UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

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

FROM EARLY REQUIREMENTS TO BUSINESS PROCESSES WITH SERVICE LEVEL AGREEMENTS

Ganna Frankova, Artsiom Yautsiukhin and Magali Seguran

June 2007

Technical Report # DIT-07-037

From Early Requirements to Business Processes with Service Level Agreements *

Ganna Frankova,[1] Artsiom Yautsiukhin,[1] and Magali Seguran[2]

[1] Dept. of Information and Communication Technology
University of Trento
Via Sommarive, 14
38050 Povo (TN)
Italy
email: {ganna.frankova,evtiukhi}@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. When designing a service-based business process employing loosely-coupled services, one is not only interested in guaranteeing a certain flow of work, but also in how the work will be performed. This involves the consideration of non-functional properties which go from execution time, costs, up to security and trust. Ideally, a designer would like to have guarantees over the behavior of the services involved in the process. These guarantees are the object of Service Level Agreements. We propose a methodology to design service-based business processes together with service level agreements that guarantee a certain quality of execution, with particular emphasis on the security aspects. Starting from an early requirements analysis modeled in the SI* formalism, we provide a set of user-guided transformations and reasoning tools which final output is a set of processes, in the form of Secure BPELs, together with a set of service level agreements to be signed by participating services. To show the potential impact on security guarantees, we illustrate the functioning of the methodology on a e-business banking example inspired by an actual industrial case.

1 Introduction

Web services features of autonomy, platform-independence, readiness to be described, published, discovered, and orchestrated are increasingly exploited by companies to build massively distributed and loosely coupled interoperable applications [27]. Enterprises not only export their services as web services, but also develop their business process to be web service-based. Since services may

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

be offered by different providers, non-functional properties become of paramount importance in defining the usability and success both of a service and of web service-based business process.

The non-functional properties of a service can be agreed a priori between the web service provider and consumer by specifying a Service Level Agreement (SLA) [25, 13, 22]. SLA for web services is a binding contract between the web service provider and consumer which specifies a collection of service level requirements that are negotiated and mutually agree upon [5]. In [2] we provide semantics of a SLA for web services considering each SLA term as containing a guarantee, i.e, a right or obligation of the signing parties. The objects of a SLA can take many forms: it can be a maximum response time, a cost per operation, or it can take more complex form such as "the service should execute in less than 5 seconds 99% of the times from 9am to 5pm". In this work we concentrate on the first case.

If on the one hand, experts from the industry state that enterprise business objectives should form the fundamental basis of the SLA [1]. On the other hand, developing an appropriate SLA that supports business goals of an enterprise is not trivial task and requires great deal of design by an expert human operator.

The present work fills the gap among the requirements engineering methodologies and the actual generation of business processes based on Service-Oriented Architectures. We propose the BP&SLA methodology for deriving service-based business processes with service level agreements from the informally specified early business requirements. The derived service-based business processes together with service level agreements are executable. The hierarchy of business processes is expressed by WS-BPEL and the related SLAs are specified by WS-Agreement. As the proposed methodology focuses on security and trust aspects, secure features business processes are implemented in Secure BPEL [9], a specification language for secure business processes.

The remainder of the paper is organized as follows. In Section 2, we introduce e-business banking case study that is used as a running example throughout the paper. Section 3 is devoted to the proposed BP&SLA methodology. Related work is discussed in Section 4. Concluding remarks are summarized in Section 5.

2 Running Example: e-Business

In any e-business scenario the quality of the execution of a process is crucial, in particular, in the context of e-banking the security aspects are of paramount importance. Let us consider the case of a typical loan origination process. The scenario is provided by the courtesy of SAP [3] and is a working scenario of the IST-FP6-IP-SERENITY project [4].

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

[4] 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.

The scenario presents the case of a customer who needs a loan in order to buy a house. The selected bank employs several business processes that have a hierarchical nature which contribute to achieving the goal of the `Loan origination` super-business process. All the processes are provided by different actors inside or outside the bank and the business processes are l`aunching of the loan origination process`, `reception of the customer`, `management of the loan origination process`, `external credit worthiness check`, and `internal credit worthiness check`.

The `bank` is responsible for launching the loan origination process but gives the responsibility of it to the `bank manager`. The loan origination process is decomposed in two processes. The reception of the customer that consists in checking his identity with the internal computer system and the management of the loan origination process. The bank manager gives the responsibility to the `post-processing clerk` on the management of the loan origination process. The management of the loan origination process is decomposed in two sub-processes check internal rating and check external rating.

When the identity is checked, a post-processing clerk sends requests to the `Credit Bureau` who is in charge of providing the business process of credit worthiness check. The credit worthiness of the customer is checked querying some Credit Bureaus. As several Credit Bureaus can be contacted, one might be more trusted than the others. For example, some agencies might be less trustworthy than other ones. Indeed, the three major credit bureaus in the United States feed information to hundreds of credit sub-stations across the United States[5]. The problem is that unless the substations update their information daily the information they retain is not always up to date. The post-processing clerk uses the results of calculation of internal rating obtained thanks to credit scoring methods and analyses the rating. If the internal rating process assigns a low risk level for the customer's application, the loan origination process moves to a next phase, i.e, price calculation process and negotiation and signature of the contract (see [9] for more detailed description of the loan origination process scenario).

From the business point of view, the process is quite common, its formal modeling and creating tools for its management in a robust and secure manner is a challenge. Next we consider a methodology to go from early requirements all the way to executable business processes with fixed quality of services regarding security.

3 The BP&SLA Methodology

Judging what is the appropriate service level agreement to sign after having defined the business objectives is far from being a straightforward task. With the Business Processes with Service Level Agreements (BP&SLA) methodology, we provide means to go from a high-level analysis of the business requirements

[5] http://www.creditclean.com/creditbureaus.htm

all the way to the definition of the processes to be executed and the service level agreements to be signed in order to guarantee certain quality of service. The methodology consists of four main phases which are, referring to Figure 1, (1) early requirements engineering, (2) business process hypergraph derivation, (3) hierarchy of business processes derivation, and (4) constraint reasoning for service level agreements derivation.

