A methodology for improving reliability of complex systems

— Synthesis of architectural design method and model checking —

Atsushi Katoh ＊, Masataka Urago and Yoshiaki Ohkami

[Translation from *Synthesiology*, Vol.3, No.3, p.197-212 (2010)]

This paper describes a methodology for decomposing a system specification into component specifications and interface specifications whose cooperative behavior is consistent with each component. The methodology is constructed by a bridge method of combining architectural design method in systems engineering standards and model checking, which have already been confirmed to be effective in developing systems. As a trial, the methodology was applied to develop an industrial robot system. The result demonstrates that the proposed methodology is effective for complex industrial systems.

Keywords : Developing methodology, systems engineering, architectural design method, model checking, bridge method, complex systems, reliability

1 Introduction

System is a combination of interacting elements organized to achieve one or more stated purposes[Term 1][1]. Through advances in technology, technological systems (or systems) such as electronic equipment systems[Term 2] or information systems[Term3] have become deeply ingrained in society. On the other hand, the systems are getting more and more complex with the sophistication of required functions and the advent of system of systems[Term 4] where a new system is formed by multiple systems with different purposes. Recently, there are many system failures due to their complexity. As seen in the accident cases of irradiation device[2], explosion of Ariane 5[3], or disruption in air traffic control system[4], the failures of complex systems have drastic influences on society. Improving the reliability of complex systems is an important issue in realizing a safe and secure society.

In the complex system, components of the system are connected and cooperate with each other. For example, in the case of the irregular-rigid-body-transport robot system which is described in chapter 5, the integrated control subsystem understands the surrounding situation based on the results of measurement by the measurement subsystem, and the robot subsystem operates accordingly. This is called cooperative behavior[Term 5] by components in this paper. In detail, processings of the system component cooperate with processings of the other system components through the interface between components in order to achieve the system function. In the complex system, it is important that the cooperative behavior by the components occurs consistently (consistency[Term 6]) according to the system specification. However, due to its complexity, errors may creep into the specifications for the cooperative behavior by the components, and the behavior may not occur consistently (inconsistency) according to the system specification. The cooperative behavior by the system components is generally tested in a system test conducted in the final phase of system development where actual products of the components are combined. In a case where inconsistency of the cooperative behavior is detected in the system test, it is necessary to return to the upstream of the system development and redesign the cooperative behavior by the components. Large amount of cost is required to correct such inconsistency. When redesigning of the cooperative behavior occurs in the final phase of system development, the reliability of the system may be compromised. Although it is necessary to design and verify the cooperative behavior by the system components surely in the upstream of system development, no method has been proposed for this purpose. The first reason is that there has been no attention paid to the cooperative behavior by the system components from the perspective of the reliability of the system. The second reason is that incorporating the quality of the system at the upstream of system development is a relatively new concept. Therefore, we study a methodology for decomposing a system specification into component specifications and interface specifications, and verifying consistency of their cooperative behavior in the system design phase[5][6]. By developing the components based on the specifications where the cooperative behavior is consistent, it is expected to improve the reliability of the complex system. This methodoloy is constructed by synthesizing architectural design method[Term 8] in systems engineering[1][Term 7] and model checking[7][Term 9].

Systems engineering is technological methodologies for achieving systems which satisfy the required quality within a given budget and time period. The research of

Graduate School of System Design and Management, Keio University 4-1-1 Hiyoshi, Kohoku-ku, Yokohama 223-8526, Japan
＊ E-mail : katoh.atsushi@z7.keio.jp

Original manuscript received January 19, 2010, Revisions received June 4, 2010, Accepted June 7, 2010

systems engineering were started mainly in the military and aerospace fields, and systems engineering evolved through the accumulating and reflecting of "best practices" of the system development. The systems engineering process is standardized as know-hows and rules independent of technological fields[8]-[10]. Architectural design method is defined as a part of the systems engineering process. Architectural designing is a method to allocate the functions and performances required of a system to the system components, and to define the specifications of the components and the interface among the components. By architectural designing according to the standardized process, the complex system can be decomposed into its components smoothly and surely. In this paper, standardized architectural design method is simply called "architectural design method".

Model checking is a method to verify whether a given property is valid or invalid in all possible state transitions which can be achieved by the models which represent the state transitions of the system, using a computer exhaustively. Model checking is one of the formal methods[11] Term 10. Model checking is already established as a verification method, and nowadays is popular in software development. According to the functional safety standard IEC 61508[12] Term 11, applying the formal method is recommended for the system development, and it is gaining attention as a method for achieving the high reliability of the system. Whether the properties which must be satisfied by the cooperative behavior is valid or not is thoroughly verified by applying model checking to the specifications for the cooperative behavior between components. As a result, it is possible to detect inconsistency of the cooperative behavior which may occur in the complex states.

Architectural design method is systematic knowledge which is formed by collecting best practices in the system design fields based on systems engineering. Model checking is a research result which improves the reliability in the system verification field, based on mathematical logic and computer science. In this research, we aim to achieve the high reliability in the complex systems, synthesizing architectural design method and model checking, and develop a methodology which utilizes the characteristic of both methods. Our research corresponds to *Type 2 Basic Research* which widely selects the knowledge of different technological fields and synthesizes them to satisfy social and economic needs.

This paper describes a methodology for decomposing a system specification into component specifications and interface specifications among components whose cooperative behavior is consistent with each component. It also describes the research process of this methodology. It is structured as follows. Chapter 2 describes the research goal and the research scenario. Chapter 3 describes architectural design method and model checking. Chapter 4 describes the synthesis process of architectural design method and model checking. Chapter 5 describes the application of an industrial use. Chapter 6 discusses the effectiveness and issues of this methodology. Chapter 7 summarizes this paper and describes the future work.

2 Research objective and research scenario

The objective of this research is to establish a methodology for decomposing a system specification into component specifications and interface specifications among components whose cooperating behavior is consistent with each component, which is not specific to particular technological systems. Figure 1 shows the research scenario. For the research scenario to achieve the research objective, the methods whose effectiveness has been fully verified are selected among the technological fields related to the system development. The reason for this is that a high-quality methodology can be established efficiently by employing methods which are already recognized as being effective for the system development. The methodology is established by synthesizing the selected methods to maximize their characteristics. The reason for this is that there is a possibility to produce a new research or technological field through developing a new technology by the synthesis of methods from different researches or technological fields. Also, the effectiveness of methodology is evaluated by applying this methodology to an actual case in industry. There are two reasons for selecting the industrial case as the application. The first reason is that in order to evaluate the practical applicability of this methodology in industry, it is necessary to take a functionally complex case as the application to consider safety, rather than a mere sample. The second reason is that by propagating the effectiveness of this methodology to industry, it may be possible to bridge the gap between the research activities and the social contributions of the research results, or the so-called valley of death.

3 Selection of methods

In establishing the methodology in this research, the functions which must be satisfied by the methodology are

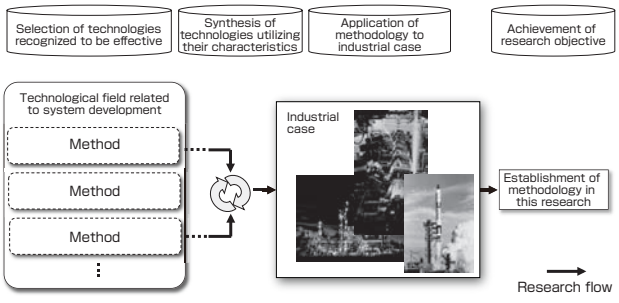


Fig. 1 Research scenario

divided as follows:

a. The function for decomposing a system specification into component specifications and interface specifications among components;
b. The function for verifying whether the cooperative behavior of the component specifications and interface specifications among components are consistent.

The system design mehod which satisfies the function "a" is selected among the system developing methods. In general, system designing is a work for defining a system specification by analyzing the user's needs, and defining specifications for functions of the components which compose the system, realization means of the components, and relationship among the components, based on the system specification. The representative method of system designing other than architectural design method includes structured analysis and structured design (SA/SD) method[13] Term 12. SA/SD method is a design method where a system is decomposed into components by focusing on data flows of the system. In SA/SD method, the system is designed by focusing on the data such as business information rather than the functions and processings, because the data is stable against changes in requirements or a technological evolution. This allows to construct systems with maintainability and expandability. However, since SA/SD method is developed primarily for technological systems such as information systems, it does not deal with control flows or processing timing[14]. Therefore, it is inappropriate for designing anything other than information systems such as embedded systems. On the other hand, architectural design method requires more efforts compared to the specific design method such as focusing on the data as in the aforementioned example, because the procedures and tasks specific to a certain designing are not defined. However, architectural design method is a general design method which is not dependent on some specific technological systems where the process for defining functions and realization means of the system are defined. Therefore, taking into account the research objective of achieving a methodology which is not specific to particular technological systems, we select the architectural design method as the system design method which satisfies function "a". Also, the representative systems engineering standards which defines architectural design method include ISO 15288[9] Term 13, ANSI/EIA 632[10] Term 14, and IEEE 1220[11] Term 15. While ISO 15288 can be applied to the entire system lifecycle process from the conceptualizing phase to the dismantling phase, the tasks and procedures of architectural designing are not finely defined. While ANSI/EIA 632 can be applied widely to the system lifecycle process from the conceptualizing phase to the transition to operation phase, the tasks and procedures for architectural designing are not finely defined. On the other hand, although IEEE 1220 limits the range of application from the system requirement analysis phase to the system test phase, the tasks and procedures for architectural designing are finely defined. Therefore, we select architectural design method defined by IEEE 1220 for our methodology.

