Web Services Security Architectures Composition and Contract Design using RBAC

D.Shravani[1], Dr.P.Suresh Varma[2], Dr.B.Padmaja Rani[3], Dr.D.Sravan Kumar[4], M.Upendra Kumar[5]
[1]Research Scholar, Rayalaseema University, Kurnool, A.P., India
[2]Principal and Professor, Department of Computer Science, Adikavi Nannaya University, Rajahmundry, A.P., India
[3]Associate Professor, Department of CSE, JNTU College of Engineering, Hyderabad, A.P., India
[4]Principal and Professor of CSE, KITE Women's College of Professional Engineering Sciences, Hyderabad, A.P., India
[5]Associate Professor, Department of CSE, MGIT, Hyderabad, A.P., India

Abstract— Service Oriented Architecture's Web Services authorization traditionally is done using common access control models like Role-Based Access Control. In thinking of a composite application that stitches together the capabilities of multiple services, any action in the composite app should ideally check the access control rules of all constituent services before initiating an action. The Web Services Access controls are categorized according to access control granularity and have two approaches: The first approach supports a negotiation-based attribute-based access control to Web Services with fine access granularity. The second approach is tailored to access control for conversation-based Web services and composite services; where in a Web Service is not considered as a set of independent operations and therefore access control must take such dependencies into account. During a Web Services invocation, a client interacts with the service, performing a sequence of operations in a particular order called conversation. In this paper, we want to propose strategies for analyzing and managing Role Based Access Control policies for designing Security Architectures for web services. We validate role-based access control with a case study, where in access decisions are based on the roles that individual users have as part of an organization. Users take on assigned roles. The process of defining roles should be based on a thorough analysis of how an organization operates and should include input from a wide spectrum of users in an organization. Access rights are grouped by role name, and the use of resources is restricted to individuals authorized to assume the associated role. For example, within a hospital system the role of doctor can include operations to perform diagnosis, prescribe medication, and order laboratory tests; and the role of researcher can be limited to gathering anonymous clinical information for studies. The use of roles to control access can be an effective means for developing and enforcing enterprise-specific security policies, and for streamlining the security management process. Under the RBAC framework, users are granted membership into roles based on their competencies and responsibilities in the organization. The operations that a user is permitted to perform are based on the user's role.

Keywords-Web Services, Security Architectures, Role-Based Access Control

I. Introduction to Web Services Security Architectures

Service Oriented Architectures Web Services authorization strategies traditionally were implemented using Role-Based Access Control. (RBAC). Authorization specifically here means, once a user is authenticated with or without the help of a directory server, an application needs to determine whether the identified user is authorized to access the functionality she is requesting. Authorization is also commonly refered to as access control. The decision to grant access may depend on multiple criteria, such as the action that is being requested, the resource on which the action is being requested, and the groups to which the authenticated user belongs or the roles that the user plays. For example, the superuser or the administrator may access all the files in a system, but a user belonging to the HR group can access only those files that are allowed for that group. In traditional, RBAC, permissions for each action on a resource are granted to one or more role. For example, in an e-learning application, a teacher role is required to grade a test. Information on what roles are granted to which users can be maintained in an LDAP directory. In the context of SOA, thinking of a composite application that stitches together the capabilities of multiple services. An action in the composite app should ideally check the access control rules of all constituent services before initialing an action. But this is only possible if the access control rules of each constituent service are also available to the composite application.. This is not possible, in general; traditionally, access control rules are built into each application in an opaque way. There is another reason why we cannot hard-core a specific access control strategy into each application. The reusability of the service may be drastically reduced for use cases that require a different access control model. Any authorization strategy for SOA Web Services will have to address these issues.

The World Wide Web Consortium (W3C) defines a Web Service as," A Software System identified by a URI whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition using XML-based messages conveyed by Internet Protocols". The most basic standard in providing security for Web Services is WS-Security (WS-Sec). Other important standards include: WS-Security Policy, Security Assertions Markup Language (SAML) and WS-Trust. WS-Secure

Conversation is a standard that enables two principals to establish a session context. The state of a session includes a secret key that they share for the duration of a session. Setting up such a session is especially efficient if two parties need to exchange several messages during a session. This reduces the overhead of creating a shared secret for every message exchanged. The idea of a session context is the application layer analogue of the session in SSL or the IPSec Security Association. WS-Trust builds on WS-Sec. Similarly, WS-Federation builds on WS-Trust. It helps broker trust between entities in different security domains. Many of the standards developed for Web Services Security are complementary to each other. A single application may use several standards to realize its security goals. At the same time, different developers may use a different subset of standards to achieve their security goals for the same application. [1]. Refer to Figure 1, which depicts the Web Services Security Architecture as defined by NIST draft.

