Visually Mapping Requirements Models to Cloud Services

Shaun Shei, Aidan Delaney, Stelios Kapetanakis and Haralambos Mouratidis

Abstract

We extend an existing visual language for requirements modelling to model the requirements of cloud services. To achieve this we demonstrate how candidate cloud services can be identified from existing visual requirements models. We further extend the meta-model of the visual language to include cloud requirements in order to migrate our candidate service to a cloud provider.

1 Introduction

Cloud computing allows the provision of a wide range of services through the abstraction of physical and virtual resources. This offers seemingly unlimited scalability, availability, flexibility and dynamic provisioning through a pay-per-use model. However, some organisations are still hesitant to fully commit to this technology due to negative publicity regarding data-breaches [1, 2], security leaks [3] as well as interoperability and compatibility issues when migrating towards cloud environments [4, 5, 6]. Cloud computing is built upon and extends several established concepts and technologies such as Service-Oriented Architecture (SOA), distributed computing and virtualization. Moreover, through extension of existing technologies we also inherit the security issues and vulnerabilities of each [7]. This creates a complicated scenario where we need to consider security from multiple perspectives. One method for tackling this problem is to adopt an agent oriented software engineering approach [8], where the focus is placed on analysing components and properties to elicit requirements for a software system. There is currently a lack of standardised modelling languages and approaches to holistically capture cloud computing environments in the context of software security. The existing notations capture aspects of different service models such as Software-as-a-Service [6], Platform-as-a-Service [9] or Infrastructure-as-a-Service [10]. There is a lack of a holistic modelling language that captures both the customer requirements and cloud services corresponds to the need for secure software systems in the industry [11]. Security is a concept that is often tacked on after the design and deployment of software systems, where security mechanisms are introduced in response to vulnerabilities as they appear. Security-by-design is a branch within recent research efforts [12], where the goal is to obtain a clear understanding of security issues early in the software development process. The Secure Tropos methodology provides a modelling language that represents security requirements through security constraints, allowing developers to model software systems and its organizational environment using actors, goals and relational links such as dependencies. The contributions presented in this paper are as follows:

- We define a pattern for service identification based on grouping a goal, plans and resources in software systems.
- We then model an initial description of properties required by services when migrated to cloud environments.

The rest of this paper is organized as follows. In section 2 we present an overview of the the Secure Tropos visual language. In section 3 we provide a standard definition of a software service and demonstrate how such services can be identified from a Secure Tropos requirements model. Section 4 extends the meta-model for Secure Tropos to incorporate cloud computing requirements as identified from the literature and describes how to adapt the identification of services towards a cloud environment. Finally, in section 5 we present our conclusion and future work.

2 Secure Tropos Notation

Secure Tropos is a requirements engineering methodology aimed at fully capturing the properties of software systems and the organizational environment, focusing on modelling security [13]. The language extends the concepts of (social) actor, goal, task, resource and social dependency from the i* modelling language

DOI reference number: 10.18293/DMS2015-015

and redefining existing concepts introduced in the Tropos language and development process [14]. The Secure Tropos methodology closely follows the software development life-cycle with emphasis on security and privacy requirements, allowing the developer to incrementally create and refine models of the system-to-be during the analysis and design stage.

The Secure Tropos notation is fully defined in [13]. Here we present an outline of the subset of the notation used in this paper. The concrete notation is presented within *views*, where each view denotes a specific phase of activity in the modelling process. We now discuss Secure Tropos Views.

2.1 Organisational View

The diagram in Figure 1 illustrates the main nodes of an organisational view of Secure Tropos. It depicts a node-link diagram enclosed in a bounding rectangle. The nodes in the node-link diagram vary in shape according to the type of Secure Tropos element that they depict. The links similarly vary.

The circular node depicts an *actor*. An actor can be a physical or abstract manifestation, with strategic goals and intentions. An example actor labelled "Lecturer" can be seen in Figure 1.

The semi-oval node depicts a *goal*. Goals represent an actors strategic interests, which can be decomposed into sub-goals and combined using Boolean operations. An example goal labelled "Get student academic achievements" can be seen in Figure 1. Goals are linked through a *Dependency* link, depicted by one semi-circles on each side of the goal element.

A *Dependency* link indicates that an actor depends on another actor in order to achieve some goal/plan or to obtain a resource, where the direction the semi-circle is pointing towards denotes the dependee. An example dependency link can be found linking the goal "Get student academic achievements" with the actor "University of Brighton" who depends on the actor "Lecturer" to achieve the goal.