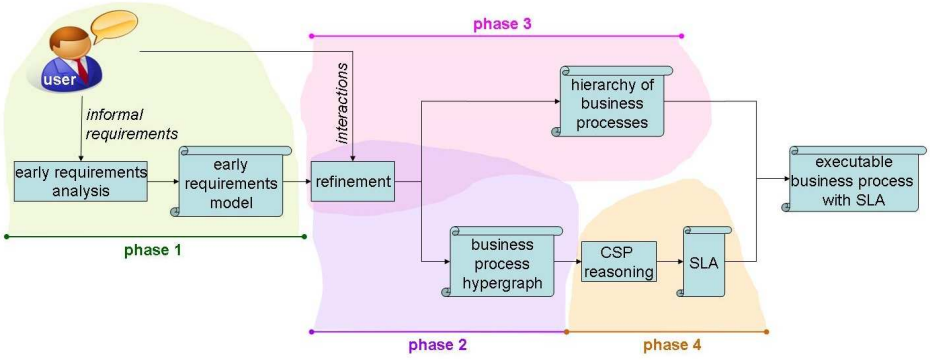

Fig. 1. The BP&SLA Methodology.

During the first phase the end user or domain expert provides informal requirements that form the seed for developing formal processes. These early requirements are formalized following the Secure Tropos methodology, an extension of the well established Tropos software engineering methodology [4, 6]. The output of this phase is an early requirement model. The model is far from being an executable entity, but rather it is a conceptual description of the actors involved in the business, their goals and their trust and security relations. To transform the model into something executable, in the second and third phase, one navigates automatically the model and asks user intervention every time that an unambiguous choice is necessary. The results of the refinement of the early requirements are an intermediate model necessary to perform the reasoning on qualities of services, the business process hypergraph, and a hierarchy of business process ready for execution, phases 2 and 3, respectively. The business process hypergraph is then further analysed to build a constraint problem which represents the relationships among the various elements of the processes regarding quality of service and security properties of the processes. By reasoning with these constraints it is possible to derive the appropriate service level agreements to be signed in order to guarantee a certain quality of service when executing the process, phase 4. The final output of the methodology is a hierarchy of business processes ready for execution together with service level agreements fulfilling a specific quality of service need. Let us consider next each of these phases individually. We do not only present the phases of the methodology, but also look

at how the application of the proposed methodology leads to a set of executable business processes and service level agreements. We consider the loan origination process scenario presented in the Section 2 as a running example.

Phase 1. Early Requirements Engineering

Early requirements engineering aims at analysing the organizational context within which a system will eventually operate [33]. During an early requirements analysis the domain actors and their dependencies on other actors for goals to be fulfilled are identified. For early requirements model elicitation in the context of security, one needs to reason about trust relationships and delegation of authority.

We employ the *SI**/Secure Tropos modelling framework [15, 23] to derive and analyse both functional dependencies and security and trust requirements. For the acquisition of the early requirements model we employ several modelling activities. Actor modelling to identify the principal stakeholders (actors) and their objectives (goals). Each goal might be refined by AND/OR goal decomposition that AND/OR decomposes a root goal onto sub-goals. It might happen that an actor does not have the capabilities to achieve his own objectives by himself. In this case that actor has to delegate the objectives to other actors that leads to their achievement outside the control of the delegator. *SI**/Secure Tropos supports two types of delegations. *Delegation of execution*, i.e, at-least delegation, means that one actor delegates to another one the responsibility to execute a service. *Delegation of permission*, i.e, at-most delegation, models the transfer of entitlements from an actor to another. We use functional dependency modelling to identify actors depending on other actors for obtaining services, and actors which are able to provide services. Permission delegation modelling is used to identifying actors delegating to other actors the permission on services. *SI**/Secure Tropos supports two types of trust dependencies. *Trust of execution*, i.e, at-least trust, means that a one actor trusts that another one will at least fulfill a service. While the meaning of *trust of permission*, i.e, at-most trust, is that an actor trusts that another actor will at most fulfill a service, but will not overstep it. Trust modelling aims at identifying actors trusting other actors for services, and actors which own services.

The early requirement model for the loan origination process case study described in Section 2 is depicted in Figure 2.

The early requirement model presents the principal entities involved, (1) actors depicted as circles and (2) interests, i.e., goals, presented as ovals. The `Bank` actor has the goal to `launch loan origination process`. The goal is delegated to the `Bank manager` actor. The delegation of execution is depicted with two lines connected by a delegation of execution (De) graphical symbol. The `Bank` actor trust the `Bank manager` actor on execution of the goal. The trust on execution is depicted with two lines connected by a trust on execution (Te) graphical symbol. In order to fulfill the goal the `Bank manager` actor refine it by an AND decomposition, depicted with a goal refinement symbol marked with AND, into goals to `receive a customer` and to `manage loan origination`. The `Bank manager`

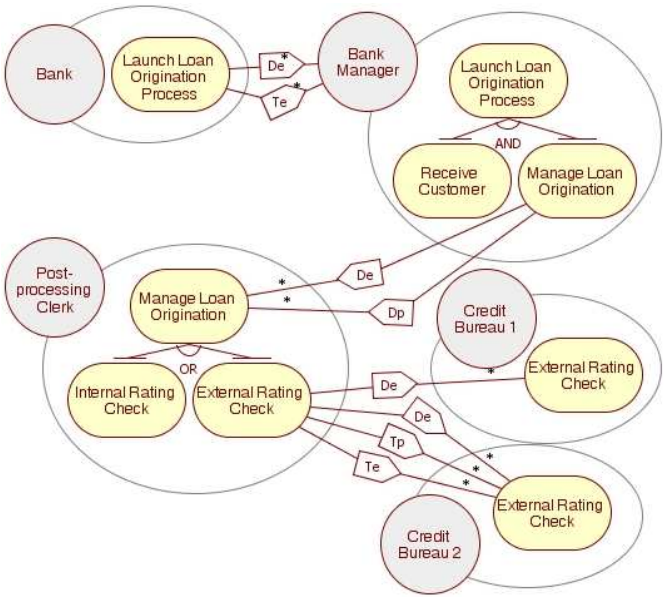

Fig. 2. The Early Requirement Model.

