Summary It has been an important issue to deal with risks in business processes for achieving companies’ goals. This paper introduces a method for applying a formal method to analysis of risks and control activities in business processes in order to evaluate control activities consistently, exhaustively, and to give us potential to have scientific discussion on the result of the evaluation. We focus on document flows in business activities and control activities and risks related to documents because documents play important roles in business. In our method, document flows including control activities are modeled and it is verified by OTS/CafeOBJ Method that risks about falsification of documents are avoided by control activities in the model. The verification is done by interaction between humans and CafeOBJ system with theorem proving, and it raises potential to discuss the result scientifically because the interaction gives us rigorous reasons why the result is derived from the verification.

Key words: risk analysis, business process, formal methods

1. Introduction

1.1 Background

It has been an important issue to deal with risks in business processes for achieving companies’ goals. Risk assessments are used for deciding how risks should be dealt with and established techniques are used practically. From the result of risk assessments, control activities are embedded in business processes. Control activities are policies (which establish what should be done) and procedures (the actions of people to carry out policies) ensure that management directives identified as necessary to address risks are carried out, such as approvals, authorization, reconciliations and segregation of duties [2], [4].

Recently, it has been more important to evaluate effectiveness of control activities while improper activities by well-known corporations have been revealed one after another. The Sarbanes-Oxley Act (2002, the US) [1] and Financial Instruments and Exchange Law (2006, Japan) [3] are promulgated for improving reliability of financial reporting, and one of requirements of these laws is that management of listed companies must issue a report as the result of the evaluation of Internal Control. Since designing control activities is one of main elements for constructing Internal Control [2], evaluation of effectiveness of control activities is increasingly important.

This paper introduces a method for precisely evaluating effectiveness of control activities. Our method shows how a formal method can be applied to evaluating effectiveness of control activities. Formal methods are mathematically based techniques for describing and verifying specifications of software systems or hardware systems precisely in the design stage. We believe that application of formal methods can be helpful for rigorous evaluations of effectiveness of control activities in business processes from a different point of view.

1.2 Research Motivation

Evaluating control activities here means checking if risks are handled by control activities as expected. In a general way of evaluating control activities, the following documents are created for each business process [24]:

- Flowchart: representation of the work flow of the objective business process
- Risk and Control Matrix: information about latent risks, specific control activities, and which control activities are dealing with which risks
- Business Process Narrative: more detailed description of the work flow

These documents are given in natural languages and graphical representations and are useful for stakeholders to understand control activities, risks, business processes, and relations between them. Which risks should be concerned to be handled in which points of the business process and which control activities are embedded in the business process are described in the flowcharts and the business process narrative. Also, which control activities are performed for handling which risks are described in the risk control matrix.

By using these documents, control activities are evaluated. However, informal descriptions like these documents can be evaluated only by humans and results of evaluation depend on the evaluators’ experiences. This fact leads to the following problems:

Inconsistency of the evaluation Results of the evaluation of the same object might be different depending on evaluators, because interpretation of risks and control activities might be different

Difficulty of evaluation of complex business processes It
Lack of scientific discussion of the result There is a lack of room for a scientific discussion on the reason why control activities are enough or not. For example, a risk control matrix gives information like “risk R is dealt with controls C1, . . . , Cn”, however it does not give which controls from C1 to Cn are really important and why risk R can be avoided or mitigated by these controls.

1.3 Aim

The aim of this study is to propose a method for precise evaluation of control activities with deeper understanding of how risks can be avoided by control activities. For this aim, we apply a formal method technique to formalise and analyse business activities, risks, and control activities. Our targets for formalisation are models of document flows in business activities, risks related to falsification of documents, and control activities, which avoid such risks. And, by verification in our method, we can evaluate control activities consistently and exhaustively. For the verification, we apply theorem proving and it raises potential to discuss the result of the evaluation scientifically, because it gives reasons why the result is derived through the verification. By this study, we can analyse relations between risks and control activities formally and it helps us for more reliable evaluation.

Applying Formal Methods

Applying formal methods to business process modeling is useful because (i) formal models do not leave any scope for ambiguity, and (ii) formal models increase the potential for formal analysis [5]. Constructing formal models helps us to understand the problems deeper, and by formal analysis, we can verify some properties of the models consistently and exhaustively. One more point we would like to emphasise is that formal models and formal verification give us potential to have scientific discussions. Informal descriptions do not have this characteristic. For formal verification, we apply theorem proving and it enables us not only to do consistent and exhaustive evaluations but also to analyse why control activities are enough or not while proving.

Focusing on Documents

We focus on document flows for a business (DFB) and analysis of risks which relate to falsification of documents. This idea is obtained from [15] and we believe that it is reasonable to model and analyse a business process in this way because documents play an important role in business processes, that is, all information created in business processes is recorded in some documents and transmission of information is done as transporting documents among divisions. We use the same model as [15], that is, the model of document flow, but the ways of formalisation are different. Formalisation for model checking is applied in [15], while this paper introduce how to formalise the model for theorem proving. More detailed information is described in Sect. 5.

In our previous work [6] we have classified patterns of control activities and risks from the material in [24] and it shows that the frequency of the occurrence of such risks is assumed as the highest. The material in [24] covers typical 38 business processes, 174 risks, and 221 controls activities in the business processes, and 169 risks and 215 control activities relate to documents. As we can see from the result of this research, most of risks and control activities are with respect to documents. From this fact, we emphasise that we can construct meaningful models of business processes with risks and control activities by focusing on DFB.

Risks to be Analysed

It is one of the most important issue in Internal Control to transmit accurate information in company [2]. Although there are other kinds of risks as objectives of risk assessments, analysis of risks related to falsification of documents are particularly important.

Therefore, in our method, we model irregular activities which might lead to risks related to falsification of documents by giving some conditions. We assume that each control activity to be modelled is strong enough, and under the assumption, it is clarified by formal verification that if the control activities as a whole are enough not to leave the unexpected falsification. “Each control activity is strong enough” means that each of them is performed as expected. For example, consistency of two documents should be judged by control activity “checking a document with another document”. In our model, the judgement by this control activity is always right, because our focus for verification is a set of control activities in a model and it is out of our scope how to realise each control activity right.

We have many assumptions for modeling. These assumptions are to ensure the strength of control activities. Once such assumptions are included in the models, we can precisely verify effectiveness of control activities as a whole in the model of document flows. Most assumptions are for ensuring the strength of approving and checking documents. An exception is setting trusted and untrusted divisions, which appear in Sect. 2.3.2 in this paper. In the real world, assumptions on who is trusted or not are made by setting segregation of duties. The segregation of duties is one of the important issues, and setting trusted and untrusted division represents a part of setting segregation of duties.