Security Constraints are depicted by the octagon node. Security constraints define security requirements through a set of restrictions that limit the way goals can be carried out. An example of a security constraint "Keep account access secure" can be found from the actor "Lecturer" to the goal "Access student records".

2.2 Security Requirements View

The diagram in Figure 2 illustrates the security requirements view, which provides a detailed analysis of the organisational view. This view depicts a node-link diagram enclosed in a bounding circle, defined by an actor that is delegated as the solution "system". Several new elements are introduced in this view.

The elongated hexagon node depicts a *Plan*. A plan specifies the details and conditions under which a goal or measure is operationalised. "Lecturer fill in form" is an example of a plan.

The rectangle node depicts a *Resource*. Resources represent a physical or virtual entity. Resources can be linked to goals using a *Requires* link. An example of a resource is "Lecturer Notes" which is linked to the goal "Get academic achievements" via a requires link. The requires link indicates that the goal requires certain resources in order to be satisfied.

The pentagon node depicts a *Threat*. A threat indicates the potential loss or problems that can put the system at risk. The "Man-in-the-middle" is an example of a threat, which is linked via the *Impacts* link to the goal "Get student details".

The *Impacts* link indicates the presence of a threat targeting a goal.

The hexagon node depicts a *Security Objective*. An example of a security objective is "Ensure data is kept private", which is linked to the security constraint "Keep personal details private" and "Keep student records private" via *Satisfies* links and the *Security Mechanism* "Secure connection" via the *Implements* link.

The *Satisfies* link indicates that the security objective satisfies the given security constraint.

The hexagon node with two parallel horizontal lines depicts a *Security Mechanism*. A security mechanism is a method or procedure that enforces security objectives. "Secure connection" is an example of a security mechanism, which is linked via the implements link to the security objective "Ensure data is kept private".

The example organisational and security views demonstrate the syntactic richness of Secure Tropos. Of which, we have provided outline explanations of a subset. We will now use this subset to identify services in Secure Tropos models.

3 Service

Before we can begin the process of determining what properties and aspects to capture when modelling cloud services, there is a need to obtain a concrete definition on what a service is. The common definition of a service can be given as "The performance of work (a function) by one for another". However, service, as the term is generally understood, also combines the following related ideas [15]:

- The capability to perform work for another

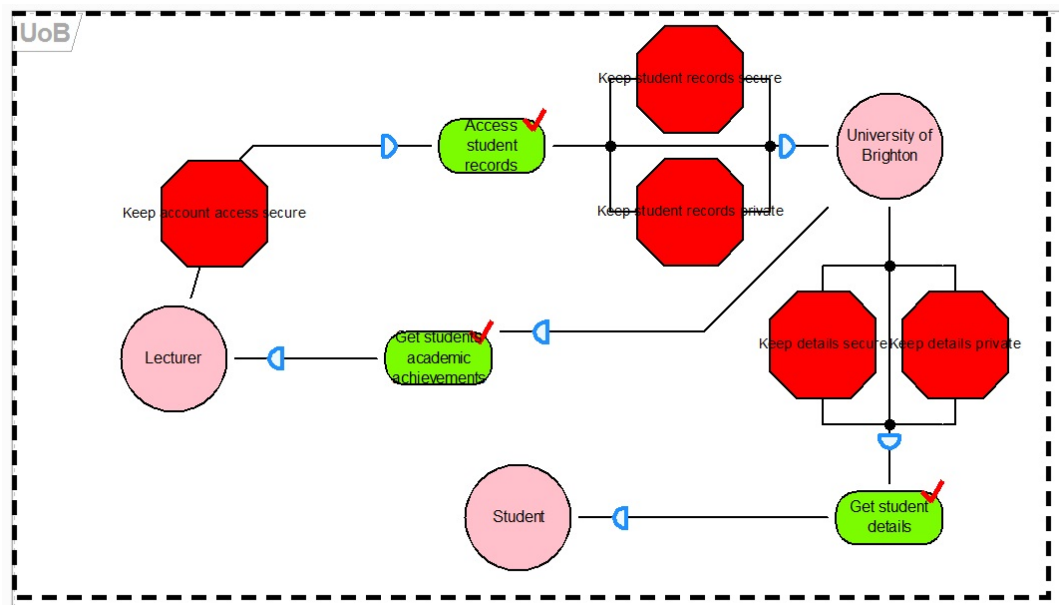

Figure 1. An example of an organisational view in Secure Tropos

- The specification of the work offered for another
- The offer to perform work for another

Our interpretation for a computing service is based on definitions provided by IT standards bodies, specifically from the "Organization for the Advancement of Structured Information Standards" (OASIS). OASIS defines a service as "A mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description" [15].

