UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

FROM EARLY REQUIREMENTS ANALYSIS TOWARDS SECURE WORKFLOWS

Ganna Frankova, Fabio Massacci and Magali Seguran

May 2007

Technical Report # DIT-07-036

From Early Requirements Analysis towards Secure Workflows *

Ganna Frankova[1], Fabio Massacci[1], and Magali Seguran[2]

[1] Dept. of Information and Communication Technologies
University of Trento
Via Sommarive, 14
38050 Trento
Italy
email: {ganna.frankova,fabio.massacci}@unitn.it
[2] SAP Labs France
SAP Research - Security and Trust
805, avenue du Dr.Maurice Donat
06254 Mougins Cedex
France
email: magali.seguran@sap.com

Abstract. Requirements engineering is a key step in the software development process that has little counterpart in the development of business processes for web services. Furthermore, the existing design methodologies for web services do not address the issue of developing secure web services, secure business processes and secure workflows. This paper presents a methodology that allows a business process designer to derive the skeleton of the concrete secure business processes from the early requirements analysis. The proposed refinement methodology, aims to obtain an appropriate coarse grained secure business process that can be further refined into workflows. We introduce a specification language for secure business processes, which is a dialect of WS-BPEL for the functional parts and abstracts away low level implementation details from WS-Security and WS-Federation specifications. To make the discussion more concrete, we illustrate the proposal with an e-business banking case study.

Keywords: Requirements Engineering, Business Processes, Security and Trust, Web Services.

1 Introduction

There are many requirements engineering frameworks for modeling and analysing security requirements, such as *SI**/Secure Tropos [12, 20], UMLsec [16], Misuse-Case [26], AntiGoals [28]. There are several methodologies aim to web services

* This work has been partly supported by the IST-FP6-IP-SERENITY project

and business processes design [24, 19, 25]. We noticed that there is a gap among the requirements engineering methodologies and the actual production of software and business processes based on a Service-Oriented Architecture (SOA). Business processes and security issues are developed separately and often do not follow the same strategy [21]. There are a number of security standards in the area of SOA. For instance, WS-Federation [2] defines the mechanisms for federating trust, WS-Trust [15] enables security token interoperability, WS-Security [22] covers the low level details such as message content integrity and confidentiality. The question we address in this paper is "How to obtain a secure workflow from the early requirements analysis?".

We address the issue of secure workflows modeling based on early requirements analysis, namely, *SI**/Secure Tropos [12, 20], by presenting a methodology that bridges the gap between early requirements analysis and secure workflows for web services development. The methodology allows a business process designer to derive the skeleton of the concrete secure business processes from the early requirements analysis. The proposed refinement methodology, aims to obtain an appropriate coarse grained secure business process that can be further refined into workflows. We introduce a specification language for secure business processes, which is a dialect of WS-BPEL for the functional parts and abstracts away low level implementation details from WS-Security and WS-Federation specifications.

The remainder of the paper is organized as follows. In Section 2, we introduce an e-business banking case study that is used as a running example throughout the paper. We provide a brief description of the *SI**/Secure Tropos framework and describe the basic concepts and diagrams that we used for the early requirements analysis in Section 3. Section 4 is devoted to the proposed methodology that allows a business process designer to derive the skeleton of the concrete secure business processes from the early requirements analysis. Furthermore, we introduce a specification language for secure business processes and show how the running example can be described by the language. Section 5 opens the 'lack of permission" problem. Related work is discussed in Section 6. Concluding remarks are summarized in Section 7.

2 Running example: a loan origination process

The general environment in which the proposed scenario takes place is the e-business organization domain. The running example is abstracted from an e-business banking scenario, more specifically, from a typical loan origination process in the context of which the activities about assignment of rights, roles, and tasks need to be carefully considered from a security point of view. The scenario is provided by the courtesy of SAP company [1] and is a working scenario

[1] http://www.sap.com.

of the IST-FP6-IP-SERENITY project [2].For more information on the scenario refer to [6].

Scenario description
John is a single 25 years old man who wants to buy a flat and needs a loan. After visiting several banks, he decides to apply for a loan of 90,000 euros to his time-proved the BBB bank.

Scene 1. John goes to the bank to ask for a loan - Peter, the pre-processing clerk receives John, checks his identity, receives clients's data for identification from the Internal Computer System and matches them with the identity of John.

Scene 2. The bank double checks the credit worthiness of John - When the identity is checked, Peter introduces John to Paul, the post-processing clerk. Paul obtains several external (conducted by the Credit Bureau) and internal (conducted by the Internal Computer System) ratings in order to check the credit worthiness of the customer.

Using a Credit Bureau - The credit worthiness is checked querying the Credit Bureau. The Credit Bureau is a third party business partner of financial institution that processes, stores and safeguards the credit information of physical individual and industrial companies. In the case of John, the Credit Bureau does not return any negative information about credit worthiness and Paul continues to process John's loan.

Using internal rating - For the internal check, the post-processing clerk analyses results of calculation of the internal rating. The internal credit scoring application assigns a low risk level to John's application and the loan origination process moves to the third phase.

Scene 3. The bank calculates the loan price - Paul queries the Pricing Engine service to compute a price of the loan taking into account the score. The result in terms of original price, customer segment special conditions, customer company special conditions, asset limit for price, is then returned to Paul. Paul is able to make a proposal to John.

Scene 4. The bank and John sign the form - If John is satisfied by the proposed product, he is going to discuss the loan in more details and to finalize the process. The representative of the bank may be Paul or Ted (the manager) according to the loan amount or the customer type. In this case, John and Paul are involved in the negotiation and signing of the contract.

The loan origination process is a business process that can be easily refined to the workflow. The different steps of the case study are depicted at the workflow diagram in Figure 1. In the rest of the work we will concentrate on those phases of the case study that are relevant to describe the security requirements such as separation of duty, compensation, and transaction.

[2] SERENITY (System Engineering for Security and Dependability) is a R&D project funded by the European Union. SERENITY aims at providing security and dependability in Ambient Intelligence systems. For more information refer to the Serenity project website http://www.serenity-project.org.

