A Platform for Secure and Trustworthy Service Composition

Michela D'Errico, Francesco Malmignati
Selex ES S.p.A.
Rome, Italy
{michela.derrico, francesco.malmignati}@guests.selex-es.com

Giovanni Fausto Andreotti
Italtel S.p.A.
Milan, Italy
fausto.andreotti@italtel.com

Abstract — **The Future Internet is moving from today's static services to an environment in which service consumers will transparently mix and match service components depending on service availability, quality, price and security attributes. This fact poses some challenges in terms of security and trustworthiness that should be guaranteed to the final users. In this paper, we present a platform for secure service design and composition based on the Activiti open-source workflow engine and Business Process Model and Notation (BPMN) extensions for expressing security needs over service specifications. The platform, developed in the realm of the Aniketos FP7 funded project, offers the capability to service designers and service providers to establish and maintain trustworthiness and secure behavior in today's constantly changing service environments. In order to demonstrate the validity of this approach, the use of the platform is shown in a real application scenario in which a security requirement on trustworthiness specified by design needs to be monitored and guaranteed during service execution.**

Keywords-service composition; service design; service deployment; security requirements; trustworthiness.

I. INTRODUCTION

A web service composition is needed when a desired functionality cannot be provided by a single web service. A service designer who wants to create a composite service has to specify the way the atomic web services have to be composed in order to fulfill his objectives. To this aim, service compositions can be modelled as business processes, namely, a set of activities that interoperate to perform a task. In this process-oriented approach, service composition is described using workflow languages and technologies.

In this paper, we present part of the work carried out in the Aniketos funded project [1] that aims to establish and maintain trustworthiness and secure behavior in an Internet service environment. As a result, a platform has been developed in order to support the service designer in performing all the steps needed to create and to manage trustworthy and secure service compositions.

Currently, many Web Service (WS-*) specifications address security concerns, but most of the focus is on secure message exchange [2], and current orchestration and choreography lack support for the specification and enforcement of the security process level requirements. These higher order interactions must become a part of composition.

The Aniketos platform overcomes this by allowing a service designer to specify security needs during the modeling of the composite service and by supporting the service discovery and composition/adaptation based on security properties and not just on the functional descriptors.

The notion of a trustworthy service at the most basic level can be taken as a service satisfying some minimum security requirements, most notably attestation and authorization of service endpoints, and the use of secure communication channels [3], but also involves a judgment about how likely that service is to perform as claimed. At present, a disconnection exists between the diverse mechanisms for managing trust. There is also a lack of solutions for availability and security aspects of dynamic binding (e.g. at runtime) of services. Although trustworthiness aspects can be defined for services, it's a major challenge to define trustworthiness aspects for composite services. Some approaches take into account various factors such as reputations and qualities of the services [4] and confidentiality and integrity [5].

Aniketos offers a way of expressing different aspects of trustworthiness and provide design time and runtime modules for evaluating and monitoring the trust level between service providers/components.

The present paper is organized as follows: Section II is dedicated to the background concepts that have been adopted and further developed to realize the platform. In Section III, an overview of the components of the Aniketos platform and the main features supported by a set of software packages is described. The application of the Aniketos design time tool-chain and service runtime management is reported in Section IV. Section V shows how the Aniketos platform can be used to implement a real case study. Finally, Section VI deals with related works and concludes the paper.

II. COMPOSITE SERVICE MODELING

In this section, components and concepts adopted for the development of the platform are presented.

A. BPMN language

The platform for web service composition developed in Aniketos uses Business Process Model and Notation (BPMN) [6], a de facto standard for the process modeling, together with the Activiti engine [7], a process server able to execute BPMN business processes. The Aniketos platform adds to the service composition the support for security and trustworthiness properties management by extending the BPMN standard language with proper security annotations.

BPMN, being a flow chart based notation, facilitates the modelling of the workflow for the service composition. The resulting business process diagram is easily understandable by technical as well as business users.

