DEPENDABLE PRIVACY REQUIREMENTS BY AGILE MODELED LAYERED SECURITY ARCHITECTURES – WEB SERVICES CASE STUDY

M.Upendra Kumar[1] Dr.D.Sravan Kumar[2] Dr.B.Padmaja Rani[3] K.Venkateswar Rao[4] A.V.Krishna Prasad[5]

[1]Research Scholar CSE JNTU Hyderabad A.P. India
uppi_shravani@rediffmail.com
[2]Principal and Professor CSE KITE WCPES Hyderabad A.P. India
dasojusravan@yahoo.co.in
[3]Professor CSE JNTU CEH Hyderabad A.P. India
padmaja_jntuh@yahoo.co.in
[3]Associate Professor CSE JNTU CEH Hyderabad A.P. India
kvenkateswarrao_jntuh@yahoo.co.in
[5]Research Scholar S.V.University Tirupathi A.P. India
kpvambati@gmail.com

ABSTRACT

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. 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 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. Agile modeled Layered Security Architectures increase the dependability of the architecture in terms of privacy requirements. We validate this with a case study of dependability of privacy of Web Services Security Architectures, which helps for secure service oriented security architecture. In this paper the major part is given to model architectural design rules using MDA so that architects and 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 premise conclusion has been given that with the system security the requirements for that system are improved. This paper 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.

Keywords

Security Architecture, Agile Modeling, Dependability, Privacy requirements, Web Services

D.C. Wyld, et al. (Eds): CCSEA 2011, CS & IT 02, pp. 232–240, 2011.

DOI: 10.5121/csit.2011.1222

1. INTRODUCTION TO AGILE MODELED LAYERED SECURITY ARCHITECTURES AND LITERATURE SURVEY

Software Engineering covers the definition of processes, techniques and models suitable for its environment to guarantee quality of results. For Software Architecture the requirements gathering and analysis is done using MDA (Model-Driven Architecture) which is a layered architecture. To provide security to the architecture various security techniques are used as agile methodologies. This states that Secure Software Architecture will improve the Requirements.

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 [1].

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 [11].

MDA: 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 [1].

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 [2].

2. DESIGNING DEPENDABLE PRIVACY REQUIREMENTS USING AGILE MODELD LAYERED SOLUTIONS

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 [9]:

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 [10].

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 [9].

3. IMPLEMENTATIONS AND VALIDATIONS – WEB SERVICES CASE STUDY

Privacy is a well recognized sticking point in the Web Services network. In this case study implementation, we explore privacy protection brings about many new security challenges. Web Services extended Cloud computing is the long dreamed vision of computing as a utility, where users can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources. By data outsourcing, users can be relieved from the burden of local data storage and maintenance. Here data (i.e., message/file) is transferred between the sender and the receiver in a extensively secure manner. Hence the communication between the sender and receiver is guaranteed by the Third Party Auditor (TPA).The software is designed in such a way that the user can easily interact with the screen because they are GUI and screen has several buttons with captions indicating the functionality like Sender details, message typed, searching for a file, keys and signatures generated, shows the encrypted data, verification of data, system name details, Receiver details. Business layer of this application are to be developed in such a way they must be easily maintainable and extensible. Software developed will able to do ant type of data transfer between sender and receiver in an authenticated, privacy of data contents.

Refer to the Figure 1, 2, 3 which provides the class diagram, sequence diagram and execution screen shot respectively of the privacy web services application implemented.