Capability The capability of a service represents the ability to do something. Capabilities are a way to meet the needs of an agent by fulfilling their requirements. A capability has no function on its own, therefore service functionalities in the service description link together with the capabilities in order to fulfill a purpose.

Interface The OASIS definition states that interfaces provide access to the capabilities of a service. The interface describes the means of interacting with a service and the actions involved in using a capability. These actions are initiated through specific protocols, commands, and information exchange specified by service functionalities in the service description. This is essentially the information that needs to be provided to the service in order to access it's capabilities and receive a result.

Service Description In order to ensure that services are published and visually available for access, the interaction with services are described in the service description in terms of inputs, outputs and associated semantics.

3.1 Service Identification

Based on the OASIS definition, we propose that in the context of our work a service can be identified and modelled through a combination of a goal, plans and resources. **Goals** provide a description of the main underlying capability of the service, which relies on the functionality provided by the plan and the attributes defined in the resource to give the service description. Each **plan** describes a functionality of the service, which is required to give purpose to the service capability through the interface. The **resources** provide a specification of the information required by the interface in order to access the capabilities of the service.

This pattern can be identified in Secure Tropos by indicating a goal, searching for dependants of the goal and including any plans and resources linked to the goal. The most basic form of the pattern includes a goal, plan and resource. As goals increase in complexity, additional plans and resources may be added in order to specify additional functionalities. A simple example denoting the identification of a service based on the goal "Get student details", plan "Student fills in form" and resource "Student Data" is shown in Figure

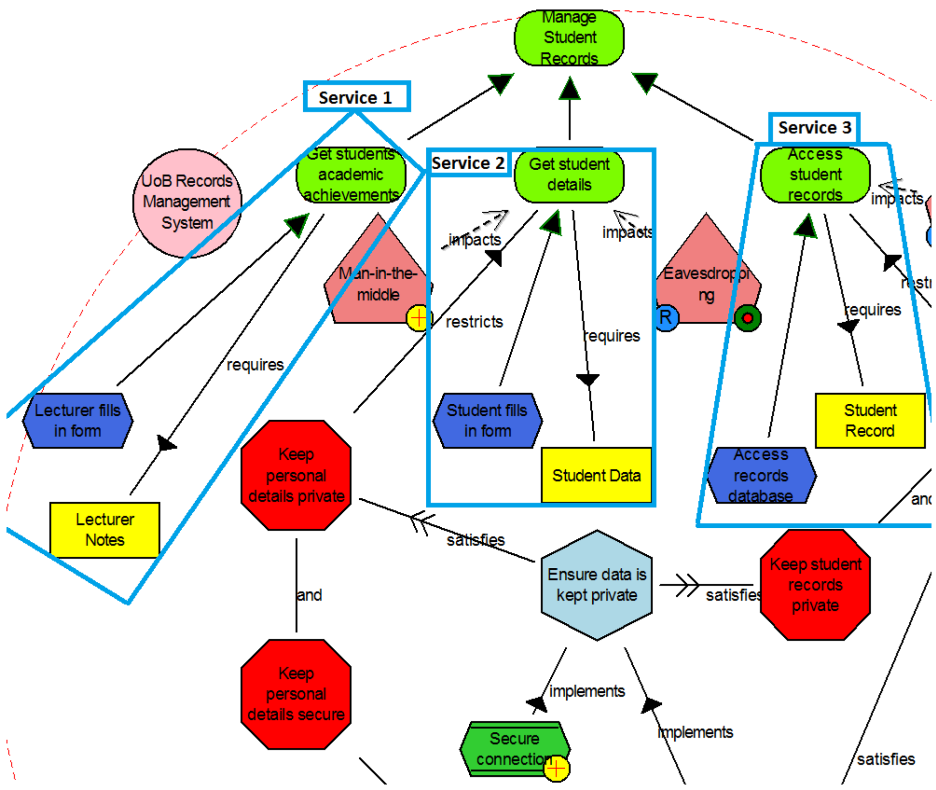

Figure 2. An example of a security requirements view in Secure Tropos.

2. We explain the key characteristics required to identify a service in the following sub-sections, justifying the goal, plan, resource pattern.

