# Dependable Solutions Design by Agile Modeled Layered Security Architectures

M. Upendra Kumar[1,*], D. Sravan Kumar[2], B. Padmaja Rani[3], K. Venkateswar Rao[4], A.V. Krishna Prasad[5], and D. Shravani[6]

[1]CSE JNTUH and CSE MGIT, Hyderabad, A.P., India
uppi_shravani@rediffmail.com
[2] CSE KITE WCPES, Hyderabad, A.P., India
dasojusravan@gmail.com
[3] CSE JNTU CEH, Hyderabad, A.P., India
padmaja_jntuh@yahoo.co.in
[4] CSE JNTU CEH, Hyderabad, A.P., India
Kvenkateswarrao_jntuh@rediffmail.com
[5] S.V. University Tirupathi, A.P., India
kpvambati@gmail.com
[6] Rayalaseema University, Kurnool, A.P., India
sravani.mummadi@yahoo.co.in

**Abstract.** Our research entitled "Designing Dependable Agile Layered Security Architecture Solutions" addresses the innovative idea and novel implementations of Security Engineering for Software Engineering using Agile Modeled Layered Security Architectures for Dependable Privacy Requirements, with a validation of an exemplar case study of Web Services Security Architectures. Securing the Software Architecture in any application at design phase is known as Security Architectures, and we focus on authentication and authorization of the user. Now a day most of the applications are developed as a Layered Security Architecture Pattern, typically we have user presentation layer, Business Logic Layer and Database access layer. Now Agile modeling is used in all applications design (but Agile Modeled Architectures are given little importance) because of shortened developed time, with customer collaborations with developers and importantly with Test Driven Development approaches. Securing Agile Modeled architectures, which being an iterative development, will provide enhanced Dependable Security Requirements in terms of Privacy of user, in its successive iterations. All this research paves a way for Secure Web Engineering.

**Keywords:** Security Architectures, Agile Modeling, Layered Pattern, Designing Solutions, Dependable Privacy Requirements, Web Services.

* Corresponding author.

N. Meghanathan et al. (Eds.): CCSIT 2012, Part I, LNICST 84, pp. 510–519, 2012.

# 1 Research Methodology for Designing Dependable Agile Layered Security Architecture Solutions—Web Services Case Study

This paper discusses the latest and advantages in secure software development process in an early stage. The primary focus is on security considerations early in the life cycle, i.e. at the system architecture stage, which has the potential to improve the requirements engineering in software system. The ultimate goal is to have a better quality product. Initially we discuss about the Research Methodology for Designing Dependable Agile Layered Security Architecture Solutions. Later on we discuss about dependability of data and application layers by Agile Modeled Security Architectures, validated with a case study of Web Services Security Architectures.

**Research on Security Architectures.** 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 Architectures 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 a Model Driven Architecture (MDA) 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. More over, 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 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".

**Research Problem Statement.** Our research entitled "Designing Dependable Agile Layered Security Architecture Solutions" addresses the innovative idea of Security Engineering for Software Engineering using Agile Modeled Layered Security Architectures for Dependable Privacy Requirements with a validation of case study of Web Services Security Architectures. The key research questions addressed are: How a failure addresses a specific security service at a specific layer impact other (interdependent) layers? Also how successful implementation of a security service had an affect on the rest of the system? [2] How can agile methods be used to generate effective security requirements? In what ways do these agile methods change the development of security requirements? How is the outcome of emergent security development different from more traditional forms? [3]

**Organization of Research Methodology.** First, we introduce to secure software engineering, security architectures design and development, introduction and overview of research title, software security architecture using Model Driven Architecture, Agile Methods, Case study of Web Services Security Architectures are discussed so that the problem statement can be designed. Second, a detailed literature survey was conducted on Secure Software Engineering, Model Driven Architecture, Agile methodology, Security Patterns for Agile Layered Security Architectures, UML 2.0, and Secure UML, Web Services Security Architectures, to find out basis for the thesis. Third, we design Agile Modeled Layered Security Architectures, with validations of case study for Web Services Security Architectures, with a initial case study validations using on simple secure Web Services Design using Agile Modeled Test Driven Development. Fourth, we design solutions using Agile Modeling for Layered Security Architectures with case study of Web 2.0 Services Security Architectures and its implementations. Fifth, Dependability (regarding Privacy requirements) Agile Modeled Layered Security Architectures with a case study of Web Services Security architectures are done, with implementation of a financial application for Secure Stock Market using Web Services.