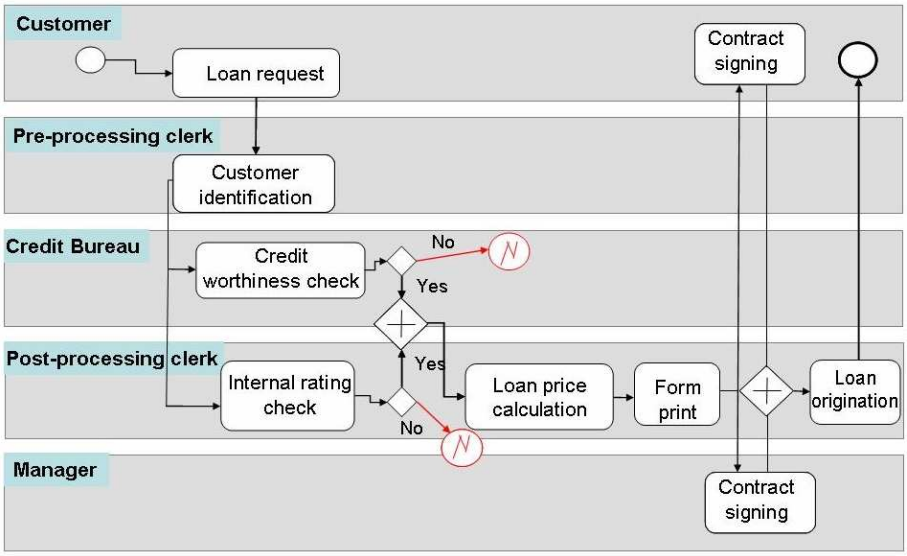
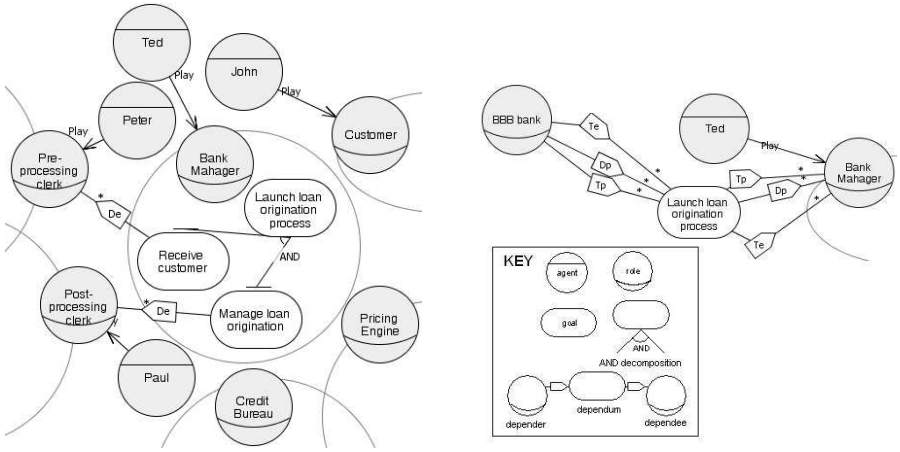


Fig. 1. Loan Origination Workflow.

3 The *SI**/Secure Tropos framework

*SI**/Secure Tropos is a formal framework and a methodology for modelling and analysing security requirements [12, 20]. In this work we employ the early security requirements analysis in order to develop a secure workflow. *SI**/Secure Tropos uses the concepts of actor and goal. Actors can be *agents* or *roles*. *SI**/Secure Tropos also supports the notion of *delegation of permission* and *delegation of execution* to model the transfer of entitlements and responsibilities from an actor (called *delegator*) to another (called *delegatee*), respectively. *Trust of permission* and *trust of execution* are used to model the expectation of one actor (the *trustor*) about the behavior and capabilities of another actor (the *trustee*). The meaning of trust of permission is that a trustor trusts that trustee will at least fulfill a service while trust of execution means that trustor trusts that trustee will at most fulfill a service, but will not overstep it.

From a methodological perspective, *SI**/Secure Tropos is based on the idea of building a model of the system that is incrementally refined and extended. Specifically, goal analysis consists of refining goals and eliciting new social relationships among actors. They are conducted from the perspective of single actors using AND/OR decomposition. In case an actor does not have the capabilities to achieve his own objectives or assigned responsibilities by himself, he has to delegate them to other actors making their achievement outside his direct control.



(a) Actors and functional dependencies. (b) Authorization and trust.

Fig. 2. *SI**/Secure Tropos Diagrams.

The above constructs allow us to capture the functional, security and trust requirements in a number of diagrams.

Actor diagram describes objectives, entitlements and capabilities of each actor which are also analyzed using goal refinement and contribution analysis techniques from the perspective of the actor.

Functional dependency diagram identifies the dependencies among actors, in particular, to which actor has been delegated the execution of which services by which actor.

Authorization diagram identifies the transfers of right among actors, in particular, to which actor has been delegated the permission, on which services and by which actor.

Trust diagram describes the expectations of actors about the behavior and capabilities of other actors in terms of trust of permission and trust of execution.

Figure 2 shows the examples of the diagrams based on the case study presented above. Actor and functional dependency diagram (see Figure 2(a)) describes the actors (agents, depicted as circles with straight lines, and roles, depicted as circles with curves); some of the bank manager's goals, depicted as ovals; goal refinement by AND decomposition, depicted with a goal refinement symbol marked with AND; and the delegation of execution dependencies among bank manager, pre-processing and post-processing clerks, depicted with two lines connected by a delegation of execution (De) graphical symbol. One of the variants of authorization and trust diagram is presented in Figure 2(b). The diagram identifies the actors, that participate, i.e, the BBB bank and bank manager, and

involved services, i.e, the "launch loan origination process" goal, in delegation of permission, trust on permission and trust of execution dependencies, depicted with two lines connected by a delegation of permission (Dp), trust on permission (Tp) and trust of execution (Te) graphical symbols.

4 From early requirements towards secure workflows

A secure business process is originated by the early requirements analysis and then is used to the development of an appropriate workflow. The process of deriving a secure workflow from early requirements is presented in Figure 3. The process includes three phases, namely, (1) early requirements engineering, (2) late requirements engineering and (3) detailed design.

