Automated Security Service Orchestration for the Identity Management in Web Service based Systems

Robert Warschofsky, Michael Menzel, Christoph Meinel
Hasso-Plattner-Institute
Prof.-Dr.-Helmert Str. 2-3
14482 Potsdam, Germany
{*robert.warschofsky, michael.menzel, meinel*}*@hpi.uni-potsdam.de*

***Abstract*—Today, there is a huge amount of security services that can be used to implement different security requirements in Web Service based systems. For example, identity management services are required for authentication and authorization whereas message logging services are necessary to achieve non-repudiation. However, the deployment and configuration of these security services usually requires expert knowledge about the systems and expert knowledge about security requirements and implementations which a person can only learn by experience. Furthermore, today's Web Service based systems become increasingly complex. Thus, implementing security requirements is a complex and error prone task, even for experts. For this paper, we analysed several service-based implementations for identity management and their differences in the service orchestration. We present an approach to derive the needed security services, their configuration, and their connections to the functional services, based on defined security requirements for a Web Service based system. Therefore, we evaluate the UML use case model of the system and apply service security pattern derived during the analysis of the identity management implementations.**

***Keywords*-Service-oriented Architectures; Web Services; Security Orchestration; Pattern-bases Security Engineering; Identity Management**

I. Introduction

Today, security in Service-oriented Architectures (SOA) is a necessary but complicated field. In the scope Web Service based SOA, several possibilities exists to secure the services, the clients, and the messages exchanged between both. Basic message security requirements, like confidentiality and integrity, can be fulfilled using methods implemented at the sender and the receiver of a message. For example, the confidentiality of a message can be assured using XML encryption and for the integrity of a message, XML signature can be used. Other security requirements, like authentication and authorisation, can be basically fulfilled using sender and receiver based implementation, too. For example, the usage of username/password tokens send within a message provides sufficient information for both requirements. However, in a SOA the usage of security services might be a more adequate solution for such requirements. For other security requirements, the usage of dedicated security services is inevitable. For example, *identity provisioning* and *single sign-on* can not be realized without a central service responsible for the identity management.

For Web Services several security services are possible and available. A Security Token Service (STS) issues identity information about its users. This information can be used for the identification and authorization of a service requester. Other security services may be used to store and distribute public keys for the encryption of messages or for the collection and safekeeping of messages to ensure the non-repudiation of certain service requests.

There exist several possibilities to actually combine security methods and security services into a Web Service based SOA. For example, an STS might be used to enable a single sign-on to several services and the actual requests to the services have to be secured using XML encryption and XML signature. The selection of a reasonable combination of security services and security methods for a certain SOA based application can be complex and error prone. In addition, even after the selection of the services and methods to be used, the orchestration of the application services and the security services has to be done in a correct manner.

To support the security design of Web Service based systems, we propose to define patterns (in terms of Gamma et al.[1]) for the orchestration of security services. Such security service orchestration patterns have to provide suggestions about the usage of distinguished security services in a certain Web Service based application to solve a certain security requirement.

We examined typical realizations of different Identity Management Models and the orchestration of the used security services. In this paper the discovered orchestration patterns for the selection and implementation of these identity management models are presented. We decided to formulate the preconditions and postconditions of the patterns in a way which allows an automated pattern selection and application. This should enable a system designer to easily create a security design model of a desired system. The preconditions of the patterns are described using an extended UML use case model. The pattern solutions describe the usage of security services using a system design model. This system design model is expressed in the *Fundamental Modelling*

Concepts (FMC [2]) language, enhanced with the security configuration model *SecureSOA* [3].

The paper is structured as follows: Section II describes the extended UML use case model as well as the FMC based system design model with SecureSOA. In Section III the examined Identity Management Models and their relationship to each other are introduced. Section IV explains the discovered patterns in detail and puts them in relation to the Identity Management models. Finally, in Section V relates work is described and in Section VI the contribution of this paper is summarized and open research questions are discussed.

II. Security enhanced design models

For our approach on security orchestration patterns, we are using two different model types. The first model type is used for the description of a Web Service based system on a conceptional use case level. The second model type represents the system design after the application of the orchestration patterns. As mentioned above, the models have to allow an automated pattern selection and application. Therefore, an automated interpretation of these models is required, which would be impossible if the patterns would only be described in natural language.