3.1.1 Capability and Functionality

Goals represent an actors strategic interest, which can be defined as the user requirements. In terms of the service definition, goals provides a description of the main underlying capability of the service. In the example shown in Figure 2, the goal "Get student details" describes the need to get student details. However on its own, this underlying capacity does not describe how this is fulfilled, nor does it specify the inputs or outputs when accessing the interface. The functionality is described by the plan, which is a description of what something does and how to achieve that goal. The plan provides parameters for a particular function in terms of behaviour and purpose. In essence, the capability of the service is described by the goal and the functionality is described by the plan. Thus by linking a function to a capability to describe how and what to do, we come closer to the definition of a service.

3.1.2 Interface

The plan describes the actions required for interacting with the service while the resource describes the information required for the interface to access the service capabilities. For example in Figure 2, the plan "Student fills in form" indicates that the interface has to have a functionality that captures the action of students filling in forms. The resource "Student Data" is also required in order to specify and store the data that was captured by the functionality of the service defined through the plan.

3.1.3 Service Description

Goals provide a conceptual part of the service description by describing what the service is supposed to accomplish and the conditions for using the service through capabilities. The service description also relies on the plan to define the semantics of interaction with the service, in addition to the resource in order to specify the attributes of the input and outputs. For example in Figure 2, the goal "Get student details" indicates that the service description will publish the fact that this service will get student details. The plan "Student fills in form" tells us the semantics of the interaction and the resource “Student data” describes the inputs and outputs of the service in the service description.

Having established a visual Goal-Plan-Resource pattern in the Secure Tropos diagram, we now proceed to incorporate cloud requirements into Secure Tropos.

4 Incorporating in Secure Tropos

We validate our proposed work through constructed examples based on existing systems. In our case study, we create a scenario based on migrating an university records management system to the cloud.

4.1 Extensions to the Meta-Model

We extend the Secure Tropos meta-model to include additional attributes to model cloud requirements. Figure 3 shows a portion of the meta-model illustrating our extensions.

Resource The resource object is extended to include fields specifying the category of the resource, type, specifications, region, owner and classification. These notions are based on requirement-level properties defined in both CloudML [16], a domain-specific modelling language that specifies the provisioning, deployment, and adaptation concerns of cloud systems at design-time [5] and Cloud Computing Ontology (CoCoOn) [17], an ontology-based system for describing cloud infrastructure. As an example consider the region property from CloudML and CoCoOn, which describes the geographical location of the hardware component and is used to determine security/privacy jurisdiction and legislation. An example of a property from CloudML that is not requirement-level is *PrivateIP*, as this is implementation-specific. The extended resource will be deployed in the physical layer when modelling cloud services, providing a fine-grained view of the infrastructure required to enact services defined in the software system layer. This also provides the foundation for performing security analysis on cloud-specific threats and vulnerabilities and the modelling of dataflow. **Service Definition** Goal, plan and resource are existing concepts in the Secure Tropos meta-model. We propose the identification of a service based on the Goal-Plan-Resource pattern and extend the meta-model as shown in Figure 3, where we indicate that a service is an aggregation of a single goal, one or many plans and one or many resources.

The migration link is indicated by the shaded box in Figure 3, which links one service to one or many cloud actors.

4.2 Organisational View

An example demonstrating the organisational view is illustrated in Figure 1, which shows the dependency relational link between different actors and the security constraints imposed based on goals. The security constraints are identified from the perspective of the dependent and dependee with regards to the goal. In this example, the *Lecturer* actor depends on the *University of Brighton* actor to satisfy the *Access student records* goal. The Lecturer has a security constraint that they should keep their account access secure while the *University of Brighton* actor has the security constraints of both keeping student records private AND secure.

4.3 Security Requirements View

The security requirements view in Figure 2 shows a wide range of elements that can be modelled in order to analyse the security requirements of a software system. The primary goal “Manage Student Records” has three sub-goals, in this example the “Get Student Details” sub-goal is examined in more detail. This sub-goal requires the resource “Student Data” and the plan “Student Fills in Form” in order to satisfy its requirements. It also has the security constraints of keeping personal details private and secure and is impacted by two threats; “Man-in-the-Middle” and “Eavesdropping”. Each of the sub-goals also define a service with its corresponding plan and resource links, as indicated by the bounding box and service labelling.

