Possibilistic Information Flow Control for Workflow Management Systems*

Thomas Bauereiss Dieter Hutter
German Research Center for Artificial Intelligence (DFKI)
Bibliothekstr. 1
D-28359 Bremen, Germany
thomas.bauereiss@dfki.de hutter@dfki.de

In workflows and business processes, there are often security requirements on both the data, i.e. confidentiality and integrity, and the process, e.g. separation of duty. Graphical notations exist for specifying both workflows and associated security requirements. We present an approach for formally verifying that a workflow satisfies such security requirements. For this purpose, we define the semantics of a workflow as a state-event system and formalise security properties in a trace-based way, i.e. on an abstract level without depending on details of enforcement mechanisms such as Role-Based Access Control (RBAC). This formal model then allows us to build upon well-known verification techniques for information flow control. We describe how a compositional verification methodology for possibilistic information flow can be adapted to verify that a specification of a distributed workflow management system satisfies security requirements on both data and processes.

1 Introduction

Computer-supported workflows and business process automation are widespread in enterprises and organisations. Workflow management systems support the enactment of such workflows by coordinating the work of human participants in the workflow for human activities as well as by automatically executing activities that can be mechanised. Graphical notations such as BPMN allow for the specification of workflows in an intuitive way. In addition to the control and data flows, there are typically various security requirements that need to be considered during the design, implementation and execution of workflows. A well-known security requirement on workflows is separation of duty for fraud prevention [7]. Confidentiality of data is another important security requirement, e.g. the confidentiality of medical data from non-medical personnel. These two can be seen as examples for different types of security requirements. On the one hand, there are security requirements on processes, i.e. constraints on the control flow and the authorisation of users, and on the other hand, there are security requirements on data, i.e. constraints on the flow of information. Several proposals to extend BPMN with graphical notations for both kinds of security requirements exist [6, 26, 33].

In this paper, we focus on the question of how the semantics of such a notation can be defined and how to use them to formally verify both types of security requirements. We do this on an abstract level without having to refer to details of enforcement mechanisms such as role-based access control (RBAC). For this purpose, we model the behaviour of a workflow as a set of traces of events, each representing a possible run of the workflow, and formalise our security requirements in a declarative way as properties of such trace sets. We map process requirements such as separation of duty to sets of allowed traces, corresponding to safety properties [3], whereas we map requirements on data to information flow properties,

*This research is supported by the Deutsche Forschungsgemeinschaft (DFG) under grant Hu737/5-1, which is part of the DFG priority programme 1496 "Reliably Secure Software Systems."

Kordy, Mauw, and Pieters (Eds.):
Graphical Models for Security 2014 (GraMSec'14)
EPTCS 148, 2014, pp. 47–62, doi:10.4204/EPTCS.148.4

which have been extensively studied [27, 15, 36, 21, 9]. This allows us to verify the absence not only of direct information leaks via unauthorised access, but also of indirect information leaks via observing the behaviour of the system. For example, if the control flow depends on a confidential data item and an unauthorised user observes which path of the control flow has been taken, they might be able to deduce the confidential value of the data item.

The relation between possibilistic information flow and safety properties is not trivial due to the refinement paradox, i.e. enforcing a safety property by removing disallowed traces might introduce new information leaks [18]. We discuss this relation for the case of separation of duty, give sufficient conditions for the compatibility with information flow properties, and show that these conditions are satisfied in our example setting.

We build upon the MAKS framework for possibilistic information flow control [15], which is suitable for formulating and verifying information flow policies at the specification level, and in which many information flow properties from the literature can be expressed. We describe how a compositional verification methodology [13] can be applied to verify our system models, which has the advantage that we can split up the verification task into separate verification tasks of the individual activities that make up the overall workflow.

Essentially, our approach allows for the formal modelling of workflows and the verification of security requirements on data and processes at a high level of abstraction. We have verified our results using the interactive theorem prover Isabelle/HOL [24]. To improve practicality, future work will focus on refinement approaches of these specifications towards concrete implementations, while preserving as much as possible of the security properties established on the abstract level. The long-term goal of our work is to facilitate the step-wise development of secure workflow management systems, starting from an abstract specification, derived from a workflow diagram, for example, then performing a series of refinement steps and eventually arriving at a secure implementation.

The rest of this paper is structured as follows. In the following subsection, we present a running example of a workflow that we will use for illustration throughout this paper. Section 2 introduces the system model. In Section 3, we elaborate on modelling confidentiality and separation-of-duty requirements, respectively. In Section 4, we describe how existing techniques for compositional verification of information flow properties can be applied and adapted for our workflow systems. Section 5 discusses related work and Section 6 concludes the paper.

1.1 Example Scenario

As a running example, we use the workflow depicted in Figure 1, adapted from an industry use-case described in an (unpublished) paper by A. Brucker and I. Hang. It models a hiring process including interviews and medical examinations. The swimlanes represent two departments of the organisation running the hiring workflow, the Human Resources (HR) and the medical department. The placement of activities in the swimlanes indicates the responsible department, and thus the authorised employees. The input and output relations of activities are depicted as directed flows of documents between activities.

We use this workflow to illustrate security requirements with respect to both data and process. On the process side, we demand separation of duty between the two medical examinations, i.e. they have to be performed by different medical officers, such that no single medical officer can manipulate the hiring process by rejecting unwanted candidates for fabricated medical reasons. Regarding confidentiality requirements, we assume that the two medical reports are highly confidential due to their potentially sensitive contents. In particular, they are confidential for the employees in the HR department, who should only be able to access information on a need-to-know basis, e.g. the CVs of applicants.