A. An extended UML use case model for preconditions

The model used to describe the preconditions of the orchestrations patterns, is basically an UML use case model [4], consisting of use cases, actors, and associations between them. Use cases can also include other use cases which is notated with the stereotype ≪*include*≫. Although, use case models provide additional concepts and stereotypes, we are not using them in our approach.

An example for the extended UML use case model is shown in Figure 1. In this example, an ≪*End User*≫ is associated to a use case which includes another use case. This second use case includes a third use case. In addition, the second and the third use case are associated to different ≪*Provider*≫. Finally, a contract is defined between these ≪*Provider*≫.

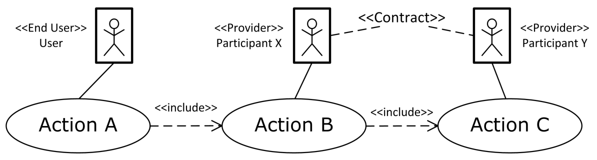


Figure 1. Example for the extended UML use case model.

To adopt the use case models to our requirements, we extend them with additional stereotypes. The stereotype ≪*Provider*≫ refines an actor. A ≪*Provider*≫ is responsible for the provisioning of a use case that represents a certain service. Another refinement of an actor is the ≪*End User*≫ that represents an actual user of a use case. In addition, we added the dependency ≪*Contract*≫ to the use case model. This dependency has to be associated to two ≪*Provider*≫ and represents a contract about the federation of identity information between these two service providers. Finally, a set of supplemental requirements can be stated for and associated with a use case in our extended use case model.

B. A security enhanced system design model

FMC Compositional Structure Diagrams (also known as FMC Block Diagrams) depict the static structure of a system and the relationships between system components. This diagram type distinguishes between active and passive components. Agents are active system components that are capable to communicate via channels and to perform activities in the system. Channels and storages are passive components used to transmit or store information.

These FMC block diagrams are the foundation for our security enhanced system design model. The elements in these diagrams are annotated with security requirements that are defined by our security modelling language SecureSOA. SecureSOA uses the integration schema defined by SecureUML [5] to enable the enhancement of system design models with security-related modelling elements.

One security requirement defined by SecureSOA is a directed *Trust relation* between two actors. The meaning of such a relation is that one actor can identify and has confidence in the other actor. An extension of this is the *Trust domain* which implies that there exists an *Trust relation* between all actors in that *Trust domain*. Another security requirement is a *Contract* between two actors. In this context, a *Contract* is a bidirectional *Trust relation*.

III. Identity Management

The identification and authentication of users are important security requirements to ensure a trustworthy communication in decentralised systems and provide the foundation to restrict the access to services. The identification, authentication and authorisation of users is performed on their representation in the digital world - a user's digital identity. A digital identity consist of a set of attributes and is managed in an account.

The *Identity Management* describes the process of establishing, representing, maintaining and provisioning a person's identity as digital identities in IT systems as shown in Figure 2.

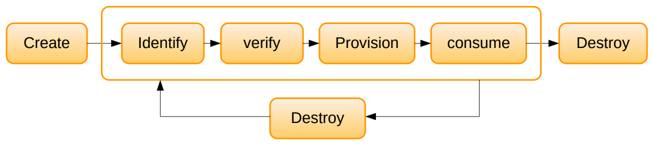
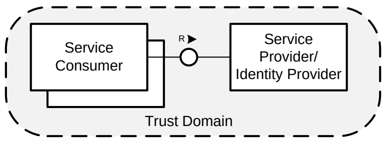


Figure 2. Life cycle of a digital identity

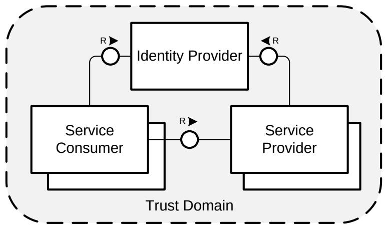
The first step in this process represents the registration of users. In order to create a user's digital identity, identity information is stored in an account that is created for the user. The usage of a digital identity at a service provider is based on four steps. A user must be identified and authenticated to verify that a specific digital identity belongs to this user. Required identity information must be provided to a service provider that results in the consumption of this information. At the end of the life cycle, a digital identity is destroyed by deleting the users account.