4.4 Defining and Migrating Services to the Cloud

Our proposed process allows the developer to identify and indicate a set of components that conceptually contribute towards a service. The constructed group is then linked via the migration relationship to a cloud service defined in another view, the cloud service view. In Figure 2, we have indicated that the goal “Get Student Details”, the plan “Student Fills in Form” and the resource “Student Data” contribute towards the definition of a service. Based on the goal description, the functionality of the service is to obtain details from students. The plan indicates that the service will include the capacity to obtain student data from forms that are filled in by the student, possibly through an form defined by the interface. The required input will be student data which the resource describes in full detail, including properties such the owner of the data, how the data is stored and the specifications of the data.

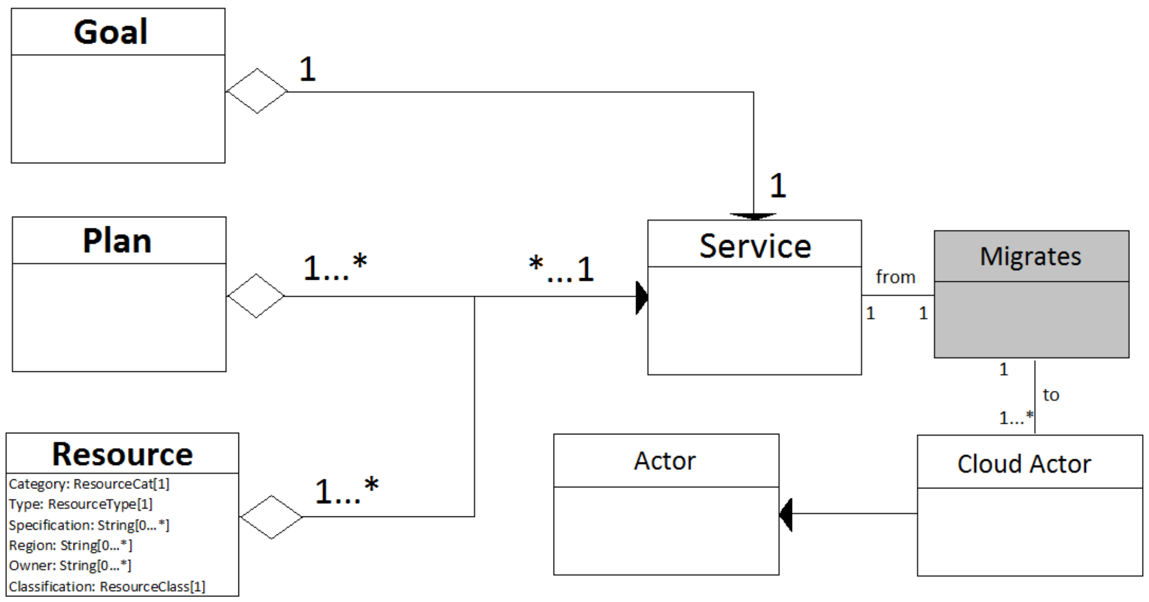

Figure 3. Extensions to the meta-model to identify services from goal, plans and resources, define the migration link from service to a cloud actor and add attributes to resource.

In order to model and analyse software systems for cloud environments, we need to create a model for describing cloud services and the components involved in the definition of these services. This would include the software applications deployed to address the problem or tasks that the service is trying to solve or achieve, specifications of the resources required to execute the software and identifying the data that will be processed to determine the flow of data. The resources can then be categorised further into network, compute and storage requirements, depending on the capabilities the service requires.

There are several possible scenarios where migrating to the cloud is applicable. In order to model cloud services, we propose a two-layer model; the software system and the physical layer. The **software system** describes the programmatic implementations of the functions offered by the services. The software system contains applications and services which provide solutions to the client requirements. Descriptions in each of the components within the software layer define a dependency link with required resources, which are provided in the physical layer. The **physical layer** contains resource elements that describe the storage, compute and network components required to satisfy the service requirements(Goal) based on the requirements of the stakeholder. Each one of the components in the physical layer describes an aspect of the infrastructure required to define a cloud computing system.

Each cloud service will include deployment models, service models and specifications for resources, based upon the user requirements and restrictions identified in the early stages of the requirements modelling. This process allows us to define the exact requirements when planning for resource provision, utilisation and optimisation.