The system verification method which satisfies function "b". is selected among the system development methods. In general, system verification is a work for verifying whether a developed system satisfies the system specification or not. The representative system verification methods other than model checking include test method Term 16 and simulation method[15] Term 17. Test method is a verification method for verifying behavior of actual products against the test cases. While it can verify the actual behavior of actual products, it is difficult to extract all of the cases which may occur and to verify the behavior in all possible cases. Simulation method is a verification method where a target to be verified and peripheral environment of the target is simulated as models on a computer, and behavior of the models is verified against the test cases. While it can verify the behavior of the target in the early phase of system development when actual products and peripheral environment do not exist, it is difficult to extract all of the cases which may occur and to verify the behavior in all possible cases, as in the test method. On the other hand, although model checking can only verify state transitions of a verification target, it can verify whether the properties to be satisfied are valid or not for all state transitions exhaustively. If there is a deadlock Term 18 in state transitions of a system, fatal accidents may occur during the system operation. Therefore, we select model checking for our methodology.

Next, architectural design method defined in IEEE 1220 and model checking are described in detail.

3.1 Architectural design method in IEEE 1220

Figure 2 shows the architectural design process. Architectural designing is composed of functional designing Term 19 and physical designing Term 20. Functional designing is a work where functions defined as a system specification are decomposed and refined, and performances defined as the system specification are allocated to the decomposed and refined functions. Physical designing is a work where system components are specified, and the functions and performances decomposed and refined in functional designing are allocated to the components. The outputs of architectural designing are component specifications and interface specifications among components.

Figure 3 shows the process of functional designing defined in IEEE 1220. The process of functional designing is defined in IEEE 1220 chapter 6 section 3 Functional analysis Term 21. Figure 4 shows the process of physical designing defined in IEEE 1220. The process of physical designing is defined in IEEE 1220 chapter 6 section 5 Synthesis Term 22. By conducting the tasks according to the numbers in Figs. 3 and 4, it is

possible to decompose a complex system into its components smoothly and surely. Architectural design method in IEEE 1220 has been used in various industrial fields, and has produced results. Therefore, a certain level of effectiveness is guaranteed[16].

3.2 Model checking

Figure 5 shows the process of model checking. The process of model checking can be categorized into four works: developing models, developing fomulae, conducting model checking, and analizing the model checking results. First, state transitions of a target to be verified are modeled based on the target specification according to the expression form of a model checking tool to be applied. Next, properties which must be satisfied by the verification target are considered. Formulae which express the properties are developed according to the expression form of the model checking tool. Then, the models and the formulae are input to the model checking tool on a computer, and model checking is conducted. Model checking verifies whether the models satisfy the properties expressed by formulae or not in all state transitions achievable by the models exhaustively. Finally, results of whether the models satisfy the properties expressed by formulae or not are analyzed based on outputs from the model checking tool. If the models satisfy the formulae, it means that the specification based on the models satisfy the properties. If the models do not satisfy the formulae, the state transitions of the models up to the state where the property is not valid are output as the counterexamples, If no errors are found in the models

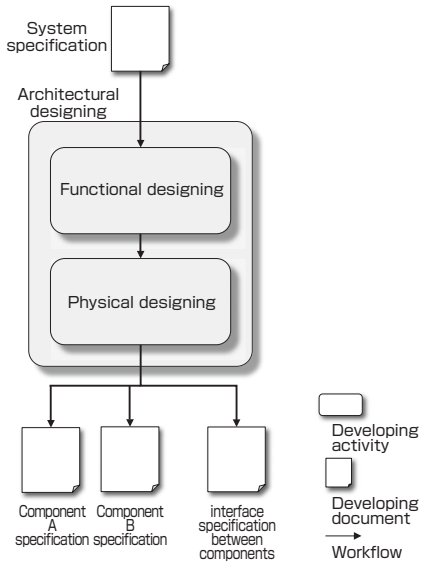


Fig. 2 Process of architectural designing

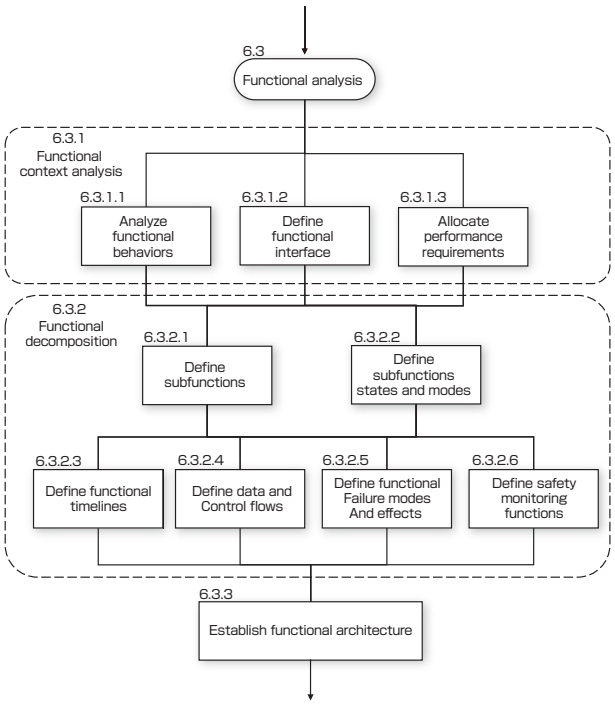


Fig. 3 Process of functional designing in IEEE 1220[11]

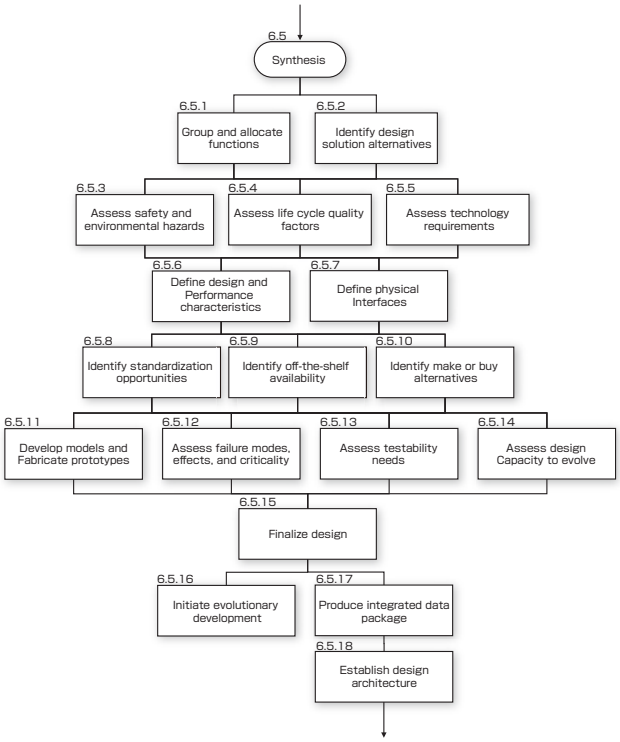


Fig. 4 Process of physical designing in IEEE 1220[11]

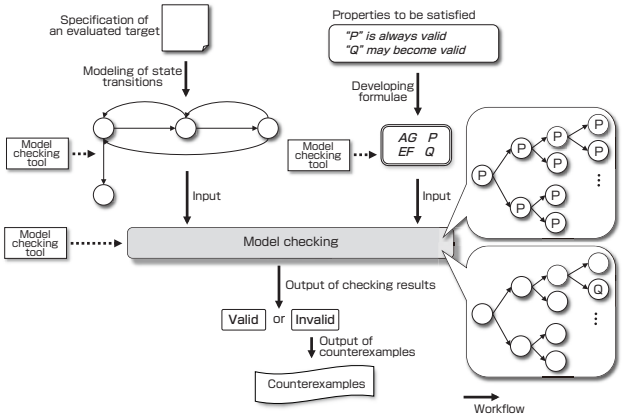


Fig. 5 Process of Model checking

when the counterexamples are analyzed, it means that the specification of the target based on the models has flaws.

The following two effects can be expected by applying model checking. First, it may be possible to reduce the cost of detecting flaws in the specification by using the counterexamples. If results of test method and simulation method come out incorrect, it is necessary to analyze the cause by hypothesizing the many causes which may lead to incorrectness. Much effort may be necessary to identify the real cause. By applying model checking, it is possible to trace the occurrence of incorrectness using the counterexamples which are output from the model checking tool automatically. Therefore, it is possible to identify the cause of incorrectness efficiently. Second, there is a possibility for detecting flaws in the specification to be verified through modeling, or the formalization of specification, when model checking is conducted.

4 Synthesis of methods

Architectural design method and model checking are synthesized while taking advantage of the characteristics of each method, to construct the methodology for this research. In this chapter, the process of synthesizing architectural design method and model checking is shown by describing workflows in this methodology.