The life cycle of a digital identity management identifies basic steps, but do not describes the architecture and components to perform this process. These components and the underlying concepts are described by the identity management models. We distinguish four models, while each model implements a specific identity management approach. As described in [6] and [7], an identity management model can be based on an domain-based approach or on an open environment approaches.

Domain-based approaches represent traditional approach that bind a digital identity to a specific security domain (e.g. a company). An identity information can be consumed in this domain solely. Identity management models implementing a domain-based approach are illustrated in Figure 3



(a) Isolated Identity Management



(b) Centralised Identity Management

Figure 3. Domain-based Identity Management Models

The *isolated identity management* model (shown in Figure 3(a)) implements a service specific management of digital identities. A service is attached to a user directory that provides the identity information for this service and enables the authentication of users. Each service that is based on this model has full control over the users. However, the users have to be registered at each service independently. Since the identification and representation of users is service-specific, the usage of this model prevents the orchestration and composition of independent services.

To enable a single-sign-on across multiple services in a domain, the *centralised identity management* model can be used as illustrated in Figure 3(b). The services are connected to a single identity provider to organise the identity management in a centralised way. The identity provider is responsible to manage the digital identities of users and to perform their authentication. Authentication decisions can be brokered to services that rely on this information. In addition, identity information required by the services can be provided as well. The usage of an identity provider requires trust relationships between the relying services and this identity provider. Since the identity provider is used in a single trust domain, these relationships are established by default.

Although domain-based identity management approaches enable service providers and organisations to control the identity information of their users, these approaches prevent the usage of identities across domain boundaries. Users have to be registered in each domain and this results in an increasing number of digital identities and accounts. For each account, users have to manage credentials that facilitate their authentication.

For example, consider the usage of web applications in the internet. Users have to manage a multitude of user name and password combinations and tend to select the same password for each account. Therefore, the application of domain-based identity management approach results in an increasing number of security risks. Furthermore, users have to keep their account consistent and identity information have to be updated in multiple accounts.

Identity management models based on *Open environment models* address these issues by enabling the usage of digital identities across trust domains. This approach is based on a set of identity providers that share and broker identity information. Since identity provider can be implemented on the basis of different technologies and protocols, an abstraction layer is required to enable the interoperable exchange of identity information.

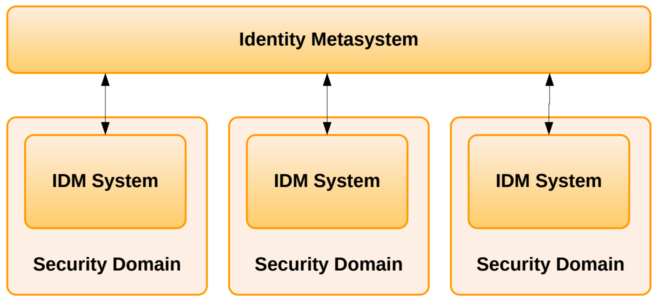
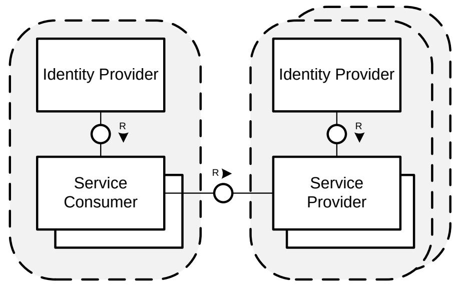


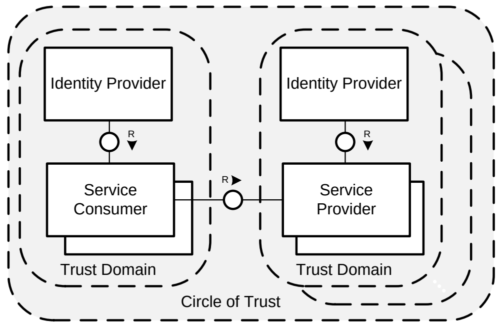
Figure 4. Identity Metasystem

The *identity metasystem* provides such an abstraction layer as illustrated in Figure 4. Standards such as WS-Trust and SAML provide interfaces and token formats and enable the usage. The identity metasystem enables the integrations of any identity management solutions to avoid the replacement of existing solutions.

