Data" source="in-003" target="in-008" />
        </relations>
    </instances>
</model>
```

```
</modelDefinition>
```

The dependability properties consist of availability, reliability, safety, integrity, consistency, and maintainability are also defined in XML. In addition, risks are defined for each nodes and relations in XML.

The XML model definition is loaded by the tool to create the assurance case based on the model. Then the following XMI information was generated to create the assurance case.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>

<ARM:Argumentation content="" description="" id="assurance case creation tool" xmi:id="38888871" xmlns:ARM=http://schema.omg.org/SACM/1.0/Argumentation xmlns:xmi=http://www.omg.org/XMI xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:version="2.0">

   <argumentElement content="assurance case creation tool satisfies dependability requirement." description="" id="G0" xmi:id="9a9aeeb6-1eb2-4b2f-a07b-1b3797cf389b" toBeSupported="" assumed="" xsi:type="ARM:Claim" />
```

··············· *omitted for the limitation of space* ···············

```
</ARM:Argumentation>
```

Fig. 3 shows the top level view of the created assurance case with a GSN editor by importing the above xmi file.

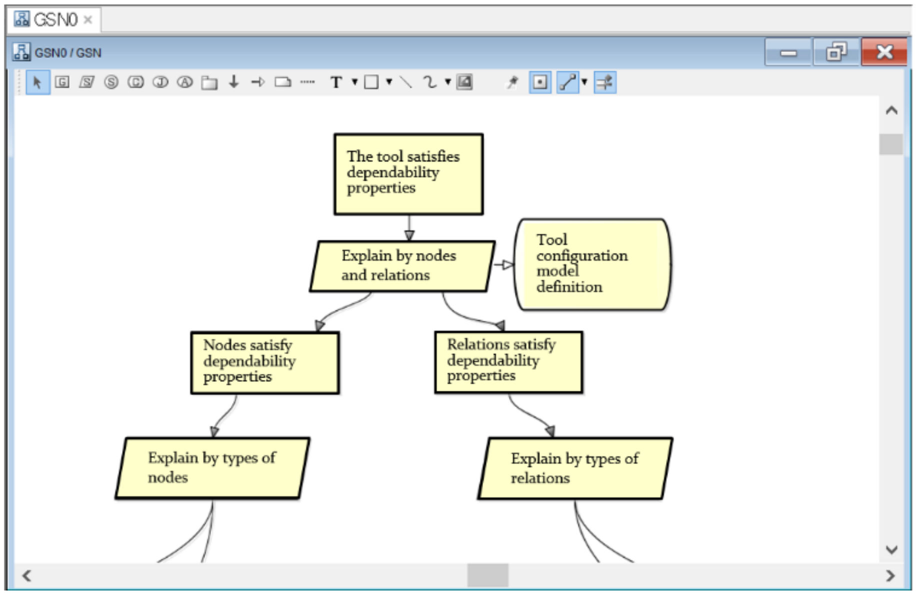

GSN0
GSN0 / GSN
The tool satisfies dependability properties
Explain by nodes and relations
Tool configuration model definition
Nodes satisfy dependability properties
Relations satisfy dependability properties
Explain by types of nodes
Explain by types of relations

**Figure.3 Example of created assurance case.**

The created assurance case consists of 218 claim nodes, 53 strategy nodes, 47 context nodes and 165 evidence. The assurance case for the tool was also developed by human with the same method proposed in [22]. The both human maid and tool made assurance cases have the same nodes.

## 4.2 Comparison of development time

The assurance case development time for using the tool contains the model definition time and the tool operation time. Table.2 shows the comparison for developing the assurance case for the case study and the model checker. The time to create the assurance case for the case study was 275 min. In contrast, by using the tool it was only 47 min. Except for the model definition, it was 19 min. to create the assurance case. Node development productivity was 5.84 sec. per node, because 483 nodes are developed in 47 min.

Model checker is a tool to confirm the validity of software model shown in [26]. It took 110 min. to create an assurance case that confirms the completeness of error handling of the tool. However, with the tool, it was possible to create an assurance case in only 13 minutes, 7.72 sec. per node.

The comparison shoed that the assurance case tool can improve the development time to create assurance case.

**Table 2. Comparison of the assurance case development time.**

| method | Work time of Case Study | Work time of Model Checker |
|---|---|---|
| Without Tool | 275 min. | 110 min. |
| XML definition | 28 min. | 8 min. |
| Tool | 19 min. | 5 min. |

## 5 Discussion

### 5.1 Effectiveness

The case study on the assurance case creation tool was executed to evaluate the effectiveness of the tool proposed. The result showed the derivation from the model of the tool in XML to assurance case is easy and traceable. This showed the effectiveness of the creation method. Although the creation was only described for the tool, it is clear the same results can be derived for other models.

The XML model template is designed so that designers can describe models in the unified manner.

Moreover, if the XML model definitions was generated by modelling tools, the model definition time can be eliminated. For the case study, approximately 93% of the assurance case development time was reduced. This shows the tool has the capability to improve the assurance case productivity largely.

The table size of UC2CT can be extended to the limit of Excel. This shows UC2CT can be used to develop large scale assurance cases. As UC2CT exhaustively decompose assurance cases by architectures and quality properties, it may necessary to reduce the number of nodes. In this case, quantitative attributes are available to reduce unimportant claims by assigning low numbers.