The Aniketos platform provides the service designer a set of features for the realization of runnable secure and trustworthy composite services. The whole design time specification process, along with runtime tools available for the management of the service execution, are detailed in the remainder of the paper.

B. BPMN elements

The BPMN specification contains a large set of object types, over 100, but a very small set of the constructs can be used to model a service composition with the Aniketos platform.

Core BPMN elements are grouped into four categories: Flow objects, Connection objects, Swim lanes and Artifacts. Throughout the paper only the elements belonging to the first two categories will be used, specifically Event, Task and Gateway as Flow objects, Sequence Flow as Connection objects. In the following, a brief description of these elements is given:

- *Start event*: represents the event that triggers the start of the service execution.
- *End event*: represents the end of the process execution.
- *Task*: represents part of the work to be done in the process that cannot be further decomposed.
- *Gateway*: is used to model forking and merging of paths.
- *Sequence* Flow: is used to connect the Flow objects and to specify in which order the tasks must be executed.

III. ANIKETOS PLATFORM

An overview of the Aniketos platform is depicted in Figure 1.

Design time support is available for service designers that use the platform in order to build secure and trustworthy service compositions. The capabilities offered in terms of analysis and composition of services are supported by the underlying components of the platform, although in this paper we will focus on the use of the front-end tools for the specification of secure and trustworthy service compositions.

The runtime support is dedicated to the monitoring of service properties during execution and adaptation in case of violation of security specifications.

The Aniketos platform [8] is composed by a number of components that have been grouped into four packages:

- Socio-Technical Security Requirements
- Secure Service Specification, Discovery & Deployment
- Secure Service Validation & Verification
- Security Monitoring & Notification.

Software packages can be potentially targeted to a variety of application domains that need security and trustworthiness, thus, aiming to effective exploitation of Aniketos' results.

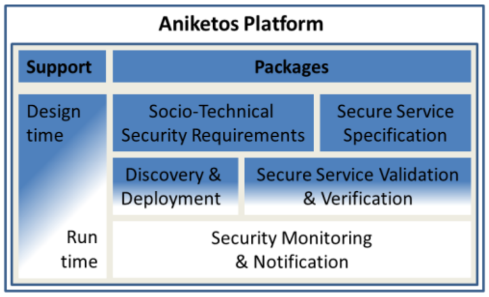

Aniketos Platform
Support
Packages
Design time
Run time
Socio-Technical Security Requirements
Secure Service Specification
Discovery & Deployment
Secure Service Validation & Verification
Security Monitoring & Notification

Figure 1: Aniketos platform overview.

The packaging takes into account the components functionalities, their licensing scheme and their role in the composite service process lifecycle.

The originality of this approach resides not only in the complete set of design time and runtime capabilities offered in an unique framework, but is also due to the possibility for the users to selectively adopt the features provided by the Aniketos packages according to their needs, thus, reducing the cost of the final realization.

A. Socio-Technical Security Requirements

This package offers a graphical tool [9] to model Socio-Technical Systems (STSs) [10][11] that are complex systems in which social actors interact with one another and with technical components to fulfill their goals. In such systems, many security issues arise from the interaction between actors and from the manipulation of the exchanged information.

A threat repository included in this package allows the designers to look for potential threats to be taken into account in the model and to acquire useful information on the threats and possible countermeasures in order to mitigate the associated risk.

In the realm of Aniketos, the package can be used to model high level security and trustworthiness requirements of a holistic application and/or parts of it. Of particular interest is the possibility to model the security requirements for a composite service process, which has to be developed from scratch or already exists and needs to conform to specific requirements.

B. Secure Service Specification, Discovery & Deployment

This package is used to model the composite service process with BPMN and to specify the security and trustworthiness requirements that the services taking part in the composition must fulfill. The BPMN is thus enriched with the consumer policies that represent the low level representation of the requirements expressed by the service designer through the Socio-Technical Security Requirements package. The package offers the possibility to publish Aniketos compliant services to the Aniketos *Marketplace*, an enriched service registry that supports discovery of atomic services and provision of security descriptors, also called agreement templates.