Evaluation Procedures

For formal verification, we must describe formal models, and for describing formal models, we must understand what should be formalised. In this study, evaluation of control activities is proceeded as follows:

1. Informative documents about the objective business process are given, and we construct a model of DFB in order to capture what should be formalised and verified.
2. After modeling DFB, we can formalise the model for formal verification
3. Then, we can verify effectiveness of control activities by proving some properties

Informative documents mean documents which explain the business process including control activities and risks, and relations between control activities and risks. We assume such documents like flowcharts, risk and control matrices, and business process narratives are given.

For formalisation and verification, we apply OTS/CafeOBJ Method[9],[19].

**Important Features of OTS/CafeOBJ Method**

In this method, systems are modeled as an Observational Transition System (OTS), which is a kind of state transition machines, and the OTS model can be naturally described in CafeOBJ [9], an algebraic specification language. An OTS specification represents behaviours of systems and can be used for verification of behaviours of systems. Verification is done by formally proving properties of the specifications by theorem proving technique with human interactions. Important features of OTS/CafeOBJ Method are as follows:

- Specifications in CafeOBJ can be described simply since CafeOBJ are based on equational logic which is easily understood
- Since CafeOBJ is an executable language, verification can be done semi-automatically with user interactions
- Verification by theorem proving can cover all possible states, while model checking cannot because it face the state explosion problem.
- In verification with OTS/CafeOBJ Method, interactions with humans can be done systematically for most of the cases

The most important feature of OTS/CafeOBJ Method for this study is that verification by interaction with CafeOBJ system can be done systematically, because it helps us to understand the problem deeper and can be done intuitively. “Systematic interaction” means that guidelines for splitting cases in a proof is given. Verification by OTS/CafeOBJ Method is done by the proof score approach. How to construct proof scores with clear structure is given in the method and it provides us a simple but proper way to split cases.

**Contributions of this Work**

This work is an application of a formal method to analysis of risks and control activities in business processes and it plays important roles as follows:

- What risks and control activities in business processes really mean can be defined precisely
- We can formally verify effectiveness of control activities by analysing all possible cases
- Through formal analysis based on good understanding of risks and control activities, we can have scientific discussions about

- why a set of control activities are necessary in order to deal with a risk, and
- why a set of control activities are not enough

**1.4 Structure of This Paper**

Structure of this paper is as follows. In Sect. 2 we introduce how we model DFB and in the end of this section we show an example. In Sect. 3 we firstly give the definition of OTS and show formalisation of the model by using the same example as Sect. 2. Section 4 explains verification procedures in OTS/CafeOBJ Method and show a verification experiment by using the example. Related work is introduced in Sect. 5, and Sect. 6 gives the conclusion and future work.

**2. Modeling DFB**

We model DFB in order to to analyse possible behaviours with respect to documents. For this purpose, we construct a model of DFB as a state transition machine. In this section, we firstly introduce how we construct state transition machines, which are models of DFB.

**2.1 Basic Concepts of Modeling**

We firstly explain how we capture a DFB as a state transition machine.

A DFB model is constructed by state, transitions, and transition rules.

**State:** A state is a set of documents.

Each document has attributes and observation values of states depend on attribute values of documents. Two states are equal if all documents in the both states have the same attribute values. In Sect. 2.2, we explain what kinds of attributes documents have.

**Transitions:** A transition is an operator which changes a state. A state is given as one of arguments of a transition and the transition returns a state.

Activities related to documents are modeled as transitions. We classify transitions into three groups, regular transitions, irregular transitions, and control transitions.

- Regular Transitions: A regular transition is a transition performed in order to proceed with the business toward the business goal. For example, creating an order document, etc.
- Irregular Transitions: An irregular transition is a transition, which changes a state to an undesirable state. Undesirable states and of desirable states are defined at the end of this subsection.
- Control transitions: A control transition is a transition, which detects undesirable states before they become risk states, or gives evidence to documents in order to help the detection. Risk states are defined in the last part of this subsection. “Checking a receipt document
with an order document" is an example of control transitions.

More detailed description of transitions is given in Sect. 2.3.

We define behaviours for a business in models of DFB as follows.

**Behaviours:** A behaviour is a sequence of transitions.

DFB models represent possible behaviours related to documents. In the models, how values of documents are changed are described as transition rules, and we can dynamically analyse behaviours by executing the models.

**Transitions Rules:** A transition rule represents how a state is changed by a transition. Since a state is a set of document in our modeling and transitions change values of documents, transition rules show how values of documents are changed.

We will explain how states are changed by transitions in Sect. 2.3.

**Scope of Modeling DFB**

For modeling DFB, we need to define what models can include by deciding how we capture the real world to model. Figure 1 intuitively represents a model of DFB for a simple sales process in a UML activity diagram. Each behaviour in this model relates to a single order. One of behaviours for a single order is as follows:

Creating order D1 with a master; Sending order D1 to sales; Approving order D1; ...

However, in the real world, not only one order is handled, but also many orders are handled concurrently. An example of behaviours of this situation is as follows:

Creating order D1 with a master; Creating order D2 with a master; Sending order D1 to client; Creating order D2 with a master; ...

Let us assume that each created document belongs to a session. A session is a set of behaviours for achieving an instance of a business goal. By a business goal we mean a purpose of business activities in a business. For example, the business goal of the simple sales process is selling a product.

Irregular transitions may occur not only in a single session but also in multiple sessions. For example, forging an order document is a kind of irregular transition, which appears in a single session, and an example of irregular transitions in multiple sessions is “handling a document created for order A by mistake although it is created for order B.”

In this paper, we model DFB which include multiple sessions, since some irregular transitions have an effect on not only one session but more than one session.

### Risk States and Undesirable States

By analysing behaviours in the models, we can evaluate effectiveness of control activities by proving that states where risks are tangible cannot be reached. We call such states risk states.

**Risk States:** A risk state is an undesirable state, which satisfies some conditions.

Desirable states and undesirable states are defined as follows:

**Desirable States:** A desirable state is a state where all documents are not forged and are handled right.

**Undesirable States:** An undesirable state is a state where one or more documents are

- forged, or
- being handled by mistake. This situation represents that the document is not used for the right purpose: e.g., the receipt for order document D1 is treated as the one for order document D2.

An example of risk states for the simple sales process described as Fig. 1 is a state where “a receipt document which has been checked with an order document is forged”. And, an example of undesirable states is a state where “a receipt document is forged”. In this case, “a receipt document has been checked with an order document” is the condition to make an undesirable state a risk state.