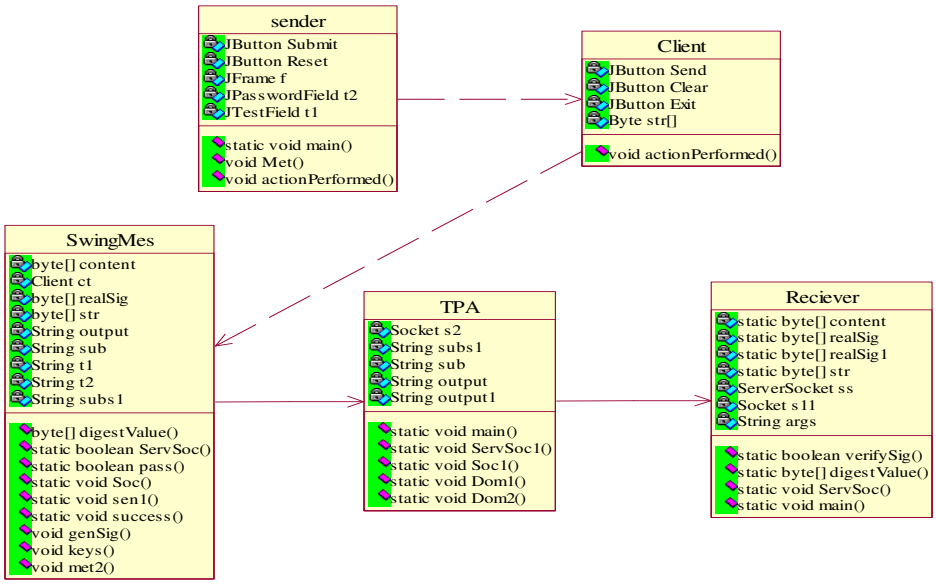

Figure 1. Class diagram of the Privacy Web Services application

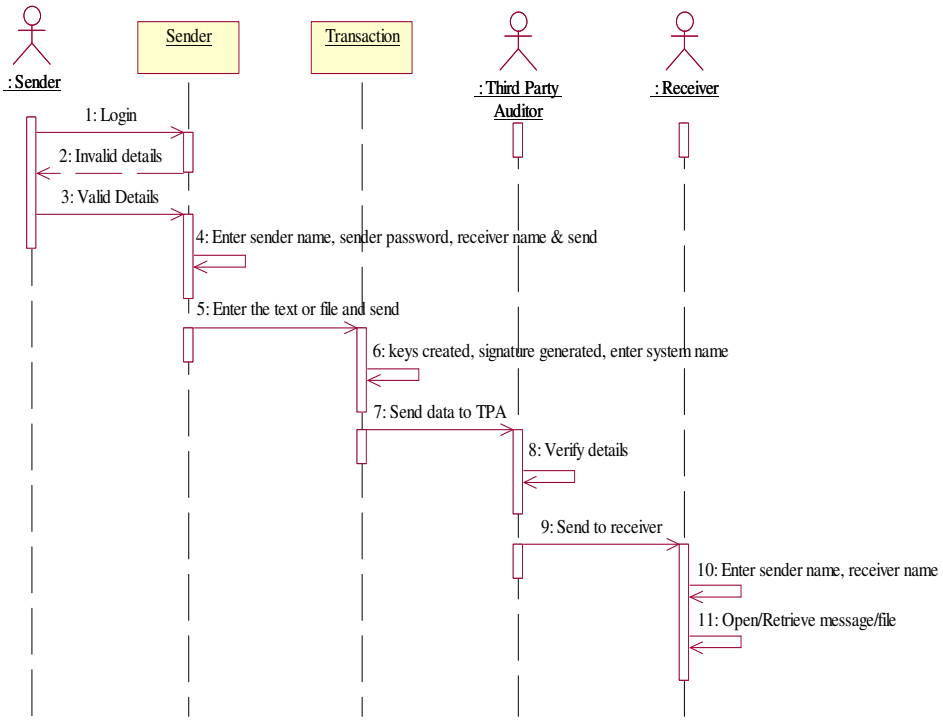

Figure 2. Sequence diagram of the Privacy Web Services application

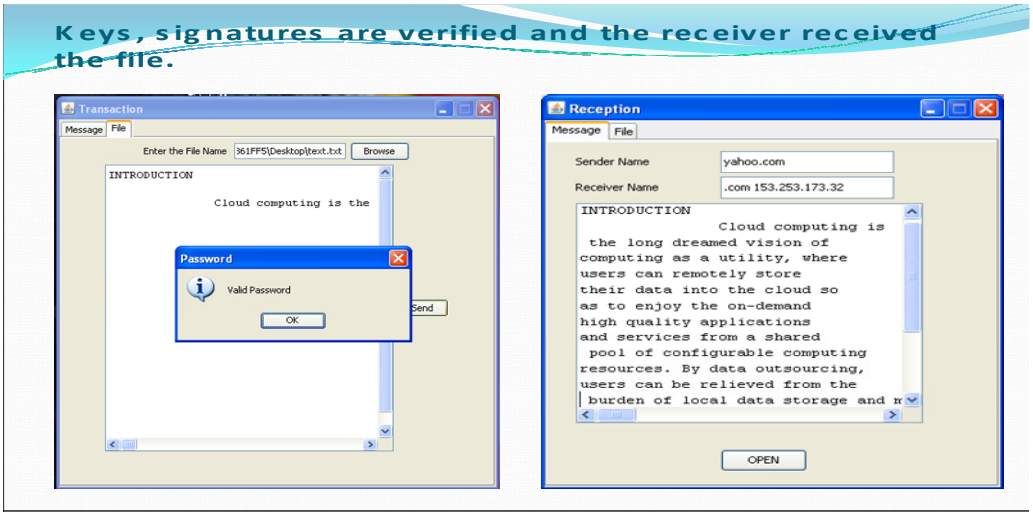

Figure 3. Execution Screen shot of the Privacy Web Services application

Validation of the Privacy Web Services Application: MDA with executable UML offers an approach that embodies all the key ingredients of the process for developing dependable systems, by offering: A uniform strategy for preserving investment in existing models built using unsupported tools, by automatically migrating them to profiled UML models for subsequent maintenance and development using state of the art UML tools; A clean separation of application behavior from the platform specific implementation using technologies such as Integrated Modular Avionics (IMA), allowing the full potential of IMA to be realized in a consistent and dependable way; A semantically well defined formalism that can be used a basis for modular certification of safety related systems; The ability to generate not only the components of the target system, but components of development tool chain, providing scope for model translation and offering "executable specifications" that can be tested early and mapped reliably onto the target, leading to greater levels of dependency.

MDA is a new approach for most organizations, and therefore carries additional training and learning curve costs and also currently the availability of production quality code generators is currently limited. MDA requires developers to work at a more abstract level than code although experience shows that most do not have any difficulty making the adjustment, there will be some who find this change of emphasis difficult to achieve. Building upon the initial success of MDA deployment so far, work is now proceeding on the enhancement of Ada code mapping rules to cover the entire xUML formalism. Work is also underway to develop a generic "adapter/router" component to provide a standard component to provide a standard way to interface re-engineered xUML components with pre-existing components. These techniques are now being applied to another avionics system in the same organization, in response to the customers need for a faster and cheaper upgrade capability. While we consider systematically all actions within a use case and analyze how they could be subverted, it produces all (or most) of the threats to a given application. While all this could be done in textual version of the use case, the use of UML activity diagrams produces a clear and more intuitive way to analyze these attacks. From the threats we derive necessary policies to stop or mitigate them. Refer to the Figures 4,5, 6 which provides the validated class diagram, sequence diagram and detailed sequence diagrams respectively of the previous implemented privacy web services application.