Finally, this package enables the deployment of the created service compositions to an application server for runtime execution.

Currently, a set of security properties are supported, namely trustworthiness, separation and binding of duty, confidentiality, non-repudiation, integrity and need-to-know. In the remainder of the paper, the trustworthiness management will be described in detail.

C. *Secure Service Validation & Verification*

This package offers verification and validation checks during design, announcement and execution of secure services.

This mechanism is built upon the matching between the consumer policies, representing the desired properties, and the services agreement templates, representing the properties that can be provided by the services.

The service validation process can be invoked when a composite service has been designed and the service developer needs to ensure that the security features of the involved services comply with the service specification.

The same check can be performed at runtime to validate that the offered security level of the composite service keeps complying with the consumer's security policy.

Furthermore, this package is used to perform a thorough security check on the properties declared by the components of a composite service.

D. *Security Monitoring & Notification*

This package enables the monitoring of execution of composite services and the generation of alerts when any malfunction in the proper service operation is identified.

Such malfunctions can refer to the violation of a service contract and/or the change in the trustworthiness level and/or detection of threat affecting the offered composite service.

The package enables subscriptions to service monitoring modules in charge of capturing and analyzing specific type of events produced by the service execution environment and of generating alerts and notifications for potential breaches at the service layer.

IV. SECURE SERVICE COMPOSITION FRAMEWORK

The *Service Composition Framework* (SCF) is the design time tool for secure and trustworthy service composition and is based on Activiti designer [7]. The SCF includes the functionalities of the *Secure Service Specification, Discovery & Deployment* package and the *Secure Service Validation & Verification* package, in order to offer an integrated and complete environment allowing the service designer to perform all the steps needed to create a composite web service. The design process starts with building the BPMN model of the composite service as a business process and ends with the deployment of the created composition plan as a web service. The deployment entails the announcement of the service to a Marketplace so that it can be made available and discovered by service providers.

A. *Business process modeling*

A composite service is a complex service made up of atomic services that can be connected in different ways, thus, providing different results based on how the services are combined.

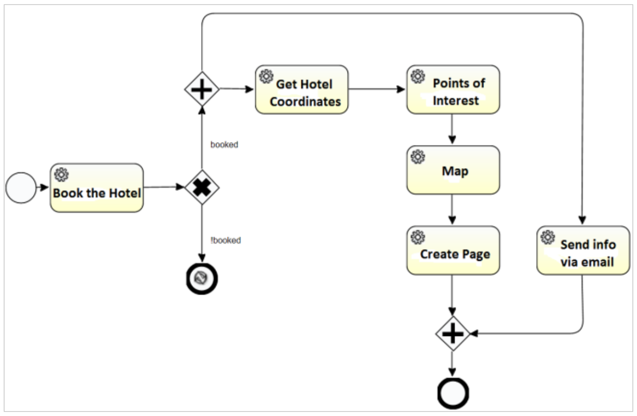


Figure 2: Service composition business process.

A business process starts with a *Start Event* element and ends with an *End Event* element. A web service composition can be modeled by using a *Start Event*, a set of service tasks connected with gateways and connection elements and an *End Event*, as shown in Figure 2. The tasks used for the service composition are service tasks that identify a piece of the process that is executed by invoking a web service.

The way the tasks are connected specifies how the final output of the composite service is built by using the output provided by the service tasks. The BPMN diagram is the graphical representation of the service composition. An example is shown in Figure 2.

B. *Binding and discovery of services*

In order to produce a runnable composition plan, each service task has to be bound to a web service. The binding has to be chosen to satisfy the functional requirements assigned to the service task. To this aim, for each service task, the *Type* parameter must be set to allow the discovery of well-suited web services. The value for the *Type* parameter is chosen from a service taxonomy agreed and shared by service designers and developers. The taxonomy is provided along with a vocabulary explaining what each service type means and entails in terms of the provided functionality.