### 2.2 Model of Documents

In this subsection, we give more detailed information about modeling documents.

We define attributes of documents and these attributes are changed by transitions. In our modeling a document has following attributes:

- **ID of a document:** This attribute is the identifier of the document.
- **Type of a document:** Each document has their own role and this attribute is for representing what kind of document it is, e.g., order, invoice, receipt.
- **Evidence history:** Seals or signatures are put on the document to show that the document is approved or checked. We call a seal or a signature a piece of evidence and we call a set of evidence in the document evidence history. We model DFBs as OTSs, and the order of events can be reasoned from effective conditions of transitions (see the definition of OTS in Sect. 3.1). Both a list and a set can be used as evidence history, and we have chosen a set.
- **Name of division:** Each document is moved from a division to another division. This information shows which division the document currently belongs to.
- **ID of session:** Each document belongs to a session. This information shows which session the document currently belongs to.

**Fig. 1** A simple sales process.
We introduce meta information of documents in addition to attributes described above. Meta information is information which cannot be seen from object level. This information is needed for representing properties to be proved. Meta information for documents is as follows:

- Authenticity of a document: This meta information shows if the document is forged or not. It cannot be recognized with seeing only one document whether the document is forged or not.
- Original session ID: This meta information shows which session the document originally belongs to. This information is used for representing if the document is in the right session or not. The original session ID of a document is fixed when it is created. In our modeling, we consider that there are two cases for creating a document, that is, the one is creating a document without referring to any documents, and another is creating a document from another document. The original session ID of the document is
  - the ID of session where the document is created if it is created without referring to any other documents, or
  - the same as the original session ID of the document which is referred when the document is created.

If the original session ID of a document is different from the ID of session in which the document currently belongs to, this situation represents that the document is being handled by mistake.

By using the meta information, we can represent undesirable states. Note that meta information is not used for defining transition rules of regular transitions. More detailed information is given at the end of Sect. 3.

2.3 Transition Patterns

We can classify transitions into some transition patterns. Each transition for a model of document flows belongs to either of these patterns. We introduce transition patterns in this subsection. Examples of concrete transitions are described in the Sect. 2.4.

2.3.1 Regular Transitions Patterns

Regular transition patterns are “creating a document” and “sending a document”.

(1) Creating a Document

There are four cases for “creating a document” as follows:

- Create-1: Creating a document without referring to any documents. The created document is not forged, and the division name, the session ID and the original session ID is the one where the document is created.
- Create-2: Creating a document from another document. The division name, the authenticity value, the session ID, and the original session ID of the new document are the same as the ones of the document referred.
- Create-m-1: Creating a document with a master without referring to any documents. Attributes of the new document are as the ones created by Create-1.
- Create-m-2: Creating a document with a master from another document. Attributes of the new document are as the ones created by Create-2.

“Creating a document with a master” in Create-m-1 and Create-m-2 means that two documents are created and one of them is the master of the another document. The document ID of a master of document D with document type T is denoted as M(D), document type as M(T), and other other attributes and meta information are equal to ones of document D respectively when they are created.

In our modeling, we assume that master documents cannot be forged, that is, master documents are assumed to be managed strictly.

(2) Sending a Document

Another regular transition pattern is “sending a document”. Documents can be sent either from a division to a division, from a division to outside of a company, or from outside of a company to a division. By instances of this transition pattern, the division name of a document is changed.

2.3.2 Irregular Transition Patterns

We assume two irregular transition patterns in this paper.

(1) Forging a Document

An irregular transition pattern is “forging a document”. By instances of this transition pattern, the authenticity value of a document is turned to “false”.

**Important Assumptions for “Forging”**

We assume that there are untrusted divisions and conditions under which a document is forged are as follows:

- A document is only forged when it is in an untrusted division. We assume that there are untrusted divisions in the objective business activities.
- We assume that evidence cannot be forged because evidence should be put on the document by trusted people. In this study verification ensures that unexpected phenomena do not occur if control activities are performed correctly. For this purpose, it is one of the key points in our modeling that evidence cannot be forged. If a document which has evidence is forged, the evidence on the document is deleted in our modeling.
- Master documents cannot be forged. A master of a document in this model is the abstraction of documents that cannot be forged in the real world. For example, it can be electronic documents which are protected by access controls for avoiding outsiders forging.
(2) Changing a Session of a Document

Another irregular transition pattern is “changing a session of a document”. This transition is caused by a mistake in handling a document. For example, in sales process, invoice document D is created for the order document D1, but by mistake, it might be handled as a document created for the order document D2. We represent this situation as a document is moved to another session.

**Important Assumptions for “Changing a Session”**

We have two assumptions related to this kind of transitions.

- A document is moved to another session if instances of this transition pattern are performed in untrusted divisions.
- The original session is not changed. The role of the original session IDs as meta information of the document is recording the session ID where the document should belong to and are used for representing properties to be proved.

2.3.3 Control Transitions

When a control transition is performed in states which satisfy some conditions for the control transition, a piece of evidence is put on the documents in the next state. The conditions are defined for detecting undesirable states, and evidence can be used for showing that the documents have been transmitted in a proper way.

Control transition patterns are “checking a document” and “approving a document”.

(1) Checking a Document

There are four cases for “checking a document” as follows:

- Check-1: checking a document with another document.
- Check-2: checking a document which has evidence with another document.
- Check-3: checking a document with another document which has evidence.
- Check-4: checking a document which has an evidence with another document which has evidence.

By checking two documents, if these two documents are consistent, then a piece of evidence is put on D1, otherwise, the evidence is not put on D1. The latter case represents that the falsification is detected. “Two documents are consistent” here means that the authenticity values and original session IDs of the two documents are the same.

Check-2 and Check-3 are a more strict check than Check-1, because either document for checking should be checked or approved before. And, Check-4 is the strictest checking. For checking by Check-2 and Check-3, one of the documents needs to have a piece of evidence given as the argument. For Check-4, both documents should have a piece of evidence respectively.

**Important Assumptions for “Checking”**

We assume that documents are forged by a single intention. It means there is only one attacker or one group of attackers. This is the reason why we introduce the meta information for authenticity as a flag to show true or false. Suppose that two documents D1 and D2 have been forged.

- In the real world, if the person who forged document D1 is different from the one who forged document D2, the content of these documents may not be consistent. In this case, a piece of evidence is not put on the document after the checking.
- However, there should be the case where both documents are forged and contents are consistent. In this case, a piece of evidence can be put on the document after checking.

For modeling “checking a document”, we assume the latter case that is more difficult to detect the falsification. A control activity itself is not always effective, and what we verify is that control activities as a whole in the objective process are effective.