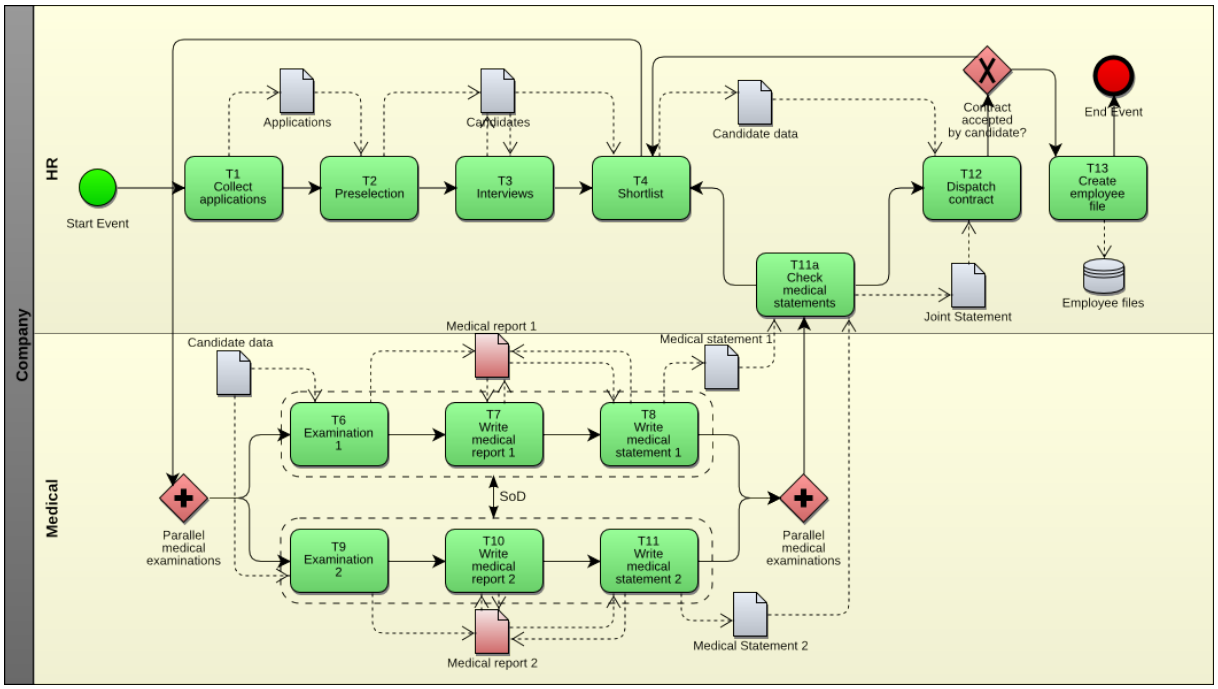

Figure 1: Example workflow (adapted from A. Brucker and I. Hang, unpublished)

In general, we assume there is a set of security domains, which are used to classify documents exchanged between activities, and a flow policy that specifies the allowed information flows between domains. We assign a domain to every document (a security classification) and to every employee (a security clearance). These classifications and clearances determine which users are allowed to participate in which activities of the workflow. For example, employees with an *HR* clearance are not allowed to participate in the activities creating the medical report; otherwise, they would have direct access to confidential medical data. We formally define the constraints regarding classifications, clearances and flow policies in Section 3.1.

Besides direct information flows via transfer of documents, we aim to control indirect flows, where confidential information is deducible from observations of the system behaviour. For example, an HR employee in the above workflow can deduce whether the two medical officers agreed about the fitness of the candidate by observing whether the workflow proceeds to activity 12 after the medical examinations or reverts to activity 4. This is acceptable and actually necessary in our scenario, as long as this is the only bit of information about the medical condition of the candidate that can be deduced by HR personnel. Our goal is to verify that the workflow indeed does not leak any additional medical information to non-medical personnel. In the next section, we begin by formally modelling the workflow, and then proceed to formalise the security requirements.

1.2 Preliminaries

We briefly recall the definitions of (state-) event systems and security predicates from the MAKS framework for possibilistic information flow [15] that we use in this paper. An event system $ES = (E, I, O, Tr)$ is essentially a (prefix-closed) set of traces $Tr \subseteq E^*$ that are finite sequences of events in the event set E.

The disjoint sets $I \subseteq E$ and $O \subseteq E$ designate input and output events, respectively. We denote the empty trace as $\langle\rangle$, the concatenation of traces α and β as $\alpha.\beta$, and the projection of a trace α onto a set E as $\alpha|_E$. In the composition $ES_1 \| ES_2$ of two event systems ES_1 and ES_2, input events of one system matching output events of the other system are connected (and vice versa) and thus become internal events of the composed system. The set of traces is the set of interleaved traces of the two systems, synchronised on events in $E_1 \cap E_2$:

$$Tr(ES_1 \| ES_2) = \{\alpha \in (E_1 \cup E_2)^* \mid \alpha|_{E_1} \in Tr(ES_1) \wedge \alpha|_{E_2} \in Tr(ES_2)\}$$

A state-event system $SES = (E, I, O, S, s_0, T)$ has a set of states S, a starting state $s_0 \in S$, and a transition relation $T \subseteq (S \times E \times S)$. The event system *induced* by a state-event system has the same sets of events and the set of traces that is enabled from the starting state via the transition relation.

The MAKS framework defines a collection of basic security predicates (BSPs). Many existing information flow properties from the literature can be expressed as a combination of these BSPs. Each BSP is a predicate on a set of traces with respect to a view $\mathcal{V}$. A view $\mathcal{V} = (V, N, C)$ on an event system $ES = (E, I, O, Tr)$ is defined as a triple of event sets that form a disjoint partition of E. The set V defines the set of events that are visible for an observer, C are the confidential events, and the events in N are neither visible nor confidential. Notable examples for BSPs, that we will use in this paper, are backwards-strict deletion (*BSD*) and backwards-strict insertion of admissible confidential events (*BSIA*)[1], defined in [19] as follows:

$$\begin{aligned} BSD_{\mathcal{V}}(Tr) \equiv & \forall \alpha, \beta \in E^*. \forall c \in C. (\beta.c.\alpha \in Tr \wedge \alpha|_C = \langle\rangle) \\ & \Rightarrow \exists \alpha' \in E^*. \left(\alpha'|_V = \alpha|_V \wedge \alpha'|_C = \langle\rangle \wedge \beta.\alpha' \in Tr\right) \\ BSIA_{\mathcal{V}}(Tr) \equiv & \forall \alpha, \beta \in E^*. \forall c \in C. (\beta.\alpha \in Tr \wedge \alpha|_C = \langle\rangle \wedge \beta.c \in Tr) \\ & \Rightarrow \exists \alpha' \in E^*. \left(\alpha'|_V = \alpha|_V \wedge \alpha'|_C = \langle\rangle \wedge \beta.c.\alpha' \in Tr\right) \end{aligned}$$

Intuitively, the former requires that the occurrence of confidential events must not be deducible, while the latter requires that the *non*-occurrence of confidential events must not be deducible. Technically, they are closure properties of sets of traces. For example, if a trace in Tr contains a confidential event, then *BSD* requires that a corresponding trace without the confidential event exists in Tr that yields the same observations. This means the two traces must be equal with respect to visible V-events, while N-events might be adapted to correct the deletion of the confidential event.

2 System Model

In order to verify that a workflow satisfies given security requirements, we need a formal model of workflows and their behaviour. We first define our notion of workflows. For simplicity, we omit aspects such as exceptions or compensation handling, but our definition suffices for our purpose of discussing the verification of security requirements for workflows.

Definition 1. *A workflow* $W = (\mathcal{A}, Docs, SF, MF, U)$ *consists of*

- *a set* $\mathcal{A}$ *of activities,*

[1] In [19], *BSIA* is defined with an additional parameter ρ that allows to strengthen the property by further specifying positions at which confidential events must be insertable. For simplicity, we choose to fix this parameter to ρ_E in the notation of [19], i.e we only require confidential events to be insertable into a trace without interfering with observations if they are in principle admissible exactly at that point in the trace.

- *a set Docs of data items,*
- *a set* $SF \subseteq (\mathcal{A} \times \mathcal{A})$ *of sequence flows, where* $(a_1, a_2) \in SF$ *represents the fact that upon completion of* a_1*, it may send a trigger to* a_2 *signalling it to start execution, and*
- *a set* $MF \subseteq (\mathcal{A} \times Docs \times \mathcal{A})$ *of message flows, where* $(a_1, d, a_2) \in MF$ *represents data item* d *being an output of activity* a_1 *and an input to* a_2*, and*
- *a set* U *of users participating in the workflow.*

The sets $\mathcal{A}$ and $Docs$ correspond to the nodes of a workflow diagram such as Figure 1, while SF and MF correspond to the solid and dashed edges, respectively.