The Web Services Access controls are categorized according to access control granularity and have two approaches: The first approach supports a negotiation-based attribute-based access control to Web Services with fine access granularity. The second approach is tailored to access control for conversation-based Web services and composite services; where in a Web Service is not considered as a set of independent operations and therefore access control must take such dependencies into account. During a Web Services invocation, a client interacts with the service, performing a sequence of operations in a particular order called conversation.

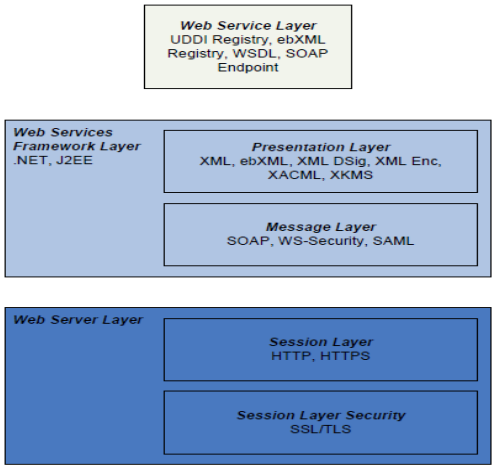

Figure 1. Web Services Security Architecture defined by NIST

A.Security Architectures for Software Security Engineering

Software Engineering covers the definition of processes, techniques and models suitable for its environment to guarantee quality of results. An important design artifact in any software development project is the Software Architecture. Software Architecture's important part is the set of architectural design rules. A primary goal of the architecture is to capture the architecture design decisions. An important part of these design decisions consists of architectural design rules.

In an MDA (Model-Driven Architecture) context, the design of the system architecture is captured in the models of the system. MDA is known to be layered approach for modeling the architectural design rules and uses design patterns to improve the quality of software system.

And to include the security to the software system, security patterns are introduced that offer security at the architectural level. Moreover, agile software development methods are used to build secure systems. There are different methods defined in agile development as extreme programming (XP), scrum, feature driven development (FDD), test driven development (TDD), etc.

Agile processing is includes the phases as agile analysis, agile design and agile testing. These phases are defined in layers of MDA to provide security at the modeling level which ensures that "security at the system architecture stage will improve the requirements for that system".

II. Model-Driven Architectures for Web Services Security Architectures

MDA (Model-Driven Architecture) is an approach to modeling that enables interoperability. MDA preserves the OMG's (Object Management Group) focus on integration and interoperability. MDA is a standard that focuses on the system or point solution and how it can be made to interoperate effectively. To enable this interoperability, it focuses on one key idea "the separation of the specification of functionality from the specification of the implementation of that functionality". With MDA, the models are the key artifacts of the system. Models- and especially models captured in the UML- are formal representations of different aspects of the system being constructed. MDA encourages the use of formal models because they are "machine processable", that is, they can be manipulated by tools because their meaning is clear and unambiguous, at the level of abstraction presented.

Key Concepts of MDA: MDA includes a number of fundamental concepts. Together, they provide a coherent view of OMG's view of how to create and maintain interoperable enterprise architectures.

Model: The OMG defines the term model to be "a description or specification of that system and its environment for some purpose". The salient aspects of a model are the following: A model is a simplification of the thing that is models. A model has a purpose or intent. The model focuses on the aspects relevant to its purpose

That is, it is common to have different models of the same thing for different purposes. MDA uses this concept to implement a well-established idea- the separation of the essential aspects of the thing from how those aspects are implemented. Refer to Figure 2 which provides class diagram for MDAauthentication using Executable UML.