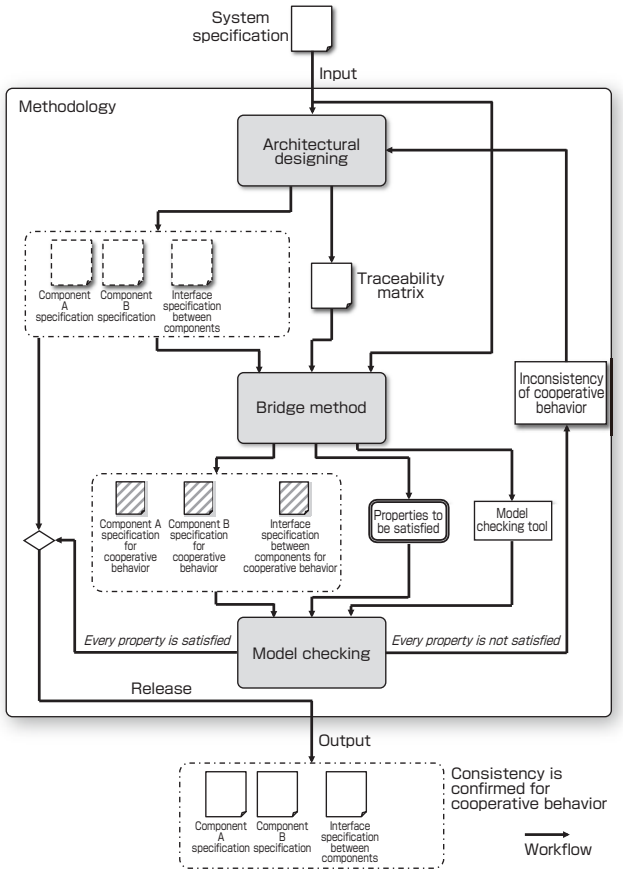


Fig. 6 Methodology in this research

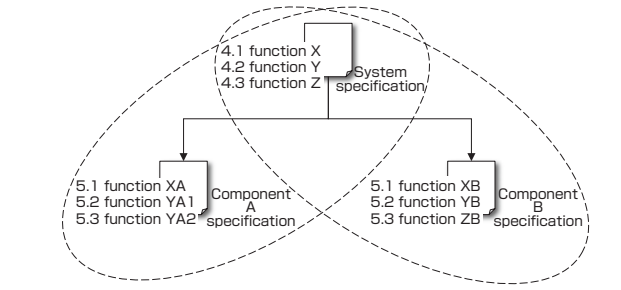
4.1 Proposed methodology

Figure 6 shows the methodology proposed for this research, whereby architectural design method and model checking are synthesized. This methodology is composed of architectural design method, model checking, and bridge method[Term23] which connects two methods.

First, a system specification is input to this methodology. Based on the system specification, architectural designing including functional and physical designing are done according to IEEE 1220. By architectural designing, the system specification is decomposed into component specifications and interface specifications among the components (dashed-line specifications in upper left in Fig. 6). A traceability matrix[Term 24] defined in IEEE 1220 is developed in the process of architectural designing. Figure 7 shows the traceability matrix. An identification number is assigned to each specification for the system specification and the component specifications. The traceability matrix summarizes the correspondence between the system specification and component specifications which are broken down from the system specification.

Next, bridge method developed in this research is applied to the component specifications, the interface specifications among components, the traceability matrix, and the system specification. By applying the bridge method, specifications related to the cooperative behavior are extracted from the component specifications and interface specifications among components (striped specifications in lower left of Fig. 6). Properties which must be satisfied by the cooperative behavior are derived. A model checking tool which is applied in the methodology is selected. Outputs of the bridge method are necessary inputs for conducting model checking.

Specifications related to the cooperative behavior extracted from the component specifications and interface specifications



Traceability matrix for system specification and component A specification

System specification	Component A specification
4.1 function X	5.1 function XA
4.2 function Y	5.2 function YA1
	5.3 function YA2
4.3 function Z	N/A

Traceability matrix for system specification and component B specification

System specification	Component B specification
4.1 function X	5.1 function XB
4.2 function Y	5.2 function YB
4.3 function Z	5.3 function ZB

Fig. 7 Traceability matrix

among components are modeled according to the expression form of the model checking tool to be applied. Also, formulae are developed from the properties which must be satisfied by the cooperative behavior, based on the expression form of the model checking tool. The developed models and the formulae are input to the model checking tool. If counterexamples are output from the model checking tool, the counterexamples are analyzed and the inconsistency of the cooperative behavior is fed back to architectural designing. Based on the inconsistency of the cooperative behavior which is fed back, architectural designing is done according to the workflows of this methodology. If no counterexamples are output from the model checking tool, the component specifications and the interface specifications among components whose cooperative behavior is consistent are released, and they are output as results of this methodology.

4.2 Bridge method

In the methodology of this research, system verification is conducted for the cooperative behavior by components in the system design phase. It is necessary to connect the outputs of architectural designing to the inputs of model checking seamlessly. We develop the method to derive the specifications related to the cooperative behavior, the properties which must be satisfied by the cooperative behavior, and the model checking tool to be applied, based on the component specifications, the interface specifications among components, the traceability matrix, and the system specification. This is called bridge method because it serves as the bridge between architectural design method and model checking. Bridge method is novel since it focuses on the cooperative behavior to present a specific means for synthesizing the systems engineering standard such as IEEE 1220 and model checking. Figure 8 shows the bridge method for architectural design method and model checking. Figure 8 corresponds to the details of the bridge method shown in the central part of Fig. 6. The inputs of the bridge method include the component specifications, the interface specifications among components, the traceability matrix, and the system specification.

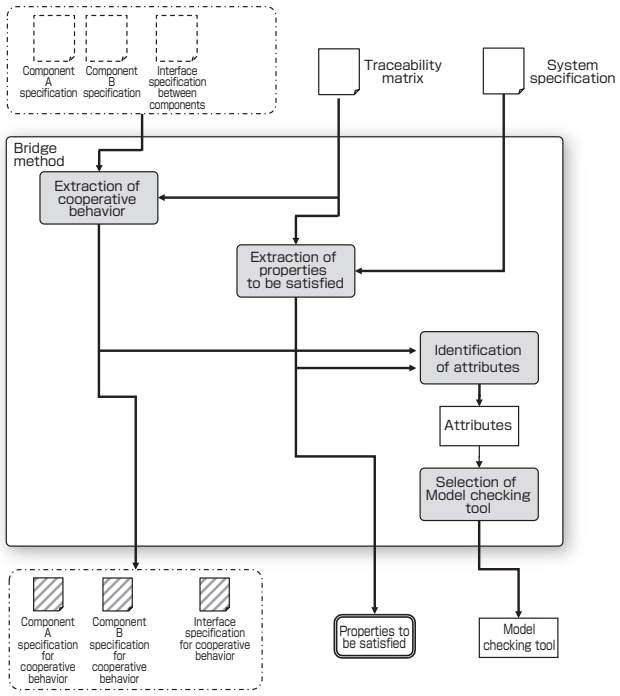


Fig. 8 Bridge method for architectural design method and Model checking

First, specifications related to the cooperative behavior are extracted from the component specifications and interface specifications among components. Figure 9 shows specifications related to the cooperative behavior in the component specifications and interface specifications among components. The traceability matrix is used when specifications related to the cooperative behavior by the component specifications are extracted. If the system specification corresponds to multiple component specifications in the traceability matrix, a function of the system is achieved when these components cooperate. In the case of Fig. 7, the 5.1 function XA of the component A specification and the 5.1 function XB of the component B specification cooperate to achieve the 4.1 function X of the system specification. In Fig. 9, this corresponds to the striped part of the component A specification and the component B specification. The specifications related to the cooperative behavior in interface specifications among components are extracted based on interface information within the component specification related to the cooperative behavior. In Fig. 9, this corresponds to the part of the message in the center.

Next, the properties which must be satisfied by the cooperative behavior of the components are extracted from the system specification based on the traceability matrix. The reason is that it is necessary for the cooperative behavior by the components extracted by the traceability matrix to satisfy the system specification achieved by the cooperative behavior. In case of Fig. 7, the cooperative behavior of the 5.1 function XA of the component A specification and the 5.1 function XB of the component B specification must satisfy the properties of the 4.1 function X of the system specification.

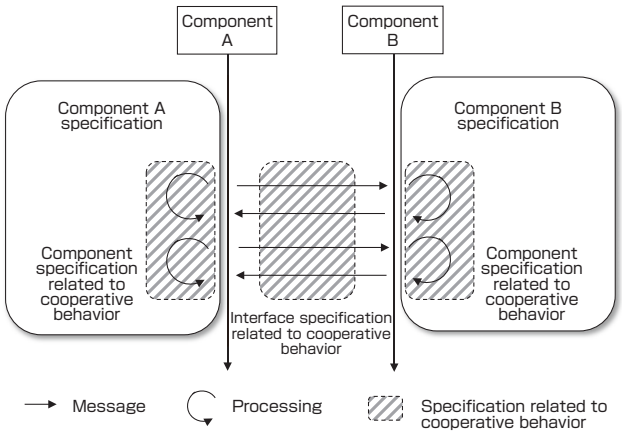


Fig. 9 Specifications related to cooperative behavior

The attributes of the cooperative behavior are determined based on the specifications related to the cooperative behavior and the properties which must be satisfied by the cooperative behavior. The attributes of the cooperative behavior can be categorized as follows:

(1) There must be no lacks and variances in messages which are sent or recieved among components and processings related to the messages;
(2) The timing of messages which are sent or recieved among components and processings related to the messages are correct.