## 2 Dependable Solutions Design by Agile Modeled Layered Security Architectures

**Software Architecture:** An important design artifact in any software development project, with the possible exception of very small Projects, is the Software Architecture. An important part of any architecture is the set of Architectural Design Rules. Architectural Design Rules are defined as the rules, specified by the architect(s) that need to be followed in the detailed design of the system. A primary role of the architecture is to capture the architectural design decisions. An important part of these design decisions consists of architectural design rules.

**Security:** Security ensures that information is provided only to those users who are authorized to possess the information. Security generally includes the following:

Identification: This assumes that system must check whether a user really is whom he or she claims to be. There are many techniques for identification and it is also called as authentication. The most widely used is “Username/Password” approach. More sophisticated techniques based on biometrical data are like retinal fingerprint scan.

Authorization: This means that the system should provide only the information that the user is authorized for, and prevent access to any other information. Authorization usually assumes defining “user access rights”, which are settings that define to which operations, data, or features of the system the user, does have access.

Encryption: This transforms information so that unauthorized users (who intentionally or accidentally come into its possession) cannot recognize it.

**Model Driven Architecture:** Model-Driven Development (MDD) is a modeling approach. The basic premise of Model-Driven Development is to capture all important design information in a set of formal or semiformal models, which are kept

consistent automatically. To realize full benefits of MDD, formalize architecture design rules, which then allow automatic enforcement of architecture on the system model. There exist several approaches to MDD, such as OMG's (Object Management Group) MDA (Model-Driven Architecture), Domain Specific Modeling (DSM), and Software factories fro Microsoft. Model-Driven Architecture prescribes that three models or sets of models shall be developed as:

The Computationally Independent Model(s) (CIM) captures the requirements of the system.

The Platform-Independent Model(s) (PIM) captures the systems functionality without considering any particular execution platform.

The Platform-Specific Model(s) (PSM) combines the specifications in the PIM with the details that specify how the system uses a particular type of platform. The PSM is a transformation of the PIM using a mapping either on the type level or at the instance level.

MDA does not directly address architectural design or how to represent the architecture, but the architecture has to be captured in the PIM or in the mapping since the CIM captures the requirements and the PSM is generated from the PIM using the mapping.

**Agile Methods:** Over the past few years, a new family of software engineering methods has started to gain acceptance amongst the software development community. These methods, collectively called Agile Methods, conform to the Agile Manifesto, which states "We are uncovering better ways of developing software by doing it and helping others does it. Through this work we have come to value: Individuals and interactions over processes and tools working software over comprehensive documentation customer collaboration over contract negotiation responding to change over following a plan That is, while there is value in the items on the right, we value the items on the left more." The individual agile methods include Extreme Programming (XP), Scrum, Lean Software Development, Crystal Methodologies, Feature Driven Development (FDD), and Dynamic Systems Development Methodology (DSDM). While there are many differences between these methodologies, they are based on some common principles, such as short development iterations, minimal design upfront, emergent design and architecture, collective code ownership and ability for anyone to change any part of the code, direct communication and minimal or no documentation (the code is the documentation), and gradual building of test cases. Some of these practices are in direct conflict with secure SDLC processes.

**Security Requirements:** Agile information systems and software methods are characterized by nimbleness to rapid changes, multiple incremental iterations and a fast development pace. Agile development is defined as a set of principles and practices that differs as a whole from traditional planned development. The major principles for agile information systems and software methods include:

Accept multiple valid approaches: A stable architecture, a tool orientation and component based development combine to enable a "fluid view" of methodology and the value of tailoring the methodology for each development project. Improvisation in development approach will help match the methodology to the constraints of the project environment. Engage the customer: Close involvement of customers in the

project enables accurate and fast requirements elicitation, and the customers again immediate satisfaction as their ideas and requirements arise in each new release.