(2) Approving a Document

Another control transition pattern is “approving a document” and there are two cases for approving a document as follows:

- Approve-1: approving a document.
- Approve-2: approving a document which has evidence.

By approving a document a piece of evidence is put on the document. For Approve-2, the document which will be approved needs to have a piece of evidence before. If there is no required evidence in the document, then evidence will not be put on the document.

**An Important Assumption for “Approving”**

As we have mentioned before, evidence cannot be forged. This assumption derives the fact: Documents which have evidence have not been forged after receiving the evidence. The documents might be forged before this transition, but, at least, evidence on documents ensures the fact as above.

2.4 Example of Models of DFB

In this subsection, we show an example of informal descriptions of models of DFB. Figure 2 shows a UML activity diagram showing activities for handling documents for whole sales process (WSP). Before formalising this model, we will explain the model informally.

**Divisions and Untrusted Divisions:** In this DFB model, client, sales division, and shipping division are included. We assume that sales division and shipping division are untrusted.

**Documents:** In this DFB model, documents of the following types are created: order, master document of order,
ack, request, invoice, master document of invoice, receipt, and report. A master document is denoted by M(T), where T is an arbitrary document type.

**Regular Transitions and Control Transitions**

Regular transitions and control transitions are as follows:

- Creating order with M(order) in client (Create-m-1).
- Sending order from client to sales (Send-order).
- Creating ack from order in sales (Create-ack).
- Approving order in sales (Approve-order).
- Sending ack from sales to client (Send-ack).
- Checking ack with M(order) in client (Check-ack).
- Creating request from order (Create-request).
- Sending request from sales to shipping (Send-request).
- Creating invoice with M(invoice) from request (Create-invoice).
- Sending invoice from shipping to client (Send-invoice).
- Checking invoice with M(order) in client (Check-invoice).
- Creating receipt from invoice in client (Create-receipt).
- Sending receipt from client to sales (Send-receipt).
- Creating report from M(invoice) in shipping (Create-report).
- Sending report from shipping to sales (Send-report).
- Checking receipt with order which is approved in sales (Check-receipt).
- Checking report with order which is approved in sales (Check-report).

Control activities are Approve-order, Check-ack, Check-invoice, Check-receipt, and Check-report.

**Irregular Transitions**

We assume that sales and shipping are untrusted divisions and irregular activities can be performed in these divisions.

- Forging a document in sales or shipping (Forge-WSP).
- Changing a session of a document in sales or shipping (ChangeSession-WSP).

In this model, we assume that no irregular transition is performed in client division. It depends on the evaluators that which divisions are trusted. We can construct a model, for example, in which “changing a session” can be performed in client division for a more strict evaluation, but for the case shown in this paper, we assume that it cannot occur in client division.

3. **Formalising DFB**

In this section, we firstly give the definition of OTS, and then show an example of formal descriptions of DFB in OTS.

3.1 **OTS (Observational Transition System)**

We introduce the definition of OTS briefly. The more precise definition is given in [19].

**Definition 3.1 (OTS):** We assume that there exists a universal state space denoted \( \Upsilon \) and that data types used in OTSs are provided. OTS \( S \) can be defined as \( \langle O, I, T \rangle \) such that

- \( O \): A finite set of observers. An observer is a function, which takes a state and values of data types as arguments and returns a value of a data type. States are characterised by return values of observations.
- \( I \): The set of initial states such that \( I \subseteq \Upsilon \), where \( \Upsilon \) is a universal state space.
- \( T \): A finite set of conditional transitions. A transition is a function, which takes a state and values of data types as arguments and returns a state. Each transition has a condition, which is called the effective condition. If the effective condition does not hold, then the transition does not change the state.

A next state of state \( s \) is called a successor state of \( s \) and defined as follows.

**Definition 3.2 (Successor States):** Given an OTS \( S \) and two states \( \nu, \nu' \in \Upsilon \), if there exists \( t \in T \) and values of data types \( X_1, \ldots, X_n \) such that \( t(X_1, \ldots, X_n, \nu) = \nu' \), we write \( \nu \leadsto_S \nu' \) and call \( \nu' \) a successor state of \( \nu \) with respect to (wrt) \( S \). \( \leadsto_S^* \) is a reflexive transitive closure of \( \leadsto_S \).

States which are reached by applications of transitions from initial states are called reachable states, and defined as follows.

**Definition 3.3 (Reachable States):** Reachable states wrt \( S \)
are inductively defined:
- each \( v_0 \in I \) is reachable wrt \( S \).
- For each \( v, v' \in I \) such that \( v \leadsto_S v' \), if \( v \) is reachable wrt \( S \), so is \( v' \).

Let \( R_S \) be the set of all reachable states wrt \( S \).

3.2 Example of Formal Description of DFB

In this subsection, we explain OTS model of DFB for WSP.

3.2.1 Observations

In OTS model, states are characterized by observations. Since a state with respect to a DFB model is a set of documents, observations are defined based on values related to documents. Observations with respect to OTS \( S_{\text{WSP}} \) for WSP can be defined as follows:

\[
O_{S_{\text{WSP}}} = \{ \text{Documentation} : \text{State} \times \text{DocumentID} \rightarrow \text{Document}, \text{Legal}?: \text{State} \times \text{DocumentID} \rightarrow \text{Bool}, \text{OriginalSID} : \text{State} \times \text{DocumentID} \rightarrow \text{SessionID}, \text{DIDSet} : \text{State} \times \text{SessionID} \times \text{DocumentType} \rightarrow \text{DocumentIDSet}, \text{UntrustedSet} : \text{State} \rightarrow \text{DivisionSet} \}
\]

State denotes a state space, and Document, DocumentID, DocumentType, DocumentIDSet, SessionID, DivisionSet, and Bool, are data types which represent documents, IDs of documents, types of document, sets of IDs of documents, IDs of sessions, sets of division names, and boolean values respectively.

Observer Documentation takes a state and the ID of a document, and returns the document with the ID in the state given as arguments. Since a document has attributes introduced in Sect. 2.2, the constructor of Document can be defined as follows.

\[
\text{mkDoc} : \text{DocumentID} \times \text{DocumentType} \times \text{Division} \times \text{EHistory} \times \text{SessionID} \rightarrow \text{Document}
\]

Division and EHistory are data types which represent names of divisions and sets of evidence respectively. If there does not exist a document with Document ID \( D \) in state \( S \), Documentation\((S, D)\) returns noDocument, which is a special constant representing there is no such document in the current state.