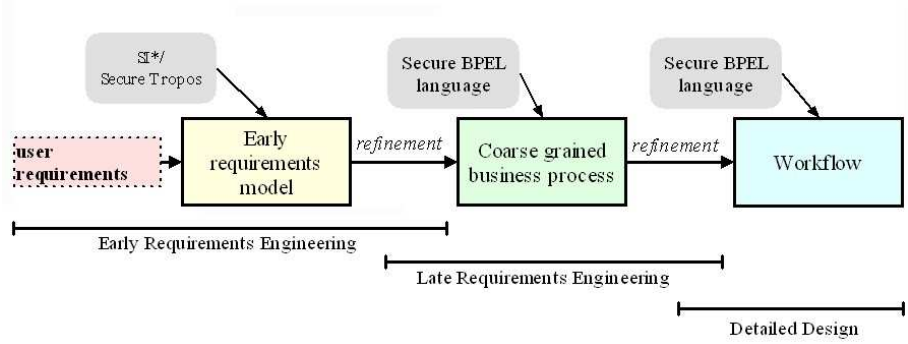


Fig. 3. Relations among early requirements, business process and workflow levels.

For the purpose of developing secure workflows based on the early requirements analysis, we propose a refinement methodology and a specification language, Secure BPEL, for secure requirements at the business process and workflow levels.

4.1 Early requirements engineering

Early requirements engineering is concerned with understanding the organizational context within which the system-to-be will eventually function. During early requirements analysis the domain actors and their dependencies on other actors for goals to be fulfilled are identified. For early requirements elicitation, one need to reason about trust relationships and delegation of authority. We employ *SI**/Secure Tropos modelling framework to derive and analyse both functional dependencies and security and trust requirements. Various activities contribute to the acquisition of the early requirements model, namely:

Actor modelling aims at identifying actors and analysing their goals.

Functional dependency modelling aims at identifying actors depending on other actors for obtaining services, and actors which are able to provide services.

Permission delegation modelling aims at identifying actors delegating to other actors the permission on services.

Trust modelling aims at identifying actors trusting other actors for services, and actors which own services.

A graphical representation of the model obtained according these modelling activities is given respectively through the actor, functional dependency, authorization, and trust diagrams, described in Section 3.

The process of the early requirements model acquisition starts from user requirements, goes thought actor, functional dependency, permission delegation and trust modelling and ends with actor, functional dependency, authorization, and trust diagrams, i.e., the requirements model obtaining (see Figure 4).

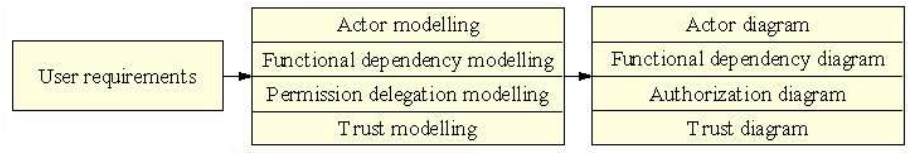


Fig. 4. Early requirements model acquisition process.

4.2 Late requirements engineering

Late requirements engineering is concerned with a definition of the functional and non-functional requirements of the system-to-be. In this work the proposed refinement methodology aims to obtain an appropriate coarse grained business process and workflow at the workflow level based on early requirements. The refinement is processed by diagrams created in the early requirements engineering phase. The methodology takes the components of the diagrams and derives a secure business process constructs from them that is described by the proposed Secure BPEL language (see Figure 5).

The relevant components of actor diagram are actors, goals and spawning of dependency relationships among actors. In functional dependency diagram, we consider dependencies among actors that delegate or are delegatees of execution of services. The components of authorization diagram are transfers of right among actors that delegate or are delegatees of permission on services. In trust diagram we consider the expectations of actors about the behavior and capabilities of other actors in terms of trust on permission and trust on execution.

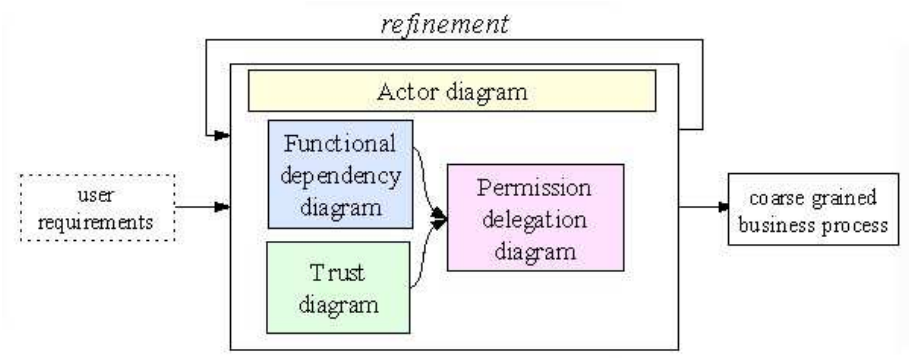


Fig. 5. Late requirements engineering.

Figure 6 presents two steps of actor diagram refinement. In first step, partners are designed based on the actors identified in the early requirements engineering stage. We assume that each actor has a single root goal that can be decomposed by AND/OR goal decomposition. Each AND/OR goal decomposition lead to operationalization phase. The second step considers partner and orchestration specification by the Secure BPEL language. Operationalization is completed with additional information to AND/OR goal decomposition on choice of sequential or parallel operation.

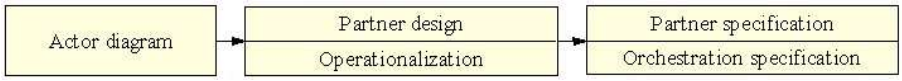


Fig. 6. Actor diagram refinement.

For the lack of space we do not present two steps of other diagrams refinement. The idea is that in first step dependencies (for functional dependency and authorization diagrams) or trust (for trust diagram) and choreography are designed and in second step choreography is specified. We consider that the level of goals is the level of services.