Accommodate requirements change: Agility means that developers quickly and easily respond to the shifting requirements driven by the changing environment for which the software is intended. Build on successful experience: The "right" people are important for project success in order to foster innovation in software development. Courage, specific knowledge, intelligence, and commitment are needed for agile development. Develop good teamwork: The right mix of people operating with the right process framework means that the right mix of knowledge and working style will be present in the project. Agile development teams must often come together quickly and be immediately effective. Agile practices include:

Develop in parallel: Releases may be completely developed in parallel, or staged onto the market such that design, development, and quality assurance are all taking place simultaneously, but sequentially on different releases. Coding may even begin before the requirements are declared. Release more often: Releases are scoped to more frequently deliver small sets of new features and fixes. Constant re-prioritization of features enables responsiveness to changing requirements and enables features to easily slip from one release to the next. Depend on tools: Heavy use of development tools and environment that speed up the design and coding process offer much of the functionality that used to be custom built. Ideally, agile developers try to avoid wasting time repetitively building features others have already developed. Implant customers in the development environment: Fast and intimate access to customer views and opinions slashes time, and ensures the high-priority features are built first. When customers participate closely in all phases of development, cycle times shorten and teams can better chuck requirements into logical releases from customer views.

Establish a stable architecture: This anchors a rapid development process that is never quite stable, yet each release has some similarity and components reuse. Assemble and reuse components: Never unnecessarily build software from scratch when it can be assembled from existing components. It is quicker and equally effective to acquire, integrate, and assemble components with wrappers, including business logic software, interfaces and back-end infrastructure.

Ignore maintenance: Building components for short life spans eliminates the need for documentation. Assembled software can be thrown away and reassembled with greater ease than maintaining complex and custom-build components. Tailor the methodology daily: Operating with an overall development framework, but allowing project teams to adjust the exact approach to the daily situation, enabled teams to meet intense demands for speed by skipping unnecessary tasks or phases. Use just enough process to be effective, and no more.

**Security requirements for Agile Security methods and Extant Security methods:** Requirements for security methods that are targeted to be integrated into agile software methods: The security approach must be adaptive to agile software development methods. They must be simple; they should not hinder to the development project. The security approach, in order to be integrated successfully with agile development methods, should offer concrete guidance and tools at all phases of development (i.e., from requirements capture to testing).

A successful security component should be able to adapt rapidly to ever changing requirements owing to a fast-paced business environment, including support for handling several incremental iterations.

**Key Security Elements in Agile Software Development.** The key security element stems from information security "meta-notation", or notation for notations, and database security. Apply these key security elements to a process aimed at developing secure software in an agile manner. This generic security process consists of these key security elements in different phases of software development (requirements analysis, design, implementation and testing). These steps are not necessarily sequential and in any case, every step is optional.

**Web Services Security Architectures Case Study.** This case study is done with a research motivation for Secure Service Oriented Analysis and Design and Secure Service Modeling. "Designing Dependable Web Services Security Architecture Solutions" addresses the innovative idea of Web Services Security Engineering through (or using) Web Services Security Architectures with a research motivation of Secure Service Oriented Analysis and Design. It deals with Web Services Security Architectures for Composition and Contract Design in general, with authentication and authorization (access control) in particular, using Agile Modeled Layered Security Architecture design, which eventually results in enhanced dependable privacy requirements, Security Policies and Trust Negotiations. All the above findings are validated with appropriate case studies of Web 2.0 Services, BPEL for Role Based Access Control, a secure stock market financial application, and their extensions for spatial mobile application for cloud. All this research paves a way for Secure Web Engineering (or) Secure Web Science. Key research questions addressed here are: How can Agile Modeled Layered Security Architectures design be used for Web Services Security Architectures with a motivation of Dependable Privacy requirements? How can we extend the above approach for Web 2.0 Services Security Architectures? How can we validate this approach for Spatial Mobile Web Services Security Architectures for Cloud case study?