The model checking tool to be applied is selected according to the identified attributes. The representative types of model checking tools are finite automaton[17] Term 25 and timed automaton[18] Term 26 which is extended based on finite automaton. When verifying point (1), the model checking tool which corresponds to finite automaton is selected. SPIN[19] Term 27 is one of the representative model checking tools for finite automaton. When temporal limitations such as the timing are verified as in point (2), the model checking tool which corresponds to timed automaton is selected. Timed automaton is an extension of finite automaton. The model checking tool based on timed automaton can also verifiy point (1). UPPAAL[20] Term 28 is one of the representative model checking tools for timed automaton. Also, each component in the system behaves in parallel. Therefore, it is necessary to select the model checking tool which can model parallel systems. SPIN and UPPAAL can model the parallel systems.

5 Application to industrial case

There is recently a rapid advancement in functions of industrial robots[21] Term 29. Many industrial robots have a strong mechanical output due to the nature of their works; therefore safety of operators must be considered. The industrial robot is one of the systems which are appropriate for the application of this methodology. As of writing this paper, we are developing an industrial robot system for transporting irregularly shaped rigid-bodies, jointly with a manufacturer of industrial robots. We select the irregular-rigid-body-transport robot system as an industrial case study, and apply our methodology.

Followings are descriptions of the irregular-rigid-body-transport robot system and results of applying this methodology.

5.1 Irregular-rigid-body-transport robot system

The irregular-rigid-body-transport robot system is an industrial robot system which engages in grasping, transporting, and placing of heavy rigid-bodies with irregular shapes and sizes. The characteristic of requirements for the irregular-rigid-body-transport robot system is that the system must have a strong autonomy in grasping and placing the irregular rigid-body. Although the area of grasping the irregular rigid-body is limited, the shape and size of the rigid-body, the position where the irregular rigid-body is grasped, and the direction of the irregular rigid-body are indefinite. The system must accurately determine the shape, size, and direction of the irregular rigid-body. Also, while the area in which the irregular rigid-body is placed is limited, the location in which the irregular rigid-body is placed within that area is indefinite. The system must accurately determine the location where other irregular rigid-bodies are not present, or the location with the lowest height in that area which is laid with other irregular rigid-bodies.

In developing the irregular-rigid-body-transport robot system, system requirement analysis is conducted based on the system needs. The system specification is defined through system requirement analysis. This methodology is applied with the defined system specification as an input.

5.2 Application of this methodology

In this section, the specific applications of this methodology are described for architectural design method, bridge method, and model checking mentioned in chapter 4.

5.2.1 Architectural design method

Architectural designing is done using the system specification of the irregular-rigid-body-transport robot system as an input. By architectural designing, the specification of the irregular-rigid-body-transport robot system is decomposed into the specifications of the measurement subsystem, the robot subsystem, and the integrated control subsystem, as well as the interface specifications among the subsystems[Term 30]. Figure 10 shows the results of architectural designing for the irregular-rigid-body-transport robot system. In architectural designing, each subsystem is designed by assuming the subsystem components[Term 31] which compose the subsystem to make the most of COTS (commercial off the shelf)[Term 32] products and existing technologies.

The measurement subsystem is composed of a laser scanner to measure a three-dimensional shape, a vertical motion mechanism for the laser scanner, and a measurement control computer which controls them. The measurement subsystem measures the shape, size, position, and direction of the irregular rigid-body when grasping it. It also measures the unevenness of the area where the rigid-body is placed.

The robot subsystem is composed of a robot arm, a robot hand, and controllers which control each subsystem component. It also has a teaching pendant[Term 33] for programming actions and emergency stop of the robot arm. The robot subsystem grasps, transports, places irregular rigid-bodies.

The integrated control subsystem is composed of an

integrated control computer which controls the measurement and robot subsystems and a console[Term 34] to input work instructions and to check system status. The integrated control subsystem controls the robot subsystem based on the results of measurements by the measurement subsystem.

Also, the traceability matrixes for the measurement, robot, and integrated control subsystems are developed for architectural designing.

5.2.2 Bridge method

Bridge method is applied to the subsystem specifications, the interface specifications among the subsystems, the traceability matrix, and the system specification for the irregular-rigid-body-transport robot system. Here, we discuss the measurement and integrated control subsystems to describe the specific application of the bridge method.

First, the specifications related to the cooperative behavior are extracted from the subsystem specifications and the interface specification between subsystems based on the traceability matrix. The specific extraction of the specifications related to the cooperative behavior follows the means described in subchapter 4.2. For the measurement subsystem specification, six items are extracted from 39 specification items, and for the integrated control subsystem, six items are extracted from 78 specification items. For the interface specification between the measurement and integrated control subsystems, 22 items are extracted from 26 specification items.

Next, the properties which must be satisfied by the cooperative behavior of the subsystems are extracted based on the traceability matrix and the system specification of the irregular-rigid-body-transport robot system. The specific extraction of the properties which must be satisfied by the cooperative behavior follows the means described in subchapter 4.2. For the measurement and integrated control subsystems, 23 items are extracted as the properties which must be satisfied by the cooperative behavior. Table 1 shows two items from the 23 properties extracted.

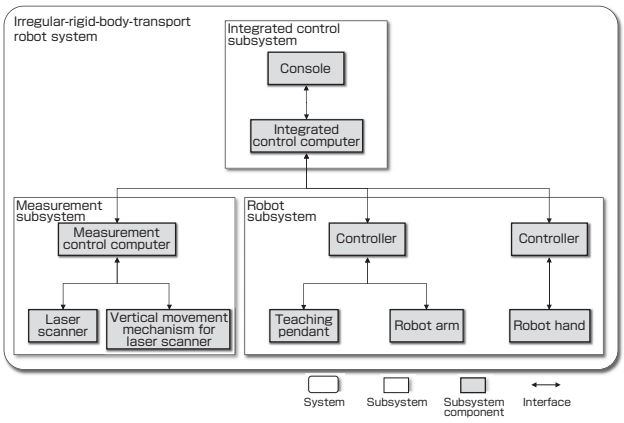


Fig. 10 Results of architectural designing for the irregular rigid-body transport robot system

Table 1 Properties to be satisfied for the cooperative behavior by the measurement and integrated control subsystems (2 out of 23 items)

No.	Property
1-5	The integrated control subsystem must receive the result of rigid-body measurement or the failure response of rigid-body measurement from the measurement subsystem, when the rigid-body measurement request is sent to the measurement subsystem.
3-3	The measurement subsystem must stop the measurement processing within 100 ms after the measurement stop request is transmitted by the integrated control subsystem.

The attributes of the cooperative behavior are determined based on the extracted specifications related to the cooperative behavior and the properties which must be satisfied by the cooperative behavior. In the cooperative behavior of the measurement and integrated control subsystems, the lacks and variances of the messages which are sent or received between subsystems and processings related to the messages are suspected. There are specifications for the timing of the messages which are sent or received between subsystems and processings related to the messages as well as the temporal limitation within 100 ms. Therefore, the two attributes, (1) and (2), as shown in subchapter 4.2, are determined.

Also the model checking tool to be applied is selected based on the identified attributes. For the cooperative behavior by the measurement and integrated control subsystems, it is necessary to verify the temporal limitation and the timing of messages which are sent or received between subsystems and processing related to the messages. Since the measurement and integrated control subsystems behave in parallel, it is necessary to select the model checking tool which can deal with the parallel systems. Therefore, UPPAAL is selected as the model checking tool which satisfies these requirements.

5.2.3 Model checking

As in the previous section, the measurement and integrated control subsystems are discussed in this section to describe the specific application of model checking.

First, the specifications related to the cooperative behavior extracted by the bridge method are modeled according to the expression form of the model checking tool. Figures 11 to 13 show the results of modeling the specifications for the cooperative behavior by the measurement and integrated control subsystems using UPPAAL. The developed model is composed of three models for the specifications related to cooperative behavior: the model corresponding to the measurement subsystem (Fig. 11); the model corresponding to the integrated control subsystem (Fig. 12); and the model corresponding to the interface between the measurement and integrated control subsystems (Fig. 13).

Next, the formulae for model checking are developed based

Table 2 Formulae of the cooperative behavior for the measurement and integrated control subsystems (2 out of 23 cases)

No.	Formula
1-5	A[] (bing_syscont_sendreq == bing_req_rod) imply (bing_syscont_req == bing_req_rod)
3-3	A[] P_BING_SCAN.BING_SCAN_REQ_STOP imply (bing_stopreq_time <= 10)

on the properties which must be satisfied by the cooperative behavior. For the measurement and integrated control subsystems, the formulae for 23 cases are developed based on the properties which must be satisfied by the cooperative behavior according to UPPAAL expression form. Table 2 shows the formulae corresponding to the two items shown in Table 1.

Model checking is conducted based on the developed models and the formulae. For the measurement and integrated control subsystems, model checking using UPPAAL is conducted based on the cooperative behavior models shown in Figs. 11 to 13 and the formulae for the 23 cases including the two cases shown in Table 2. The results of model checking are analyzed. For the measurement and integrated control subsystems, the results show that the models do not satisfy some formulae. When the counterexamples output by UPPAAL are analyzed, six cases of inconsistency of the cooperative behavior are detected including the results of the formulae shown in Table 2. Table 3 shows the results of model checking corresponding to the two cases in Table 2.