The table in Figure 7 presents the mentioned diagrams to Secure BPEL language notions refinement. Considering actor diagram, the notion of actor is refined into partner in Secure BPEL, a root goal is refined into business process while AND/OR goal decomposition with delegation are refined into orchestration. The notions of delegation of execution and delegation of permission presented in dependency and authorization diagrams are refined into choreography of services and authorization respectively. As for trust diagram, trust on execution and permission are refined into choreography of attestation that is further

refined into attestation of integrity for the notion of trust on execution and attestation of reporting for trust on permission.

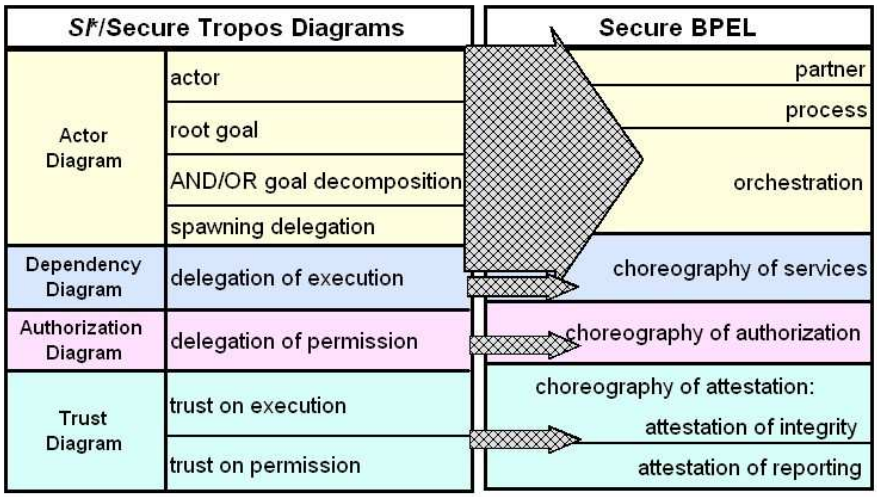


Fig. 7. Diagrams to Secure BPEL refinement.

4.3 Secure BPEL Language

Secure BPEL language is an extension of Web Services Business Process Execution Language (WS-BPEL) [23] that allows for secure workflows specification. Hence, if a business process designer is familiar with WS-BPEL processes, he simply needs to understand the additional constructs introduced by Secure BPEL. We suffix each new or refined construct with the keyword "S" to clearly distinguish them.

Refining Actor Diagram

Actors identification consists in identifying all actors, i.e., agents and roles, involved in a business process and all roles played by all the agents identified.

The concept of actors at the business process level is refined as partners and specified in Secure BPEL by the `<partnerS>` construct (see Figure 8).

To ease the language specification we provide a slight extension to the WS-BPEL standard by retaining the `<partner>` construct from the Business Process Execution Language for Web Services (BPEL4WS) [3]. While such extension is not necessary for actually writing down the workflow solution (because each partner role is specified on every individual invocation), it is extremely convenient at the requirements level because it offers a compact view of who is doing

```
<partnersS>
    <partnerS nameS = "agentName">+
      roles played by agent
    </partnerS>
</partnersS>
```

Fig. 8. Actor identification.

what. Further, at this stage, we also need to identify which agent has to run which process and hence the addition of the `nameS` attribute.

Each partners interaction at the business process level is specified by the `<partnerLinkS>` construct [3] and specifying all roles played by a partner (see Figure 9). The role of the partner itself is indicated by the attribute `myRole` and the role of the companion is indicated by the attribute `partnerRole` within the `<partnerLinkS>` construct. When there is only one role, one of these attributes is omitted as appropriate. The partner is identified by the `partnerNameS` attribute. Each `partnerLinkS` is named and this name is used for all service interactions via that `partnerLinkS`.

```
<partnerLinksS>
    <partnerLinkS name="partnerLinkName"
      myRole = "myRoleName"?
      partnerNameS = "agentName"?
      partnerRole = "partnerRoleName"?>+
    </partnerLinkS>
</partnerLinksS>
```

Fig. 9. Actor description.

Example 1. According to the first scene of the case study presented in Section 2, in the actor identification step, two agents (specified by the `<partnersS>` construct) are identified, namely, John and Peter (the `nameS` attribute of the `<partnerS>` construct). For the partners interaction at the business process level (the `<partnerLinkS>` construct), the agents roles are described (the `myRole` / `partnerRole` attribute within the `<partnerLinkS>` construct). Partner John has a role customer. John is a partner of Peter whose role is a pre-processing clerk. In such manner it is possible to identify and describe all actors presented in the case study.

[3] As we work at a high level of abstraction, at this point some workflow details are not considered. Most notably, we do not specify partner link types that characterizes the conversational relationship between two partners by defining the roles played by each of the partners in the conversation.

The concept of actor is specified in the *SI**/Secure Tropos metamodel [27] as an agent can play several roles. Figure 10 shows the portion of the Secure BPEL metamodel where the concept of partner and role are specified. One secure business process can be composed of several partners. While to each partner can be associated one or more partner links that specify all roles played by a partner. The role of a partner itself is specified by the `myRole` attribute and the role of the companion is indicated by the `partnerRole` attribute.

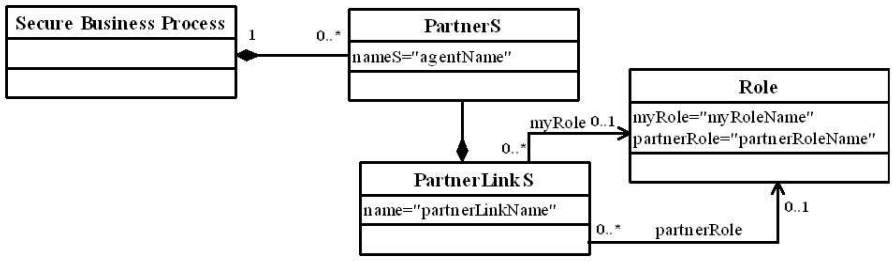


Fig. 10. The Secure BPEL metamodel specifying the partner concept.

Structured activities is a basis of orchestration specification and consist of a sequential/parallel composition and branching statement. The notion of sequential and parallel composition corresponds to a refinement of the concept of AND goal decomposition. Branching statement is a refinement of the concept of OR goal decomposition.