**Further Extension of this Research Work.** Mining approach for Business Intelligence to improve insights of Web Engineering applications deals with an innovative idea of Mining for Web Engineering with a case study of Business Intelligence Web application. Next generation Business Intelligence web application development uses integrated technologies like Web 2.0, Agile Modeling, and Service-orientation (using Web Services). We initially validated the Web 2.0 Services and Agile Modeling, for insights of Web application security in terms of authentication and authorization for Web Engineering. Applying Mining strategies to Web Services will provide valuable insights in terms of Service discovery, Service dependency, Service composition etc. This approach provides insights of Web application security for Web Engineering. These insights are important in maintenance of these developed applications and also in their scalability purposes. We validate our approach with a suitable exemplar Secure Web Services for Stock Market application.

## 3 Implementations and Validations

The section discusses about Implementations and Validations on Web Services Security Architectures Case Study.

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 1 which provides class diagram for Web Services Design Application.

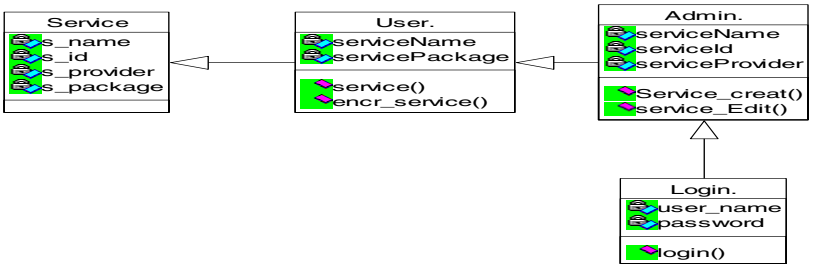


**Fig. 1.** Class Diagram for Web Services Application Design

**Role Based Access Control for Web Services Security Policies**

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 2 and Figure 3which provides respectively class diagram and sequence diagram and execution screen shot for Role-based access control for Web Services policies.

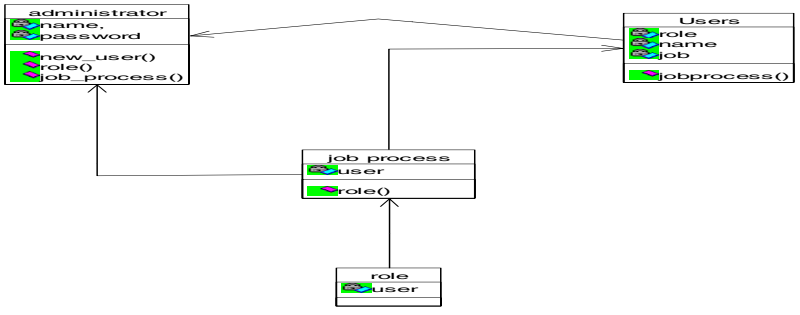


**Fig. 2.** Class Diagram for RBAC Web Services Security Policies

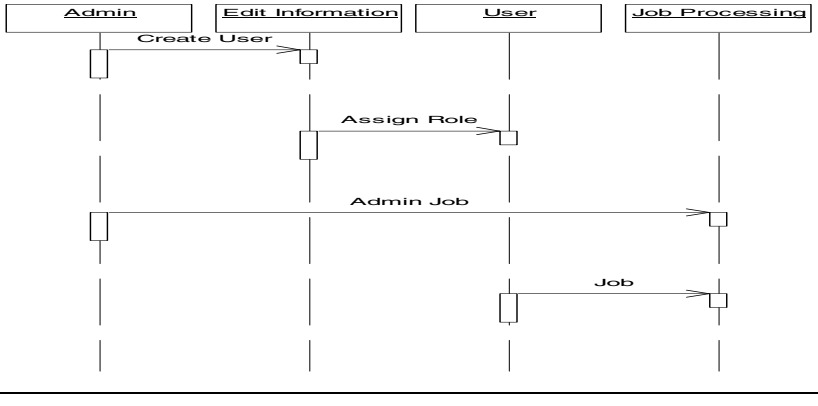


**Fig. 3.** Sequence Diagram for RBAC Web Services Security policies

## 4 Conclusions

This paper had discussed about Research Methodology on Designing Dependable Agile Layered Security Architecture Solutions – Web Services Case Study. In this research the major part is given to model architectural design rules using Model Driven Architecture (MDA) so that architects and the developers are responsible to automatic enforcement on the detailed design and easy to understand and use by both of them. This MDA approach is implemented in use of agile strategy in three different phases covering three different layers to provide security to the system. With this procedure a conclusion has been given that with the system security the requirements for that system are improved. This research summarizes that security is essential for every system at initial stage and upon introduction of security at middle stage must lead to the change in the system i.e. an improvement to system requirements.