In order to help the service designer to set the service *Type*, the SCF provides in a pop-up a Type Cloud showing the set of the service types that are available in the *Marketplace*.

Once specified the service *Type* for each service task the SCF is ready to execute the service discovery, which will return the set of operations provided by web services belonging to the service *Type* category specified. The SCF shows, for each operation, the input required and the output provided. For each service task, the service designer has to select one of the available web service operations and has to specify a valid input in terms of process variable or directly defining it with a plain value.

Moreover, for each service task, the service designer has to create the variable that will contain the result of the task, that is the output provided by the selected web service operation.

The set of process variables available for each service task contains the output variables of all the service tasks executed before in the process and the variables provided by the Start Event. The set of input added to the Start Event can be seen as global variables accessible by any task in the process.

C. *Security and trustworthiness requirements*

The binding of service tasks with web services has the aim to fulfill the functional requirements. The security and trustworthiness requirements can be specified once the binding has been completed. As for the properties specification at BPMN level, we will focus on confidentiality, integrity and trustworthiness.

The service designer can add confidentiality property if data to be transmitted between service tasks have to be kept confidential and thus the use of an encryption algorithm is required. The service designer can also specify which data (input, output or both) are required to be enciphered.

The integrity property can be added to specify that the service designer needs that a “sender” service has to apply a mechanism enabling the “receiver” service to detect whether data has been corrupted or modified by an attacker. In this way, the designer can specify the mechanism and the algorithm to be used by the “sender” service.

The trustworthiness property is specified by setting a threshold value that is computed [12] by taking into account a set of parameters such as service provider’s reputation and Quality of Service (QoS) [13].

The specified properties represent the consumer policy, namely the desired properties that have to be compared with the provided security properties to establish whether the web services selected during the discovery phase satisfy the security and trustworthiness requirements. The offered properties are encoded in *ConSpec* language [14] and written in a file called agreement template, associated to each web service in the Aniketos *Marketplace*.

To enable the comparison, the consumer policies have to be encoded in *ConSpec* language as well by using an editor available in the SCF. In this operation, the SCF helps the service designer with a *ConSpec* editor that will use the properties specified in the previous step to automatically fill out the consumer properties files.

D. *Creation and validation of composition plans*

After binding service tasks with web services and specifying security requirements, the SCF is used to create a set of composition plans. The operation selected for service task S_k during the discovery phase can be offered by $N_{ws}(k)$ different web services, this means that N_{cp} composition plans can be created satisfying the functional requirements. The number N_{cp} of possible composition plans is given by the following formula:

$$N_{cp} = \prod_{1}^{N} N_{ws}(k) \qquad (1)$$

where N is the number of service tasks in the composition and $N_{ws}(k)$ is the number of web services providing the web operation selected for the binding of the kth service task.

At this point, the service designer can use the SCF to start the security and trustworthiness verification of the composition plans. The result of this verification will be the set of composition plans having web services whose offered properties match the consumer policies.

E. *Rules definition and service deployment*

Then, the service designer has to select one of the composition plans and has to specify the rules for the management of events that can occur during runtime execution. A set of rules can be defined to handle specific events, such as a threat detection or a violation of the security and trustworthiness properties specified during design. For each rule, the service designer can specify the constraints for the event to fire the rule and the action to be performed once the rule is fired. For example, an action we introduce here is the recomposition, a mechanism that replaces the web service offering the operation selected during the binding. The result of this action will be a different runnable composition plan satisfying the same security and trustworthiness properties as the substituted web service.

Having specified the rules, the designer can complete the design time process by deploying the composite service. The deployment entails the exposure of the composition as a web service and its announcement in the Aniketos *Marketplace*, thus, allowing other service designers to use it as part of other service compositions.