Legal? and OriginalSID are observers which is for observing meta information of documents. If document \( D \) is forged in state \( S \), Legal?\((S, D)\) returns false, otherwise true. OriginalSID\((S, D_1:\text{DocumentID})\) returns the ID of the session where \( D_1 \) is created if it is created without referring to any other document, or the same value as OriginalSID\((S, D_2:\text{DocumentID})\) if \( D_1 \) is created from \( D_2 \).

Every document belongs to a session, and DIDSet\((S, I:\text{SessionID}, T:\text{DocumentType})\) returns IDs of documents which belong to session \( I \) and whose type is \( T \) in state \( S \). UntrustedSet\((S)\) returns a set of names of divisions which are not trusted. In this case study, we assume that untrusted divisions are sales and shipping.

3.2.2 Initial States

We assume that initial states are not undesirable states. No document is forged and each document created is in the original sessions.

Initial states with respect to OTS \( S_{\text{WSP}} \) for WSP can be defined as follows:

\[
I_{S_{\text{WSP}}} = \{ \text{init} \in \text{State} \mid \forall D : \text{DocumentID}, \text{Legal}?(\text{init}, D) = \text{true} \land \text{OriginalSID(\text{init}, D)} = \text{getSID(Documentation(\text{init}, D))} \land \text{UntrustedSet(\text{init})} = \{ \text{sales, shipping} \} \}
\]

Term OriginalSID\((\text{init}, D)\) = getSID\((\text{Documentation}(\text{init}, D))\) means that every document belongs to the right session. Function getSID is a selector for getting ID of a session where a document currently belongs to, and its signature is as follows:

\[
\text{getSID} : \text{Document} \rightarrow \text{SessionID}
\]

And, it is defined in CafeOBJ as follows:

\[
\text{eq getSID(mkDoc}(D, T, V, H, I)) = I .
\]

eq getSID\((\text{noDocument})\) = noSessionID .

\( D, T, V, H, \) and \( I \) are variables of DocumentID, DocumentType, Division, EHistory, and SessionID, respectively. noSessionID is a special constant representing that no session exists for the document of the argument.

3.2.3 Transitions

Transitions in WSP introduced in Sect. 2.4 have the following signatures.

\[
T_{S_{\text{WSP}}} = \{ \text{Create-m-order} : \text{State} \times \text{SessionID} \times \text{DocumentID} \rightarrow \text{State}, \text{Send-order} : \text{State} \times \text{DocumentID} \rightarrow \text{State}, \text{Create-ack} : \text{State} \times \text{DocumentID} \times \text{DocumentID} \rightarrow \text{State}, \text{Approve-order} : \text{State} \times \text{DocumentID} \times \text{DocumentID} \rightarrow \text{State}, \text{Send-ack} : \text{State} \times \text{DocumentID} \times \text{DocumentID} \rightarrow \text{State}, \text{Check-ack} : \text{State} \times \text{DocumentID} \times \text{DocumentID} \rightarrow \text{State}, \text{Create-request} : \text{State} \times \text{DocumentID} \times \text{DocumentID} \times \text{DocumentID} \rightarrow \text{State}, \text{Send-request} : \text{State} \times \text{DocumentID} \rightarrow \text{State}, \text{Create-m-receipt} : \text{State} \times \text{DocumentID} \times \text{DocumentID} \rightarrow \text{State}, \text{Send-receipt} : \text{State} \times \text{DocumentID} \\
\rightarrow \text{State}, \text{Create-report} : \text{State} \times \text{DocumentID} \times \text{DocumentID} \times \text{DocumentID} \rightarrow \text{State}, \text{Send-report} : \text{State} \times \text{DocumentID} \rightarrow \text{State}, \text{Check-receipt} : \text{State} \times \text{DocumentID} \times \text{DocumentID} \rightarrow \text{State}, \text{Check-report} : \text{State} \times \text{DocumentID} \times \text{DocumentID} \rightarrow \text{State}, \text{Forge-WSP} : \text{State} \times \text{DocumentID} \rightarrow \text{State}, \text{ChangeSession-WSP} : \text{State} \times \text{DocumentID} \times \text{SessionID} \rightarrow \text{State} \}
\]

Each transition is either a regular transition, an irregular transition, or a control transition. Transition rules can be defined by instantiating transition patterns.
Instantiation of Transitions

We have formal descriptions of transition patterns and they can be used as subsidiary functions for defining transition rules of specific transitions in WSP.

As an example of formal descriptions of the transition rules of regular transition patterns, the transition rule of Create-2 can be described as follows:

\[
\text{eq Documentation}(\text{Create-2}(S, V, D1, T, D2), D3) = \begin{cases} 
\text{mkDoc}(D1, T, V, \text{emptyE}, \text{getSID}(\text{Documentation}(S, D2))) & \text{if } (D1 = D3) \\
\text{Documentation}(S, D3) & \text{if } (D1 \neq D3)
\end{cases}
\]

\[
\text{eq Legal?}(\text{Create-2}(S, V, D1, T, D2), D3) = \begin{cases} 
\text{IF}(D1 = D3) & \text{if } (\text{getSID}(\text{Documentation}(S, D2)) = I) \\
\text{Legal?}(S, D3) & \text{if } (D1 \neq D3)
\end{cases}
\]

\[
\text{eq OriginalSID}(\text{Create-2}(S, V, D1, T, D2), D3) = \begin{cases} 
\text{getSID}(\text{Documentation}(S, D2)) & \text{if } (D1 = D3) \\
\text{OriginalSID}(S, D3) & \text{if } (D1 \neq D3)
\end{cases}
\]

\[
\text{eq DIDSet}(\text{Create-2}(S, V, D1, T, D2), I, T2) = \begin{cases} 
\text{mkDoc}(D1, T, V, \text{emptyE}, \text{getSID}(\text{Documentation}(S, D2))) & \text{if } (T1 = T2) \\
\text{DIDSet}(S, I, T1) & \text{if } (D1 \neq D3)
\end{cases}
\]

\[
\text{eq UntrustedSet}(\text{Create-2}(S, V, D1, T, D2)) = \text{UntrustedSet}(S)
\]

By application of Create-2 to State S, Division V, Document IDs D1 and D2, and Document Type T, document D1 is created from document D2 and its attributes and meta information is as follows:

- ID of the document : D1
- Type of the document : T
- Name of division : V
- Evidence history : empty
- Session ID : the same as the session ID of D2
- Authenticity : the same as the value of D2
- Original session ID : the same as the original session ID of D2

For describing the formal description of Create-request, we can use Create-2 as a subsidiary function for defining transition rules as follows.

\[
\text{ceq Create-request}(S, D1, D2, D3) = \text{Create-2}(S, \text{sales}, D1, \text{request}, D2) \quad \text{if } \text{c-Create-request}(S, D1, D2, D3)
\]

\[
\text{ceq Create-request}(S, D1, D2, D3) = S \quad \text{if } \text{not}(\text{c-Create-request}(S, D1, D2, D3))
\]

c-Create-request denotes the effective condition of Create-request. We omit the description of c-Create-request. The condition in natural language is as follows: D1 is not used for any other documents, document D2 is order document in sales, document D3 is ack document in the same session as D2, and has been checked with H(order) document (see Fig. 2). If the effective condition holds for the arguments of Create-request, the transition rule is defined by invoking the Create-2, otherwise the state is not changed.