Inconsistency of the cooperative behavior is fed back to architectural designing. For the measurement and integrated control subsystems, architectural designing is done again, based on the six cases of inconsistent cooperative behavior

Table 3 Model checking results of the cooperative behavior for the measurement and integrated control subsystems (2 out of 6 cases)

No.	Model checking result
1-5	Depending on the messages which are sent or recieved between subsystems, or the timing of processings related to the messages, the response for rigid-body measurement request/the measurement request for placement area/the measurement stop request issued before n-th time may be returned to the rigid-body measurement request issued on the n-th time.
3-3	The measurement processing may not stop within 100 ms from receiving the measurement stop request. Also, the measurement subsystem itself may stop by receiving the measurement stop request.

including the results of Table 3. As a result of re-architectural designing, the measurement subsystem specification, the integrated control subsystem specification, and the interface specification between the measurement and integrated control subsystems whose inconsistency of the cooperative behavior is corrected, are developed.

5.3 Application results

The methodology in this research is applied to the system specification of the irregular-rigid-body-transport robot system. As a result, the measurement subsystem specification, the integrated control subsystem specification, the robot subsystem specification and the interface specifications among the subsystems are derived from the system specification of the irregular-rigid-body-transport robot system. It is also possible to detect inconsistency of the cooperative behavior, as shown in Table 3, from the measurement and integrated control subsystems before fixing these specifications. Later, it is possible to develop the subsystem specifications and interface specifications between subsystems whose cooperative behavior is consistent. We are able to apply this methodology to the

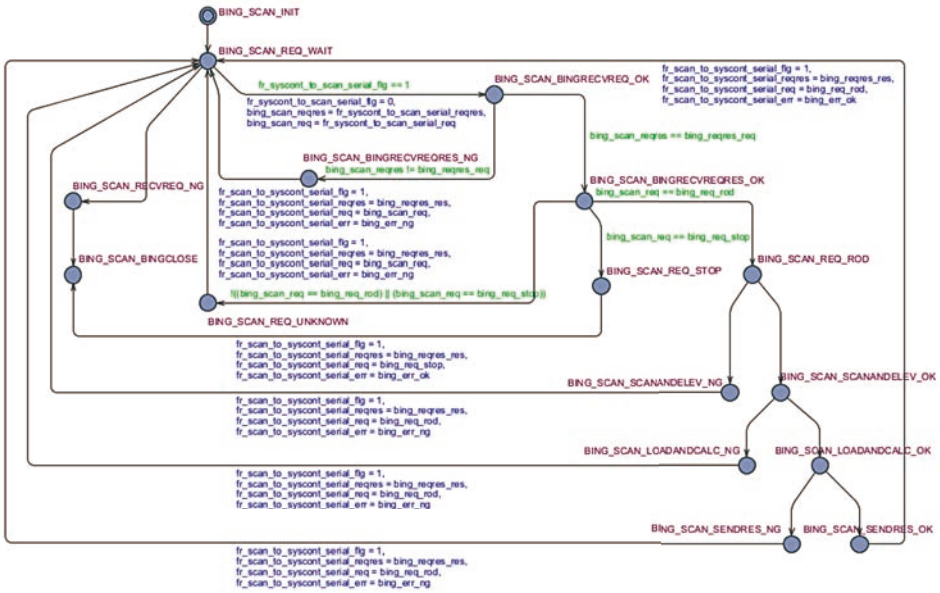


Fig. 11 Cooperative behavior model for the measurement subsystem

actual product development of this industrial robot, and contribute in increasing the reliability of it. It is reconfirmed that this research corresponds to *Type 2 Basic Research*. As of writing this paper, the irregular-rigid-body-transport robot system is being developed based on the specifications whose cooperative behavior is consistent to meet the needs of industry.

6 Discussion

In this chapter, the effectiveness of this methodology is demonstrated. Issues of this methodology are also described.

6.1 Effectiveness of this methodology

The effectiveness of the methodology in this research is discussed based on the application results to the industrial case shown in chapter 5. In demonstrating the effectiveness, we focus on the QCD (quality, cost and delivery)[Term 35] of the case study.

First, this methodology is considered from the perspective of the quality of a development target. The methodology is applied to the functionally complex industrial case of the irregular-rigid-body-transport robot system. As a result, it is possible to develop the measurement and integrated control subsystems specifications whose cooperative behavior is consistent by feeding back inconsistency of the cooperative behavior by the measurement and integrated control subsystems to architectural designing in one cycle. Particularly, inconsistencies of the cooperative behavior shown in Table 3 are flaws which are difficult to detect by human beings only, but it is possible to detect them because of model checking where the validity of the properties to be satisfied is verified exhaustively. The fact that such flaws are detected in the early phase of system development demonstrates the effectiveness of this methodology.

Table 4 Man-hours required to apply the methodology to industrial case

Work	Man-hour
Architectural designing	189
Bridge method	5※
Model checking	12※

※Values in the measurement and integrated control subsystems

Next, this methodology is considered from the perspective of the development cost and delivery time. Table 4 shows the man-hours for applying this methodology to the industrial case. For bridge method and model checking, the values for the measurement and integrated control subsystems described in chapter 5 are shown. For architectural designing, 189 man-hours are required. For system development, there are many previous researches on the effectiveness of applying systems engineering method including architectural designing[22]-[25]. In the previous researches, it was shown that it was possible to shorten the cost and delivery time

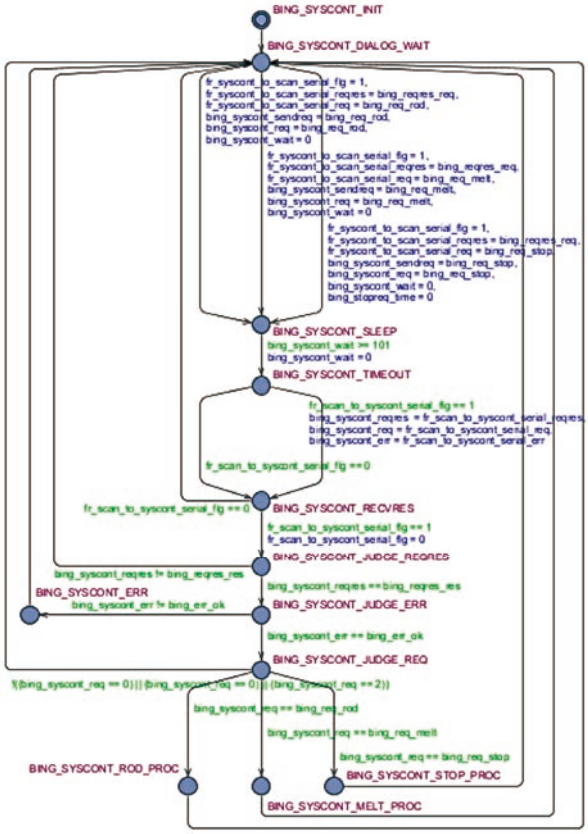


Fig. 12 Cooperative behavior model for the integrated control subsystem

of the system development by appropriately applying the systems engineering methods. In this paper, it is not possible to present the degree of reducing the cost and delivery time by the application of architectural design method in the industrial case. However, considering the fact that certain results are obtained for architectural designing, there is a high possibility that the cost and delivery time can be reduced for the system development. For bridge method and model checking, as shown in Table 4, 5 man-hours and 12 man-hours are required for the measurement subsystem and the integrated control subsystem, respectively. Here, we discuss inconsistency of the cooperative behavior detected by conducting model checking through the bridge method. Inconsistency of the cooperative behavior between subsystems can generally be detected in the system test where the subsystems are combined, conducted at the final phase of system development. If inconsistency of the cooperative behavior among subsystems is detected in the system test, large amount of cost and time are required for correcting. Boehm, in Reference[26], analyzed that if the cost of detecting and correcting the requirement flaws at the phase of defining requirement specifications were set as 1, the cost would be 2 in a small-scale system if the requirement flaws were detected in the test, and the cost would be 20 in a large-scale system. Inconsistencies of the cooperative behavior shown in Table 3 are flaws which are highly likely to be missed unless model checking is applied in the phase when the required specifications are defined. Considering the whole system development, the man-hours required for bridge method and model checking are highly cost-effective.

6.2 Applicability of this methodology

The methodology in this research is not specialized to the development of particular technological systems, but it can be applied universally to the development of any technological system. The reason is that in designing a system, achieving a function of the system by finding components which compose the system based on the system specification and having the components cooperate is a common concept for all technological systems. Also, this methodology can deal with unique problems related to the cooperative behavior in an applicable system. The reason is that it has the bridge method where attributes and model checking tools are selected according to the characteristics of the cooperative behavior.

However, attentions must be paid in the application of this methodology. This methodology employs model checking to verify whether the cooperative behavior by the components is consistent. Model checking is a method to verify whether given properties are valid or invalid in all possible state transitions which can be achieved by the models using a computer exhaustively. In cases where there are numerous states of the models in model checking, state explosion may occur where model checking does not get completed since the number of state combinations is too large. This means there is a possibility that the verification of the cooperative behavior by model checking may not get completed when the cooperative behavior by the components becomes extremely complex as a result of architectural designing. In this case, it is necessary to conduct re-architectural designing to prevent the cooperative behavior by the components from getting extremely complex, or reduce the number of the states in the models for the specifications related to the cooperative behavior.

6.3 Issues

We are able to comfirm the effectiveness of this methodology on an industrial case study. However, there are some issues which must be solved in the future.