The identity management models based on the open environment approach are shown in Figure 5. Both models are based on the identity metasystem to integrate multiple identity providers. Service providers and identity providers rely on identity information and authentication decisions that is brokered by other identity providers. The brokering of this information across trust domains enables a single-sign-on across organisational borders. The brokerage of identity information is based on established trust relations. Both models differ from each other in the establishment of these trust relations.



(a) User-Centric Identity Management



(b) Federated Identity Management

Figure 5. Identity Management Models

The *Federated Identity Management* model illustrated in Figure 5(b) is based on a circle of trust that is shared by multiple organisations. Identity providers rely on assertions issued by other identity providers in the federation. Contracts are concluded between the participating organisations to establish a federation.

Whereas, the *User-centric Identity Management* model is not based on predefined trust relations. The user is in the centre of the interactions between identity and service provider. The user can select an identity provider that asserts the authentication and required identity information. The identity provider issues a security token that can be used to access a service. The service provider has to decide, whether a token can be accepted from this source.

IV. Security Orchestration Pattern

The decision which Identity Management Model should be used in a SOA-based application can be made on the basis of the properties of these models. It certainly does make sense, to use the model that fits best to the application and its environment. Finding the best fitting Identity Management Model can be supported using orchestration patterns that describe how to orchestrate security services in a SOA for a certain security goal. The patterns presented here are the results of the analysis of typical Identity Management Model implementations and represent best practises.

These patterns can be applied to an extended Use Case model (see Section II-A) of the desired SOA-based application and will produce a security enhanced system design model expressed in FMC and SecureSOA (see Section II-B). This system design model can further be transformed into concrete service implementations and security configurations, as described in [8].

In this section the patterns are described in detail. First, the general structure of the patterns is explained (Section IV-A). In the Sections IV-B to IV-E, the patterns are explained in relation to the Identity Management Models presented in Section III. Finally, Section IV-F illustrates the exemplary application of multiple patterns.

A. General structure of a pattern

As described in [9], the mandatory elements of a pattern in general are *Name*, *Context*, *Forces*, *Problem*, and *Solution*. These elements are shortly explained in Table I. Since the Security Orchestration Patterns described in this paper are applied to extended Use Case models, the *Context* of these patterns is always a Use Case model. In addition, all presented patterns cover the selection of an appropriate Identity Management model and therefore, the *Problem* of these patterns is always *Identity Management*.

Name	is a label that identifies the pattern and reflects the intention of the pattern.
Context	describes the environment before the application of this pattern.
Forces	are conditions that exists within the context. They affect the problem and might represent trade-offs or preconditions.
Problem	describes a problem that occurs within the context.
Solution	is a proven solution for the problem within the context.

Table I
MANDATORY ELEMENTS OF PATTERNS (TAKEN FROM [9]).

For the application of all pattern presented in this paper, the use case model of the desired system is analysed and an corresponding system design model is created. Therefore, for each use case a service is created and for each ≪*include*≫ dependency a service call is realized. In addition, for each ≪*Provider*≫ a trust domain is created that contains all services created from the use cases associated with this ≪*Provider*≫.

B. Isolated Identity Management

The **context** of the *Isolated Identity Management* pattern (*IIM*) consists of only two actors and two use cases. The

forces for this pattern define that one of the actors is an ≪*End User*≫ and the use case associated with this ≪*End User*≫ includes the other use case (Figure 6(a)). Since the user only requests one service of one provider, the service itself can be enabled to authenticate all its users.

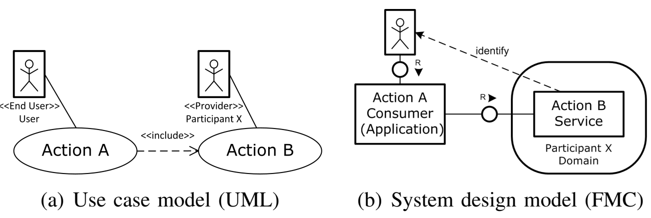


Figure 6. *Isolated Identity Management* pattern (*IIM*).