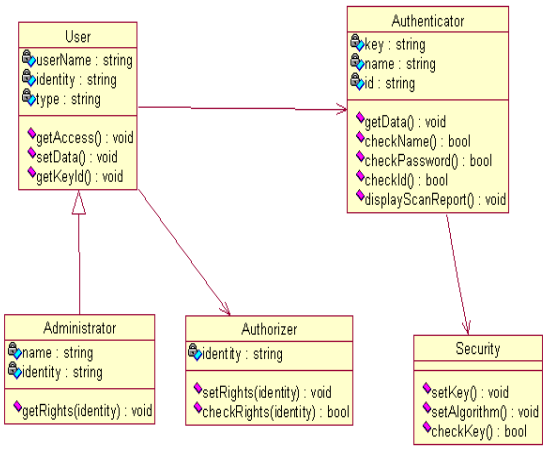

Figure 2. Class Diagram for MDA Authentication using Executable UML

III. Agile Modeling for Web Services Security Architectutes

Software design is a process that defines the solution to software implementation at the early stages of the software development process. It comprises software requirements and software architecture modeling. A modern concept of design modeling is performed through the design patterns. Agile Methodologies (AMs) focus on incremental development without a single and large upfront design. Namely, they adopt the Big Design Upfront Anti-pattern (BDUFA) that embraces changes adopting envisioning modeling of design (requirements and architecture) just when needed. AMs use design patterns when the language of programming is object oriented. AMs share the common principle of reusing working objects from previous projects or project iterations to avoid waste and useless activities.

A design practice peculiar to development with the AMs concerns testing. AMs extensively use the concept of *acceptance test*. Some of the design approaches in Agile Methods are:

Extreme programming (XP): The initial light design is defined with the *System Metaphor*. The System Metaphor is a short description of the system describing how it works. The design is implemented with the use of models like user stories, acceptance tests, and CRC cards. These models are refined in each iteration. As the work is incremental the additions of new system components or user's requirements are also modeled iteratively.

SCRUM: SCRUM is a management, enhancement and maintenance methodology for an existing systems or prototypes. SCRUM is more oriented to the orchestration, SCRUM uses backlogs to organize and design the work. Backlogs are dynamically and iteratively filled by the different stakeholders of the development. The priority is based on customer's needs and team ability.

Test Driven Development (TDD): In TDD the stress is on testing. It includes design patterns like the Positive Feedback loop, for which tests needed to be isolated, to be returned soon starting from the assert, to use realistic data, to relate input with output. In TDD the mock objects are used in testing.

Feature Driven Development (FDD): FDD consists of six phases as, *Requirements analysis, Develop an overall model, Build a feature list, Plan by feature, Design by feature, Build by feature.* According to FDD, developers capture requirements with use cases, in each use case actors are potential security subjects. Then an overall model is derived from the use cases. Then designers specify the abuse case scenario to include countermeasures to prevent the abuse case from occurring. Then designers put forth a development plan that guides the order of the features to be developed. Next the attributes are added to the classes. At last, the developers code and test the information system or software under development.

To arm agile methodologies with security features, to restrain reduction of agility nature a method has been proposed consists of first, security activities are extracted from existing processes and guidelines, and then agility degree of activities is defines to measure their nimbleness, integration issues of agile and security activities are handled and an algorithm to integrate security activities with organization's agile process is introduced, finally agility reduction tolerance (ART) parameter and its optimum value are discussed. Refer to Figure 3 which provides class diagram for Agile Modeling with Security activities.

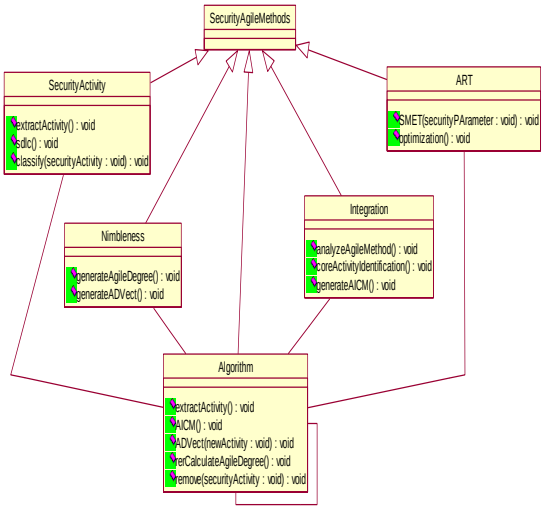

Figure 3. Class Diagram for Agile Modeling with Security Activities

Refer to Figure 4 which provides sequence diagram for Agile Modeling predecessor activities.