The first issue is a problem of the bridge method in this methodology. In bridge method, the properties which must be satisfied by the cooperative behavior are extracted. In general, the properties to be satisfied by a target of model checking are liveness and safety[27]. Liveness is a property where “the verification target will eventually reach a desirable state”. Safety is a property where “the verification target will never reach an undesirable state”. The liveness property is often the specifications which must be achieved

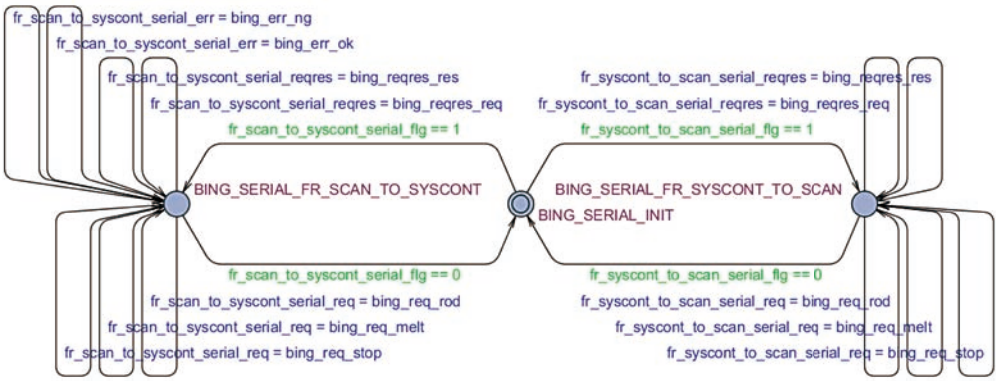


Fig. 13 Cooperative behavior model for interface specification between the measurement and integrated control subsystems

by the verification target. Therefore, the liveness of the verification target can be extracted from the specifications of the verification target or the specifications based on the verification target. However, most of the property related to safety of the verification target is not defined in the specifications of the verification target or the specifications based on the verification target. Extracting safety of the verification target depends highly on the experience and skill of engineers who use this methodology. Particularly, extracting the safety property which must be satisfied by the cooperative behavior tends to be dependent on engineers due to the complexity of the cooperative behavior.

The second issue is a problem of model checking in this methodology. In model checking, the models are developed based on the specifications related to the cooperative behavior extracted from the component specifications and interface specifications among components. The component specifications and the interface specifications are often described in natural language. Therefore, in many cases, the extracted specifications are modeled manually and errors may creep into the models. This is an issue for model checking in general.

The third issue is also a problem of model checking in this methodology. In model checking, the fomulae are developed based on the properties which must be satisfied by the cooperative behavior according to the expression form of the model checking tool. The formulae of the model checking are expressed by temporal operator[Term 37] (G: globally, F: finally) called temporal logic[Term 36], path quantifier[Term 38] (A: all, E: exists), and logic operator[Term 39] (OR, AND, NOT). However, since specialized knowledge is needed to use temporal logic, it is difficult to develop the formulae based on the properties to be satisfied by the cooperative behavior. This is also an issue for model checking in general.

7 Summary and future work

In this paper, we present the methodology where the system specification is decomposed into the component specifications and the interface specifications among components for the consistent cooperative behavior among the components which compose the system. Bridge method is developed to synthesize architectural design method and model checking, and the actual example of the bridge method is presented. The results of applying this methodology to an industrial robot are shown. From the application results, it is demonstrated that this methodology is effective for the complex system used in industry. We plan to present this methodology to the industrial fields through the Graduate School of System Design and Management, Keio University[28] (Keio SDM) to which the authors belong. The Keio SDM is a graduate school where people of various fields of both humanities and sciences study. Engineers who work in the frontline of product development on systems of aerospace, information, robot, and electronics attend Keio SDM. By presenting this methodology to the engineers attending Keio SDM, we can make contribution to industry. However, there are rooms for improvements in this methodology. In the future, we will solve issue 1 which is listed in this paper. We will aim to develop a methodology of higher quality.

Terminologies

Term 1. System: a combination of interacting elements organized to achieve one or more stated purposes.

Term 2. Electronic equipment system: a system equipped with multiple processors which digitally process information (e.g. cellular phone).

Term 3. Information system: a system for business activities where multiple computers engaging in data processing are connected with a network (e.g. business system).

Term 4. System of systems: a composite system which is formed by multiple systems with different purposes.

Term 5. Cooperative behavior: a cooperation with processings of the system component and the other system components through the interface among components.

Term 6. Consistency of cooperative behavior: a consistent implementation of the cooperative behavior by the components in correspondence to the system specification.

Term 7. Systems engineering: technological methodologies to achieve systems which satisfy the required quality within a given budget and time period. It is standardized as know-hows and rules independent of the technological fields.

Term 8. Architectural design method: a design method to allocate functions and performances required of a system to the system components, and to define the specifications of the components and the interface among the components.

Term 9. Model checking: a verification method to verify whether a given property is valid or invalid in all possible state transitions which can be achieved by the models which represent the state transitions of the system, using a computer exhaustively.

Term 10. Formal method: a developing and verification technology where specifications are expressed using the language of mathematical logic to guarantee the correctness of the specification.

Term 11. IEC 61508: an international standard which determines compliance items needed to build the functional safety in the process industry, machine manufacturing, traffic transportation, medical device, and others, using electrical, electronic, and programmable electronic systems.

Term 12. Structured analysis and structured design (SA/SD) method: a design method where a system is decomposed into components by focusing on data flows of the system.

Term 13. ISO 15288: one of the systems engineering standards. The tasks and procedures are defined for each process of the entire system lifecycle from the conceptualizing phase to the dismantling phase.

Term 14. ANSI/EIA 632: one of the systems engineering standards. The tasks and procedures are defined for each process of the system lifecycle from the conceptualizing phase to the transition to operation phase.

Term 15. IEEE 1220: one of the systems engineering standards. The tasks and procedures are defined for each process of the system lifecycle from the system requirement analysis phase to the system test phase.

Term 16. Test method: a verification method for verifying behavior of actual products against the test cases.

Term 17. Simulation method: a verification method where a target to be verified and peripheral environment of the target are simulated as models on a computer, and behavior of the models is verified against the test cases.

Term 18. Deadlock: a state where two or more processing units wait for each other to complete each processing, and as a result, all processings fail to move on further.

Term 19. Functional designing: a work where functions defined as a system specification are decomposed and refined, and performances defined as the system specification are allocated to the decomposed and refined functions.

Term 20. Physical designing: a work where system components are specified, and the functions and performances decomposed and refined in functional designing are allocated to the components.

Term 21. Functional analysis: the process which corresponds to functional designing, defined in chapter 6 section 3 in IEEE1220.

Term 22. Synthesis: the process which corresponds to physical designing, defined in chapter 6 section 5 in IEEE1220.

Term 23. Bridge method: a method which is presented in this paper to connect architectural design method and model checking seamlessly.

Term 24. Traceability matrix: a table which summarizes the correspondence of upper and lower level specifications.

Term 25. Finite automaton: a behavior model composed of finite number of combinations of the state, transition, and operation.

Term 26. Timed automaton: a behavior model where temporal variables are incorporated into finite automaton. It allows modeling of the time passage as transition conditions.

Term 27. SPIN: a model checking tool based on finite automaton. State transitions of the system are modeled using PROMELA (process meta language) which is a language similar to C. It can be downloaded from <http://spinroot.com/>.

Term 28. UPPAAL: a model checking tool based on timed automaton. State transitions of the system can be modeled in an intuitive manner using GUI (graphical user interface). It can be downloaded from <http://www.uppaal.com/>.

Term 29. Industrial robot: an industrial-use machine with the auto-control functions for manipulation or transportation. It can be programmed to conduct various work routines.

Term 30. Subsystem: an entity possessing the structure of a distinct, local system, while being part of a system.

Term 31. Subsystem Component: an element or a part which composes the subsystem.

Term 32. COTS (commercial off the shelf): software and hardware products which are available on the market.

Term 33. Teaching pendant: a device used for programming acitons and emergency stop of an industrial robot.

Term 34. Console: an input and output device used for operating the system. It is composed of input device such as a keyboard and output device such as a monitor display.

Term 35. QCD: an abbreviation for quality, cost, and delivery of development.

Term 36. Temporal logic: a theory of rules and expressions to understand and express the problem in relation to time. Temporal operator, path quantifier, and logic operator are combined to express the properties such as “P is always valid” or “Q is eventually valid.”

Term 37. Temporal operator: operators to express “G: globally” and “F: finally” in temporal logic.

Term 38. Path quantifier: operators to express “A: all” or “E: exists” in temporal logic.

Term 39. Logic operator: symbols which express logic operation. It includes “NOT: negation”, “AND: logical product” and “OR: logical sum.”

References

[1] International council on systems engineering (INCOSE): *INCOSE Systems Engineering Handbook version 3.1*, 1.5 of 6, INCOSE, USA (2007).

[2] N. G. Leveson: *SAFEWARE: System Safety and Computers*, 515-553, Addison-Wesley Professional, USA (1995).

[3] H. Shimizu: Arian 5 no bakuhatsu jiko to sofuto uea anzensei ni kansuru kokusai kikaku (Explosion of Ariane 5 and the international standard for software safety), *Anzen Kogaku (Journal of the Japan Society for Safety Engineering)*, 41 (1), 39-42 (2002) (in Japanese).