Meta Information in Effective Condition

Since meta information cannot be seen from object level, we assume that meta information cannot be used in effective conditions of regular transitions. For example, we have shown that one of elements in c-Create-request is “document D3 is ack document in the same session as D2”. It is possible that effective condition is satisfied although original session of D2 and D3 are different. When modelling the original session, we assume that there is a mechanism which enable us to check if two documents are originally related, like managing documents with ordered document IDs.

We assume that the original relation of two documents are cared only when they are checked or approved. So, meta information can be used only in effective conditions of control transitions.

4. Verification as Evaluation of Control Activities

A formal description of a DFB model includes which control activities are in which points in the model and what functionality each control activity has. And, by verification of the formal description, we can prove that the model includes enough control activities at proper points to avoid risk states wrt the model to be reachable although possible irregular transitions occur. And, if we cannot prove it, we can analyse, from the result of the verification, how to improve the model by adding some control activities or changing the points where some control activities are assigned.

In this section, we firstly introduce what kind of properties are to be proved and how we can prove them. After that, we show an example of verification by using formal model introduced in Sect. 3.2.

4.1 Invariant Properties and Verification Procedures

In our method, we prove invariant properties to verify the effectiveness of control activities. In this section, we give a definition of invariant properties and verification procedures in OTS/CafeOBJ Method.

Invariant Properties

Predicates whose types are \( \forall x_j : D_j \) are called state predicates. We suppose that each state predicate includes a finite number of logical connectives. We also suppose that all variables in state predicates except for one whose type is \( \forall x_j \) are universally quantified. That is, the form of a state predicate \( p \) can be \( \forall x_{j_1} : D_{j_1} \cdots \forall x_{j_n} : D_{j_n}, \text{P}(v, x_{j_1}, \ldots, x_{j_n}) \).
Properties to be proved in this paper are only invariant properties. That is the reason why state predicates are confined to universal quantified ones. In OTS/CafeOBJ Method, liveness properties can be proved [22], but we focus on invariant properties in this paper, and it will be future work to prove liveness properties for models of documents flows.

Definition 4.1 (Invariants): Any state predicate \( p : \mathcal{T} \rightarrow \text{Bool} \) is called an invariant with respect to an OTS \( S \), if \( p \) holds in all reachable states of \( S \), i.e., \( \forall v : \mathcal{R}_S.p(v) \).

Verification Procedure

In OTS/CafeOBJ Method, verification is done by proofs by induction with interactions between users and CafeOBJ system. It helps us not only to guarantee that the model satisfies some properties, but also to ensure that the model is described right. Formal verification in OTS/CafeOBJ Method gives us deeper understanding of the problems and what we have described.

Assume that \( P : \times D_1 \times \ldots \times D_n \rightarrow \text{Bool} \) is the predicate which represents the property to be proved, where \( S \) is a state space wrt an OTS \( S \) and \( D_1, \ldots , D_n \) are data types. An invariant property is a property which holds in all reachable states, and proof that shows \( P \) is an invariant property wrt OTS \( S \) is constructed by the following procedure.

1. For each \( v_0 \in IS \) and \( d_1 \in D_1, \ldots , d_n \in D_n \), prove that \( P(v_0, d_1, \ldots , d_n) \) is true. If it does not hold, we cannot prove that \( P \) is an invariant property wrt \( S \).
2. For each \( v, v' \in S \) and \( d_1 \in D_1, \ldots , d_n \in D_n \), such that \( v \leadsto_S v' \), prove that \( P(v, d_1, \ldots , d_n) \) implies \( P(v', d_1, \ldots , d_n) \) is true. If it does not hold, we have to consider the followings:

   a. Consider whether there are any sequences of transitions which can reach to \( v \) from initial states. If such sequences can be found, we cannot prove that \( P \) is an invariant property wrt \( S \), otherwise, go to the next step as described below.

   b. Consider proving that \( v \) is not a reachable state wrt \( S \). For that purpose, we construct one or more lemmas as predicates which do not hold for \( v \). If we can prove that these lemmas are invariant properties wrt \( S \), then it means that \( v \) is not a reachable state wrt \( S \). Otherwise, we cannot prove that \( P \) is an invariant property wrt \( S \).

In order to prove that lemmas are invariant properties wrt \( S \), we iterate the procedure as described above.

CafeOBJ system is used for reasoning the result of application of predicates to arguments. Case splitting for covering all possibilities in the model can be done with interactions between users and CafeOBJ system.

4.2 Example of Verification

In this section, we show an example of verifications by using the formal description of WSP shown in the previous section.

4.2.1 Property to Be Proved

One of risks in the document flows model of WSP is states which satisfy that “there exists one or more reports which are forged after checking the report with order which is approved in the sales division”. And, one of properties we want to prove in the DFB model of WSP is that “all reports which are in sales division and have been checked with order are not forged”. This property says reports are not forged after all of regular and of control transitions have been performed in WSP. It does not always hold that all transitions can be performed, because control transitions might stop the process in order to avoid risks. By proving that this property is an invariant property, we can show the effectiveness of control activities in WSP. The property can be described as follows in CafeOBJ in module INV.

\[
\text{mod INV} \{ \\
\quad \text{pr ( SALE-AND-SHIP ) } \\
\quad \text{op inv1 : State DocumentID } \rightarrow \text{Bool} \\
\quad \text{var S : State} \\
\quad \text{var D : DocumentID} \\
\quad \text{eq inv1(S, D)} \\
\quad \quad = \left( \text{getDType(Documentation(S, D)) = report} \right) \land \left( \text{getDivision(Documentation(S, D)) = sales} \right) \land \left( \text{in?(ch(order), getEHistory(Documentation(S, D)))} \right) \implies \left( \text{Legal?(S, D) = true} \right). \}
\]

getDType(Documentation(S, D)) = report represents that the type of document D is report, getDivision(Documentation(S, D)) = sales represents that document D is in sales division, in?(ch(order), getEHistory(Documentation(S, D))) represents the predicate which checks if evidence ch(order) is in the evidence history of document D, and Legal?(S, D) = true represents the predicate which checks if document D is not forged.