actor delegates the last goal to the `Post-processing clerk` actor. Here not only at-least delegation of execution, but also at-most delegation of permission is used. The delegation of permission is depicted with two lines connected by a delegation of permission (Dp) graphical symbol. The `Post-processing clerk` actor refines the `manage loan origination` goal into the `internal rating check` and `external rating check` goals. The goal is refined by an OR decomposition, depicted with a goal refinement symbol marked with OR. The `external rating check` goal is delegated to the `Credit Bureau 1` and `Credit Bureau 2` actors. While the `Post-processing clerk` trusts both on delegation and on permission to the `Credit Bureau 2` actor on processing of external credit check, there is no trust relation between the actor and the `Credit Bureau 1` actor. The trust on permission is depicted with two lines connected by a trust on permission (Tp) graphical symbol.

Phase 2. Business Process Hypergraph Derivation

The second phase of the BP&SLA methodology is devoted to creating an intermediate structure to reason about the business processes and their qualities. This intermediate structure is an hypergraph, which we define as follows.

Definition 1 A **business process hypergraph** $\mathcal{B}$ is a pair $\langle B, H \rangle$ where B is a set of business processes and H is a set of hyperarcs. A *hyperarc* is an ordered pair $\langle N, t \rangle$ from an arbitrary nonempty set $N \subseteq B$ (source set) to a

single node $t \in N$ (target node). Each hyperarc is associated with a vector of functions $\varphi = [\varphi_1\langle N, t\rangle, ..., \varphi_n\langle N, t\rangle]$ which calculate value of a target node taking as arguments source nodes.

The business process hypergraph is obtained by navigating the early requirement model and refining it eventually resorting to user interaction. This is performed algorithmically according to the procedure presented in Figure 3.

The algorithm takes the early requirements model, the actor with its goal and the vector of QoS parameters as an input. Each node of the business process hypergraph is a business process that corresponds to a goal in the early requirements model. As we consider the goals to be operational. Each hyperarc in the business process hyperarc corresponds to the goal refinement (a hyperarch is drawn as a solid line) or delegation dependency (a hyperarch is drawn by a dashed line) in the early requirements model.

In the business process hypergraph construction algorithm, we use the `addHyperArc (sourceNode, targetNode)` function to add one hyperarc in the business process hypergraph from a single source node to the target node. While the `addHyperArcForAll (sourceSetOfNodes,targetNode,aggregationFunction)` function adds one hyperarc in the business process hypergraph from a source set of nodes, i.e., `nodei[...]` to the target node. Where `aggregationFunction` is a vector of aggregation functions $\varphi = [\varphi_1\langle N, t\rangle, ..., \varphi_n\langle N, t\rangle]$ assigned to the business process hyperarc. The aggregation functions design takes into account the structural activity associated to the corresponding business processes, i.e, the source set of nodes, and the QoS parameter. Each aggregation function calculates the value of a target node taking as arguments source nodes (with the structural activity associated) for a particular QoS parameter.

The concept of AND goal decomposition from the early requirements model is refined as sequential or parallel business process composition in the business process hypergraph. Sequential business process composition corresponds to the sequence flow structural activity and the aggregation function for sequential aggregation of QoS parameters is applied. The parallel flow structural activity is used in case of parallel business process composition and the aggregation function for parallel aggregation of QoS parameters is applied. The concept of OR goal decomposition in the early requirements model is refined as branching statement in the business process hypergraph. If the structural activity is non-deterministic choice, the aggregation function for choice aggregation of QoS parameters is applied. In case of the design choice structural activity, the nodes corresponding to the business processes are connected by different hyperarc with the target node. The design choice structural activity appears in case of presence of different alternatives for the same business process, e.g., the same business process might be delegated to different partners that have different service level agreement offers. The refinement of the concept of AND/OR goal decomposition from the early requirements model can not be completely automated, but only supported as it happens in model-driven architectures. For instance, in case of AND goal decomposition, the system can provide assistance in refining the decomposition into sequence flow or parallel flow structural ac-

```
BPHC (SI*, actor, goal, QoS)
begin
   if goal is not a leaf goal
      currentNode = node (goal)
      for each children in AND
         nodei = BPHC (SI*, actor, childGoal, QoS)
         interactWithUser (sequence | parallel)
         if sequence
            addHyperArcForAll (nodei[...], currentNode, sequence)
         if parallel
            addHyperArcForAll (nodei[...], currentNode, flow)
      end for
      for each children in OR
         interactWithUser (non deterministic choice | design choice)
         if non deterministic choice
            nodei = BPHC (SI*, actor, childGoal, QoS)
            addHyperArcForAll (nodei[...], currentNode, switch)
         end if
         if design choice
            nodei = BPHC (SI*, actor, childGoal, QoS)
            addHyperArc (nodei, currentNode)
         end if
      end for
      for each delegated child
            nodei = BPHC (SI*, actor, childGoal, QoS)
            addHyperArc (nodei, currentNode)
      end for
      if trust dependency
         trustLevel(currentNode) = 1
         for each children
            trustLevel(childNode) = 1
      else
         trustLevel(currentNode) = 0
         for each children
            trustLevel(childNode) = 0
   return currentNode
   end if
   if goal is a leaf goal
      return node (goal)
   end if
end
```

Fig. 3. Business Process Hypergraph Construction.

tivity. OR goal decomposition might be refined into non-deterministic or design choice structural activity. While determining the proper structural activity is the domain dependence task that involves the user interactions. In the business process hypergraph construction algorithm we use the `interactWithUser (option1 | option2 ... | optionk)` function to support the interaction with the users with the aim to decide which structural activity to apply to for a particular goal decomposition. The users determine the proper structural activity based on the proposed options where the only one option has to be selected.