5 Conclusion

We have discussed the current progress and challenges for modelling secure software system in this paper, emphasising the need for a modelling language that is able to holistically capture properties that define a cloud computing system based on client requirements. To address this gap, we define the properties and attributes required to model software services and cloud services. We define a pattern to group interdependent properties for the migration towards cloud services in a cloud computing environment, based on the strategic interests of stakeholders. These properties consist of the primary goal which defines what the main functionality of the service should be, plans that tells us what the capabilities of the service should be in order to fulfil the functionalities defined in the goal and the resources that are required by the service in order to perform its functions. We validate our proposed research through extensions to the security requirements modelling language Secure Tropos and provide a case study based on modelling the migration of an university records management system to a cloud computing environment.

In order to obtain a holistic view of security vulnerabilities and threats, we will build on the cloud service view to examine each component within the cloud service based on different cloud models. We can further extend the security attacks view to include cloud specific security vulnerabilities, threats and mechanisms to mitigate these attacks. The nature and approach of attacks changes dynamically according to a wide variety of parameters, such as the service model, deployment model and in scenarios involving deployment of services to multiple clouds or service providers.

References

[1] T. H. Depot, "The home depot reports findings in payment data breach investigation," 2014.

[2] A. Pavel, "Amazon.com server said to have been used in sony attack," May 2011.

[3] Cloud Security Alliance, "Security research alliance to promote network security," *Network Security*, vol. 1999, no. 2, pp. 3–4, 1999.

[4] A. Bergmayr, H. Brunelière, J. L. C. Izquierdo, J. Gorroñogoitia, G. Kousiouris, D. Kyriazis, P. Langer, A. Menychtas, L. Orue-Echevarria, C. Pezuela, and M. Wimmer, "Migrating legacy software to the cloud with ARTIST," *Proc. European Conference on Software Maintenance and Reengineering, CSMR*, pp. 465–468, 2013.

[5] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, "Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems," *Proc. IEEE Sixth International Conference on Cloud Computing*, pp. 887–894, 2013.

[6] S. Frey and W. Hasselbring, "The cloudmig approach: Model-based migration of software systems to cloud-optimized applications," *International Journal on Advances in Software*, vol. 4, no. 3 and 4, pp. 342–353, 2011.

[7] M. Armbrust, O. Fox, R. Griffith, A. D. Joseph, Y. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "Above the clouds: A Berkeley view of cloud computing," *University of California, Berkeley, Tech. Rep. UCB*, pp. 07–013, 2009.

[8] N. R. Jennings, "Agent-oriented software engineering," in *Multiple Approaches to Intelligent Systems*, pp. 4–10, Springer, 1999.

[9] E. Kamateri, N. Loutas, D. Zeginis, J. Ahtes, F. D'Andria, S. Bocconi, P. Gouvas, G. Ledakis, F. Ravagli, O. Lobunets, and K. a. Tarabanis, "Cloud4SOA: A semantic-interoperability paas solution for multi-cloud platform management and portability," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8135 LNCS, pp. 64–78, 2013.

[10] R. G. Cascella, C. Morin, P. Harsh, and Y. Jegou, "Contrail: A reliable and trustworthy cloud platform," in *Proc. 1st European Workshop on Dependable Cloud Computing*, p. 6, ACM, 2012.

[11] W. Madsen, *Trust in Cyberspace*, vol. 1999. 1999.

[12] P. Devanbu, S. Stubblebine, and S. S. Premkumar T. Devanbu, "Software engineering for security - a roadmap," *Icse*, pp. 227–239, 2000.

[13] H. Mouratidis and P. Giorgini, "Secure Tropos: A Security-Oriented Extension of the Tropos methodology," *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, pp. 285–309, Apr. 2007.

[14] A. Bandara, H. Shinpei, J. Jurjens, H. Kaiya, A. Kubo, R. Laney, H. Mouratidis, A. Nhlabatsi, B. Nuseibeh, Y. Tahara, T. Tun, H. Washizaki, N. Yoshioka, and Y. Yu, "Security patterns: comparing modeling approaches," 2010.

[15] C. M. MacKenzie, K. Laskey, F. McCabe, P. Brown, and R. Metz, "Reference Model for Service Oriented Architecture," *Oasis*, 2006.

[16] E. Brandtzæg, S. Mosser, and P. Mohagheghi, "Towards cloudml, a model-based approach to provision resources in the clouds," in *8th European Conference on Modelling Foundations and Applications (ECMFA)*, pp. 18–27, 2012.

[17] M. Zhang, R. Ranjan, A. Haller, D. Georgakopoulos, M. Menzel, and S. Nepal, "An ontology-based system for cloud infrastructure services' discovery," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, pp. 524–530, IEEE, 2012.