We define the behaviour of workflows, not in a monolithic way, but in terms of the behaviours of components representing activities communicating with each other. As we will show in Section 4, this simplifies the verification, because it allows us to use the decomposition methodology of [13] to verify the security of the overall system by verifying security properties of the subcomponents. We believe that such a decomposition approach can help in scaling up verification of information flow properties to larger systems.

Each activity a is therefore modelled as a state-event system $SES_a = \left(E_a, I_a, O_a, S_a, s_a^0, T_a\right)$ analogously to Definition 3 of [13]. The set of events E_a consists of events of the form

- $Start_a(u)$, starting the activity a and assigning it to the user $u \in U$,
- $End_a(u)$, marking the end of the activity,
- $Send_a(a', msg)$ and $Recv_a(a', msg)$, representing the sending (or receiving, respectively) of a message msg from activity a to activity a' (or vice versa),
- $Setval_a(u, i, val)$ and $Outval_a(u, i, val)$, representing a user $u \in U$ reading (or setting, respectively) the value val of data item i, and
- a set of internal events τ_a.

We denote the set of events of a given activity $a \in \mathcal{A}$ as E_a, and the set of all events in a workflow as $E_W = \bigcup_{a \in \mathcal{A}} E_a$. We denote the set of events of a given user $u \in U$ as $E_u = \{Start_a(u) \mid a \in \mathcal{A}\} \cup \{Setval_a(u, i, val) \mid a \in \mathcal{A}, i \in Docs, val \in Val\} \cup \{Outval_a(u, i, val) \mid a \in \mathcal{A}, i \in Docs, val \in Val\} \cup \{End_a(u) \mid a \in \mathcal{A}\}$, and the set of all user interaction events as $E_U = \bigcup_{u \in U} E_u$. The messages between activities can have the form

- $Trigger$, used to trigger a sequence flow to a successor activity in the workflow,
- $Data(i, v)$, used to transfer the value v for data item i, and
- $AckData(i)$, used to acknowledge the receipt of a data item.

Using separate messages for data and sequence flows is inspired by the BPMN standard, which describes its (informal) execution semantics in terms of tokens that are passed from one activity to the next, representing control flow separately from data flows. In addition, this separation simplifies the modelling of confidentiality, as it becomes straightforward to classify events transporting $Data$ messages into confidential or non-confidential events based on the classification of the data items they transport.

The local states of the activities include program variables such as a program counter and a mapping $Mem\colon Docs \to Val$, storing the values of data items. After initialisation, the activity waits for messages from other activities, transferring input data or triggering a sequence flow. When one (or more) of the incoming transitions have been triggered, the activity internally computes output messages (possibly via interaction with users), sends them via the outgoing data associations, and triggers outgoing sequence flows. In Appendix A, we formally specify two types of activities as examples, namely user activities that allow users to read and write data items, and gateway activities that make a decision on the control flow based on the contents of their input data items.

Each of these state-event systems SES_a induces a corresponding event system ES_a. The overall system then emerges from the composition of these event systems ES_a for every activity $a \in \mathcal{A}$, together

with a communication platform ES_P:

$$ES_W = (\|_{a \in \mathcal{A}} ES_a) \| ES_P$$

We call ES_W the *workflow system* for the workflow W. We reuse the communication platform of [13], which is formally specified in Section 2.3 of [13]. It asynchronously forwards messages between the activities. As we do not assume that it provides guarantees regarding message delivery, its specification is very simple.[2] Upon composition with the platform, the communication events between the activities become internal events of the composed system. Only the communication events with users remain input and output events. These events form the user interface of the workflow system.

A simple version of our example workflow can be represented as a composition of instances of the activity types specified in Appendix A. We can represent the activity T11a in Figure 1 as a gateway that decides on the control flow based on the results of the medical examinations: If they are positive, the workflow continues with dispatching the contract, otherwise it goes back to selecting another candidate from the shortlist. The other activities essentially consist of users reading and generating documents, so we can represent them as user activities. Of course, these activities can be enriched with further details, e.g. the interviews can be expanded to subprocesses of their own, but we assume that this is handled in a subsequent refinement step and consider only the abstract level in this paper.

3 Security Policies

3.1 Confidentiality

We assign security domains from a set $\mathcal{D}$ of domains to the data items exchanged between the activities of the workflow. We denote this domain assignment function by $dom\colon Docs \to \mathcal{D}$. A flow policy is a reflexive and transitive relation on domains and specifies from which domains to which other domains information may flow.[23] Note that, even though we focus on confidentiality in this paper, also integrity requirements can be seen as a dual to confidentiality and handled using information flow control. For example, in [23] a lattice of combined security levels is built as a product of a confidentiality lattice and an integrity lattice. For our example workflow, we only require two confidentiality domains HR and Med. The medical reports $MedReport1$ and $MedReport2$ created by activities T6–T8 and T9–T11 in Figure 1 are assigned to the Med confidentiality domain, and other data items to the HR domain. The example flow policy states that information may flow from HR to Med, but not vice versa, i.e. $HR \rightsquigarrow Med$ and $Med \not\rightsquigarrow HR$ (see Figure 2).

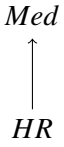

Figure 2: Flow policy

Users read and write the contents of data items via the inputs and outputs of activities they participate in. In order to exclude unwanted direct information flows, we have to make sure that the classifications of the data items that users work with are compatible with their clearances. A straightforward approach is to enforce a Bell-LaPadula style mandatory access control. This can be formulated in terms of classifications that are assigned to activities based on the classifications of their inputs and outputs:

Definition 2. *An activity classification $cl_{\mathcal{A}}\colon \mathcal{A} \to \mathcal{D}$ is an assignment of domains to activities such that*

[2] However, it is possible to adapt the decomposition methodology to other communication models, e.g. providing some notion of reliability of message delivery or means for synchronous communication. For example, see Appendix A for some remarks on guaranteeing a notion of ordered message delivery.

1. *for all input data items i of an activity a, $dom(i) \rightsquigarrow cl_{\mathcal{A}}(a)$, and*
2. *for all output data items i of an activity a that may be assigned to untrusted users, $cl_{\mathcal{A}}(a) \rightsquigarrow dom(i)$.*

We allow users to participate in an activity of a given classification only if they have a matching clearance. We denote the mapping of users to clearances as $cl_U : U \to \mathcal{D}$. The conditions in Definition 2 correspond to the Simple Security and $*$-Property of the Bell-LaPadula model, respectively. Note that we relax the $*$-Property by allowing trusted users to downgrade data items. Otherwise, we would not be able to assign a classification to the activities T8 and T11 in our example workflow, because they have high inputs (the medical reports) and low outputs (the statements about the final result of examinations). However, this specific flow of information in the example is acceptable and necessary, because the output should contain only the non-confidential final decision of the medical officer[3] required by the HR department, while the detailed content of the medical reports remains classified as confidential. Essentially, we admit inputs and outputs of trusted users to act as a channel for declassification that is not formally controlled by our information flow analysis. It would be possible to model declassification more explicitly, e.g. using intransitive flow policies [17], but for simplicity we choose this solution for this paper. The same approach is followed in [33], for example. See [32] for an early discussion of this approach to downgrading and [28] for a general overview of principles and dimensions of declassification.