Each node in the business process hypergraph is assigned with a vector of QoS parameters and a trust level value (TL). The values of the QoS parameters correspond to the QoS that can be achieved by the business process. The trust level value denotes the level of trust between the truster and the trustee on the fulfilling of the business process (here we employ only at-least trust). In [10], we propose a methodology that identifies the concrete business process providing the highest quality of service and protection among all possible design alternatives. The idea is to take into account the level of trust of service providers and adjusts the expected quality value correspondingly. In spite of the fact that the approach to use the notion of trust as weighting factor is promising, the author do not clarify how the trust values are decided. Instead in our approach the trust level is determined from the reasoning on the presence/absence of trust dependencies in the early requirements model. Further, the determined trust level of service providers might be employed when there is a possibility to choose one business process from the several alternatives suggested by different providers.

The problem of finding service level agreements for business processes is then a problem of reasoning on the business process hypergraph.

The hypergraph corresponding to the case depicted in Figure 2 is shown in Figure 4. Each goal of the early requirement model is associated with a node of the hypergraph. Each node of the hypergraph is a business process.

The nodes `Receive Customer` and `Manage Loan Origination` are connected by one hyperarc with the top node `Launch Loan Origination Process`, that means that the business processes `Receive Customer` and `Manage Loan Origination` contribute to satisfaction of the global goal `Launch Loan Origination Process`. The dashed hyperarc leads from the delegated (here we employ only at-least delegation) business process `Manage Loan Origination` to the target one. The nodes in the business process hypergraph are assigned with vectors of QoS parameters and trust level values. The trust level is determined from the reasoning on the presence/absence of at-least trust dependencies in the early requirement model presented in Figure 2. A vector of aggregation functions $\varphi = [\varphi_1\langle N, t\rangle, ..., \varphi_n\langle N, t\rangle]$ is assigned to the hyperarc. The aggregation function takes into account the structural activity associated to the `Receive Customer` and `Manage Loan Origination` business processes and the QoS parameter. The notion of sequential and parallel composition corresponds to a refinement of the concept of AND goal decomposition. If the structural activity is sequence flow, the aggregation function for sequential aggregation of QoS parameters is applied.

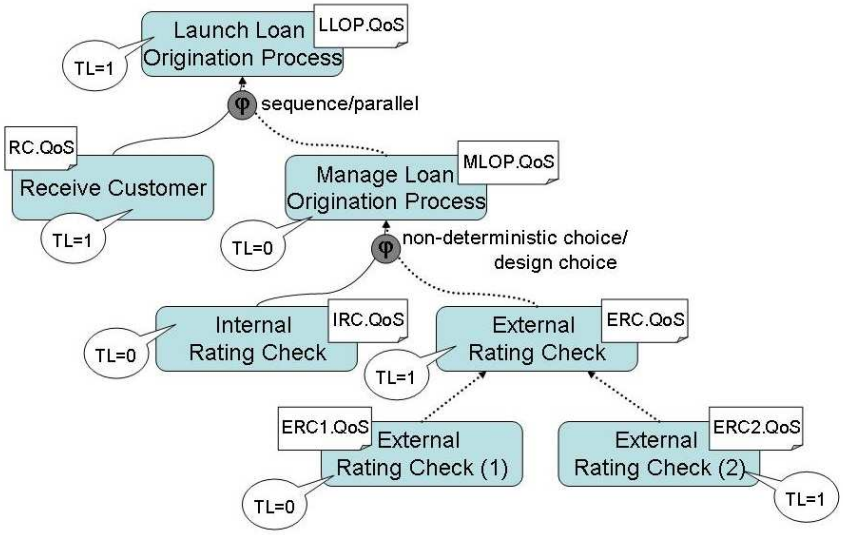

Fig. 4. Business Process Hypergraph.

If the structural activity is parallel flow, the aggregation function for parallel aggregation of QoS parameters is used.

The nodes `Internal Rating Check` and `External rating Check` are connected by one hyperarc with the target node `Manage Loan Origination`. The business process `External Rating Check` is delegated and is expressed by the dashed hyperarc. A vector of aggregation functions φ is assigned to the hyperarc. The aggregation function takes into account the structural activity associated to the `Internal Rating Check` and `External Rating Check` business processes and the QoS parameter. Branching statement is a refinement of the concept of OR goal decomposition. If the structural activity is non-deterministic choice, the aggregation function for choice aggregation of QoS parameters is applied. If the structural activity is design choice, the nodes `Internal Rating Check` and `External rating Check` are connected by different hyperarc with the target node `Manage Loan Origination`.

The nodes `External Rating Check (1)` and `External rating Check (2)` are connected by two hyperarc with the target node `External Rating Check`. Both the business process `External Rating Check (1)` and `External rating Check (2)` are delegated that is expressed by the dashed hyperarcs.

Phase 3. Hierarchy of Business Processes Derivation

The third phase of the BP&SLA methodology is dedicated to hierarchy of business processes construction. We build the hierarchy of business processes with the aim to use it for obtaining a set of executable business processes. These

are created following the Secure BPEL specifications [9]. Where each delegated business process is labeled with a service level agreement derived in the phase 4.

The hierarchy of business processes, as well as the business process hypergraph, is derived by refining the early requirements model. As we build the hierarchy to obtain executable business processes with service level agreements, we must clearly determine (i) the business processes, (ii) which partner proceeds which business process, and (iii) delegation and trust dependencies among the involved partners.