4.2.2 Understanding Model while Proving

Induction cases are split by transitions and inv1(s, d) implies inv1(s’, d) does not hold in some cases, where s is an arbitrary variable of State, d is ID of document, and s’ is a successor state of s. Since case splitting in the proof constructed by interactions between CafeOBJ system and users covers all possible cases which the model can represent, states in some cases are not reachable states. In order to prove inv1 is an invariant property, we need a lemma, which shows that certain states are not reachable states wrt OTS \( S_{WSP} \) for WSP. Lemma discovery is helpful to understand the model, since it gives users new facts about the model.

By case splitting, we can find candidates of lemmas. An example of candidates is that “if order document approved in sales division is forged, receipt is not created”.

\[
\frac{\text{getDType(Documentation(S, D)) = report and getDivision(Documentation(S, D)) = sales} \land \text{in?(ch(order), getEHistory(Documentation(S, D)))}}{\text{inv1(S, D)}}
\]

\[
\frac{\text{Legal?(S, D) = true}}{\text{inv1(S', D)}}
\]

\[
\quad \text{inv1(S, D)} \quad \text{inv1(S', D)}
\]
While proving a property, we can understand the model by discovering lemmas, that is, we can obtain the knowledge about what should be done in the model to prove the property. If the proof is done successfully, we can analyse why control activities are enough for avoiding risks, otherwise we can analyse why control activities are not enough.

As the result of the verification, \( \text{inv1} \) is not satisfied in the DFB model for WSP. We have tried to prove \( \text{inv1} \) for the two cases described as follows:

- We assume that there is only one session for WSP, that is, session IDs of all documents are the same in initial states and ChangeSession does not occur.
- We assume that there are more than one session for WSP, that is, ChangeSession occurs in untrusted divisions.

In the former case, the proof is successfully done, however, in the latter case, it is not done successfully. In the latter case, by combinations of forging a document and changing a session, two documents have the same type and they cause failure of the proof. The lemmas that we have defined in the proof give us a deeper understanding of the relations between risks and control activities.

### 4.2.3 Improvement of Model of Document Flows

By modifying the WSP as follows, we can prove \( \text{inv1} \) for the OTS of new WSP described in Fig. 3.

- Approve order document in client before sending. This transition is an instance of Approve-1.
- Approve order document which has a piece of evidence of approval by client, after ack document is created in sales. This transition is an instance of Approve-2.

Other transitions are the same as WSP. Since two ack documents are created from the same order document in some behaviours defined in OTS for the original WSP, \( \text{inv1} \) is not satisfied. To avoid this situation, we have modified the WSP as above. Since evidence cannot be forged and the evidence in client is needed to approve order document in sales, according to the key assumptions of our modeling in Sect. 2.3.2, evidence is deleted if the document is forged. And client is a trusted division in this case, so the order document which is forged in sales division is never approved since it does not have evidence which shows approval by client. By putting these control activities, the proof to show OTS for the new WSP satisfies \( \text{inv1} \) is done successfully.

### 5. Related Work

It is common knowledge that business process modeling is effective for the development of enterprise information systems [16]. Some modeling notations, such as BPMN [14] and activity diagram of UML [13] based diagram notations are commonly used for modeling business process. These diagram notations are intuitively understood and easy to learn, however descriptions using this kind of notations might include ambiguity and lack strictness. Therefore, precise specifications and formal verification techniques have been proposed, such as Petri-net based approach [5]. The motivation of this paper is different from the work about formal modeling and verification of business processes. In our work, we focus on how business activities with control activities can achieve the business goal regardless of irregular activities performed. Our work is motivated by verification of cryptographic protocol such as [10] and [23], which assume existence of attackers, while conventional business process modeling and verification are focusing on correctness of sequences of actions which are performed for achieving business goals without attackers.

Techniques for formally modeling and verifying business processes are proposed from several points of view. We introduce some research about business process modeling. Some relate to document-based modeling and others relate to Internal Control issues, mainly compliance checking.

The work which has the most strong connection to our work is [15]. The idea of modeling business processes by focusing on flows of documents is based on [15]. In their work, business processes are specified in Maude [8] and verification of control activities against falsification of documents is done by model checking. In our work, verification is done by theorem proving with human interactions. Model checking needs much less interaction. However, model checkers face the state explosion problem, while theorem proving can check all reachable states during verification.
Since we have started from the same base, that is, modeling business processes focusing on document flows and verifying effectiveness of control activities, it would be the next challenge to research how the collaboration of our work can help to solve the problems in this area.

[25] gives semantics of documents by using DTL (Document Tree Logic), which is similar to computational tree logic in program analysis. By using documents and giving semantics, sequences of approval of a document can represent some workflow, and authorization can be analysed.

[7],[20],[21] are about modeling business processes based on business artifacts (or simply artifacts). Artifacts mean the information entities that capture process goals and allow for evaluating how thoroughly these goals are achieved. This modeling concept is similar to our approach, which is modeling business processes based on documents that record the information generated during business activities.

[17],[18] introduce a pattern based approach for modeling Internal Controls. They show control patterns used for implementation of control activities on business processes. Their work focuses on how to design control activities in business processes. On the other hand, we focus on formal descriptions and verification of existing business processes together with control activities.

[11],[12] explain how the logic behind the obligations and permissions of a business process and contracts can be made explicit in terms of deontic concepts to check compatibility between business processes and business contracts. While this work focuses on risks concerning the violation of contracts, we focus on the risks that appear in handling documents in business processes.

6. Conclusion and Future Work

In this paper, we aim to develop a method for precisely evaluating effectiveness of control activities. The evaluation can be done consistently and exhaustively, and the result of the evaluation can be discussed scientifically.

In our method, we apply a formal method to formalise and verify effectiveness of control activities. For formalisation, we focus on behaviours of documents flows for a business to model risks and control activities in business activities. And, for formal verification, we apply theorem proving for not only consistent and exhaustive evaluation, but also for giving scientific reasons of the results.

What we have done in this paper is

- defining how to construct DFB models,
- defining how to formalise DFB models based on OTS/CafeOBJ Method, and
- showing an example of verification of the properties of a DFB model for whole sales process and analysis of risks. The verification corresponds to rigorous evaluation of effectiveness of control activities.