The **solution** is that for the action associated to the ≪*End User*≫ a consumer application is created in the system design model and for the included action an corresponding service is created which the consumer has to request. Since the service is responsible for the identity management of all its requesters, an identify relation is added between the service and the application user (Figure 6(b)). For example, the requester can use a username/password combination to identify itself.

C. Centralized Identity Management

A system design model with a *Centralized Identity Management* is created by the *Service Refinement* pattern (*SR*). This pattern describes the usage of composed services where all services are offered by the same provider. The **context** of this pattern contains at least three use cases and one actor. The **forces** define, that one of these use cases includes the two other use cases and all three use cases are associated to the same actor. This actor has to be a ≪*Provider*≫ which represents a company and not an ≪*End User*≫ (Figure 7(a)).

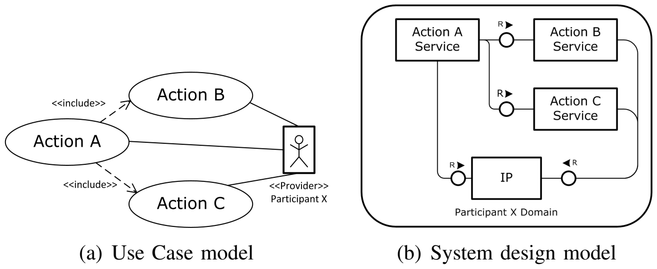


Figure 7. *Service Refinement* pattern (*SR*).

The **solution** is that for each of these use cases a corresponding service is created in the system design model. The service created for the including use case has to request the other two service. Since all use cases are associated to the same provider, all created services are contained in the same trust domain. The identity management for all services is handled by an Identity Provider residing in this trust domain to avoid duplicate user data in each service.

D. User-centric Identity Management

For the *User-centric Identity Management Model* we present two patterns. The *Inter-Domain Service Call* pattern (*IDS*) defines a *User-centric Identity Management* solution. This pattern describes the usage of composed services where the services are offered by different providers. The **context** consists of at least two use cases and two actors. The **forces** define, that both use cases are associated to different actors and one use case includes the other use case. In addition, the including use case is associated to at least one ≪*Provider*≫ (Figure 8(a)), which differences the forces of this pattern and the forces of the of the *Isolated Identity Management* pattern.

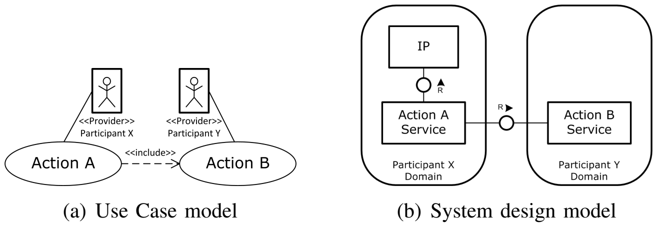
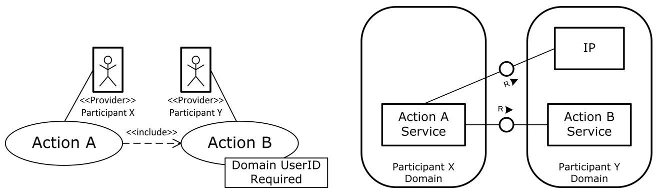


Figure 8. *Inter-Domain Service Call* pattern (*IDS*).

The **solution** of this pattern describes the creation of two services corresponding to the two use cases. Since both use cases are associated to different ≪*Provider*≫, the created services are in different trust domains. The identity management is handled by an identity provider in the trust domain of the requesting service. The identity provider is created due to the fact, that the actor associated to the including use case represents a company which should use some kind of central identity management. As mentioned in the description of the *User-centric Identity Management Model*, the requester can select its identity provider and the requested service has to decide whether to accept the information from this provider.

The *Identified Inter-Domain Service Call* pattern (*IDS2*) is similar to the *Inter-Domain Service Call* pattern. The **forces** of this pattern define an additional requirement for the included use case. This use case, when included, requires the provision of identity information originated from the domain of the use case (Figure 9(a)). This requirement is satisfied with a **solution** as shown in Figure 9(b). For the use cases corresponding services in different trust domains are created and in the domain of the called service an identity provider is created. This identity provider is requested by the calling service to obtain a valid identification in the domain of the called service. The identification is then used to call the service.