## References


1. Peterson, G.: Security architecture Blueprint. Arctec Group LLC (2007)
2. Tillwick, H., Olivier, M.S.: A Layered Security Architecture Blueprint. In: Proceedings of the Fourth Annual Information Security South Africa Conference (2004)
3. Baskerville, R.: Agile Security for Information Warfare: A call for research. Georgia State University, USA
4. Anderson, R.: Security Engineering: A guide to building Dependable Distributed Systems, 2nd edn. Wiley Publishers, USA (2008)
5. Bishop, M.: Computer Security. Art and Science. Pearson Education (2003)
6. Mao, W.: Modern Cryptography: Theory and Practice. Pearson Education (2004)
7. Gupta, V., et al.: Sizzle. A standard based end to end security architecture for the Embedded Internet. In: Pervasive and Mobile Computing. Elsevier (2005)
8. Pandian, D., et al.: Information Security Architecture – Context aware Access control model for Educational Applications. International Journal of Computer Science and Network Security (2006)
9. Whitmore, J.J.: A method for designing secure solutions. IBM Systems Journal 40(3), 747–768 (2001)
10. Smetters, D.K., Grinter, R.E.: Moving from the design of usable security technologies to the design of useful secure applications. In: ACM New Security Paradigms Workshop, pp. 82–89 (2002)
11. Cheng, B.H.C., Konrad, S., Campbell, L.A., Wasserman, R.: Using security patterns to model and analyze security requirements
12. Hunt, J.: Agile Software Construction. Springer, Heidelberg (2006)
13. Zelster, L.: Security Architecture cheat sheet for Internet applications
14. Harman, M., Mansouri, A.: Search based Software Engineering. Introduction to the special issue of the IEEE Transactions on Software Engineering, 737–741 (November/December 2010)
15. Spiekermann, S., Cranor, L.: Engineering Privacy. IEEE Transactions on Software Engineering 35(1), 67–82 (2009)
16. Mattsson, A., Lundell, B., Lings, B., Fitzgerald, B.: Linking Model-Driven Development and Software Architecture: A Case Study. IEEE Transactions on Software Engineering 35(1) (2009)
17. Douglass, B.P.: Real-time agility, the Harmony/ESW Method for Real-time and Embedded Systems Development. Pearson Education Inc. (2009)
18. Russo, B., Scotto, M., Silliti, A.: Agile Technologies in Open Source Development. IGI Global publishers (2010)
19. Keramati, H., Mirian-Hosseinabadi, S.-H.: Integrating Software Development Security Activities with Agile Methodologies. IEEE (2008)
20. Lazar, I., Parv, B., Motogna, S., Czibula, I.-G., Lazar, C.-L.: An Agile MDA approach for Executable UML Structured Activities. Studia Univ. Bases LII(2) (2007)
21. Gueheneuc, Y.-G., Antoniol, G.: DeMIMA: A Multilayered Approach for Design Pattern Identification. IEEE Transactions on Software Engineering 34(5) (2008)
22. Halkidis, S.T., Tsantalis, N., Chatzigeorgiou, A., Stephanides, G.: Architectural Risk Analysis of Software Systems Based on Security Patterns. IEEE Transactions on Dependable and Secure Computing 5(3) (2008)
23. Gamma, E.: Design Patterns Elements of Reusable Object Oriented Software. Addison Wesley Publishers (2009)
24. Siponen, M., Baserville, R., Kuivalainen, T.: Extending Security in Agile Software Development Methods, pp. 143–157
25. Peeters, J.: Agile Security Requirements Engineering, psu.edu 10.1.1.91.4183