F. *Runtime support*

The platform includes a Service Runtime Environment (SRE) that is in charge of executing the services and enforcing the rules specified by the service designer. To this aim the SRE must interact with the modules that monitor the services security and trustworthiness. Specifically, based on the events specified in the rules, the SRE subscribes to the modules that are able to detect and notify about those specific events. This mechanism allows the SRE to trigger the action specified in the rule when the related event is received.

Support for runtime management is provided by the *Security Monitoring & Notification* package.

V. APPLICATION TO A REAL CASE STUDY

In this section, we describe the application of the Aniketos design time and runtime support to a real example taken from an industrial case study [15]. The aim is to create a *HotelBookingService* that takes as input the user and hotel data and returns the reservation detail. The service will then be used to build a web application for hotel reservations. The process is split into the following main phases:

a) Design time: the designer creates a composite service with a trustworthiness security requirement. Before deploying the service the designer specifies that, if at runtime the requirement is not fulfilled, the service has to be recomposed. Then, the service is deployed and integrated into a web application that is available to the user.

b) Runtime: the system performs a service adaptation (in this specific case a recomposition) of the service in case of security violations detected during service execution, in a manner that is totally transparent to the user.

B. *Scenario definition*

The service designer aims to create a hotel reservation service that takes in input user's preferences and data that are necessary for booking the hotel and returns a web page with Points of Interest (POIs) related to the hotel location plus a confirmation mail. The scenario is depicted in Figure 3.

The composite service includes an atomic service that provides a map showing a set of locations. The designer selects the service offered by Service Provider A (SP-A) and the resulting composite service is deployed and integrated into a web application.

An incident happens to the servers of the SP-A: this affects the reputation of the service provider and thus the trustworthiness associated to it. The final result is that the trustworthiness becomes lower than the threshold value set by the designer at design time implying that the service properties do not match anymore the consumer policies and thus a recomposition is needed. The result will be the substitution of the atomic service provided by SP-A with a similar service provided by another Service Provider (SP-B) matching the trustworthiness requirement.

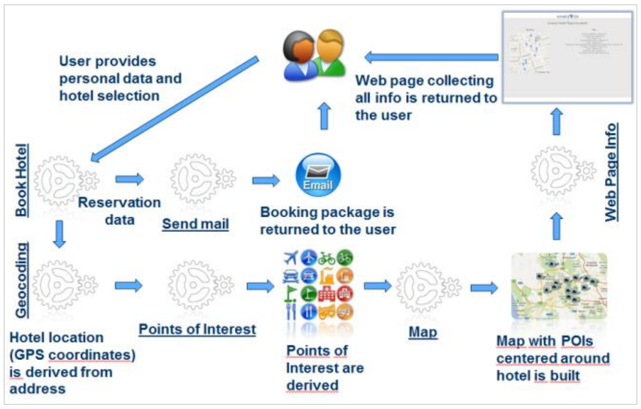

User provides personal data and hotel selection
Web page collecting all info is returned to the user
Book Hotel
Reservation data
Send mail
Email
Booking package is returned to the user
Web Page Info
Geocoding
Hotel location (GPS coordinates) is derived from address
Points of Interest
Points of Interest are derived
Map
Map with POIs centered around hotel is built

Figure 3: Application scenario.

C. *Creation of Composite Service*

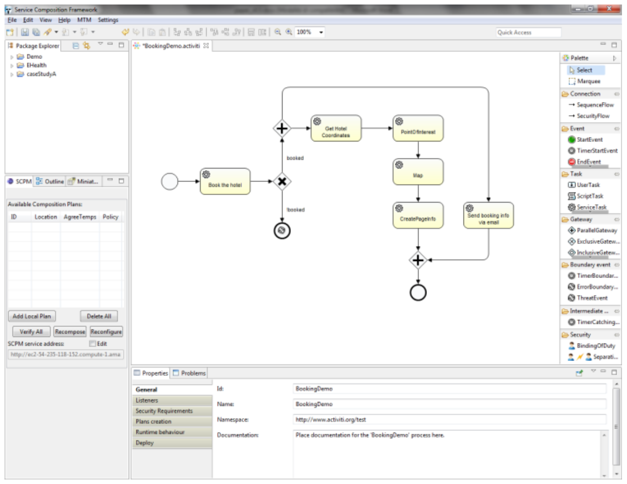
This section aims to describe the use of the Aniketos front-end tools following the process described in Section IV for the design of a secure and trustworthy composite service in the application scenario.