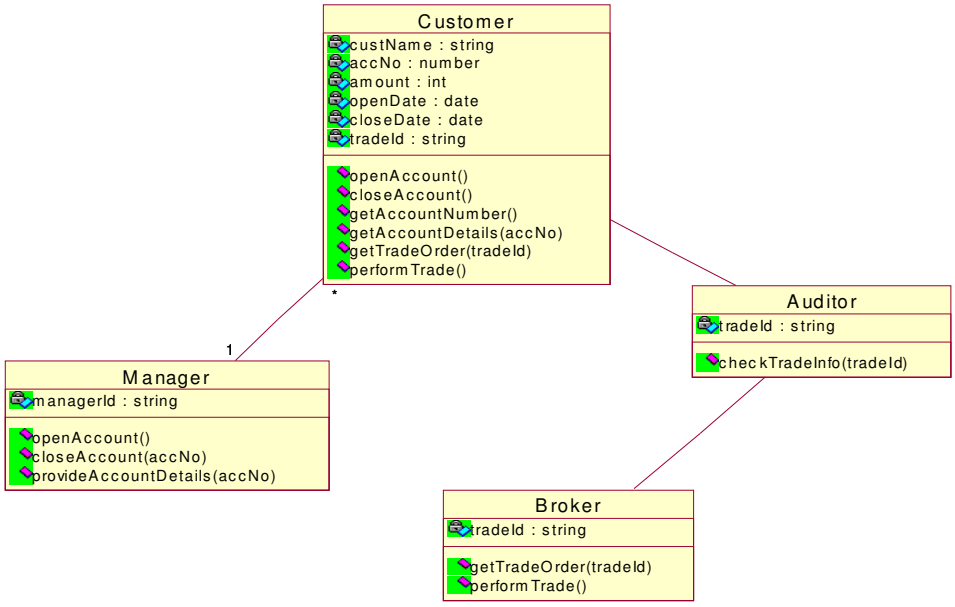

Figure 4. Class diagram of the validation of implemented case study

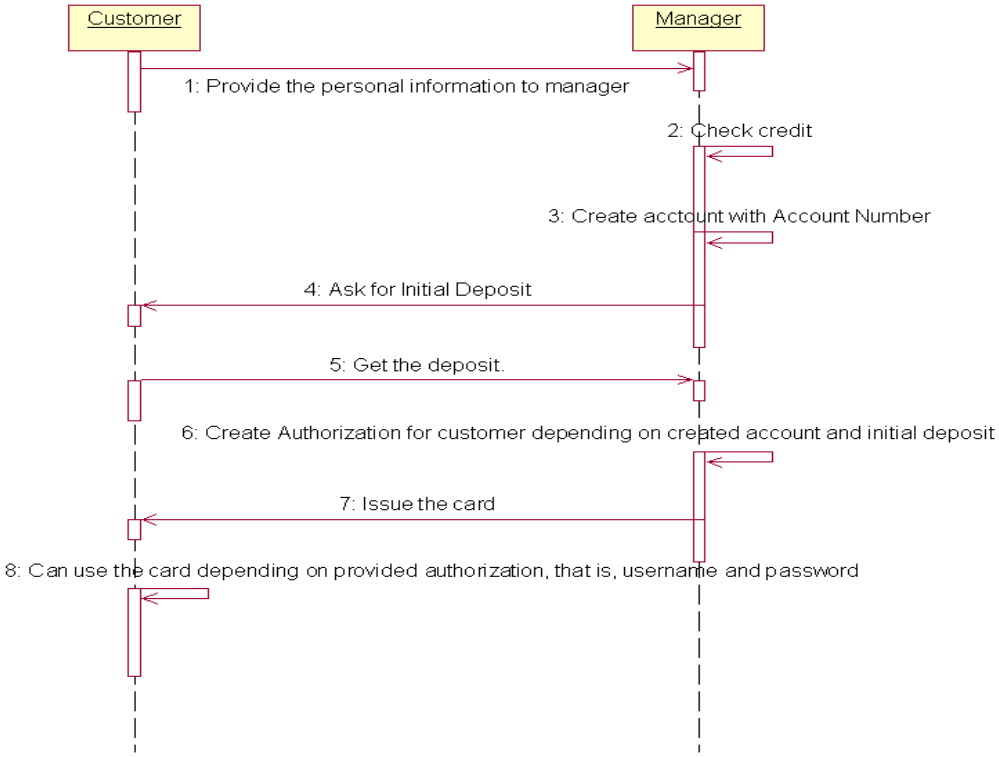

Figure 5. Sequence diagram of the validation of implemented case study

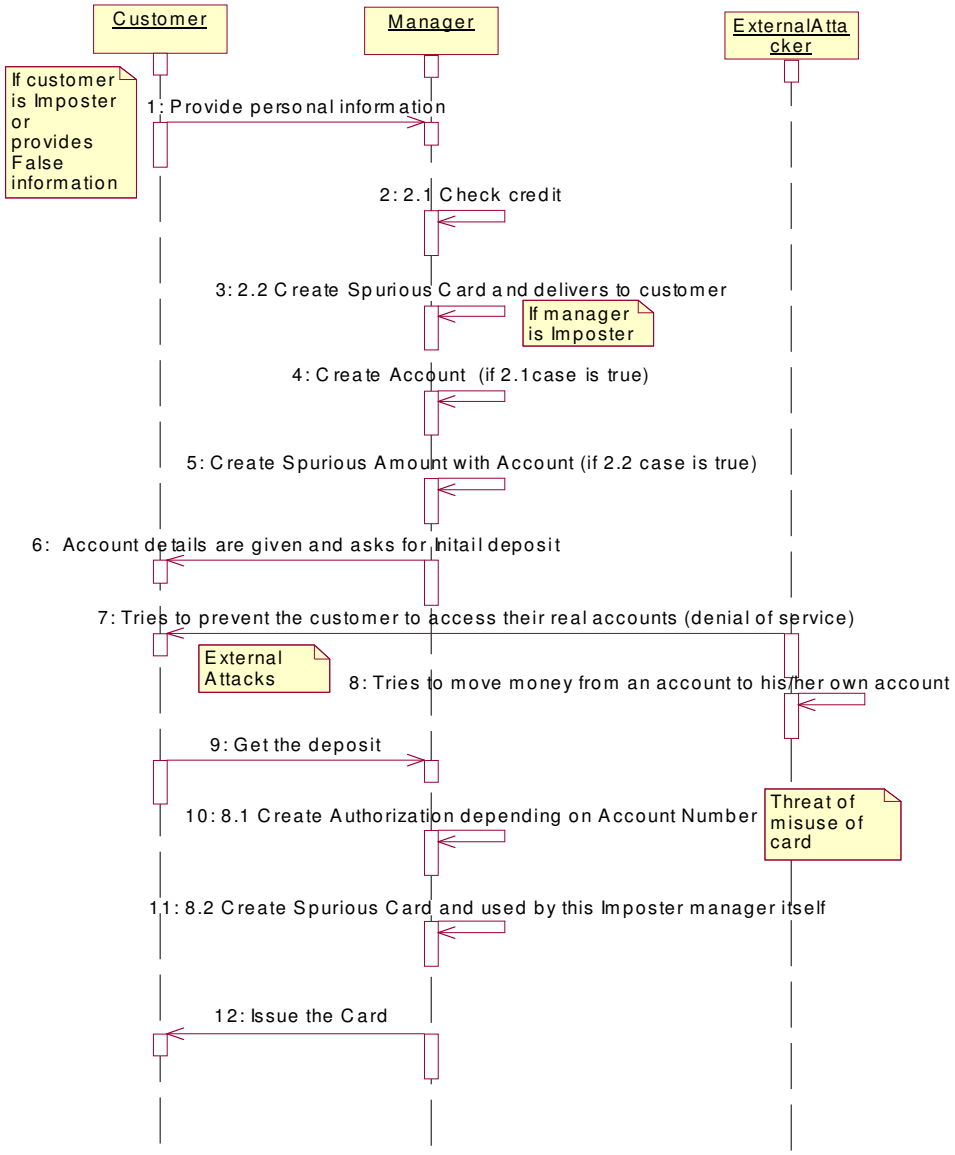

Figure 6. Detailed Sequence diagram of the validation of implemented case study

4. CONCLUSIONS

In this paper the major part is given to model architectural design rules using MDA so that architects and 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 paper 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.

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

REFERENCES

[1] Anders Mattsson, Bjorm Lundell, Brian Lings, and Brian Fitzgerald, (2009) "Linking Model-Driven Development and Software Architecture: A Case Study", 2009, IEEE Transactions on Software Engineering, vol. 35, no. 1.

[2] "Real-time agility, the Harmony/ESW Method for Real-time and Embedded Systems Development"

[3] "Design Approaches", Agile open source.

[4] Hossein keramati, Seyed-Hassan Mirian-Hosseinabadi, "Integrating Software Development Security Activities with Agile Methodologies", 2008, IEEE.

[5] I. Lazar, B. Parv, S. Motogna, I.-G. Czibula, C.-L. Lazar, "An Agile MDA approach for Executable UML Structured Activities", Studia Univ. Bases, vol. LII, No. 2, 2007.

[6] Yann-Gael Gueheneuc, Giuliano Antoniol, "DeMIMA: A Multilayered Approach for Design Pattern Identification", 2008, IEEE Transactions on Software Engineering, vol. 34, no. 5.

[7] Spyros T. Halkidis, Nikolaos Tsantalis, Alexander Chatzigeorgiou, George Stephanides, "Architectural Risk Analysis of Software Systems Based on Security Patterns", 2008, IEEE Transactions on dependable and secure computing, vol. 5, no. 3.

[8] Erich Gamma, "Design Patterns".

[9] M. Siponen, R. Baserville, T. Kuivalainen, "Extending Security in Agile Software Development Methods", pp 143-157.

[10] Johan Peeters, "Agile Security Requirements Engineering".