The hierarchy of business processes construction algorithm is presented in Figure 5.

The algorithm takes the early requirements model and the actor with its goal as an input. As for the business process hypergraph construction, we consider the level of goals in the early requirements model to be the level of business processes in the hierarchy of business processes. Further, the business process(es) proceeded by one actor are grouped and marked with the actor that proceeds them. In the hierarchy of business processes construction algorithm we use the `addActorLabelToNode (node)` function to mark a business processes with the actor that proceeds it. When several business processes are proceed by one actor, we consider that there is some relation among them. The `addRelation (nodes, relationType)` function of the algorithm adds a link that describes relation among business processes in the hierarchy of business processes where `relationType` is {sequence, parallel, switch, OR}. As in the case of business process hypergraphs, determining the proper type of relation, that depends on the structural activity, is the domain dependence task that involves the user interactions.

The notions of delegation the SI*/Secure Tropos methodology supports allows us not only to present the delegation dependencies among partners in the hierarchical structure of business processes, but also to label with service level agreements only the business processes that are delegated. In this work, we adopt only the delegation of execution dependencies, but not the delegation of permission ones. We consider the fact that one needs to sign a service level agreement with the partner only in case of transfer of responsibilities to the partner, i.e., the business process is delegated to the partner and the partner processes it. While if there is only a fact of transfer of entitlements, i.e., the business process is delegated to the partner and the partner has permissions to processes the business process, but do not actually does it, there are no reasons for a service level agreement signing. Furthermore, we employ the notion of trust dependency to show the trust relations between the actors in the hierarchical structure of business processes. The `addDelegationArc (currentNode, childActor)` function adds a link (drawn by dashed line) that describes the delegation dependency in the hierarchy of business processes. While the `addTrustArc (actor, childActor)` function adds a link (drawn by solid line) that describes the trust dependency in the hierarchy of business processes.

The specification language for secure business processes Secure BPEL [9] is a dialect of WS-BPEL for the functional parts and abstracts away low level

```
HBP (SI*, actor, goal)
begin
   if goal is not a leaf goal
      currentNode = node (goal)
      for each children in AND
         nodei = BPHC (SI*, actor, childGoal)
         interactWithUser (sequence | parallel)
         if sequence
            addRelation (nodei[...], sequence)
         if parallel
            addRelation (nodei[...], flow)
      end for
      for each children in OR
         interactWithUser (non deterministic choice | design choice)
         if non deterministic choice
            nodei = BPHC (SI*, actor, childGoal)
            addRelation (nodei[...], switch)
         end if
         if design choice
            nodei = BPHC (SI*, actor, childGoal)
            addRelation (nodei[...], OR )
         end if
      end for
      for each delegated goal
            nodei = BPHC (SI*, actor, childGoal)
            addDelegationArc (currentNode, childActor)
            addActorLabelToNode (nodei)
            addActorLabelToNode (currentNode)
      end for
      for each trusted goal
            nodei = BPHC (SI*, actor, childGoal)
            addTrustArc (actor, childActor)
            addActorLabelToNode (nodei)
            addActorLabelToNode (currentNode)
      end for
   return currentNode
   end if
   if goal is a leaf goal
      return node (goal)
   end if
end
```

Fig. 5. Hierarchy of Business Processes Construction.

implementation details from WS-Security and WS-Federation specifications. Secure BPEL allows us to describe delegation (both delegation of execution and delegation of permission) and trust (both trust on execution and trust on permission) relations among all the partners that execute sub-business processes in the context on the global business process.

The hierarchy of business processes corresponding to the early requirement model for the loan origination process case study is shown in Figure 6.

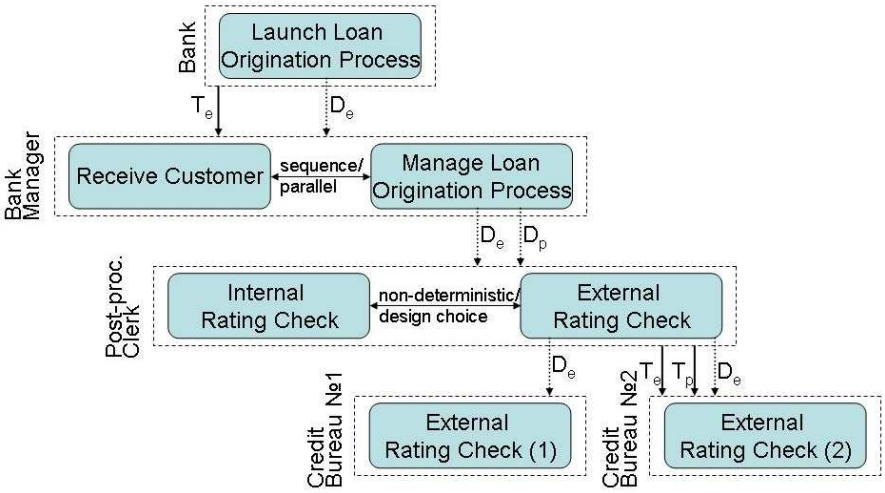

Fig. 6. Hierarchy of Business Processes.

Each goal is associated with a business process, represented by a rounded-corner rectangle in the hierarchy. Dashed rectangles are used in order to represent the actors that proceed the business processes. In our case these actors are the `Bank`, the `Bank Manager`, the `Post-processing Clerk`, the `Credit Bureau 1`, and the `Credit Bureau 2`.