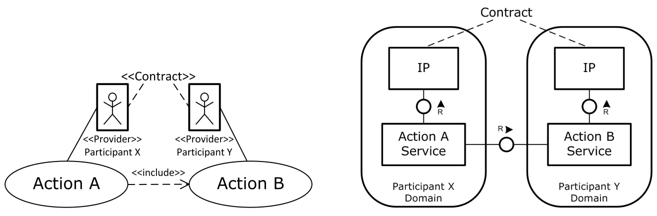


(a) Use Case model (b) System design model

Figure 9. *Identified Inter-Domain Service Call* pattern (*IDS2*).

E. Federated Identity Management

A system design model with a *Federated Identity Management* solution is created using the *Federated Inter-Domain Service Call* pattern (*FIDS*). This pattern describes the federated usage of identity information and service provisioning between two companies. The **context** contains at least two use cases and two actors. The **forces** describe that the two use cases are associated with different ≪*Provider*≫ and that between these two ≪*Provider*≫ a ≪*Contract*≫ is defined (Figure 10(a)).

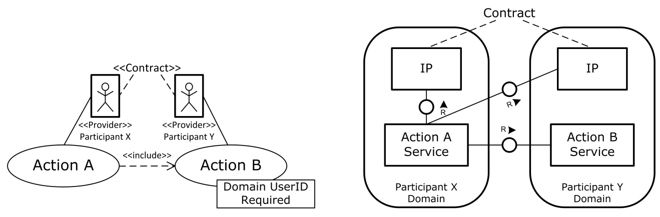


(a) Use Case model (b) System design model

Figure 10. *Federated Inter-Domain Service Call* pattern (*FIDS*).

The **solution** describes that two services are created corresponding to the two use cases. These services are contained in different trust domains and in each trust domain, there is an responsible identity provider. In addition, there is a contract defined between both identity providers. The services communicate only with the identity provider in their own trust domain and with each other. To transform the identity of the calling service from the first domain to the identity in the second domain, the service of the second domain has to request its identity provider.

The *Identified Federated Inter-Domain Service Call* pattern (*FIDS2*) is comparable to the The *Federated Inter-Domain Service Call* pattern. Similar to the *Identified Inter-Domain Service Call* pattern, the **forces** of this pattern define the additional requirement for the included use case that identity information originating from the domain of the use case provider are needed (Figure 11(a)). The **solution** shown in Figure 11(b) demonstrates the fulfilment of this additional requirement. Corresponding to the two use cases, two services are created that are contained in different trust domains and in each trust domain, there is an responsible identity provider. In addition, there is a contract defined



(a) Use Case model (b) System design model

Figure 11. *Identified Federated Inter-Domain Service Call* pattern (*FIDS2*).

between both identity providers. In contrast to the *FIDS* pattern, the requesting service communicates with both identity providers to first receives the identity information from its own identity provider and then translates the identity information for the requested service using the other identity provider.

F. Composed Example

As we have shown, for some Identity Management Models there exist multiple security orchestration patterns. The usage of additional requirements for one use case can result in the selection of a different pattern with a different solution. Nevertheless, this selection can be done in an automated way, even for complex use case models.

As an example for the application of multiple patterns, a travel agency scenario is illustrated in Figure 12. In this scenario, an ≪*End User*≫ is associated to the use case *Booking Travel*. This use case includes the use case *Select Travel* associated with the *Travel Agency* and the use case *Pay Travel* associated with the *Bank*. The use case *Select Travel* includes the use cases *Select Flight* associated to a *Flight Agency*, *Select Hotel* associated to a *Hotel Agency*, and *Select Car* associated to a *Car Agency*. In addition, the use cases *Pay Travel* and *Select Flight* have the requirement to be provided with identity information originated from their own domains to be included successfully. Finally, there are contracts defined between the *Travel Agency* and the *Flight Agency*, the *Hotel Agency*, and the *Car Agency*.