The process starts in Figure 4 with drawing the business process representing the composite service. Then, the following operations are performed in sequence: discovery, configuration and specification of the security and trustworthiness requirements over the services involved in the composition.

The requirements that are needed in this case are: *confidentiality*, *scope of usage* and *trustworthiness*.

- Confidentiality will be needed to transmit data related to user's information in a secure manner.
- Scope of usage (e.g., need-to-know) will guarantee that user's data will be used only in the scope of the service and not for any other purposes.
- Trustworthiness indicates that a (minimum) level of trustworthiness value for elementary services (*BookingSrv and MapSrv*) that belong to the composition is required.

Since trustworthiness will be the only requirement to be monitored in this scenario, it's worth to describe that in Aniketos the trustworthiness of the composite service is evaluated by using the weakest link principle. A specific module of the Aniketos platform (Trustworthiness module) evaluates the trustworthiness value for each service taking part in the composition [16]: the lowest value is returned as the trustworthiness value of the composite service. In Aniketos, the trustworthiness value is a combination of cognitive and non-cognitive measure of trust [17][18].



SCPM service address: http://ec2-54-235-118-152.compute-1.ama

Id: BookingDemo
Name: BookingDemo
Namespace: http://www.activiti.org/test
Documentation: Place documentation for the 'BookingDemo' process here.

Figure 4: Service Composition Framework.

When the modeling process is complete, the composition plans are generated and validated in terms of trustworthiness.

Finally, the service designer makes a selection and deploys a specific composition plan in order to make it available to other service providers through the Marketplace.

D. *Service-based application*

This section describes how the deployed service will be used in the final application.

A web service is provided with a Web Service Description Language (WSDL) [19] that exposes the operations offered by the service and gives information on how the operations can be invoked by a client. This means that the service result cannot be taken as it is but has to be integrated, for instance, into an application. For the hotel reservation application, the service client is integrated in a web page. The input required by the service is asked through a form that is submitted by the user. Thus, the composite service execution is triggered and the result is presented in a new web page returned by the composite service.

E. *Security management at runtime*

This section shows the runtime execution of the composite service including the monitoring for detection of contract violations. In this scenario, when the level of trustworthiness is no longer guaranteed by an atomic service belonging to the composition, an event detecting this contract violation is sent to the runtime environment.

According to the rule set at design time, a recomposition is triggered and leads to the substitution of the atomic service with another one matching the trustworthiness requirement. Figure 5 shows the application rendering when the atomic service that provides the map is replaced.

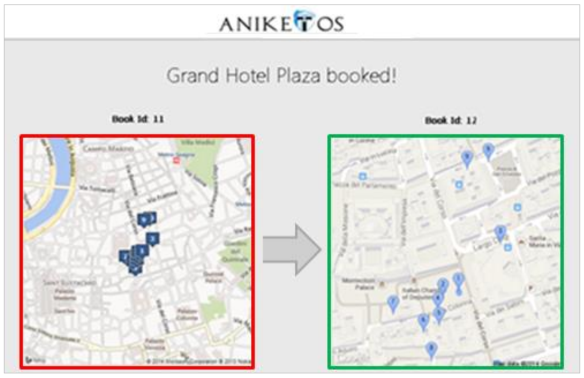


Figure 5: Application rendering.