The dependencies among actors, i.e., delegation and trust, are represented as dashed and solid lines correspondingly. The `Bank` actor delegates the `Launch Loan Origination Process` business process to the `Bank Manager` actor. The delegation of execution dependency is depicted by dashed line marked with the delegation of execution (De) symbol. The delegation of execution line connects the delegated business process, i.e., the `Launch Loan Origination Process` business process with the delegatee, the `Bank Manager` actor. The `Bank` actor trust the `Bank Manager` actor to fulfill the `Launch Loan Origination Process` business process. The trust on execution dependency is depicted by line marked with the trust on execution (Te) symbol. The trust on execution line connects the

trusted business process, i.e., the `Launch Loan Origination Process` business process, with the trustee, the `Bank Manager` actor.

The relation among business processes proceeded by the `Bank Manager` actor is defined by the structural activity associated to the `Receive Customer` and `Manage Loan Origination` business processes. The notion of sequential and parallel composition corresponds to a refinement of the concept of AND goal decomposition. If the structural activity is sequence flow, the sequence relation is applied. If the structural activity is parallel flow, the relation is the parallel one.

The relation among business processes proceeded by the `Post-proceeding Clerk` actor is defined by the structural activity associated to the `Internal Rating Check` and `External Rating Check` business processes. Branching statement is a refinement of the concept of OR goal decomposition. If the structural activity is non-deterministic choice, the non-deterministic choice relation is applied. If the structural activity is design choice, the design choice relation is applied. The `Bank Manager` actor delegates the `Manage Loan Origination` business process to the `Post-processing Clerk` actor. The delegation of execution and delegation of permission lines connects the delegated business process, i.e., the `Manage Loan Origination` business process with the delegatee, the `Post-processing Clerk` actor. The delegation of permission dependency is depicted by dashed line marked with the delegation of permission (Dp) symbol. There are no trust dependencies between the `Bank Manager` and the `Post-processing Clerk` actors on the `Manage Loan Origination` business process.

The `External Rating Check` business process is delegated to the `Credit Bureau 1` and the `Credit Bureau 2` actors. The delegation of execution lines connect the delegated business process, i.e., the `External Rating Check` business process with the delegatee, the `Credit Bureau 1` and the `Credit Bureau 2` actor. There are no trust dependencies between the `Bank Manager` and the `Credit Bureau 1` actors on the `External Rating Check` business process. While trust on execution and trust on permission lines connects the trusted business process, i.e., the `External Rating Check` business process, with the trustee, the `Credit Bureau 2` actor. The trust on permission dependency is depicted by line marked with the trust on permission (Tp) symbol.

Phase 4. Constraint reasoning for service level agreements derivation

In the last phase of the BP&SLA methodology service level agreements for business processes are derived by reasoning on the business process hypergraph. The reasoning technique we employ in this work is constraint programming. The key idea is to state the relationships among the qualities of processes and their activities as a set of constraints.

Formally, the constraint satisfaction problem is defined as follows:

- a set of variables $\{x_1,.., x_n\}$,
- for each variable x_i a finite set D_i (its domain) of possible values,
- a set of constraints, i.e., relations or expressions, restricting the values that the variables can simultaneously take [32].

We build a constraint systems by recursively navigating the business process hierarchy and hypergraphs. The algorithm is presented in Figure 7.

```
CSPEC (BPH, node, CSP, QoSDomain)
begin
   if node is not a leaf node
      addToCSP (Var node ∈ QoSDomain)
      if decomposition = AND
         for all nodes
            expr = expression (nodes, flow/sequence)
         end for
         addToCSP (Var node = expr)
      end if
      if decomposition = OR
         for all nodes
            if non deterministic choice
               expr = exression (nodes, switch)
            end if
            if design choice
               expr = expression (node, mult xi)
               where xi = 0 or 1 and sum(xi) = 1
            end if
         end for
         addToCSP (Var node = expr)
      end if
   end if
   for every node
      CSPEC (BPH, node, CSP, QoSDomain)
   end for
   if node is a leaf node
      addToCSP (Var node ∈ QoSDomain)
   end if
end
```

Fig. 7. Constraint System Building.

The algorithm takes the business process hypergraph, the node to start with, and the problem domain as an input, and it builds a constraint expression for every level of the hypergraph. Intuitively, the expression represents the quality of service for that level. For each level a new fresh variable is added and its range is restricted to the domain of the quality of services. Depending on what kind of children are available for that level different kind of expressions are built. If the children are connected with AND, the expression is built as an aggregation of the variables representing the children nodes. In the case of choice, there are

different expressions for each child and an additional expression represents the fact that only one child will contribute to the execution (the sum of x_i).

Once the constraint expressions are built, the algorithm proceeds recursively on all children. If the node is a leaf node, then one simply adds a variable for that node and a constraint on the domain of the variable.

Once the constraint system is in place, one can perform constraint propagation to find the solution space for acceptable qualities of services. If then one desires to have service level agreements to attach to the business processes, it is simply a matter of performing a labeling of the solution space and obtaining satisfying values for the qualities of services. We remark that such a solution might not exists. in this case, the result of the methodology will be a set of processes, but with no quality guarantees.

Here we show the generation of service level agreements based on given quality of service requirements for the execution of the business process using the loan origination process case study.

In order to obtain the quality constraint expressions, we need to be given the domain over which the quality of services range, e.g., integers for costs or real numbers for response time. In the case of the proposed methodology, the QoS Domain is a vector of QoS with corresponding possible values for the parameters. The example of the QoS Domain we consider is the following vector: [Execution Time (ET) $\in$ N, Availability (Av) $\in$ N, Time to Recover after an attack (TR) $\in$ N].