The general structure of this use case model can be used to create a general system design model, as shown in Figure 13. For each use case a corresponding service is created and for the ≪*include*≫ relations between the uses cases corresponding service calls are added. Finally, the security domains of the ≪*Provider*≫ actors are added and the corresponding services are attached to these domains. Using the use case model and the created system design model, the security orchestration patterns, presented in this paper, can be applied.

The application of the orchestration patterns adds the identity management components to the system design model. For example, the application of the *Federated Inter-Domain Service Call* pattern creates the *Identity Provider* in the domain of the *Travel Agency*, the *Flight Agency*, the *Hotel*

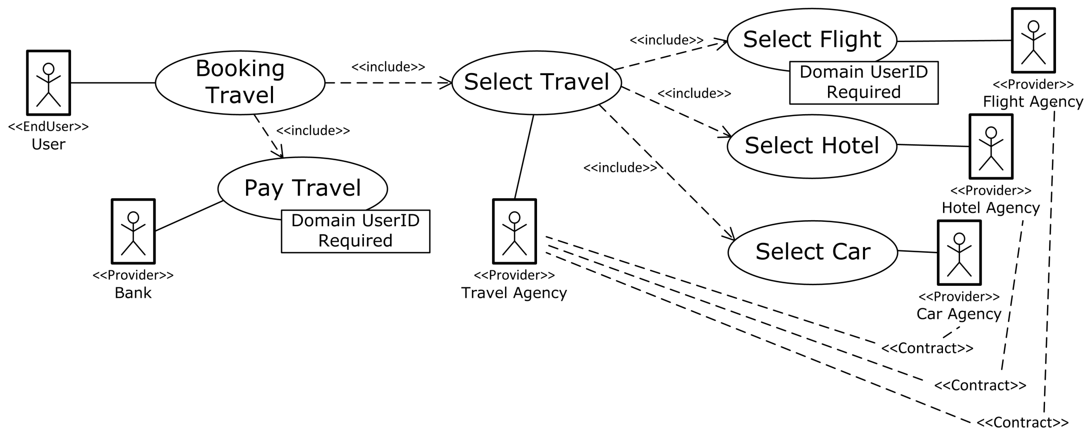


Figure 12. Travel agency use case model.

Agency, and the *Car Agency*. This pattern also creates the *Contract* relations between these *Identity Providers* and the requests between the services and the *Identity Provider* in one trust domain. The *Identified Inter-Domain Service Call* pattern creates the *Identity Pattern* in the domain of the *Bank* and adds the service calls between the *Booking Travel Application*, the *Pay Travel Service*, and the *Identity Provider* in the *Bank* domain.

V. Related Works

There already exist some approaches concerning Identity Management and service security pattern.

S. Rieger [6] provides approaches for the realization of user-centric and federated Identity Management. However, an automated selection of the best fitting Identity Management Model for an application is not defined.

A. Jøsang et al.[7] defines different Identity Management Models and additional trust requirements for each of these models. The support for an automation of the model selection is also not provided.

N. Delessy et al.[10] proposed a pattern language for identity management. These patterns are designed to be used in the software development cycle. However, they define the patterns using natural language which makes it impossible to use these patterns in an automated way.

VI. Conclusion

In this paper, we presented a approach to describe security orchestration patterns for Identity Management Models in a way that they can be selected and applied automatically. The forces of these patterns are described using extended UML use case models. Using these models, it is possible to describe the functional purpose of a desired service based system. The application of the the orchestration pattern creates a system design model that contains the required functional services as well as all necessary security services used to ensure certain identity management requirements. The created system design models are expressed using the modelling language FMC and the security modelling extension SecureSOA. One advantage of SecureSOA is the capability of an automated generation of security configurations for all services in the system design mode [8].

To be able to create a completely secured system design model, further service security orchestration patterns have to be defined for different security topics. One such security topic is the provisioning and distribution of public keys for the encryption of messages between services. A security service implementing the Needham-Schroeder public-key protocol [11] would be an appropriate candidate to realize a solution for this topic. Another security topic concerns the realization of non-repudiation. For this subject, pattern could be useful which foster the application of a message logging service provided by a trusted third party. To summarize, the application of service security patterns simplifies the design of secure service-based system by providing service security solutions.

References

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[2] A. Knöpfel, B. Gröne, and P. Tabeling, *Fundamental Modeling Concepts*. John Wiley & Sons, Mar. 2006.