Regardless of whether trusted users are present or not, we want to verify that the system itself does not leak information about data items i with classification $dom(i) \not\rightsquigarrow d$ to users with clearance d. The set of confidential events for a domain d thus consists of events setting or reading values of these data items, while events of activities whose classification is allowed to flow into d are considered to be potentially observable for users in domain d:

Definition 3. *Let $d \in \mathcal{D}$ be a domain. The security view on a workflow system ES_W for d is defined as $\mathcal{V}_d = (V_d, N_d, C_d)$, where*

$$
\begin{aligned}
V_d &= \bigcup_{cl_{\mathcal{A}}(a) \rightsquigarrow d} E_a \\
C_d &= \{Setval_a(u,i,val) \mid \exists u \in U, i \in Docs, v \in Val.\ dom(i) \not\rightsquigarrow d \wedge cl_{\mathcal{A}}(a) \not\rightsquigarrow d\} \\
&\quad \cup \{Outval_a(u,i,val) \mid \exists u \in U, i \in Docs, v \in Val.\ dom(i) \not\rightsquigarrow d \wedge cl_{\mathcal{A}}(a) \not\rightsquigarrow d\} \\
N_d &= E \setminus (V_d \cup C_d)
\end{aligned}
$$

The set C_d contains the confidential input and output events.[4] Note that we assume that confidential information enters the system only via user input or output, and that the system does not generate confidential information by itself (e.g. by generating cryptographic key material). If that were the case, the corresponding system events would have to be added to the set C_d. Moreover, it is worth pointing out that we consider certain other types of information to be *non-confidential*. In particular, the information whether an activity has been performed or not, or the information which user has performed which activity is considered to be non-confidential. Again, such requirements could be captured by formulating the security view accordingly. For our setting, the above view reflects our security requirement that the values of confidential data items should be kept secret. Hence, we use this view for the rest of this paper.

In Section 4, we describe how to verify that a given workflow system satisfies the security predicate $BSD_{\mathcal{V}_d} \wedge BSIA_{\mathcal{V}_d}$ with respect to this view for every domain d. It expresses that confidential user inputs and outputs can be deleted or inserted without interfering with the observations of users in domain d.

[3] Which can be further enforced by allowing only Boolean values as content of the low output.

[4] The set C_d only contains events of activities a with $cl_{\mathcal{A}}(a) \not\rightsquigarrow d$, because activities with $cl_{\mathcal{A}}(a) \rightsquigarrow d$ are considered to be visible, and the set of visible and confidential events must be disjoint.

3.2 Separation of Duties

As discussed in the introduction, separation of duties is another common security requirement in workflow management systems. Separation of duties can be formally defined as a safety property [3]. The "bad thing" happens when the same user participates in two activities constrained by separation of duty, hence we only allow traces where this does not occur.

Definition 4. *Let $a, a' \in \mathcal{A}$ be two activities. We call the set of traces*

$$\left\{\alpha \in E_W^* \mid \forall u, u' \in U.\ \forall e_1, e_2 \in \alpha.(e_1 \in (E_a \cap E_u) \wedge e_2 \in (E_{a'} \cap E_{u'})) \rightarrow u \neq u'\right\}$$

a separation-of-duty property $P_{SoD}^{a,a'}$.

As we have modelled user assignment explicitly as events, this property can also be characterised by requiring that 1. constrained activities are assigned to different users, and 2. users may participate in an activity only after they have been assigned to it:

$$\begin{aligned} P_{SoD}^{a,a'} \supseteq &\left\{\alpha \in E_W^* \mid \forall u, u' \in U.\ Start_{a_1}(u) \in \alpha \wedge Start_{a_2}(u') \in \alpha \longrightarrow u \neq u'\right\} \\ &\cap \left\{\alpha \in E_W^* \mid \forall a \in \mathcal{A}, u \in U, e \in (E_a \cap E_u).\ Start_a(u) \notin \alpha \longrightarrow e \notin \alpha\right\} \end{aligned}$$

A system with a set of traces Tr and events E satisfies such a property iff $Tr \subseteq P_{SoD}^{a,a'}$. In our example workflow, there are separation of duty constraints between the activities belonging to the two medical examinations ($T6$–8 in Figure 1 one the one hand, and $T9$–11 on the other hand). Hence, we want to enforce $P_{SoD}^{a,a'}$ for the pairs $(a,a') \in \{T6, T7, T8\} \times \{T9, T10, T11\}$.

Similarly, other runtime-enforceable security policies [30] can be modelled as safety properties. In this paper, we focus on the above notion of separation of duty as an example and investigate its relation to information flow in Section 4.2.

4 Verification

4.1 Information Flow Security

To ease the verification of the security of a workflow system, we decompose it into the individual activities of the workflow and make use of the methodology presented in [13] to verify the resulting distributed system. For each domain $d \in \mathcal{D}$, we verify that users in that domain can learn nothing about information that is confidential for them. The first step of the methodology [13] is to partition the activities into a set of low activities $\mathcal{A}_L^d = \{a \in \mathcal{A} \mid cl_{\mathcal{A}}(a) \rightsquigarrow d\}$ that are (potentially) visible in domain d and a set of high activities $\mathcal{A}_H^d = \{a \in \mathcal{A} \mid cl_{\mathcal{A}}(a) \not\rightsquigarrow d\}$ that are not visible and may handle confidential information.[5] It follows that $\mathcal{V}_d$ from Definition 3 is a global security view as defined in [13, Definition 13], i.e. the visible events are exactly the events of the low activities, the set of confidential events is a subset of the events of the high activities, and the remaining events are non-visible and non-confidential.

The second step is finding suitable local views $\mathcal{V}_d^a$ for high activities $a \in \mathcal{A}_H^d$ in order to verify that they do not leak confidential information to low activities. Hence, we cannot generally treat communication events of these activities as N-events, as we did in the global view, but we have to consider some of them as V-events (e.g. a high activity sending a trigger or a declassified data item to a low activity)

[5] In [13], the set $\mathcal{A}_L^d$ is called the set of observers, while $\mathcal{A}_H^d$ is called the set of friends. This might be a bit counterintuitive in our setting for some readers, as the friends would be the activities that are *not* visible. To avoid confusion, we simply speak of low and high activities, respectively.

and some of them as C-events (e.g. a high activity receiving a confidential data item). Intuitively, this means we split each of these activities into a part that visibly interacts with low activities and a part that handles confidential data, and verify that the latter does not interfere with the former. Technically, these local views satisfy certain constraints that allow us to instantiate the compositionality result of [13], as we discuss below.