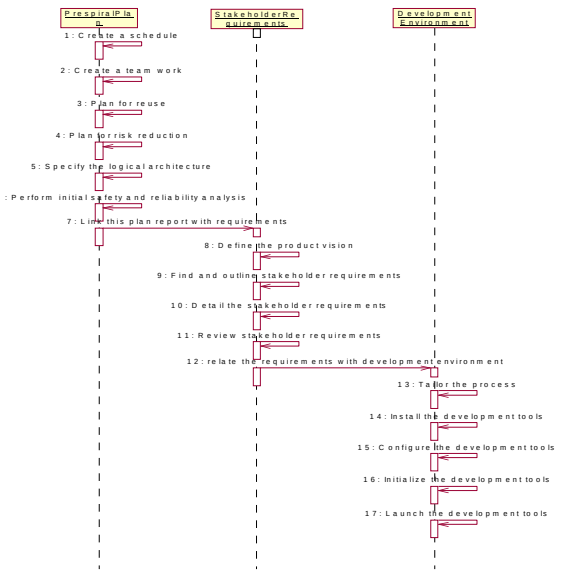

Figure 4. Sequence Diagram for Agile Modeling predecessor activities

Refer to Figure 5 which provides sequence diagram for Agile Modeling Analysis, Design and Testing..

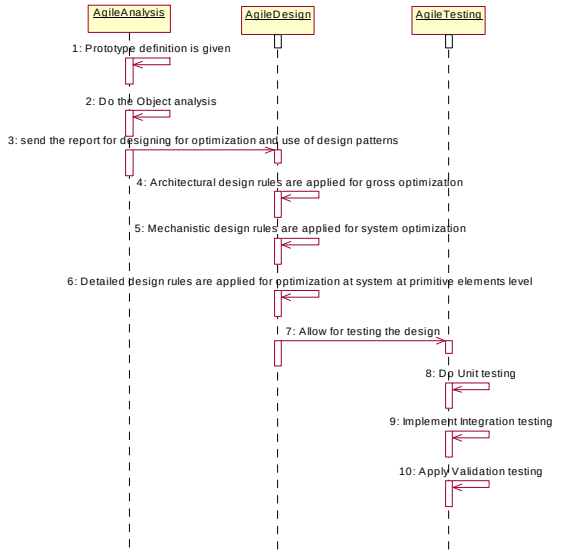

Figure 5. Sequence Diagram for Agile Modeling Analysis, Design and Testing

IV. Implementations and Validations

A. Design of Web Services

SERVICE-ORIENTED computing (SOC) is an emerging paradigm for designing distributed applications. SOC applications are obtained by suitably composing and coordinating (that is, orchestrating) available services. Services are stand-alone computational units distributed over a network and are made available through standard interaction mechanisms. Composition of services may require peculiar mechanisms to handle complex interaction patterns (for example, to implement transactions) while enforcing nonfunctional requirements on the system behavior, for example, security, availability, performance, transactional, quality of service, etc. From a methodological perspective, Software Engineering should facilitate the shift from traditional approaches to the emerging service-oriented solutions. Along these lines, one of the goals of this paper is to strengthen the adoption of formal techniques for modeling, designing, and verifying SOC applications. In particular, we propose a SOC modeling framework supporting history-based security and call by contract.

The execution of a program may involve accessing security-critical resources and these actions are logged into histories. The security mechanism may inspect these histories and forbid those executions that would violate the prescribed policies. Service composition heavily depends on which information about a service is made public, on how those services that match the user's requirements can be chosen, and on their actual runtime behavior. Security makes service composition even harder. Services may be offered by different providers which only partially trust each other. On the one hand, providers have to guarantee that the delivered service respects a given security policy in any interaction with the operational environment, regardless of who actually called the service. On the other hand, clients may want to protect their sensitive data from the services invoked.