Sequential composition is specified by the `<sequence>` construct. The construct defines a collection of activities to be performed sequentially, in the lexical order in which they appear within the construct. Parallel composition is specified by the `<flow>` construct. The construct defines one or more activities to be performed concurrently. While branching statement is specified by the `<if>` construct that is used to select exactly one activity for execution from a set of choices. Refer to WS-BPEL [23] specification for the syntax of the constructs.

Example 2. Following the case study presented in Section 2, all the main activities are sequential. The following activities: customer identification, check rating, calculation of the price and signature of the contract are done in a sequential way. At the business process level, the process defining these activities in the sequential order, is implemented by the `<sequence>` construct.

Example 3. In the second scene of the case study presented in Section 2, the process of checking the credit worthiness is divided in two parallel subprocesses: the external part (provided by Credit Bureau) and the internal one (based on internal scoring). Nevertheless the internal one is stopped when the results coming from the Credit Bureau are negative. At the business process level, the process is implemented by the `<flow>` construct.

Refining Functional Dependency Diagram
Dependencies derived from a functional dependency diagram are notably delegation of execution. The refinement process starts with abstract goals and ends up with concrete atomic activities at the business process level, while those activities can be further refined at the workflow level. Here we consider that the level of goals is the level of services. Atomic activities is a basis of choreography specification and consist of the service invocation activities and the response to a service invocation activities.

Considering one particular dependency, invocation of a service by a depender is specified by the `<invoke>` construct (see Figure 11(a)). Responding to a service invocation by a dependee is specified by the `<pick>` construct (see Figure 11(b)). The construct allows to block and wait for a suitable message to arrive, i.e., a message of service invocation. When the message arrives, the associated activity, i.e., service execution, is performed and the pick completes.

```
<invoke
    partnerLink = "partnerLinkName"
    operation = "operationName">
</invoke>
```

(a) Service invocation.

```
<pick>
    <onMessage partnerLink = "agentName">
      execute service
    </onMessage>
</pick>
```

(b) Response to service invocation.

Fig. 11. Atomic activities.

The concept of delegation of execution at the business process level is refined as a process that consists of invocation of a goal (service), from one partner, i.e., a depender and other partner's, i.e, dependee, acceptance of the delegation and execution of the goal.

Example 4. The concept of delegation of execution is seen in some scenes of the case study presented in Section 2. In particular, in the first scene, John as a bank customer delegates the function of processing the loan origination to the BBB bank. Then the BBB bank delegates the identification of the customer to Peter, the pre-processing clerk, and delegates the managing of the loan origination process to Paul, the post-processing clerk. In the second scene, Paul delegates the credit worthiness check, in particular, external rating analysing, to the Credit Bureau. At the business process level the delegation process of credit worthiness check to the Credit Bureau is a follows. At the delegater side, the partner Paul invokes the operation "credit worthiness check" (by the `<invoke>` construct) from the partner Credit Bureau. While at the delegatee side, the partner Credit Bureau, the delegatee responds to a service invocation (see the `<pick>` construct) accepting the message of service invocation and executes the goal "credit worthiness check".

Refining Authorization and Trust Diagrams
Interactions with partners can be more complicated than delegation of execu-

tion represented by the atomic activities. There is a set of activities to represent the *SI**/Secure Tropos concepts of delegation of permission, trust on execution and trust on permission at the business process level. This set includes request/response for authentication token, authorization token, attestation of integrity and attestation of reporting.

The concept of attestation characterizes the process of vouching for the accuracy of information [13]. In this work we use two types of attestation, i.e, attestation of integrity and attestation of reporting. Attestation of integrity provides proof that an actor can be trusted to report integrity and performed using the set or subset of the credentials associated with the actor. Attestation of reporting is the process of attesting to the contents of integrity reporting.

The `<RequestSecurityServiceS>` construct is used to request a token for the purpose of authentication, authorization, attestation of integrity and attestation of reporting. The syntax for the construct is presented on Figure 12.

```
<requestSecurityServiceS>
    <typeS>
     typeS="Authentication|Authorization|
            Attestation-Integrity|Attestation-Reporting"
    </typeS>
    <purposeS>
     goalName+
    </purposeS>
    <participantsS>+
     <participantS nameS = "agentName">
     <participantS>
    </participantsS>
    <onBehalfOfS>... </onBehalfOfS>
    <usageS> ... </usageS>
</requestSecurityServiceS>
```

Fig. 12. Request security service.

The following describes the attributes and elements listed above:

/requestSecurityServiceS/typeS This element describes the type of security service requested, i.e., authentication, authorization, attestation of integrity and attestation of reporting. That is, the type of the service that will be returned by the `<requestSecurityServiceResponseS>` construct.

/RequestSecurityServiceS/purposeS This element specifies the scope for which the security service is desired, i.e., the goal to which the service applies.

/RequestSecurityServiceS/participantsS/ This element specifies the participants sharing the security service. This attribute is used by the requestor to clarify the actual parties involved.

/RequestSecurityServiceS/participantsS/participantS This element specifies participant (or multiple participants) that play a role in the use of the service or who are allowed to use the service.

/RequestSecurityServiceS/onBehalfOfS This element indicates that the requestor is making the request on behalf of another.

/RequestSecurityServiceS/usageS This element specifies a policy (as defined in WS-Policy) that indicates desired settings for the requested service such as `<delegatable> true|false </delegatable>`.

The `<RequestSecurityServiceResponseS>` construct is used to return a security service or response to a security service request. It should be noted that any type of parameter specified as input to a service request may be present on response in order to specify the exact parameters used by the issuer. The syntax for this construct is similar to the one presented on Figure 12. The only difference is in the additional `<requestedSecurityServiceS>` element that is used to return the requested security service.

Example 5. As we shown in the Example 4 the concept of delegation of execution is seen in some scenes of the case study presented in Section 2. This example aims to show the concept of delegation of permission by using the first scene of the case study. The BBB bank delegates the identification of the customer to Peter the pre-processing clerk. At the business process level, from the delegator side, the type of security services requested is authorization (specified with the `<typeS>` element), the purpose is "customer identification" (by the `<purpose>` element) and the participant is Peter (by the `<participant>` element), see Figure 12. From the delegatee side, the `<requestSecurityServiceResponseS>` construct is used to respond to the security service request with the purpose (by the `<purpose>` element) "customer identification" and the participant is Ted (by the `<participant>` element).