Definition 5. *Let $d \in \mathcal{D}$ be a domain, and $a \in \mathcal{A}^d_H$ be a high activity for d. Furthermore, let $Docs^C_d = \{i \in Docs \mid dom(i) \not\rightsquigarrow d\}$ denote the set of data items that are confidential for d. The* local view for *a is defined as $\mathcal{V}^a_d = (V^a_d, N^a_d, C^a_d)$ with*

$$V^a_d = (I_a \cup O_a) \setminus \bigcup_{i \in Docs^C_d} E_i$$

$$C^a_d = \bigcup_{i \in Docs^C_d} (E_i \setminus \{Send_a(b,m) \mid \exists v.\ m = Data(i,v) \vee m = AckData(i)\})$$

$$N^a_d = E_a \setminus (V^a_d \cup C^a_d)$$

where the set E_i of high communication events containing data item i is defined as

$$E_i = \big\{e \mid \exists b \in \mathcal{A}^d_H, m, u, v.\ (m = Data(i,v) \vee m = AckData(i)) \\ \wedge (e = Send_a(b,m) \vee e = Recv_a(b,m) \vee e = Setval_a(u,i,v) \vee e = Outval_a(u,i,v))\big\}$$

Combining these local views, we define the composed view for *d as $\mathcal{V}_{d^+} = (V_{d^+}, N_{d^+}, C_{d^+})$ where*

$$V_{d^+} = \bigcup_{a \in \mathcal{A}^d_H} V^a_d \cup \bigcup_{a \in \mathcal{A}^d_L} E_a \qquad C_{d^+} = \bigcup_{a \in \mathcal{A}^d_H} C^a_d \qquad N_{d^+} = E_W \setminus (V_{d^+} \cup C_{d^+})$$

Note that the combined view $\mathcal{V}_{d^+}$ is *stronger* than our global view $\mathcal{V}_d$ in the sense that more events are considered confidential or visible for an observer in domain d. Theorem 1 of [19] tells us that $BSD_{\mathcal{V}_{d^+}} \wedge BSIA_{\mathcal{V}_{d^+}}$ for the stronger view implies $BSD_{\mathcal{V}_d} \wedge BSIA_{\mathcal{V}_d}$.

Also note that all communication events with low activities are considered visible, and that the forwarding of confidential data items from one high activity to another is considered non-confidential. The justification for this is that secrets enter and leave the subsystem of high activities through communication with users and low activities, and the forwarding between high activities can be considered as internal processing. Hence, we can use communication events between high activities for correcting perturbations caused by inserting or removing confidential user inputs. We make use of this fact in the proof of the following theorem, which states the security of activities as we have specified them in Appendix A in terms of the transition relations T^{gen}_a, T^{user}_a and $T^{gw(Cond)}_a$.

Theorem 1. *Let W be a workflow, $d \in \mathcal{D}$ a domain and SES_a for $a \in \mathcal{A}$ an activity. If the transition relation of SES_a is*

- *$T^{gen}_a \cup T^{user}_a$, or*
- *$T^{gen}_a \cup T^{gw(Cond)}_a$ and Cond does not depend on confidential data for d,*

then $BSD_{\mathcal{V}^a_d}(Tr_a) \wedge BSIA_{\mathcal{V}^a_d}(Tr_a)$ holds.

The proof of this and the following theorems can be found in the extended version of this paper [5]. We use the unwinding technique [16] for the proof. Note that since the generic transition relation T^{gen}_a and the activity-specific transition relations are disjoint, we can partition this proof into a generic part that covers the events and states used in T^{gen}_a, and an activity-specific part. Therefore, if we want to use a different kind of activity than the ones specified in this paper, and we reuse the generic part T^{gen}_a of the transition relation, then we can also reuse most of this proof.

The next step is to instantiate the compositionality result of [13], which states that the security of the overall system with respect to the global security view is implied by the security of the subsystems with respect to their local views. However, our local views do not quite satisfy the requirement of being *C-preserving* in the sense of Definition 18 of [13], because that definition disallows *N*-events in the communication interface between subsystems. Hence, we slightly adapt the notion C-preserving views, allowing *Send* events to be in *N*:

Definition 6. *Let $\mathcal{A}_H^d \subseteq \mathcal{A}$ and $C \subseteq E_{\mathcal{A}_H^d}$. A family $(\mathcal{V}^a)_{a\in\mathcal{A}_H^d}$ of views $\mathcal{V}^a = (V_a, N_a, C_a)$ for E_a is* C-preserving for *C iff*

1. *$a \in \mathcal{A}_H^d$ and $b \notin \mathcal{A}_H^d$ implies $\forall m.Send_a(b,m) \in V_a \wedge Recv_a(b,m) \in V_a$.*
2. *$a, a' \in \mathcal{A}_H^d$ implies*
   (a) *$Recv_{a'}(a,m) \in C_{a'}$ iff $Send_a(a',m) \notin V_a$ and*
   (b) *$Recv_{a'}(a,m) \in V_{a'}$ iff $Send_a(a',m) \in V_a$*
3. *$C \cap E_a \subseteq C_a$ for all $a \in \Phi$.*

As can be easily seen, our local views are C-preserving for the set of global confidential events C_d from Definition 3: communication with low activities is visible, corresponding *Recv* and *Send* events are either visible or non-visible (where non-visible *Recv* events need to be confidential, while the corresponding *Send* events are allowed to be treated as *N*-events), and events that are confidential in the global view are confidential for the local views.

It turns out that the compositionality result of [13] still holds for our weakened notion of C-preserving local views; a sufficient (but not necessary) condition is that the subsystems satisfy not only *BSD* (as in [13]), but *BSD* and *BSIA*, which our activities happen to do.

Theorem 2. *Let W be a workflow, $ES_W = (\|_{a\in\mathcal{A}} ES_a) \| ES_P$ be a workflow system, $\mathcal{V}_d$ be a global security view for domain d, and $\left(\mathcal{V}_d^a\right)_{a\in\mathcal{A}_H^d}$ be a family of local views that is C-preserving for C_d. If for all $a \in \mathcal{A}_H^d$, ES_a satisfies $BSD_{\mathcal{V}_d^a} \wedge BSIA_{\mathcal{V}_d^a}$, then ES_W satisfies $BSD_{\mathcal{V}_{d^+}} \wedge BSIA_{\mathcal{V}_{d^+}}$ and, therefore, $BSD_{\mathcal{V}_d} \wedge BSIA_{\mathcal{V}_d}$.*

Note that, if other kinds of activities than the ones from Appendix A should be part of the workflow, it is only required to prove that their specifications also satisfy the security predicates for the local views, in order to show that the overall workflow satisfies the information flow security predicates.

We have formalised and verified our model and proofs using the interactive theorem prover Isabelle [24]. Our development is based on a formalisation of the MAKS framework developed by the group of Heiko Mantel at TU Darmstadt (unpublished as of this writing). We intend to make our formalisation publicly available when the MAKS formalisation is released.

Conceptually, the main difference between our workflow management systems and the shopping mall system described in [13] lies in the relation between users and the system. In the shopping scenario, there is a one-to-one correspondence between users and software agents running in the system. Communication with the users happens only during initialisation, when users write their preferences into the initial memory of their agents, which run autonomously thereafter. In our workflow systems, the interaction is much more dynamic, as multiple activities can be assigned to the same user at runtime and there is ongoing communication between users and the system. This has impact on the system model — we introduced additional events for user interaction — and the construction of views. The partitioning into high and low activities is based on classifications of data items and activities, and access control has to ensure that only users with a matching clearance can participate in an activity, so that our security views are actually in line with the possible runtime observations of users. Despite these differences, we have seen that the methodology of [13] can be applied with small technical adjustments.