Our methodology for designing and composing services is to create new services, and to sell it by a package base through a secured media. In particular, we are concerned with Safety properties of service behavior. Services can enforce security policies locally and can invoke other services that respect given security contracts. This call-by-contract mechanism offers a significant set of opportunities, each driving secure ways to compose services. We discuss how we can correctly plan service compositions in several relevant classes of services and security properties. With this aim, we propose a graphical modeling framework in this project. Our formalism features dynamic and static semantics, thus allowing for formal reasoning about systems. Static analysis and model checking techniques provide the designer with useful information to assess and fix possible vulnerabilities.

Several approaches have been developed to support the verification of service-oriented systems. For example, dynamic bisimulation-based techniques have been adopted to analyze the consistency between orchestration and choreography of services whereas state-space analysis has been exploited to check the correctness of service orchestration. Our approach allows for synthesizing and checking the correctness of the orchestration statically.

In proposed system, we introduced a UML-like graphical language for designing and verifying the security policies of service oriented applications. Another feature offered by our framework is that of mapping high-level service descriptions into more concrete programs. This can be done with the help of simple model transformation tools. Such model-driven transformation would require very little user intervention. Here one new framework is introduced called Service Component Architecture (SCA). This framework aims at simplifying implementations by allowing designers to focus only on the business logic while complying with existing

standards. Our approach complements the SCA view, providing a full-fledged mathematical framework for designing and verifying properties of service assemblies. It would be interesting to develop a (model-transformation) mapping from our formal framework to SCA. Refer to Figure 6 which provides class diagram for Web Services Design Application.

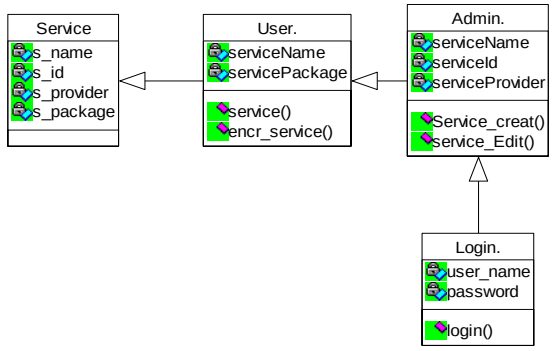

Figure 6. Class Diagram for Web Services Application Design

Refer to Figure 7 which provides execution screen shot for Web Services Design Application.

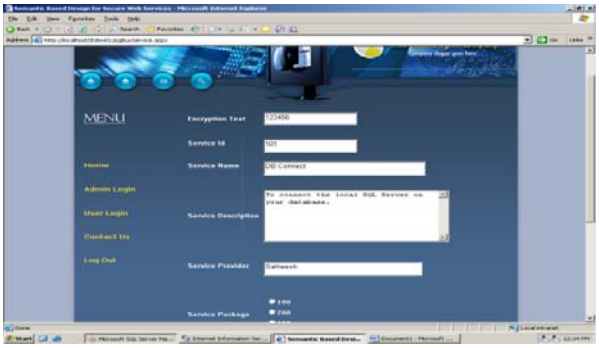

Figure 7. Execution Screen shot for Web Services Application Design

B. Design of Agile Pair Programming

In this implementation, efficiency of pairs in program design tasks is identified by using pair programming concept. Pair programming involves two developers simultaneously collaborating with each other on the same programming task to design and code a solution. Algorithm design and its implementation are normally merged and it provides feedback to enhance the design. Previous controlled pair programming experiments did not explore the efficacy of pairs against individuals in program design-related tasks. Variations in programmer skills in a particular language or an integrated development environment and the understanding of programming instructions can cover the skill of subjects in program design-related tasks. Programming aptitude tests (PATs) have been shown to correlate with programming performance. PATs do not require understanding of programming instructions and do not require a skill in any specific computer language. By conducting two controlled experiments, with full-time professional programmers being the subjects who worked on increasingly complex programming aptitude tasks related to problem solving and algorithmic design. In both experiments, pairs significantly outperformed individuals, providing evidence of the value of pairs in program design-related tasks. Refer to Figure 8 and Figure 9 which provides respectively class diagram and sequence diagram for Agile Pair Programming design application.