[4] Ministry of Land, Infrastructure, Transport and Tourism: FDP shisutemu no shogai no gen’in chosei no kekka (Result

of the cause adjustment of FDP system failure), MLIT (online), http://www.mlit.go.jp/kisha/kisha03/12/120312_.html (2009-09-22) (in Japanese).
[5] A. Katoh, N. Kohtake, S. Haruyama and Y. Ohkami: Moderu kensa o mochiite kumikomi shisutemu ni okeru sofuto uea to hado uea no kyocho dosa ni kansuru yokyu shiyo no fuseigo o kenshutsu suru shuho (A Model Checking Methodology for Detecting Inconsistency of Specifications concerned with Cooperating Behavior between Software and Hardware in Embedded System), *IPSJ Symposium Series*, 2009, 65-70 (2009) (in Japanese).
[6] A. Katoh and Y. Ohkami: FPGA to sofuto uea ni okeru kyocho dosa no seigosei ni kansuru hyoka shuho no teian (An Approach for Verifying Correctness of Co-operations between FPGA and Software in Electronic System), *IPSJ SIG Notes*, 2009 (31), 105-112 (2009) (in Japanese).
[7] E. M. Clarke, O. Grumberg and D. E. Long: Model checking and abstraction, *ACM Transactions on Programming Languages and Systems*, 16 (5), 1512-1542 (1994).
[8] Institute of Electrical and Electronics Engineers (IEEE): *IEEE standard for system and software engineering - System life cycle processes*, IEEE 15288-2008 (2008).
[9] American National Standard Institute (ANSI)/ Electronic Industries Alliance (EIA): *ANSI/EIA Standard for Process for Engineering a System*, ANSI/ EIA 632-1999 (1999).
[10] Institute of Electrical and Electronics Engineers (IEEE): *IEEE standard for application and management of the systems engineering process*, IEEE 1220-2005 (2005).
[11] E. M. Clarke and J. M. Wing: Formal methods: State of the art and future directions, *ACM Computing Surveys*, 28 (4), 626-643 (1996).
[12] International Electrotechnical Commission (IEC): *IEC standard for functional safety of electrical/electronic/programmable electronic safety-related systems*, IEC 61508-SER Ed. 1.0 (2005).
[13] T. DeMarco: *Structured Analysis and System Specification*, Yourdon Press, USA (1978).
[14] M. Arisawa and T. Saitoh: *Moderu Shimyureshon Giho (Model Simulation Methodology)*, 16-17, Kyoritsu Shuppan (1997) (in Japanese).
[15] V. K. Rompaey, D. Verkest, I. Bolsens and D. H. Man: CoWare – A design environment for heterogeneous hardware/software systems, *Proceedings of the European Design Automation Conference*, 252-257 (1996).
[16] T. Doran: IEEE 1220: For practical systems engineering, *IEEE Magazines Computer*, 39 (5), 92-94 (2006).
[17] M. Sipser: Course Technology Ptr (SD), *Introduction to the Theory of Computation*, 29-90, The Netherlands (1996).
[18] R. Alur and D. L. Dill: A theory of timed automata, *Theoretical Computer Science*, 126 (2), 183-235 (1994).
[19] G. J. Holzmann: The model checker SPIN, *IEEE Transaction on Software Engineering*, 23 (5), 279-295 (1997).
[20] K. G. Larsen, P. Pettersson and W. Yi: UPPAAL in a nutshell, *International Journal on Software Tools for Technology Transfer*, 1 (1-2), 134-152 (1997).
[21] Japanese Industrial Standards Committee: *Sangyo yo manipyuretingu robotto - yogo (Industiral Manipulating Robot – Terminology)*, JIS B0134 (2008) (in Japanese).
[22] J. P. Elm: A study of systems engineering effectiveness – Initial results, *Proceedings of the Systems Conference 2008 2nd Annual IEEE*, 1-7 (2008).
[23] B. Boehm, R. Valerdi and E. Honour: The ROI of systems engineering: Some quantitative results for software-intensive systems, *Systems Engineering*, 11 (3), 221-234 (2008).
[24] E. C. Honour: Understanding the value of systems engineering, *Proceedings of the INCOSE International Symposium*, 1-16 (2004).
[25] A. K. Kludze: The impact of systems engineering on complex systems, *Proceedings of Conference on Systems Engineering Research* (2004).
[26] B. W. Boehm: *Software Engineering Economics*, 38-40, Prentice-Hall, USA (1981).
[27] B. Alpern and F. B. Schneider: Defining liveness, *Information Processing Letters*, 21, 181-185 (1985).
[28] Graduate School of System Design and Management, Keio University: Homepage, Graduate School of System Design and Management, Keio University (online), http:// www.sdm.keio.ac.jp/ (see 2010-04-18).

Authors

Atsushi Katoh

Completed the courses in Electric System at the Graduate School of Science and Technology, Kumamoto University in 2000. Joined an electric appliance company and engaged in research and development of a small network device in a ubiquitous environment. Worked on independent verification and validation (IV&V) of spacecraft software in space development field since 2006. Also engaged in research on systems engineering at Keio SDM. Won the Research Award of the Information Processing Society of Japan in 2009. Member of the Information Processing Society of Japan. In this paper, was in charge of the research plan, selection and synthesis of the methods, application to the industrial case, and discussion.

Masataka Urago

Completed the doctoral program at the Department of Mechanical Sciences and Engineering, Graduate School of Engineering, Tokyo Institute of Technology in 1998. Doctor (Engineering). Became an assistant at the Tokyo Institute of Technology in 1998. Appointed to associate professor of Keio SDM in 2008. Engages in research on systems engineering as well as computer modeling and mathematical computation of engineering problems. Member of the Japan Society of Mechanical Engineers and the International Council on Systems Engineering (INCOSE). In this paper, was in charge of the selection and synthesis of the methods.

Yoshiaki Ohkami

Completed the doctoral program at the Department of Electrical Engineering, Graduate School of Engineering, Tokyo Institute of Technology in 1968. Doctor (Engineering). NASA Fellow, professor of the Tokyo Institute of Technology, professor of Keio University, and Research Director, Japan Aerospace Exploration Agency (JAXA). Became the Director of Keio SDM in 2008. Engages in research on dynamics and control of complex systems and strategic systems engineering. Fellow of the Japan Society of Mechanical Engineers and the International Council on Systems Engineering (INCOSE). Member of the Japan

Society for Aerospace and Space Sciences, Institute of Electrical and Electronics Engineers, Inc. (IEEE), and American Institute of Aeronautics and Astronautics (AIAA). In this paper, was in charge of research strategy planning and research integration.

Discussions with Reviewers

1 Novelty of the issue and outcomes

Question (Kanji Ueda, AIST)

In chapter 2, you say the reason for synthesizing the methods of different research or technological fields is that you can expect to produce a new research or technological field as some new technologies are generated from the synthesis. What kind of results did you obtain from this research?

Comment (Motoyuki Akamatsu, Human Technology Research Institute, AIST)

This paper claims novelty, but the readers outside of this specialty cannot understand immediately whether it is novel. I assume that there had been methods proposed to improve the reliability of the system, and I think you can emphasize the novelty if you explain the situation before this research was carried out. Similarly, I think you should explain why such critical technology was never pursued before, and why it had been difficult.

Answer (Atsushi Katoh)

I would like to explain the background of this research by focusing on the cooperative behavior by the system components. Normally, the cooperative behavior by the system components is verified in the system test that is conducted at the final phase of system development. When any inconsistency of the cooperative behavior is detected there, it is necessary to return to the upstream of the system development for correcting, and correction is often quite costly. Also, re-designing in the final phase may compromise the reliability of the system. Therefore, the cooperative behavior by system components must be designed and verified thoroughly in the upstream of the system development. At that point, there was no proposal for a methodology to achieve this. The reason is that the cooperative behavior by system components was not viewed from the perspective of system reliability, and the incorporation of system quality in the upstream of the system development was a relatively new concept. I shall add these points in chapter 1.

As results of conducting this research, the following four outcomes were obtained. First is the establishment of this methodology where the system specifications are decomposed into the component specifications and interface specifications among components for consistent cooperative behavior. Second is the clarification that bridge method is needed to synthesize architectural design method and model checking, and the presentation of an actual example for the bridge method for the cooperative behavior. Third is the expansion of the ranges of application and research of model checking, by applying the method that was mainly used in software development to system development. Fourth is the proposal of this methodology to the robot industry.

2 Cooperative behavior

Comment (Kanji Ueda)

You use the expressions "cooperative behavior by the components" several times, but I think the general readers may have difficulty understanding the meaning. Also, you use "consistency/inconsistency of the cooperative behavior" as self-explanatory terms. Please state the definition or the meaning of the cooperative behavior, and provide explanations.

Comment (Motoyuki Akamatsu)

You do not describe the means for determining the attributes of the cooperative behavior, extracting the properties to be satisfied by cooperative behavior, and for considering safety. Therefore, I don't think the robot engineers, for example, can see whether they can use this. I think you should provide explanations that give some hints for the people of the robot industry to use this methodology.

Answer (Atsushi Katoh)

In this paper, the cooperative behavior by components is defined as "a cooperation with processings of the system component and the other system components through the interface among components". The consistency of cooperative behavior is defined as the state where "a consistent implementation of the cooperative behavior by the components in correspondence to the system specification is conducted". The inconsistency of cooperative behavior is defined as the state where "the definition of the consistency of the cooperative behavior is not satisfied". These are added to chapter 1.