4.2 Compatibility with Separation of Duties

As described in Section 3.2, we can formalise constraints such as separation of duty as safety properties. Having established information flow security of our workflow system, we now ask whether these security properties are preserved when enforcing separation of duty constraints. In general, this is not the case. Altering a system such that it satisfies a safety property can be seen as a refinement, and it is well-known that possibilistic information flow security is not preserved under refinement in general [18]. Consider, for example, the security predicate *BSIA*. Repeatedly inserting confidential events of different users into a trace can exhaust the possible user assignments that would satisfy the separation of duty constraints, thus deadlocking the process and making further visible observations impossible. We can, however, try to find sufficient conditions under which information flow properties are preserved:

Theorem 3. *Let $ES = (E, I, O, Tr)$ be an event system and $\mathcal{V} = (V, N, C)$ be a view for ES. Let $E_a, E'_a \subseteq E$ be two disjoint sets of events corresponding to activities a and a', and let $P_{SoD}^{a,a'}$ be an SoD property. Let $E_u \subseteq V \cup C$ be the communication events with a user u and $E_U = \bigcup_{u \in U} E_u$ the set of all user events. If*

1. *user assignment is non-confidential, i.e. there is a set $E^{assign} \subseteq E \setminus C$ of assignment events, and a user u may only participate in an activity after having been assigned to it via an event from $E^{assign} \cap E_u$, or*
2. *only confidential or only visible user I/O events of activities a and a' are enabled in ES, i.e. there is a set $E^{disabled} \subseteq E$ of events that never occur in a trace of ES, and $V \cap (E_a \cup E'_a) \cap E_U \subseteq E^{disabled}$ or $C \cap (E_a \cup E'_a) \cap E_U \subseteq E^{disabled}$ holds, or*
3. *the SoD constraint between a and a' is already enforced by ES, i.e. $Tr \subseteq P_{SoD}^{a,a'}$,*

then $BSD_{\mathcal{V}}(Tr) \wedge BSIA_{\mathcal{V}}(Tr)$ implies $BSD_{\mathcal{V}}(Tr \cap P_{SoD}^{a,a'}) \wedge BSIA_{\mathcal{V}}(Tr \cap P_{SoD}^{a,a'})$.

In our running example, we can choose $E^{assign} = \{Start_a(u) \mid u \in U\}$ and apply the first case of the theorem for the workflow system ES_W and a view $\mathcal{V}_{d^+}$, because only the details of the results of the medical examinations are confidential, not the information who carried out the examinations. Furthermore, in case $cl_{\mathcal{A}}(a) \neq cl_{\mathcal{A}}(a')$, the mandatory access control described in Section 3.1 already enforces SoD statically, so the third condition also applies. In general, Theorem 3 gives us sufficient conditions for the compatibility of SoD constraints and information flow properties, taking into account the classifications of events that are relevant for enforcing SoD. Similar results could be developed for other classes safety properties that are of interest in workflows, but we leave this as future work. Note that Theorem 3 is not specific to workflow systems as specified in this paper. It can be applied to any system where users perform different activities in the presence of separation of duty constraints.

5 Related Work

We build upon the MAKS framework for possibilistic information flow control [15], which is suitable for formulating and verifying information flow policies at the specification level. We have focused on confidentiality of data from unauthorised employees within the organisation, but in principle information flow control can be adapted to different attacker models and security policies by choosing the security views appropriately. Furthermore, approaches have been proposed to take into account factors such as communication over the Internet [12] or encrypted communication channels [14]. In [25], a connection between role-based access control (RBAC) and mandatory access control is drawn, which might be adapted to enforce the mandatory access control we described in Section 3.1 using RBAC mechanisms.

Early examples for workflow management systems with distributed architectures include [2, 22, 31]. Later, computing paradigms with a similar spirit have emerged, e.g. service-oriented architectures or cloud computing. We see these techniques and standards as complementary to our work, as they can be used for the implementation of our abstract specifications.

BPMN extensions to annotate business process diagrams with security annotations can be found in [6, 26, 33]. Closest to the security requirements considered by us comes the notation proposed in [33] that supports both the annotation of activities with separation of duty constraints and the annotation of documents and process lanes with confidentiality and integrity classifications or clearances, respectively.

Several proposals for a formal semantics of workflow specifications can be found in the literature. For example, [34] maps BPMN diagrams to CSP processes and describes how the formal semantics can be leveraged to compare and analyse workflow diagrams, e.g. with respect to consistency. It focuses on the control flow and does not model data flows. In [35], workflows are represented as statements in a workflow description language, which is mapped to a representation as hierarchical state machines. An information flow analysis algorithm is described, but the actual information flow property that it checks is not stated in a declarative, mechanism-independent way. [1] represents workflows as Petri nets and describes an approach for information flow analysis. The focus is on keeping the occurrence of tasks confidential, whereas our work focuses on the confidentiality of the data that is processed in the workflow. In [4] and [29], workflows are formalised as transition systems and model-checking is employed to verify properties specified as LTL formulas. This is suitable to verify safety or liveness properties, whereas the information flow predicates considered by us can be seen as hyperproperties [8].

6 Conclusion

Graphical notations such as BPMN are widely used for workflow specification. We have presented an approach to formally model both the behaviour of a workflow and the associated security requirements, and described how to apply the decomposition methodology of [13] and how to verify a distributed workflow management system with ongoing user interaction. We have shown that, even though possibilistic information is in general not refinement-closed, the enforcement of separation of duty is compatible with the information flow security of the system under certain assumptions.

We have sketched how a simple version of our example workflow can be represented as a composition of instantiations of the activity types specified in Appendix A. As we have shown the security of these activities in Theorem 1, we can use Theorem 2 to derive the security of the composed system from the security properties of the individual activities. This demonstrates how instantiations of a type of activities that has been proven secure once can be plugged into larger workflows in a secure way. Hence, we believe that this compositional approach can help in making verification techniques for information flow scale to larger workflow systems. However, more work is needed before this approach can actually be applied to realistic systems. For example, tool support for translating a more realistic subset of BPMN to our system model would be a major step in this direction, which would also help us evaluate our approach with a sample of existing workflows.

Moving from an abstract specification towards the implementation level is another important direction of future work. This paper deals with workflows on a high level of abstraction. We intend to work on notions of security-preserving refinement that allow us to expand abstract activities in a workflow into more concrete subprocesses and refine the behaviour of atomic activities towards an executable implementation. There is a large body of existing work that we can build upon for this purpose, such as action refinement for replacing atomic events on the abstract level with sequences of more concrete events [11],

switching between event-based and language-based notions of information flow [20], or directly generating executable code from specifications [10]. In the long term, we hope that these decomposition and refinement techniques will contribute to making the step-wise development of secure workflow systems from workflow diagrams to executable code more scalable and efficient.

Acknowledgements We thank Richard Gay, Sylvia Grewe, Steffen Lortz, Heiko Mantel and Henning Sudbrock for providing a formalisation of the MAKS framework in Isabelle/HOL that allowed us to verify our main results in Isabelle, and the anonymous reviewers for helpful comments on the paper.

References

[1] Rafael Accorsi & Andreas Lehmann (2012): *Automatic Information Flow Analysis of Business Process Models*. In: *BPM*, pp. 172–187, doi:10.1007/978-3-642-32885-5_13.