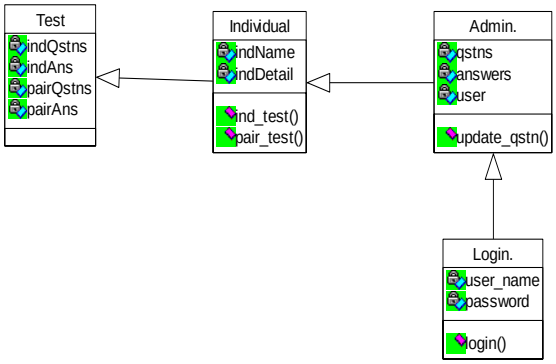

Figure 8. Class diagram for agile pair programming application

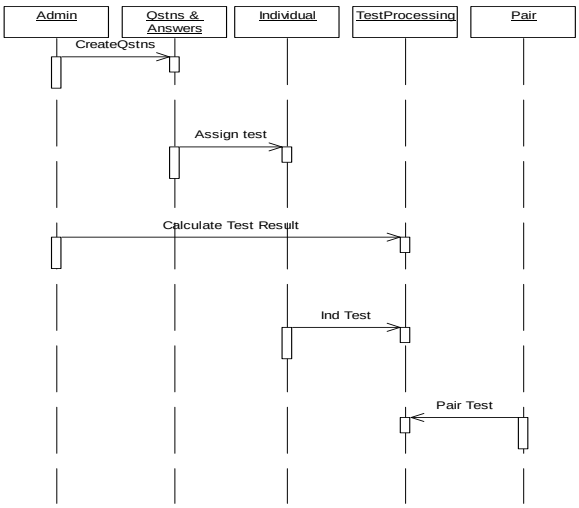

Figure 9. Sequence diagram for agile pair programming application

In the computerized world all the data are saved on electronically. It also contains more sensitive data. In computer systems security, role-based access control is an approach to restricting system access to authorized users. It is a newer alternative approach to mandatory access control and discretionary access control. Security critical business processes are mapped to their digital governments. It needs different security requirements, such as healthcare industry, digital government, and financial service institute. So the authorization and authentication play a vital role. Authorization constraints help the policy architect design and express higher level organizational rules. Access is the ability to do something with a computer resource (e.g., use, change, or view). Access control is the means by which the ability is explicitly enabled or restricted in some way

(usually through physical and system-based controls). Computer- based access controls can prescribe not only who or what process may have access to a specific system resource, but also the type of access that is permitted. These controls may be implemented in the computer system or in external devices. Refer to Figure 10, Figure 11 and Figure 12 which provides respectively class diagram, sequence diagram and execution screen shot for Role-based access control for Web Services policies.