### 5.2 Applicability

The applicability of the assurance creation tool to ArchiMate is clear by the above discussions. The BA, AA, and TA described in ArchiMate models can be easily defined in the form of XML template proposed by this paper. Any architecture models in ArchiMate contain nodes and relationships among nodes. Therefore, the decomposition hierarchy defined by Table 1 can be applied to any models consists of nodes and relationships.

In addition, every graph G can be represented by nodes and relationships among nodes. Nodes and their relationships may have categories. It is necessary to validate every instance of nodes and relationships according to the sort of categories, if we validate the G. Therefore, the proposed approach can be applicable for any models to assure the dependability properties.

## 5.3 Limitation

This paper only examines the effectiveness of the proposed method for two example architecture. More evaluations are necessary to generalize the effectiveness of the proposed tool and the method.

**ACKNOWLEDGMENT**

This work has been conducted as a part of "Research Initiative on Advanced Software Engineering in 2015" supported by Software Reliability Enhancement Center(SEC), Information Technology Promotion Agency Japan(IPA).

## REFERENCES

[1] Real-Time and Embedded Systems, “Dependability through Assuredness™ (O-DA) Framework,” Open Group Standard, 2013.

[2] D. Jackson, "Software for dependable systems– sufficient evidence?," NATIONAL RESEARCH COUNCIL, 2008.

[3] DEOS project, http://www.crest-os.jst.go.jp, 2013.

[4] DEOS project, JST White Paper DEOS-FY2011-WP-03J, www.dependable-os.net/ja/topics/file/White_Paper_V3.0J.pdf , 2011.

[5] M. Tokoro, eds., "Open Systems Dependability, Dependability Engineering for Ever-Changing Systems," CRC Press, 2012.

[6] Avizienis, Laprie, J., Randell, B., Landwehr, C., "Basic concepts and taxonomy of dependable and secure computing," IEEE Transactions on Dependable and Secure Computing, vol.1. No.1, pp.11-33, 2004.

[7] Josely, A., "TOGAF® Version 9.1 A Pocket Guide," 2011.

[8] Josely, "ArchiMate®3.0, A Pocket Guide," The Open Group, Van Haren8 Publishing, 2016.

[9] T. Kelly, "A Six-Step Method for the Development of Goal Structures," York Software Engineering, 1997.

[10] T. Kelly, J. McDermid, "Safety Case Construction and Reuse using Patterns," University of York, 1997.

[11] T. Kelly, "Arguing Safety, a Systematic Approach to Managing Safety Cases," PhD Thesis, Department of Computer Science, University of York, 1998.

[12] J. McDermid, "Software safety: where's the evidence?, " in SCS '01: Proceedings of the Sixth Australian workshop on Safety critical systems and software, pp. 1-6, Darlinghurst, Australia, Australian Computer Society, Inc., 2001.

[13] T. Kelly, and R. Weaver, "The Goal Structuring Notation – A Safety Argument Notation," Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases, 2004.

[14] R. Bloomfield, and P. Bishop, "Safety and Assurance Cases: Past, Present and Possible Future," Safety Critical Systems Symposium, Bristol, UK, 2010.

[15] G. Despotou, and T. Kelly, "Extending the Concept of Safety Cases to Address Dependability," in proceedings of the 22nd International System Safety Conference (ISSC), Providence, RI USA, 2004.

[16] Hauge, and K. Stolen, "A Pattern-Based Method for Safe Control Systems Exemplified within Nuclear Power Production," SAFECOMP 2012, LNCS 7612, pp.13-24, 2012.

[17] Wardzinski, "Safety Assurance Strategies for Autonomous Vehicles, "SAFECOMP 2008, LNCS 5219, pp.277-290, 2008.

[18] S. Yamamoto, and Y. Matsuno, "An evaluation of argument patterns to reduce pitfalls of applying Assurance Case," Assure2013.

[19] R. Alexander, T. Kelly, Z. Kurd, and J. McDermid, "Safety Cases for Advanced Control Software: Safety Case Patterns," Technical report, University of York, 2007.

[20] P. Graydon, and T. Kelly, "Assessing Software Interference Management When Modifying Safety-Related Software," in Proceedings of the Next Generation of System Assurance Approaches for Safety-Critical Systems (SASSUR) Workshop, SAFECOMP 2012, Springer, 2012.

[21] Ruiz, I. Habli, and H. Espinoza, "Towards a Case-Based Reasoning Approach for Safety Assurance Reuse," SAFECOMP 2012 Workshops, LNCS 7613, pp. 22–35, 2012.

[22] R. Hawkins, I. Habli, I., D. Kolovos, R. Paige, and T. Kelly, "Weaving an Assurance Case from Design: A Model-Based Approach," HASE15, pp.110-117, 2015.

[23] S. Yamamoto, "An approach to assure Dependability through ArchiMate," SAFECOMP 2015 Workshops, LNCS 9338, PP.50-61, Assure 2015, DOI: 10.1007/978-3-319-24249-1_5.

[24] Shuichiro Yamamoto and Nobuhide Kobayashi, Mobile Security Assurance through ArchiMate, Vol. 4, No. 3 of IT Convergence Practice, pp.1-8, (INPRA), 2017, http://inpra.yolasite.com/vol4no3.php

[25] Shuichiro Yamamoto, Assuring Security through Attribute GSN, ICITCS 2015, 5th International Conference on IT Convergence and Security (ICITCS), pp.1-5, 2015

[26] Nobuhide Kobayashi, Assurance case development method using SPRME on software review, ER2016, 2016.

[27] OMG, UML, http://www.uml.org/

[28] OMG, SysML, http://www.omgsysml.org/