Several examples of the aggregation functions for QoS are presented in [17]. The authors provide aggregation functions for such numerical QoS parameters as cost, execution time, etc. Aggregation functions for such QoS parameters as maximal execution time (Max ET), availability (Av) and maximal time to recover after an attack (Max TR) for sequential, parallel and choice structural activities are the following:

Str. activities	**Max ET**	**Av**	**Max TR**
sequence	$\varphi = \sum_{p=1}^{k} p_i$	$\varphi = \Pi_{p=1}^{k} p_i$	$\varphi = \sum_{p=1}^{k} p_i$
parallel	$\varphi = \sum_{p=1}^{k} p_i$	$\varphi = \Pi_{p=1}^{k} p_i$	$\varphi = \sum_{p=1}^{k} p_i$
choice	$\varphi = \max(p_1, ..., p_k)$	$\varphi = \min(p_1, ..., p_k)$	$\varphi = \max(p_1, ..., p_k)$

The expressions we obtain navigating the business process hypergraph from Figure 4 according to the algorithm presented in Figure 7 are the following.

Maximal Execution Time (ET)
LLO=LLO.ET+sum(RC.ET,MLO)
MLO=MLO.ET+max(IRC.ET,ERC)
ERC=ERC.ET+ERC1.ET*x_1+ERC2.ET*x_2 when $x_i \in 0,1$ and $\sum x_i = 1$.
Availability (Av)
LLO=LLO.Av*Π(RC.Av,MLO)
MLO=MLO.Av*min(IRC.Av,ERC)
ERC=ERC.Av*(ERC1.Av*x_1+ERC2.Av*x_2) when $x_i \in 0,1$ and $\sum x_i = 1$.
Maximal Time to Recover after an attack (TR)
LLO=LLO.TR+sum(RC.TR,MLO)
MLO=MLO.+max(IRC.TR,ERC)
ERC=ERC.TR+ERC1.TR*x_1+ERC2.TR*x_2 when $x_i \in 0,1$ and $\sum x_i = 1$.
where LLO stands for Launch Loan Origination, RC to Receive Customer, MLO to manage Loan Origination, IRC to Internal Rating Check, ERC to External Rating Check, ERC1 and ERC2 to External Rating Check(1) and External Rating Check(2) business processes.

Suppose the constraint propagation for maximal execution time QoS property for the super-process in order to achieve execution time less then 35 seconds is performed and we get the following satisfying values for the qualities of services: ERC1.ET=2 s, ERC2.ET=4 s, ERC.ET=10 s, IRC.ET=1 s, MLO.ET=5 s, RC.ET=7 s, and LLO.ET=8 s.

The SLAs for the delegated business processes are the following.

SLA(LLO)=LLO=LLO.ET+sum(RC.ET,MLO)=8+7+19=34 s
SLA(MLO)=MLO.ET+max(IRC.ET,ERC)=19 s
SLA(ERC)=ERC.ET+ERC1.ET*x_1+ERC2.ET*x_2=14 s
when $x_i \in 0,1$ and $\sum x_i = 1$
when MLO=MLO.ET+max(IRC.ET,ERC)=5+max(1,14)=19 s
ERC=ERC.ET+ERC1.ET*x_1+ERC2.ET*x_2=10+2*x_1+4*x_2=10+4=14 s
when $x_i \in 0,1$ and $\sum x_i = 1$.

Note that while choosing the business process among two alternatives External Rating Check(1) and External Rating Check(2) we rely on the trust levels of the providers Credit Bureau 1 and Credit Bureau 2 correspondingly. As the trust level of Credit Bureau 1 is 0 and the one of Credit Bureau 2 is 1, we choose the last option.

4 Related Work

Requirements engineering for business processes in the context of web services is gaining increasing attention in the Software Engineering and Service-Oriented Communities. This is witnessed by a number of relevant research proposal, that we review next. Lau and Mylopoulos [20] propose a design methodology for web services adapted from the Tropos [6, 4] 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. Penserini, Perini, Susi, and Mylopoulos [28] 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. On the negative side, Tropos is not tailored specifically to web service design. Therefore the proposed methodologies do not address the issue of integration of Web Service Business Process Language in order to specify actual behaviour of participants in a business interaction. 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 [11]. The agent-oriented methodology Tropos is used for analysing web service requirements by Aiello and Giorgini in [3]. 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. On the contrary, our work aims not only to obtain business processes from an early requirements analysis, but also to provide them with service level agreements. Furthermore, the work involves Tropos that does not support the notion of trust dependability while the Secure Tropos does.

Georg, Ray and France [12] 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 [16] 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 [7] 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 neg-

ative site, the approaches do not support the design of software in general and service-based business processes in particular.

Service level agreement issues are gaining attention and have been addressed in a number of recent works. However, the research mainly focuses on service level agreement specification, negotiation and monitoring of SLAs [31, 25]. Several languages for SLA specification have been proposed, most notably, WSLA [22], WS-Agreement [13], SLAng [19], WSOL [30]. Presently, SLA negotiation is mainly a manual process and full or partial automation is needed. Theoretical bases of SLA negotiation are provided by Demirkan, Goul and Soper [8]. The authors identify negotiation support system requirements. The critical issue is a common understanding of the terms among negotiating parties [24]. Using templates is a proposed solution [29]. Gimpel at al. [14] propose PANDA - Policy-driven Automated Negotiations Decision-making Approach. The approach automates decision-making within negotiation. Fundamental concepts of non-functional SLA monitoring are presented in [26] which discusses on the separation of the computation and communication infrastructure of the provider, service points of presence and metric collection approaches. The WS-Agreement is supported by the definition of a managing architecture: CREMONA - An Architecture and Library for Creation and Monitoring of WS-Agreement [21].

The issue of service level agreement generation we touch upon in the present work, is not well developed. An approach proposed by Cappiello, Comuzzi, and Plebani [5] presents a negotiation model to support the automatic generation of service level agreement on-the-fly. The authors developed a model to express web service quality, provider capabilities, and user requirements that is further employed in the negotiation model to generate service level agreement. In our approach, we tie business processes with service level agreements. We do not focus on SLA negotiation, while we take into account early requirements the user of the business process provide, the structure of the business process and security and trust concepts.

5 Concluding Remarks

The service level agreements an enterprise has with its service providers must support the business goals it wants to achieve. The issue of making a SLA that favors the business objectives requires significant commitment of resources from the enterprise side.