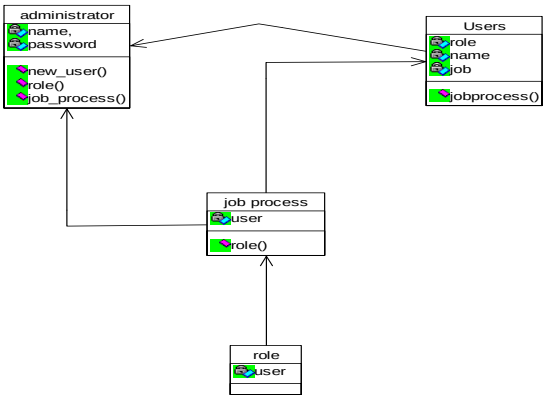

Figure 10. Class diagram for RBAC Web Services policies

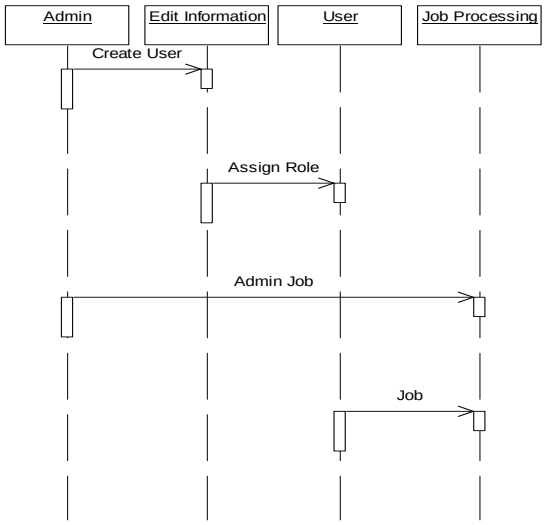

Figure 11. Sequence diagram for RBAC Web Services policies

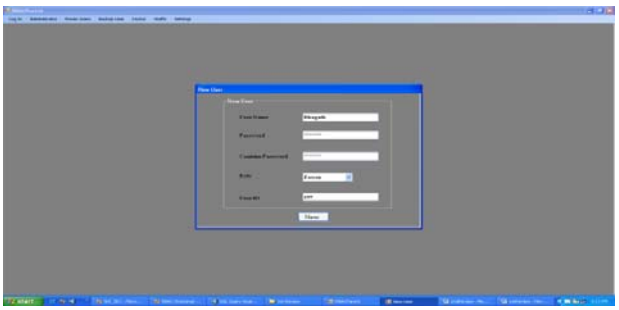

Figure 12. Execution screen shot for RBAC Web Services policies

D. Model-Driven Web Requirements

Web engineering is a new research line in software engineering that covers the definition of processes, techniques, and models suitable for Web environments in order to guarantee the quality of results. The research community is working in this area and, as a very recent line, they are assuming the Model-Driven paradigm to support and solve some classic problems detected in Web developments. However, there is a lack in Web requirements treatment. This paper presents a general vision of Navigational Development Techniques (NDT), which is an approach to deal with requirements in Web systems. It is based on conclusions obtained in several comparative studies and it tries to fill some gaps detected by the research community. This paper presents its scope, its most important contributions, and offers a global vision of its associated tool: NDT-Tool. Furthermore, it analyzes how Web Engineering can be applied in the enterprise environment. NDT is being applied in real projects and has been adopted by several companies as a requirements methodology. The approach offers a Web requirements solution based on a Model-Driven paradigm that follows the most accepted tendencies by Web engineering.

Most approaches in Web engineering are focused on analysis and design phases. They usually propose using classic requirements techniques, such as use cases, in order to capture and define requirements on the Web. Metamodels do not offer specific artifacts to deal with the Web environment since only an approach for classic requirements treatment is offered.

This project presents the power of tools that support metamodels because they are suitable for any approach defined using metamodels NDT is a methodological approach which deals with requirements in Web environments. NDT was proposed in order to support the requirements engineering and the analysis phase of Web systems and is based on the MDE paradigm. Refer to Figure 13 which consists of execution screen shot of Model-Driven Web requirements.

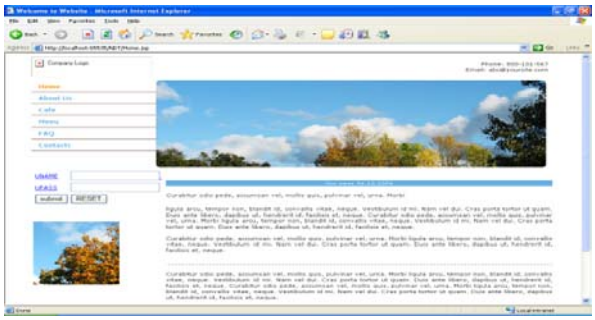

Figure 12. Execution screen shot for Model-driven web requirements

V. Designing Secure Services Applications Using Patterns

I Designing an effective authorization strategy is important for the security and reliability of our service application. Failure to design a good authorization strategy can leave our application vulnerable to information disclosure, data tampering, and elevation of privileges. Consider the following guidelines when designing an authorization strategy. Set appropriate access permissions on resources for users, gropus, and roles; and apply granular level granular level authorization across all trust boundaries. Execute services under the most restrictive account that is appropriate. Consider using Uniform Resource Locator (URL) authorization and/or file autorization when protecting URL-and file-based resources. When appropriate, restrict access to publicly accesible service methods using declarative principle permisssions demands.

Conclusions and Future Work

In this paper, we proposed role-based access control policies for Web Services using Layered Model-driven architectures and Agile modeling security principles for enhancing security requirements. Further work includes Securer Web Service contract design and versioning for SOA. Security for Workflow and Business Processing focuses on an important component that makes it possible to build and manage complex applications. In a Web-based environment, business processes or workflows can be built by combining Web Services through the use of a process specification language. Such languages basically allow us to specify which tasks have to be executed and the order in which those tasks should be executed. One such language is WS-BPEL, which provides syntax for specifying business processes based on Web Services.