Example 6. Following the second scene of the case study presented in Section 2, the post processing clerk trusts the Credit Bureau to give trustworthy external rating, i.e, trust on permission concept. At the business process level, from the truster side, the type of security service requested is authentication (specified with the `<typeS>` element) with the goal check external rating (with the `<purpose>` element) and participant Credit Bureau (with `<participant>`). From the trustee side, the `<requestSecurityServiceResponseS>` construct is used to answer to the security service request with the `<purpose>` check external rating and the `<participant>` post-processing clerk. After this step, from the truster side, the type of security service is attestation of integrity (specified with the `<typeS>` element) with the goal check external rating (with the `<purpose>` element) and participant Credit Bureau (with `<participant>`). From the trustee side, the `<requestSecurityServiceResponseS>` is used to answer to the security service request with the `<purpose>` check external rating and the `<participant>` post-processing. The concept of trust on execution is considered in the second scene of the case study. At the business process level, the process is very similar to the one presented above for trust on permission. The only

one difference is the type of the security service involved, which is attestation of reporting in the second step.

Figure 13 shows the portion of the Secure BPEL metamodel where the concept of activity is specified. Activity is composed of partner activity and structured activity. Partner activity, in its turn, consists of the `invoke`, `pick`, `request security serviceS`, and `request security service responseS` activities. While structured activity is composed of the `sequence`, `flow`, and `if` activities.

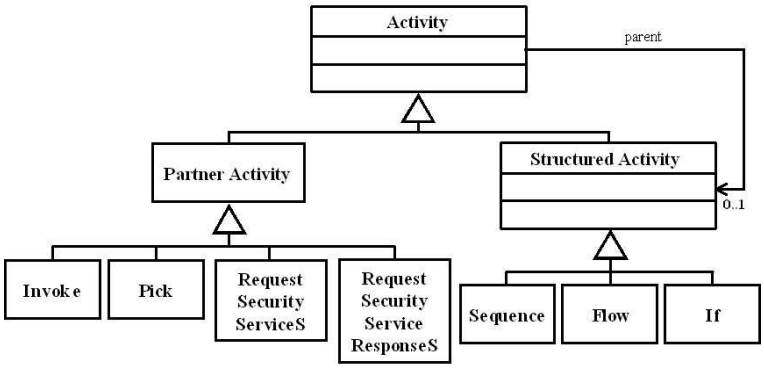


Fig. 13. The Secure BPEL metamodel specifying activity hierarchy.

Refining Delegation and Trust At the *SI**/Secure Tropos level it is possible to check the presence of trust or delegation chains by means of the satisfaction of axioms. Axiomatization for build a delegation chain is as follows:
$delegate(A, B, G) \longrightarrow delegateChain(A, B, G)$.
$delegate(A, C, G) \cap delegateChain(C, B, G) \longrightarrow delegateChain(A, B, G)$.
Delegation transitivity rule:
$delegate(A, C, G) \cap delegate(C, B, G) \longrightarrow delegate(A, B, G)$.
Axiomatization for build a trust chain is as follows:
$trust(A, C, G) \longrightarrow trustChain(A, B, G)$.
$trust(A, C, G) \cap trustChain(C, B, G) \longrightarrow trustChain(A, B, G)$.
Trust transitivity rule:
$trust(A, C, G) \cap trust(C, B, G) \longrightarrow trust(A, B, G)$.

Delegation/trust transitivity rules appear only at the formal level. At the business process level the properties should be checked. In order to check the satisfaction of the properties, i.e, to construct a delegation/trust chain, for every rule instance one needs a delegation/trust broker that builds up a delegation/trust chain. At this point the notion of brokering delegation/trust appear and the delegation/trust brokering service can be implemented by security token services (see WS-Trust [15]). The security token services play the role of authorities and

form the basic of trust by evaluating requests and issuing a rage of security tokens that are used to broker trust relationships between different trust domains. There are several mechanisms to delegation/trust bootstrap. The simplest one is if an actor, i.e, the recipient, has a fixed set of delegation/trust relationships. Then the mechanism will evaluate all requests to determine if they contain security tokens from one of the trusted roots. The second mechanism is based on the first one and it relevant for our work. An actor may choose to allow hierarchies of trust so long as the trust chain eventually leads to one of the known trust roots. The mechanisms for federating trust are defined in WS-Federation [2].

Example 7. In the case study presented in Section 2, the BBB banks delegates the function of providing a loan to the general director and the general director delegates the function to the manager. The manager delegates the customer identification to pre-processing clerk and the management of the loan origination process to the post-processing clerk. Taking into account the axioms for build a delegation chain and the delegation transitivity rule, one needs a delegation broker that builds up a delegation chain. The process is as follows: from the fact that the BBB banks delegates the function of providing a loan to the general director and the general director delegates the function to the manager implies the fact that the BBB banks delegates the function of providing a loan to the manager.

5 "Lack of permission" Problem

The proposed methodology aims to derive the skeleton of the concrete secure business processes from the early requirements analysis. Besides that, the methodology allows to further refine a coarse grained secure business process into secure workflow.

While deriving the concrete secure business processes from the early requirements analysis, one can face the situation we named as "lack of permission". The "lack of permission" situation appears when there is a chain of delegation/trust of execution with no corresponding chain of delegation/trust of permission. Each delegator/trustor of execution delegates/trusts on execution of a goal to the corresponding delegatee/trustee, which, in its turn, plays the role of delegator/trustor of execution and delegates/trusts on execution of the goal to other delegatee/trustee, etc. As there is no corresponding chain of delegation/trust of permission, i.e., the root delegator/trustor of execution (actor A) delegates/trusts permission of the goal (goal G) only to the leaf actor that actually executes the goal (actor C), while all the other nodes (actor B) faces the "lack of permission" problem (see Figure 14).