This simple scenario demonstrates the effectiveness of the Aniketos approach that is to establish and maintain the security and trustworthiness properties of the service during service execution, in a totally transparent manner to the end-user.

VI. CONCLUSION AND FUTURE WORK

A platform for designing and ensuring secure and trustworthy service compositions has been presented. This approach, developed in the realm of the Aniketos project, covers all the phases in the service development chain, ranging from modeling and specification of security needs to the actual operation of the delivered services.

The Aniketos platform uses design time descriptions to establish trust and verify safe and secure service behaviour among several different service providers. Runtime monitoring and automatic adaptation of services are needed due to an evolving environment of threats and operating conditions. Down-time is costly; a composed service must be able to operate even during an attack (with possible limitations or change of behaviour), taking risks and adaptation costs into account.

Further improvements are needed to make discovery and security verification more mature, and to express what the compositions should do in case of attacks. Although some existing security modelling and verification techniques allow the service composer to specify security properties, the number of services satisfying these requirements may be large. It will be therefore important to enhance the Aniketos platform and offer a scalable approach to service compositions based on security properties.

Aniketos already delivers a functional service runtime environment that supports reception of warnings from a notification service and dynamically adapts the composition in a risk-reducing manner. However, further work is needed to enlarge the set of monitored security properties.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no. 257930 [1].

REFERENCES

[1] Aniketos: Ensuring Trusworthiness and Security in Service Composition, http://www.aniketos.eu [retrieved: April, 2014].

[2] A. Charfi et al., "Reliable, Secure, and Transacted Web Service Compositions with AO4BPEL," in Proc. of the European Conference on Web Services, 2006, pp. 23-34.

[3] Z. Jianwu et al., "On achieving trustworthy SOA-based Web Services," SAM'06, Las Vegas, NV, USA, 2006, pp. 341-347.

[4] H. Elshaafi, J. McGibney and D. Botvich, "Trustworthiness monitoring and prediction of composite services", ISCC, 2012, pp.580–587.

[5] B. Zhou et al, "Secure service composition adaptation based on simulated annealing", ACSAC, 2012, pp. 49–55.

[6] OMG, Business Process Model and Notation (BPMN), 2011, http://www.omg.org/spec/BPMN/2.0.

[7] Activiti BPM Platform, http://www.activiti.org [retrieved: April, 2014].

[8] Aniketos Deliverable 5.3 "Final Aniketos Platform integration", March 2014.

[9] Socio-Technical Security modeling language and tool, http://fmsweng.disi.unitn.it/sts [retrieved: April, 2014].

[10] F. E. Emery and E. L. Trist, "Socio-Technical Systems", Management Sciences, Models and Techniques, editors Churchman, C. W. Pergamon, London, 1960.

[11] F. Dalpiaz et al, "Security requirements engineering via commitments", IEEE STAST, 2011, pp. 1-8.

[12] H. Elshaafi, and D. Botvich, "Aggregation of trustworthiness properties of BPMN-based composite services.", IEEE CAMAD, 2012, pp. 383-387.

[13] E. Maximilien and M. Singh, "Agent-based trust model involving multiple qualities", Proc. 4th Int. Conf. on AAMAS, 2005.

[14] I. Aktug and K. Naliuka, "ConSpec - A Formal Language for Policy Specification", Electr. Notes Theor. Comput. Sci., (197) 1, 2008, pp. 45-58.

[15] Aniketos Deliverable 6.1 "Initial analysis of the industrial case studies", July 2011.

[16] H. Elshaafi, J. McGibney, and D. Botvich, "Trustworthiness monitoring and prediction of composite services", IEEE ISCC, 2012, pp. 580-587.

[17] Aniketos Deliverable 2.1 "Models and methodologies for embedding and monitoring trust in services", July 2011.

[18] Aniketos Deliverable 2.4 "Models and methologies for integrated security and trust paradigm for service contracts", June 2013.

[19] E.Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1", World Wide Web Consortium (W3C), 2001.