Acknowledgements

The authors wish to thank the following students of CSE, MGIT for implementing these concepts: B.Preethi, Kavitha, Anuradha, R.Nitesh Kumar, Ch.Nitesh Reddy, M.Sundeep and A.Prithvi Srikanth

For detailed implementations, source code, UML diagrams and documentation, please refer to the website http://sites.google.com/site/upendramgitcse

References

[1] Bernard Menezes,"Network Security and Cryptography", Cengage Learning India Pvt. Ltd., 2010 pp. 393-413.

[2] Elisa Bertino, Lorenzo D.Martino, Federica Paci, Anna C.Squicciarini," Security for Web Services and Service-Oriented Architectures", Springer-Verlag Berlin Heidelberg 2010

[3] Anders Mattsson, Bjorm Lundell, Brian Lings, and Brian Fitzgerald, "Linking Model-Driven Development and Software Architecture: A Case Study", IEEE Transactions on Software Engineering, vol. 35, no. 1. pp. 83-93 January/February 2009.

[4] Spyros T. Halkidis, Nikolaos Tsantalis, Alexander Chatizigeorgiou and George Stephanides, "Architectural Risk Analysis of Software Systems Based on Security Patterns," *IEEE Transactions on Dependable and Secure Computing*, vol. 5 no. 3, pp. 129–142, July-September 2008.

[5] Mouratidis and Giorgini, *Integrating Security and Software Engineering: Advances and Future Vision*. Idea Group Publishing Inc., 2007.

[6] Heiko Tillwick and Martin S Olivier, "A Layered Security Architecture: Design Issues", in Proceedings of the Fourth Annual Information Security South Africa Conference (ISSA2004), July 2004.

[7] Asoke K. Talukder and Manish Chaitanya, *Architecting Secure Software System*. CRC Press, 2009.

[8] Massimo Bartoletti, Pierpaolo Degano, Gian Luigi Ferrari and Roberto Zunino, "Semantics Based Design for Secure Web Services," *IEEE Transactions on Software Engineering*, vol. 34 no. 1, pp. 33–49, January-February 2008.

[9] Anoop Singhal and Theodore Winograd, *Guide to Secure Web Services*. NIST Draft (800-95), September 2006.

[10] Daniel Jackson, *A Direct path to Dependable Software*. Computer Science and Artificial Intelligence Software, Massachusetts Institute of Technology.

[11] Hossein Keramati, Seyed-Hassan Mirian-Hosseinabadi, "Integrating software development security activities with agile methodologies," aiccsa, pp.749-754, 2008 IEEE/ACS International Conference on Computer Systems and Applications.

[12] I. Lazar, B. Parv, S. Motogna, I.-G. Czibula, C.-L. Lazar, "An Agile MDA approach for Executable UML Structured Activities", Studia Univ. Babes-bolyai, Informatica, vol. LII, No. 2, 2007, pp.111-114

[13] Yann-Gael Gueheneuc, Giuliano Antoniol, "DeMIMA: A Multilayered Approach for Design Pattern Identification", IEEE Transactions on Software Engineering, vol. 34, no. 5. pp. 667-684, September/October 2008.

[14] Johan Peeters, "Agile Security Requirements Engineering", on Requirements Engineering for Information Security, 2005.

[15] Ramarao Kanneganti, Prasad Chodavarapu, "SOA Security", Manning Publishers, 2007, ISBN 978-1932394689.

[16] Sylvain Halle, Roger Villemaire, Omar Cherkaoui,"Specifying and Validating Data-Aware Temporal Web Servies Properties", IEEE Transactions on Software Engineering, Vol. 35, No. 5, pp. 669-682 September/October 2009.

[17] George Spanoudakis and Andrea Zisman, "Discovering Services during Service-Based System Design Using UML", IEEE Transactions on Software Engineering, Vol 36, No.3, May/June 2010, PP 371 – 389.

[18] Joao Antunes, Nuno Neves, Miguel Correla, Paulo Verissimo, Rui Neves,"Vulnerability Discovery with Attack Injection", IEEE Transactions on Software Engineering, Vol. 36, No. 3, pp. 357-369, May/June 2010.