[2] Gustavo Alonso, Roger Günthör, Mohan Kamath, Divyakant Agrawal, Amr El Abbadi & C. Mohan (1996): *Exotica/FMDC: A Workflow Management System for Mobile and Disconnected Clients*. *Distributed and Parallel Databases* 4(3), pp. 229–247, doi:10.1007/BF00140951.

[3] Bowen Alpern & Fred B. Schneider (1987): *Recognizing safety and liveness*. *Distributed Computing* 2(3), pp. 117–126, doi:10.1007/BF01782772.

[4] Wihem Arsac, Luca Compagna, Giancarlo Pellegrino & Serena Elisa Ponta (2011): *Security Validation of Business Processes via Model-Checking*. In: *Engineering Secure Software and Systems*, *LNCS* 6542, Springer, pp. 29–42, doi:10.1007/978-3-642-19125-1_3.

[5] Thomas Bauereiss & Dieter Hutter (2013): *Possibilistic information flow security of workflow management systems*. Technical Report. Available at `http://bauereiss.name/papers/WorkflowSecurity_TR.pdf`.

[6] Achim D. Brucker, Isabelle Hang, Gero Lückemeyer & Raj Ruparel (2012): *SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes*. In: *SACMAT 2012*, ACM, pp. 123–126, doi:10.1145/2295136.2295160.

[7] David D. Clark & David R. Wilson (1987): *A Comparison of Commercial and Military Computer Security Policies*. *IEEE Symposium on Security and Privacy*, pp. 184–194, doi:10.1109/SP.1987.10001.

[8] Michael R. Clarkson & Fred B. Schneider (2010): *Hyperproperties*. *Journal of Computer Security* 18(6), pp. 1157–1210, doi:10.3233/JCS-2009-0393.

[9] Riccardo Focardi & Roberto Gorrieri (1995): *A Classification of Security Properties for Process Algebras*. *Journal of Computer Security* 3(1), pp. 5–33, doi:10.3233/JCS-1994/1995-3103.

[10] Florian Haftmann & Tobias Nipkow (2007): *A code generator framework for Isabelle/HOL*. In: *Theorem Proving in Higher Order Logics: Emerging Trends*. Available at `http://es.cs.uni-kl.de/events/TPHOLs-2007/proceedings/B-128.pdf`.

[11] Dieter Hutter (2006): *Possibilistic Information Flow Control in MAKS and Action Refinement*. In: *ETRICS*, *LNCS* 3995, Springer, pp. 268–281, doi:10.1007/11766155_19.

[12] Dieter Hutter (2007): *Preserving Privacy in the Web by Using Information Flow Control*. In Andreas U. Schmidt, Michael Kreutzer & Rafael Accorsi, editors: *Long-Term and Dynamical Aspects of Information Security: Emerging Trends in Information and Communication Security*, Nova Science.

[13] Dieter Hutter, Heiko Mantel, Ina Schaefer & Axel Schairer (2007): *Security of multi-agent systems: A case study on comparison shopping*. *Journal of Applied Logic* 5(2), pp. 303–332, doi:10.1016/j.jal.2005.12.015.

[14] Dieter Hutter & Axel Schairer (2004): *Possibilistic Information Flow Control in the Presence of Encrypted Communication*. In: *ESORICS*, *LNCS* 3193, Springer, pp. 209–224, doi:10.1007/978-3-540-30108-0_13.

[15] Heiko Mantel (2000): *Possibilistic Definitions of Security - An Assembly Kit*. In: *CSFW*, IEEE Computer Society, pp. 185–199, doi:10.1109/CSFW.2000.856936.

[16] Heiko Mantel (2000): *Unwinding Possibilistic Security Properties*. In: *ESORICS*, *LNCS* 1895, Springer, pp. 238–254, doi:10.1007/10722599_15.

[17] Heiko Mantel (2001): *Information Flow Control and Applications - Bridging a Gap*. In: *FME*, *LNCS* 2021, Springer, pp. 153–172, doi:10.1007/3-540-45251-6_9.

[18] Heiko Mantel (2001): *Preserving Information Flow Properties under Refinement*. In: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 78–91, doi:10.1109/SECPRI.2001.924289.

[19] Heiko Mantel (2002): *On the Composition of Secure Systems*. In: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 88–101, doi:10.1109/SECPRI.2002.1004364.

[20] Heiko Mantel & Andrei Sabelfeld (2003): *A Unifying Approach to the Security of Distributed and Multi-Threaded Programs*. *Journal of Computer Security* 11(4), pp. 615–676. Available at `http://iospress.metapress.com/content/r0pr0ma4kv8wa542/`.

[21] J. McLean (1996): *A general theory of composition for a class of "possibilistic" properties*. *IEEE Transactions on Software Engineering* 22(1), pp. 53–67, doi:10.1109/32.481534.

[22] Peter Muth, Dirk Wodtke, Jeanine Weissenfels, Angelika Kotz Dittrich & Gerhard Weikum (1998): *From Centralized Workflow Specification to Distributed Workflow Execution*. *Journal of Intelligent Information Systems* 10(2), pp. 159–184, doi:10.1023/A:1008608810770.

[23] Andrew C. Myers, Andrei Sabelfeld & Steve Zdancewic (2006): *Enforcing Robust Declassification and Qualified Robustness*. *Journal of Computer Security* 14(2), pp. 157–196. Available at `http://iospress.metapress.com/content/EYT2D3ERKY3A2H25`.

[24] Tobias Nipkow, Lawrence C Paulson & Markus Wenzel (2002): *Isabelle/HOL: a proof assistant for higher-order logic*. *LNCS* 2283, Springer, doi:10.1007/3-540-45949-9.

[25] Sylvia Osborn, Ravi Sandhu & Qamar Munawer (2000): *Configuring role-based access control to enforce mandatory and discretionary access control policies*. *ACM Trans. Inf. Syst. Secur.* 3(2), p. 85–106, doi:10.1145/354876.354878.

[26] Alfonso Rodríguez, Eduardo Fernández-Medina & Mario Piattini (2007): *A BPMN Extension for the Modeling of Security Requirements in Business Processes*. *IEICE Transactions* 90-D(4), pp. 745–752, doi:10.1093/ietisy/e90-d.4.745.

[27] A. Sabelfeld & A.C. Myers (2003): *Language-based information-flow security*. *IEEE Journal on Selected Areas in Communications* 21(1), pp. 5–19, doi:10.1109/JSAC.2002.806121.

[28] Andrei Sabelfeld & David Sands (2009): *Declassification: Dimensions and principles*. *Journal of Computer Security* 17(5), pp. 517–548, doi:10.3233/JCS-2009-0352.

[29] Andreas Schaad, Volkmar Lotz & Karsten Sohr (2006): *A model-checking approach to analysing organisational controls in a loan origination process*. In David F. Ferraiolo & Indrakshi Ray, editors: *SACMAT*, ACM, pp. 139–149, doi:10.1145/1133058.1133079.

[30] Fred B. Schneider (2000): *Enforceable security policies*. *ACM Trans. Inf. Syst. Secur.* 3(1), p. 30–50, doi:10.1145/353323.353382.