As future work, we need more experience of analysis. Further experience of analysis would reveal correspondence between DFB models and the real world more precisely, possibly with an identification of more proper assumptions that would increase the reliability of results of the evaluation. It would also help us to find relations between risks control activities more deeply and such relations might indicate a more efficient way of the verification. It is also important to improve our method for larger scale applications such as [26] in the future.

References

[1] Pub. L. 107-204, 116 Stat. 754, Sarbanes Oxley Act (2002)
[2] The Committee of Sponsoring Organizations of the Treadway Commission (COSO), Internal Control - Integrated Framework, http://www.snaei.edu.cn/service/library/book/
0-Framework-final.pdf
[3] Financial Service Agency, Financial Instruments and Exchange Law, http://law.e-gov.go.jp/htmldata/S23/S23HO025.htm
[4] Financial Service Agency, On the Setting of the Standards and Practice Standards for Management Assessment and Audit Concerning Internal Control Over Financial Reporting (Council Opinions): Practice Standards, http://www.fsas.go.jp/en/news/2007/20070420.pdf
[5] W.M.P. van der Aalst, “Three good reasons for using a petri-net-based workflow management system,” in S. Navathe and T. Wakayama, ed., Proc. International Working Conference on Information and Process Integration in Enterprises (IPIC’ 96), Cambridge, Massachusetts, Nov. 1996.
[6] Y. Arimoto, Y. Watanabe, M. Kudo, and K. Futatsugi, “Checking assignments of controls to risks for internal control,” Proc. 2nd International Conference on Theory and Practice of Electronic Governance, IECGOV 2008, ACM press, Cairo, Egypt, Dec. 2008.
[7] K. Bhattacharya, C. Gerege, R. Hull, R. Liu, and J. Su, “Towards formal analysis of artifact-centric business processes models,” Business Process Management 2007, vol.4714/2007, Springer-Verlag, 2007.
[8] M. Clavel, F. Duran, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and C. Talcott, “All about maude — A high-performance logical framework,” Lect. Notes Comput. Sci., vol.4350, Springer-Verlag, 2007.
[9] R. Diaconescu and K. Futatsugi, AMAST Series in Computing vol.6, CafeOBJ report, World Scientific, 1998.
[10] S. Escobar, C. Meadows, and J. Meseguer, “A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties,” Theor. Comput. Sci., vol.367, no.1, pp.162–202, 2006.
[11] S. Goedertier and J. Vanthienen, “Designing compliant business processes with obligations and permissions,” Business Process Management Workshops, vol.3649/2005 of LNCS, pp.285–301, Springer-Verlag, 2006.
[12] G. Governatori, Z. Milosevic, and S. Sadiq, “Compliance checking between business processes and business contracts,” in Proc. 10th IEEE International Enterprise Distributed Object Computing Conference, IEEE, 2006.
[13] ISO/IEC 18501 Standard (2005), Information Technology — Open Distributed Processing — Unified Modeling Language (UML), Version 1.4.2.
[14] Object Management Group (2006), Business Process Modeling Notation Specification, Final Adopted Specification dtc/06-02-01.
[15] S. Iida, G. Denker, and C. Talcott, “Document logic: Risk analysis of business processes through document authenticity,” J. Research and Practice in Information Technology, vol.43, no.1, Australian Computer Society, 2011.
[16] S. Morimoto, “A survey of formal verification for business process modeling,” ICCS 2008, vol.5102/2008 LNCS, Springer-Verlag, 2008.
[17] K. Namiri and N. Stojanovic, “Pattern-based design and validation of business process compliance,” In OTM Conferences, LNCS, vol.4803, Springer-Verlag, 2007.
[18] K. Namiri and N. Stojanovic, “Using control patterns in business processes compliance,” in WISE 2007 Workshops, LNCS, vol.4832, Springer-Verlag, 2007.
[19] K. Futatsugi, J.A. Goguen, and K. Ogata, “Verifying design with proof scores,” Proc. 1st VSTTE, LNCS 4171, Springer, pp.277–290, 2008.
[20] C.E. Gerede and J. Su, “Specification and verification of artifact behaviours in business process models,” ICSOC 2007, LNCS, vol.4749/2007, Springer-Verlag, 2007.
[21] C.E. Gerede and K. Bhattacharya, “Static analysis of business artifact-centric operational models,” Proc. IEEE International Conference on Service-Oriented Computing and Applications (2007), IEEE, 2007.
[22] K. Ogata and K. Futatsugi, “Proof score approach to verification of liveness properties,” IEICE Trans. Inf. & Syst., vol.E91-D, no.12, pp.2804–2817, Dec. 2008.
[23] L.C. Paulson, “The inductive approach to verifying cryptographic protocols,” J. Computer Security Archive, vol.6, no.1-2, IOS Press, Amsterdam, 1998.
[24] M. Sasano, Naibutousei no Nyuumon to Jissen (Introduction and Practice of Internal Control), 2nd ed. Chuou Keizaisha, 2007.
[25] H. Sato, “Analyzing semantics of documents by using a program analysis method,” Proc. 33rd Annual IEEE International Computer Software and Applications Conference, IEEE, 2009.
[26] O. Takaki, I. Takeuti, T. Seino, N. Izumi, and K. Takahashi, “Incremental verification of consistency properties of large-scale workflows from the perspectiveness of control flow and evidence lifecycle,” International Journal on Advances in Software, vol.2, no.1, pp.147–161, 2009.

Kokichi Futatsugi

Kokichi Futatsugi has been a professor of JAIST since 1993. Before getting a full professorship at JAIST, he worked at ETL (Electrotechnical Laboratory), MITI (Ministry of International Trade and Industry), Japanese Government and was appointed the Chief Researcher of ETL in 1992. He worked also at SRI International (Stanford Research Institute), California for 1983-1984 as a visiting researcher. His research interests include formal specification languages, formal methods, systems verifications, software requirements/specifications, and his current research activities are centered around formal specification and verification based on CafeOBJ (an executable formal specification language). He got Fellow Award of Japan Society for Software Science and Technology in 2008, and is now advise/editor of international journals like IJSI (www.ijsi.org), JAL (www.elsevier.com/locate/jal), JHOSC (www.springerlink.com/content/102420/), JOT (www.jot.fm).

Yasuhito Arimoto

Yasuhito Arimoto is a Ph.D. candidate of School of Information Science at Japan Advanced Institute of Science and Technology. His research interests include formal methods, business process modeling, and domain modeling.

Shusaku Iida

Shusaku Iida is a professor of School of Network and Information at Senshu University in Kawasaki, Kanagawa, Japan. He received his PhD in Information Science from Japan Advanced Institute of Science and Technology in 1999. His research interest include algebraic specification language, formal models of social systems, and business process modeling.