In the Secure BPEL language both delegation and trust are modeled by invocation. The delegation of execution concept is modeled as invocation of an operation by one partner from the other partner. The concepts of delegation/trust of permission/execution are modeled as different types of security services invocation. In order to address the "lack of permission" problem one needs of

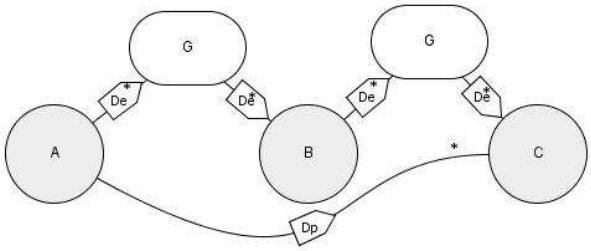


Fig. 14. "Lack of permission" problem.

introducing special types of invocation. The new invocation should allow the data to be protected, i.e, allows message confidentiality and integrity.

6 Related Work

Requirements engineering methodology for business processes in the context of web services is gaining increasing attention in the Software Engineering and Service-Oriented Communities. Next, we overview not only approaches aimed to use requirements engineering methodologies in the context of web service design, but also specifications developed in order to build secure web services.

Lau and Mylopoulos [19] propose a design methodology for web services adapted from the Tropos [7, 5] project. The work is based on the use of goals to determine the space of alternative solutions to satisfy the goals. The key point is that the solutions are represented by web services. The generated web services design is expected to accommodate as many of those solutions as possible rendering the design usable by a broader class of applications. On the negative side, Tropos is not tailored specifically to web service design. Therefore the proposed methodology does not address the issue of integration of Web Service Business Process Language in order to specify actual behaviour of participants in a business interaction. Furthermore, the design methodology for web services does not aim to design secure web services.

Kazhamiakin, Pistore, and Roveri [18] propose a methodology for business requirements modelling that uses the Tropos framework to capture the strategic goals of the enterprise. The proposed methodology enables to produce a concrete business process expressed by BPEL4WS description. The concrete business process is elicited from the description of business process notions with Tropos concepts extended with formal annotation called Formal Tropos [10]. On the contrary, our work aims to design secure business processes. The requirements expressed by the proposed language are security and dependability requirements that are not temporal properties Formal Tropos deals with.

Penserini, Perini, Susi, and Mylopoulos [25] address the issue of refining the Tropos methodology and tailoring it to the design of web services. The Tropos

design process is extended to support a revised notion of capability that explicitly correlates actor plans with stakeholders needs and environmental constraints. The agent capability is considered as a service. Furthermore, the authors sketch how Tropos design-tome models can support service discovery and composition by relating stakeholder goals to sets of services available. Even if, the idea is feasible, the work is in an the early stage and there is a need for more precise mapping of agent capability that is considered as a service. Furthermore, there is no secure web services design support.

Georg, Ray and France [11] propose the use of aspects for designing a secure system. The work illustrates how an aspect-oriented approach to modeling allows to encapsulate the concerns of security, availability of services and timeliness so they can be woven into a secure system design. The weaving strategy identifies security aspects based on the kinds of possible attacks and the mechanisms that allows the detection, prevention, and recovery from such attacks. Haley, Laney, and Nuseibeh [14] represent security requirements as crosscutting threat descriptions using aspect-oriented software development crosscutting concepts and problem frames. Security requirements are seen as constraints on functional requirements intended to reduce the scope of vulnerabilities. This allows to analyze secure requirements along with other constraints when producing specification for the problem. Cheng, Konrad, Campbell, and Wassermann [8] propose the use of security patterns for modeling and analysing secure systems. The authors describe a collection of security patterns using a template that addresses difficulties inherent to the development of secure-critical systems. Employing the patterns, it is possible to gain insight into the issue of modeling and analysing security concerns starting from the requirements engineering phase. On the negative site, the approaches do not support the design of software and business processes based on a service-oriented architecture.

In addition, the use of User Requirements Notation for business process modelling is proposed by Weiss and Amyot [29]. The research works of Colombo, Mylopoulos, and Spoletini [9] presents a methodological framework that supports the modelling and formal analysis of requirements for service composition through a social and process perspective. While Kaabi, Souveyet, and Rolland [17] propose a goal driven approach to service elicitation, distribution and orchestration. The agent-oriented methodology Tropos is used for analysing web service requirements by Aiello and Giorgini in [1]. In the approach the authors do not model every individual web service as an agent, but rather model the whole set of interacting services as a multi-agent system where different dependent strong and soft goals coexist. None of these methodologies aims to design *secure* web services or supports *secure* business processes.

WS-Security [22] specifies enhancements to SOAP messaging that while building secure web services can be used in order to implement message content integrity and confidentiality. WS-Security does not address the issues of interoperability between SOAP client and SOAP service. The standard does not specifies how a SOAP client and a SOAP service can agree on the nature and characteristics of the security tokens. WS-Security begins with the assumption

that, if one of the partners uses a particular type of security token, then the other partner will be able to interpret and process the token. While the guarantee that both partners who wish to use WS-Security to secure their SOAP messages support the security token they will be able to understand and process is needed. WS-Trust [15] enables security token interoperability by defining a request/response protocol by which SOAP actors can request of some trusted authority that a particular security token be exchanged for another. WS-Trust supports broker trust relationships and therefore can be used to build delegation and trust chains between partners. A semantic of the main mechanisms of WS-Trust and typical protocols, relying on these mechanisms are modelled in [4]. The core security properties of the specification are proved and some limitation and potential vulnerabilities are discussed. Designing secure business processes is out of the scope of this work as it focuses at the lower level, i.e, protocols modelling and verification.

7 Concluding Remarks

One of the most thought provoking issues in requirements engineering is that of designing web services and developing of business processes and workflows for web services. The research on web services design is well under way, but most of the design methodologies for web services do not address the issue of developing secure web services, secure business processes and secure workflows.

The main contribution of the paper is to bridge the gap between early requirements analysis and secure workflows for web services development. In particular, we have proposed a methodology that allows a designer of a business process to derive the skeleton of the concrete secure business processes based on the early requirements analysis. Furthermore, the secure business processes are refined in order to obtain the appropriate secure workflows that can be described by the proposed specification language for secure requirements called Secure BPEL.