[31] Hans Schuster, Stefan Jablonski, Thomas Kirsche & Christoph Bussler (1994): *A Client/Server Architecture for Distributed Workflow Management Systems*. In: *PDIS*, IEEE Computer Society, pp. 253–256, doi:10.1109/PDIS.1994.331708.

[32] Daniel F. Stork (1975): *Downgrading in a Secure Multilevel Computer System: The Formulary Concept*. Technical Report, DTIC Document. Available at `http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA011696`.

[33] Christian Wolter & Christoph Meinel (2010): *An approach to capture authorisation requirements in business processes*. *Requir. Eng.* 15(4), pp. 359–373, doi:10.1007/s00766-010-0103-y.

[34] Peter Y. H. Wong & Jeremy Gibbons (2008): *A Process Semantics for BPMN*. In: *ICFEM*, *LNCS* 5256, Springer, pp. 355–374, doi:10.1007/978-3-540-88194-0_22.

[35] Ping Yang, Shiyong Lu, Mikhail I. Gofman & Zijiang Yang (2010): *Information flow analysis of scientific workflows*. Journal of Computer and System Sciences 76(6), pp. 390–402, doi:10.1016/j.jcss.2009.11.002.

[36] Aris Zakinthinos & E. Stewart Lee (1997): *A General Theory of Security Properties*. In: IEEE Symposium on Security and Privacy, IEEE Computer Society, pp. 94–102, doi:10.1109/SECPRI.1997.601322.

A Specification of Activities

In this appendix, we give a formal specification of the behaviour of our activities using PP-statements. In this formalism, the transition relation of a state-event system is specified by listing pre- and post-conditions on the state for each event (see Section 2.1 of [13] for a formal semantics).

$Recv_a(b, Data(i,v))$; **affects:** Mem, $AQueue$
Pre: $pc = 0$, $(b,i,a) \in MF$, $(b,i) \notin AQueue$
Post: $Mem'(i) = v$, $AQueue' = AQueue \cup \{(b,i)\}$

$Send_a(b, AckData(i))$; **affects:** $AQueue$
Pre: $pc = 0$, $(b,i) \in AQueue$
Post: $AQueue' = AQueue \setminus (b,i)$

$Recv_a(b, Trigger)$; **affects:** $TriggeredBy$
Pre: $pc = 0$
Post: $TriggeredBy' = b$

τ_a^{Active}; **affects:** pc
Pre: $pc = 0$, $TriggeredBy \neq \bot$, $AQueue = \emptyset$
Post: $pc' = 1$

$\tau_a^{SendData}$; **affects:** pc, $MQueue$
Pre: $pc = 2$
Post: $pc' = 3$, $MQueue' = \{(b,i) \mid (a,i,b) \in MF \wedge Mem(i) \neq \bot\}$

$Send_a(b, Data(i,v))$;
affects: $MQueue$, $AQueue$
Pre: $pc = 3$, $(b,i) \in MQueue$
Post: $MQueue' = MQueue \setminus \{(b,i)\}$, $AQueue' = AQueue \cup \{(b,i)\}$

$Recv_a(b, AckData(i))$; **affects:** $AQueue$
Pre: $pc = 3$, $(b,i) \in AQueue$
Post: $AQueue' = AQueue \setminus \{(b,i)\}$

$\tau_a^{AckTimeout}$; **affects:** $AQueue$
Pre: $pc = 3$
Post: $AQueue' = \emptyset$

$\tau_a^{SendTriggers}$; **affects:** pc, $SQueue$
Pre: $pc = 3$, $MQueue = \emptyset$, $AQueue = \emptyset$
Post: $pc' = 4$, $SQueue' = \{b \mid (a,b) \in SF\}$

$Send_a(b, Trigger)$; **affects:** $SQueue$
Pre: $pc = 4$, $b \in SQueue$
Post: $SQueue' = SQueue \setminus \{b\}$

Figure 3: PP-statements of generic transition relation T_a^{gen}

We specify the behaviour of our activities in two parts. The PP-statements in Figure 3 specify the *generic* part of the behaviour of activities, i.e. the communication with other activities in order to exchange data items and trigger sequence flows. For this purpose, it maintains program variables $MQueue$ (which data items still have to be sent), $AQueue$ (which data items still have to be acknowledged), $SQueue$ (which triggers still have to be sent), $TriggeredBy$ (whether and from where a trigger has been received), and $User$ (to which user this activity is assigned). The program counters 0, 3 and 4 correspond to the phases of waiting for inputs and triggers, sending outputs, and sending triggers, respectively.

When the program counter reaches 1, an *activity-specific* transition relation takes over in order to perform the actual activity. In our simple example workflow, we only need two kinds of activities, namely

user input/output and gateways (deciding on the control flow based on a condition *Cond* on input data). The latter continues the workflow with that activity b for which $Cond(b, Mem)$ evaluates to true. These two kinds of activities are specified in Figures 4 and 5, respectively. We denote the transition relations induced by the PP-statements in Figures 3, 4, and 5 as T_a^{gen}, T_a^{user}, and $T_a^{gw(Cond)}$, respectively. The overall transition relation of an activity is the union of T_a^{gen} and an activity-specific transition relation.

$Start_a(u)$; **affects:** $User$
Pre: $pc = 1, User = \bot, cl_U(u) = cl_{\mathcal{A}}(a)$
Post: $User' = u$

$Setval_a(u,i,v)$; **affects:** Mem
Pre: $pc = 1, User = u$
Post: $Mem'(i) = v$

$Outval_a(u,i,v)$; **affects:**
Pre: $pc = 1, User = u, Mem(i) = v$

$End_a(u)$; **affects:** pc
Pre: $pc = 1, User = u$
Post: $pc' = 2$

Figure 4: PP-statements of transition relation T_a^{user} for user activities

$Send_a(b, Trigger)$; **affects:** pc
Pre: $pc = 1, Cond(b, Mem) = \top, (a,b) \in SF$
Post: $pc' = 5$

Figure 5: PP-statement of transition relation $T_a^{gw(Cond)}$ for gateways

After completion of the activity has been signalled by setting the program counter to 2, the generic transition relation takes control again and starts sending output data items to the designated receivers. It makes sure that they have been received by waiting for acknowledgements, and afterwards proceeds by sending triggers to the successor activities in the workflow. An exception to this rule is if a receiver fails to send an acknowledgement; in this case the $\tau_a^{AckTimeout}$ event can be used to signal a timeout and proceed with the workflow. This is important for security, because otherwise a confidential activity could block the progress of the workflow by refusing to acknowledge a data item.

Of course, other modelling decisions are possible to solve this problem. As an alternative, we have also modelled and verified a system specification where the communication platform guarantees causal delivery of messages, i.e. messages from one activity to another are always received in the order that they are sent. This would make acknowledgements unnecessary, because an activity could always be sure that a trigger message is received after all data items, if the messages are sent in this order. However, this shifts complexity from the individual activities to the communication platform and the interface, and it turns out that this makes the proof of compositionality more laborious. Essentially, we had to prove an additional security predicate *FCIA* for the platform and the activities together with several additional side conditions on the local views in order to obtain compositionality. In this paper, we therefore present the above model with explicit acknowledgements for simplicity. However, we intend to further investigate the implications of different guarantees provided by the communication platform in future work.