[3] M. Menzel and C. Meinel, “SecureSOA - Modelling Security Requirements for Service-oriented Architectures,” in *IEEE International Conference on Services Computing (SCC 2010)*, Jul. 2010, pp. 146–153.

[4] OMG, “The UML 2.0 Superstructure Specification,” Object Management Group, Specification Version 2, 2004. [Online]. Available: http://www.omg.org/docs/formal/07-11-04.pdf

[5] D. Basin, J. Doser, and T. Lodderstedt, “Model Driven Security: from UML Models to Access Control Infrastructures,” *ACM Transactions on Software Engineering and Methodology*, vol. 15, no. 1, pp. 39–91, January 2006.

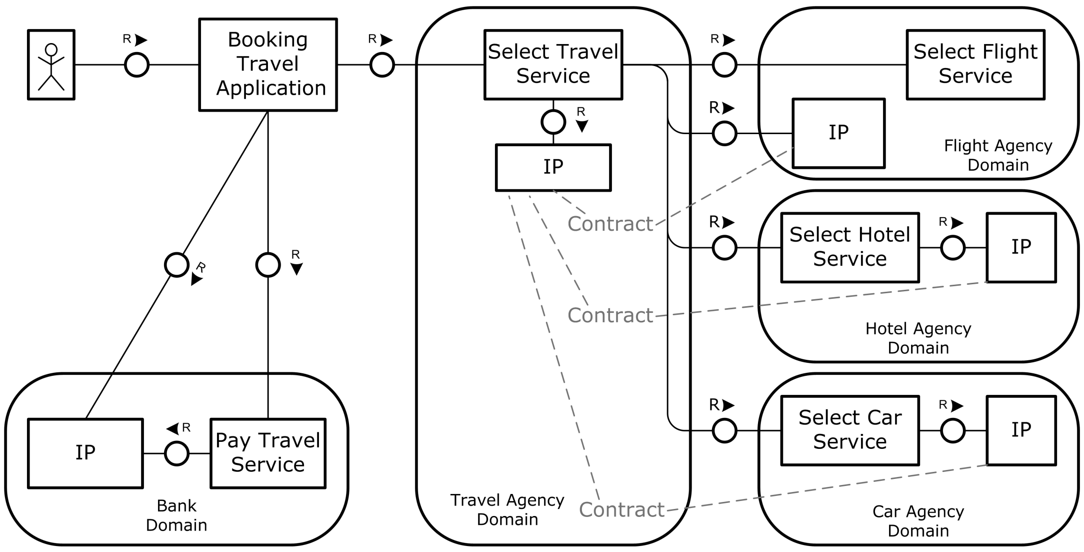


Figure 13. Travel agency system design model using FMC and SecureSOA.

[6] S. Rieger, “User-Centric Identity Management in Heterogeneous Federations,” in *ICIW’09: International Conference on Internet and Web Applications and Services*. IEEE Computer Society, 2009, pp. 527–532.

[7] A. Jøsang, J. Fabre, B. Hay, J. Dalziel, and S. Pope, “Trust Requirements in Identity Management,” in *ACSW Frontiers 05: Proceedings of the 2005 Australasian workshop on Grid computing and e-research*. Australian Computer Society, Inc., 2005, pp. 99–108.

[8] M. Menzel, R. Warschofsky, and C. Meinel, “A Pattern-driven Generation of Security Policies for Service-oriented Architectures,” in *ICWS ’10: IEEE International Conference on Web Services*. IEEE Computer Society, Jul. 2010, pp. 243–250.

[9] G. Meszaros and J. Doble, *A pattern language for pattern writing*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997, pp. 529–574.

[10] N. A. Delessy, E. B. Fernandez, and M. M. Larrondo-Petrie, “A Pattern Language for Identity Management,” in *ICCGI ’07: Proceedings of the International Multi-Conference on Computing in the Global Information Technology*. Washington, DC, USA: IEEE Computer Society, 2007, p. 31.

[11] R. M. Needham and M. D. Schroeder, “Using encryption for authentication in large networks of computers,” *Communications of the ACM*, vol. 21, pp. 993–999, December 1978.