The presented work proposes the BP&SLA methodology for designing service-based business processes with service level agreements attached. The contribution allows for bridging the gap between the informally specified early business requirements the user provides and the executable business process together with the service level agreements the enterprise benefits from and achieves its business objectives. As the activities about assignment of responsibilities on business processes, need to be carefully considered from the security point of view, the proposed methodology focuses on security and trust aspects. The framework

supports the Secure BPEL language that allows for secure business processes specification.

Considering current web service standards stack, the output of the methodology is related to the following two standards: WS-BPEL to express the hierarchy of business processes and WS-Agreement for expressing the related SLAs. Secure BPEL [9], a specification language for secure business processes, is a dialect of WS-BPEL for the functional parts and abstracts away low level implementation details from WS-Security, WS-Trust and WS-Federation standards. Secure BPEL allows to describe delegation and trust relations among all the partners that execute sub-business processes in the context on the global business process.

The trust dependencies among involved parties are represented in the structures of the methodology, but are not exploited. As next step, we plan to refine and explore how the trust level of the provider affects the choice of the business process. Furthermore, we will focus on the multi-requirement analysis, which requires the identification of a decision-making function that selects the more preferable set of attributes. Also we extend the approach to *ensure* that the obtained business process with service level agreements is compliant with the stated requirements. Concerning the implementation of the methodology, a tool supporting the early requirements model refinement into the executable secure business process implemented by the Secure BPEL language is currently being developed in the framework of the IST-FP6-IP-SERENITY project.

Acknowledgments

The authors thank Marco Aiello, Fabio Massacci, and Fabio Casati for fruitful discussion. Ganna Frankova thanks the University of Groningen for hosting her while part of the presented research was performed.

References

1. SLA: Getting it Right. *Voice&Data*, March 05 2005.
2. M. Aiello, G. Frankova, and D. Malfatti. What's in an Agreement? An Analysis and an Extension of WS-Agreement. In *Proceedings of the 3rd International Conference on Service-Oriented Computing*, 2005.
3. 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.
4. 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.
5. C. Cappiello, M. Comuzzi, and P. Plebani. On Automated Generation of Web Service Level Agreements. In *Proceedings of the International Conference on Advanced Information Systems Engineering*, 2007.
6. J. Castro, M. Kolp, and J. Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems*, 27(6):365–389, September 2002.

7. 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.
8. H. Demirkan, M. Goul, and D. S. Soper. Service Level Agreement Negotiation: A Theory-based Exploratory Study as a Starting Point for Identifying Negotiation Support System Requirements. In *Proceedings of the 38th Hawaii International Conference on System Sciences*, 2005.
9. G. Frankova, F. Massacci, and M. Seguran. From Early Requirements Analysis towards Secure Workflows. Technical report, University of Trento, 2007. http://eprints.biblio.unitn.it.
10. G. Frankova and A. Yautsiukhin. Service and Protection Level Agreements for Business Processes. European Young Researchers Workshop on Service Oriented Computing, 2007.
11. 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.
12. 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*, 2002.
13. GGF. Web Services Agreement Specification (WS-Agreement), September 2005.
14. H. Gimpel, H. Ludwig, A. Dan, and B. Kearney. PANDA: Specifying Policies for Automated Negotiations of Service Contracts. In *Proceedings of the First International Conference on Service Oriented Computing*, 2003.
15. 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.
16. 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*, 2004.
17. M.C. Jaeger, G. Rojec-Goldmann, and G. Mühl. QoS Aggregation in Web Service Compositions. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service*, 2005.
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*, 2004.
19. D.D. Lamanna, J. Skene, and W. Emmerich. SLAng: A Language for Defining Service Level Agreements. In *Proceedings of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems*, 2003.
20. D. Lau and J. Mylopoulos. Designing Web Services with Tropos. In *Proceedings of IEEE International Conference on Web Services*, 2004.
21. H. Ludwig, A. Dan, and R. Kearney. CREMONA: an Architecture and Library for Creation and Monitoring of WS-Agreements. In *Proceedings of the Second International Conference on Service-Oriented Computing*, 2004.
22. H. Ludwig, A. Keller, A. Dan, R.P. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification. Version 1.0. IBM Corporation, January 2003.
23. F. Massacci, J. Mylopoulos, and N. Zannone. An Ontology for Secure Socio-Technical Systems. *Handbook of Ontologies for Business Interaction*, 2007.
24. Strbel Michael. *Engineering Electronic Negotiations*. Kluwer Academic Publishers, New York, 2002.

25. C. Molina-Jimenez, J. Pruyne, and A. van Moorsel. The Role of Agreements in IT Management Software. *Architecting Dependable Systems III*, pages 36–58, 2005.
26. C. Molina-Jimenez, S.K. Shrivastava, J. Crowcroft, and P. Gevros. On the Monitoring of Contractual Service Level Agreements. In *Proceedings of the First IEEE International Workshop on Electronic Contracting*, 2004.
27. M. P. Papazoglou and D. Georgakopoulos. Service Oriented Computing. *Communications of the ACM*, 46(10), 2003.
28. 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*, 2006.
29. D. M. Reeves, M. P. Wellman, and B. N. Grosof. Automated Negotiation from Declarative Contract Descriptions. *Computational Intelligence*, 18(4):482–500, November 2002.
30. V. Tosic. WSOL Version 1.2. Technical Report SCE-04-11, Department of Systems and Computer Engineering, Carleton University, July 2004.
31. J.J.M. Trienekens, J.J. Bouman, and M. van der Zwan. Specification of Service Level Agreements: Problems, Principles and Practices. *Software Quality Journal*, 12(1):43–57, March 2004.
32. E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, London, 1995.
33. E. Yu. Towards Modeling and Reasoning Support for Early Requirements Engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, 1997.