The attributes of the cooperative behavior are determined based on the properties to be satisfied by the cooperative behavior and the specifications related to the cooperative behavior. For the measurement and integrated control subsystems in the applied case, we were concerned about the lacks in the messages and processings of the specifications related to the cooperative behavior. Also, there were temporal limitation specifications such as within 100 ms and timing of messages and processings, among the properties to be satisfied by the cooperative behavior and the specifications related to the cooperative behavior. Therefore, the two points presented in subchapter 4.2 were determined as the attributes. We will add these to section 5.2.2.

The properties that must be satisfied by the cooperative behavior are extracted using the traceability matrix that summalizes the correspondence between the system specifications and the component specifications obtained by decomposing the system specifications.

Confirming safety of the cooperative behavior is the topic of this research. Safety means a property where the system will never reach an undesirable state. Since it is difficult to identify "all the states where the system is undesirable", it cannot be denied that there may be a fault in the safety confirmation for the cooperative behavior. We plan to investigate the solution to this issue by combining a safety analysis method such as fault tree analysis (FTA) with this methodology.

3 Selection of methods

Question (Kanji Ueda)

There are many expressions of "according to IEEE 1220…". What is its relationship to the methodology developed in this research? Also, the reason why you employed IEEE 1220 is not clear.

Comment (Motoyuki Akamatsu)

Please explain what other methods there were of system design methods other than architectural designing. Please also explain methods other than IEEE 1220 that you did not employ for architectural designing. Also, to clarify the scenario for selecting IEEE 1220 among several other architectural design methods, it will be easier to understand if you describe what disadvantages there were in the other methods.

It is explained in chapter 1 that model checking is a method to exhaustively verify the models that express the state transitions of the system. Please explain whether model checking method used here was originally developed in this research or is an application of an existing method.

It is written that the checking tool based on finite automaton was selected as the model checking tool, and you describe the

advantages. Please describe what other tools there were and what their disadvantages were. Similarly, if there were other candidates in the selection of UPPAAL, please describe them.

Answer (Atsushi Katoh)

In this research, we employed the research policy of selecting the methods for which the effectiveness has been already established, to efficiently establish a high-quality methodology. In constructing this methodology, we applied architectural design method that has been recognized as being effective and is standardized. Other system design methods include structured analysis and structured design (SA/SD) method. While architectural deign method is not appropriate for system design focusing on specific technological elements such as data or service, it is a universal design method independent of any particular technological systems. Therefore, we selected architectural design method as our system design method. The representative system engineering standards for architectural design method include ISO 15288, ANSI/EIA 632, and IEEE 1220. Since the tasks and procedures are finely defined for each process in IEEE 1220, we employed IEEE 1220. These standards are compared in chapter 3.

Model checking is a method that has already been established as a verification method. The research was started in the early 1980s, and today, it is widely used in software development. Model checking is "a verification using models", but it has become a proper noun for the verification method. We will add these points to chapter 1. The system verification methods other than model checking include test method and simulation method. However, model checking can exhaustively verify whether the properties to be satisfied for the state transitions are valid or not. Therefore, we employed model checking as our system verification method. We will compare these in chapter 3.

There is a verification method called theorem proving in the formal method. Theorem proving is a method where the system specifications and designs are described in a language that is mathematically defined in terms of semantics, and are given exact proof. While theorem proving enables exact verification of a system, great efforts are needed because some parts must be done interactively with humans. The authors think that theorem proving is not a verification method that has been applied in industry and the effectiveness has not been demonstrated. Therefore, we excluded it from the candidates of verification methods of this research.

The representative types of model checking include finite automaton and timed automaton, which is an extended finite automaton. Model checking and the tools to be applied must be selected according to the characteristic of the verified target. In a case of verifying general state transitions where the situation change is triggered by some external event, we select a model checking tool corresponding to finite automaton. SPIN is a representative model checking tool. In a case of verifying state transitions including temporal limitations, we select a model checking tool that corresponds to timed automaton. UPPAAL is another representative model checking tool. If the target that can be verified by model checking for finite automaton, it can also be verified using a model checking tool for timed automaton. However, it does not work the other way around. We will add these points to chapter 4. In the applied case, it was necessary to verify the temporal limitations of the cooperative behavior. Also, for the model checking tool corresponding to timed automaton, the authors determined that only UPPAAL possesses the applicable quality in the industrial case.

4 Industrial application of the outcomes

Comment (Kanji Ueda)

I think the significance as *Type 2 Basic Research* will increase if you address how the result of this research was actually used in the industrial application, what is the prospect, and the limit of application if it is used.

Comment (Motoyuki Akamatsu)

The important point of this research is that this methodology must be used by the people in industry who actually work to develop the system. In that sense, I would like to see an explanation on what targets these outcomes can be used, and the range of its application. Also, please state your thoughts on the ways of diffusing this methodology.

Answer (Atsushi Katoh)

I think the methodology in this research can be applied to the development of any technological systems, without specializing in particular technological systems. The reason is that when a system is designed, the function is achieved by finding the components that compose the system based on the system specifications, and then have the components behave cooperatively, and this is a common concept for all technological systems. Also, this methodology can deal with the unique issues of cooperative behavior in the applied system. The reason is that this methodology has the bridge method where the attributes and the model checking tool are selected according to the characteristics of the cooperative behavior.

Currently, the authors are planning and developing a new product for the industrial robot jointly with an industrial robot company. The industrial case study in chapter 5 describes these efforts. As of writing of this paper, the industrial robot is being developed according to the specifications that were verified for consistent cooperative behavior by applying the methodology of this research. I think this research amounts to *Type 2 Basic Research* because of the contribution to increase the reliability of industrial robot. These points are added to chapter 5.

However, cautions must be taken when applying this methodology. This method employs model checking to verifiy consistency of the cooperative behavior by components. When there are numerous states in model checking, state explosion may occur where model checking cannot be completed due to the huge number of combinations of the states. This means that if the cooperative behavior by components becomes extremely complex due to architectural designing, the verification of the cooperative behavior may not get completed in model checking. In such cases, it is necessary to do re-architectural designing to ensure the cooperative behavior by components will not become extremely complex, or the number of the states in the models for the specifications related to the cooperative behavior must be reduced. We will describe these in the newly added section "Applicability of this methodology" in chapter 6.

This methodology will be spread throughout industry by Keio SDM to which the authors belong. We will add this to chapter 7.

5 Advantages and disadvantages of the selected methods

Question (Motoyuki Akamatsu)

In chapter 3, you describe the selection of the methods. You mention architectural design and SA/SD methods for achieving function "a", and state that you selected architectural design method because it is a universal design method, although it is not suitable for system design focusing on specific technological elements such as data or service. Since the elemental technologies are selected according to the research goal, can you clearly state the goal, and then explain that you selected architectural design method as a result of comparing the advantages and disadvantages?

The goal of this research, I assume, is to construct a universal methodology, but when universality is set as a goal, you will also have a problem that the methodology cannot be applied to

individual problems. I think the solution to this is the bridge method, but if this is so, please explain this clearly (this is also relevant to discussion 3).

Answer (Atsushi Katoh)

I shall clarify the objective (goal) of the research. The objective of this research is to establish a methodology for decomposing a system specification into component specifications and interface specifications among components whose cooperating behavior is consistent with each component, which is not specific to particular technological systems. We will add this to chapter 2. The system design method with function "a" mentioned in chapter 3 includes SA/SD and architectural design methods. In the SA/SD method, the system design is done focusing on the data (such as business information) that are stable against the changes in the system environment. This enables the construction of a system with maintainability and expandability. However, because it is a method developed primarily for information systems, it is not very suitable for designing anything other than the information system. On the other hand, architectural design method has no procedures or tasks specifically defined for a certain designing, and therefore requires more efforts compared to specific design methods. However, architectural design method is a universal design method independent of some particular technological systems. Therefore, considering the research objective of developing a methodology not specific to particular technological systems, we selected architectural design method as the system design method with function "a". The process of selecting architectural design method from the system design methods, and the advantages and disadvantages of the SA/SD and architectural design methods are revised in chapter 3.

As you indicated, I do think there is a problem that this methodology will be difficult to apply to individual problems because it is not specific to any technological systems. For this methodology, the attributes and the model checking tools are selected according to the characteristics of the cooperative behavior in the applied system using the bridge method. The issues unique to the applied system for the cooperative behavior are handled in this manner. We will add these to subchapter 6.2.

6 Bridge method

Comment (Motoyuki Akamatsu)

You mention that one of the outcomes of this research is that you clarified the fact that the bridge method is necessary. Please describe the research scenario for the bridge method, such as why the bridge method is necessary, what requirements it has to satisfy, and why you named it bridge method.

Since architectural design method and model checking were developed based on two different ways of thinking, I assume that the outputs from architectural design method were insufficient for model checking, and I think it is natural that you needed a technology to convert each other to connect the two items with different concepts. Therefore, to clarify the originality of this method, please address whether this is simply a conversion method, or a method developed to verify the cooperative behavior.

Answer (Atsushi Katoh)

In this research, the system verification is conducted for the cooperative behavior by components at the phase of system design. Therefore, we focused on the cooperative behavior, and saw it was necessary to seamlessly connect the outputs of architectural design and the inputs of model checking. Therefore we developed the method to derive the component specifications and interface specifications among components related to the cooperative behavior, the properties to be satisfied by the cooperative behavior, and the model checking tool to be applied. This technology is called bridge method because it bridges architectural design method and model checking. I believe the bridge method is novel because it focuses on the cooperative behavior, and we clarified the specific method for synthesizing the system engineering standards such as IEEE 1220 and model checking. We will add these to subchapter 4.2.