The research presented in this work is still in progress. This work prods for more investigation of the proposed Secure BPEL language. In the framework of the IST-FP6-IP-SERENITY project, the Workflow Security Analysis Tool is currently underway. The tool supports workflow security requirements implementation by Secure BPEL specification language. In the next future, we plan to dive into the details of the low level secure requirements of transaction, messages integrity and confidentiality that will be included in the next release of the language.

References

1. M. Aiello and P. Giorgini. Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities of Services. *CEPIS Upgrade - The European journal of the informatics professional*, 5(4):20–26, 2004.

2. S. Bajaj, G. Della-Libera, B. Dixon, M. Dusche, M. Hondo, M. Hur, C. Kaler, H. Lockhart, H. Maruyama, A. Nadalin, N. Nagaratnam, A. Nash, H. Prafullchandra, and J. Shewchuk. Web Services Federation Language (WS-Federation) 1.0, July 2003. `http://specs.xmlsoap.org/ws/2003/07/secext/`.
3. BEA, IBM, Microsoft, SAP AG, and Siebel Systems. Business Process Execution Language for Web Services Version 1.1, May 2003. `http://www.ibm.com/developerworks/library/ws-bpel/`.
4. K. Bhargavan, R. Corin, C. Fournet, and A. D. Gordon. Secure Sessions for Web Services. In *Proceedings of Workshop on Secure Web Service*, Fairfax, VA, USA, October 24 2004.
5. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. TROPOS: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
6. S. Campadello, L. Compagna, D. Gidoin, S. Holtmanns, V. Meduri, J.-C. Pazzaglia, M. Seguran, and R. Thomas. A7.D1.1 - Scenario Selection and Definition. SAP, April 2006. Report.
7. J. Castro, M. Kolp, and J. Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems*, 27(6):365–389, September 2002.
8. B. H. C. Cheng, S. Konrad, L.A. Campbell, and R. Wassermann. Using Security Patterns to Model and Analyze Security Requirements. In *IEEE Workshop on Requirements for High Assurance Systems*, Monterey, California, USA, September 2003.
9. E. Colombo, J. Mylopoulos, and P Spoletini. Modeling and Analyzing Context-aware Composition of Services. In *Proceedings of International Conference on Service-Oriented Computing*, Amsterdam, The Netherlands, December 13–15 2005.
10. A. Fuxman, L. Liu, j. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso. Specifying and Analyzing Early Requirements in Tropos. *Requirements Engineering*, 9(2):132–150, May 2004.
11. G. Georg, I. Ray, and R. France. Using Aspects to Design a Secure System. In *Proceedings of IEEE International Conference on Engineering of Complex Computer Systems*, Greenbelt, Maryland, USA, December 2 - 4 2002.
12. P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Requirements Engineering for Trust Management: Model, Methodology, and Reasoning. *International Journal of Information Security*, 5(4):257–274, October 2006.
13. Trusted Computing Group. Tcg specification architecture overview revision 1.2, April 2003. `https://www.trustedcomputinggroup.org/groups/TCG_1_0_Architecture_Overview.pdf`.
14. C.B. Haley, R.C. Laney, and B. Nuseibeh. Deriving Security Requirements from Crosscutting Threat Descriptions. In *Proceedings of International Conference on Aspect-Oriented Software Development*, Lancaster, UK, March 2004.
15. IBM, BEA Systems, Microsoft, Layer 7 Technologies, Oblix, VeriSign, Actional, Computer Associates, OpenNetwork Technologies, Ping Identity, Reactivity, and RSA Security. Web Services Trust Language (WS-Trust).
16. J. Jürjens. *Secure Systems Development with UML*. Springer-Verlag, 2004.
17. R.S. Kaabi, C. Souveyet, and C. Rolland. Eliciting Service Composition in a Goal Driven Manner. In *Proceedings of International Conference on Service Oriented Computing*, New York, NY, USA, November 15-18 2004.
18. R. Kazhamiakin, M. Pistore, and M. Roveri. A Framework for Integrating Business Processes and Business Requirements. In *Proceeding of the Enterprise Distributed Object Computing Conference*, California, USA, September 20-24 2004.

19. D. Lau and J. Mylopoulos. Designing Web Services with Tropos. In *Proceedings of IEEE International Conference on Web Services*, San Diego, USA, July 6-9 2004.
20. F. Massacci, J. Mylopoulos, and N. Zannone. An Ontology for Secure Socio-Technical Systems. *Handbook of Ontologies for Business Interaction*, 2007. To appear.
21. T. Neubauer, M. Klemen, and S. Biffl. Secure Business Process Management: A Roadmap. In *Proceedings of International Conference on Availability, Reliability and Security*, Vienna, Austria, April 2006.
22. OASIS. Web Services Security: SOAP Message Security 1.1 (WS-Security).
23. OASIS. Web Services Business Process Execution Language Version 2.0, August 2006. Public Review Draft, `http://docs.oasis-open.org/wsbpel/2.0/`.
24. M.P. Papazoglou and J. Yang. Design Methodology for Web Services and Business Processes. In *Proceedings of the International Workshop on Technologies for E-Services*, Hong Kong, China, August 2002.
25. L. Penserini, A. Perini, A. Susi, and J. Mylopoulos. From Stakeholder Needs to Service Requirements. In *Proceeding of International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements*, Minneapolis, Minnesota, USA, September 12 2006.
26. G. Sindre and A.L. Opdahl. Eliciting Security Requirements with Misuse Cases. *Requirements Engineering*, 10(1):34–44, January 2005.
27. A. Susi, A. Perini, J. Mylopoulos, and P. Giorgini. The Tropos Metamodel and its Use. *Informatica*, 29:401–408, 2005.
28. A. van Lamsweerde, S. Brohez, R. De Landtsheer, and D. Janssens. From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering. In *Proceedings of Workshop on Requirements for High Assurance Systems*, Monterey Bay, California, USA, September 9 2003.
29. M. Weiss and D. Amyot. Business Process Modeling with URN. *International Journal of E-Business Research*, 1(3):63–90, 2005.