Design and Implementation of Agile Security Architecture Model

M.Upendra Kumar

Associate Professor CSE MGIT Hyderabad India

Abstract *In this Paper, it addresses Design and Implementation of proposed model for Agile Modelled Layered Security Architectures for Security Requirements. We had validated on case study Next Generation Secure Web Engineering Applications, using Agile Modelling for Web Services, Web 2.0 Services Authentication and Authorization.*

Keywords — *Design, Implementation, Agile Modelling, Security Architectures*

I. INTRODUCTION

1 PROPOSED MODEL FOR AGILE LAYERED SECURITY ARCHITECTURE DESIGN

For any Software Development, Designing Software Architecture is the important significant work. Designing Architecture is based on Architectural Design Rules. In MDA, Architectural Design Rules are formed in the form of models only. This design is done using design patterns concepts and System is de3veloped through Agile Processing. With this proposed model, for any software system security, the requirements are refined in the iterations to follow.

In the existing system, there is a link between Model Driven Development and Software Architecture, but no specification is made how the architectural rules are used to develop the architecture. More over there are no security measurements used for the system development. In proposed system, using architectural design rules architecture is developed as Model Driven Architecture. For this architecture security is introduces using patterns and it is developed using Agile Processing techniques.

Model Driven Architecture is a three layer process containing of

- CIM (Computation-Independent Model): As the name suggests it focus on the required functionalities of the system but not on the computation process that is needed. This is designed by system analysts and its main key elements are Use Cases. Here the analysis is done.
- PIM (Platform- Independent Model): The essential parts of the system and the essential behaviours of the system are focused but not about the platform to be used. For a CIM there is a PIM given.
- PSM (Platform-Specific Model): Using the transformations the behaviours are modelled in this layer using one specific platform. For a single PIM there can be multiple PSM's exists. The main modeling concept lies in this layer so there is a technique to design this layer.

Design of PSM

PSM is designed in three design approaches based on the levels of the system. Those are

- Architectural Design: The gross level designing is done3 i.e. at Entire system level.
- Mechanistic Design: For each requirement, use case collaborations are given and designing at this collaboration level is the mechanistic design.
- Detailed Design: As the name implies, individual class or subsystem or component design is explained in detailed design.

Transformations: There are transformations used between the PIM and PSM layers. These transformations are:

- Model Transformations: To transform model to code this is used. First mappings are done to transform from PIM to PSM.
- Meta model Transformations: Meta model is defined as model-on-model.
- Design Patterns Transformations: Using design pattern identification the mapping is done.

Design

To identify the design pattern that is appropriate to the system development, the concept of Design Motif Identification Multilayer approach (DEMIMA) is used as design phase in proposed model. Design pattern is defined as problem solving approaches for recurring, recursive problems. It is defined with four parts: Name, Purpose, Solution and Consequences (Pros and Cons). A design pattern will have motive, structure, implementation, sample code, applications, collaborations etc. to identify. DEMIMA follows a three layered approach to identify a pattern that is perfectly matching the requirement and this pattern is unique.

- In the First layer, the source code is taken and based in its structure is identified which will be a model of Unified Model Language (UML) language. Here, Program Model is defined as set of Entities (Classes and Interfaces) and as set of Elements (Attributes, Operations and Relations)
- In Second layer the maintainers choose a Design Pattern from structure whose motif is embedded by architecture. Here, idioms are defined as higher level of abstraction. Idioms are low-level patterns specific to some program languages and implementation of particular characteristics of classes and their relationships.
- In last layer, maintainers deduce intent and motivation of overall system based on chosen

design pattern. That is, describes model of design motif with same language used for idiom model.

Implementation

The development of architecture is based on agile process. The agile process is one of the software processing techniques containing the same phases as waterfall model (or) Rational Unified Process (RUP) (Inception, Elaboration, Construction, Transition) but implemented in iterative manner. The phases are:

1.1 AGILE ANALYSIS

Micro cycle (Program running in incremental model) requirements are analyzed in two activities: Prototype definition (template or rough definition for given requirement), this comes across CIM layer of MDA since it involves no computation process) and Object analysis (rough objects are identified from prototype and analysis is done, this comes under PIM layer of MDA since behavior of objects are known but not concentrating on their implementation)

1.2 AGILE DESIGN

The PSM layer of MDA is designed using agile design of three levels as Architectural Design, Mechanistic Design and Detailed Design.

1.3 AGILE TESTING

Test cases and test workflows are given as Unit Test (White Box), Integration test (Gray box) and Validations (Black box). Test Driven Development is used as agile methodology to process step.

In this proposed model, Security is provided to system at architectural level. This is because of the identification of proper design pattern or design motif for construction of MDA. Because of this identification there will be no confusion or overlap of design criteria for any requirement that comes to architecture as design rule.

When a comparison is made between existing and proposed systems, there were benefits with MDA as Portability (because of PIM), Reusability (because of PSM) and isolation from technology churn (moves or turns to latest technology easily). Initial security measures are defined. Since models are used to develop architecture, it is to understand and use by architects and developers. With design pattern based designing security is provided at architectural level only so if any requirement is tested at this level that is verified by its rules so saying a refinement to requirements. Therefore in future scope, security level must be increased as well as improvement to DEMIMA process is done since as of now it is able to identify few design patterns only.

1.4 METHODOLOGY FOR PREDECESSOR ACTIVITIES OF AGILE METHODOLOGY

Figure 4.1 provides Sequence diagram for Predecessor activities of Agile Methodology for Security. Important objects specifying significant tasks are: Pre spiral Plan, Stakeholder requirements, and Development Environment. Steps involved iteratively across various iterations are creating a Schedule, Creating a team model, planning for reuse, planning for Risk (dependability) reduction, Specifying logical architecture, Perform initial safety and reliability analysis, and finally link this report with requirements. After invoking of Stakeholder requirements phase the significant steps involved are: Define the product vision, Find and outline stakeholder requirements, Detail the stake holder requirements, Review stake holder requirements, and finally relate the requirements with development environment. Eventually significant steps in Development Environment phase are: Tailor the process, Installation of development tools, Configure the development tools, Initialize the development tools and Launch the development tools.

Appendix 1 figure 1 provides Sequence diagram for Predecessor activities of an Agile Methodology.

1.5 METHODOLOGY FOR AGILE IMPLEMENTATION

Figure 4.2 provides Sequence diagram for Agile Security Implementation. The significant objects or phases for Secure Agile Implementation are: Agile analysis has important steps like prototypes base line is figured out and its repestive objects and their internal interactions among other objects are analyzed. This is followed by phase Agile Design based on its obtained agile anaysis specification so that optimization of usage of design patterns is done. The steps involved in agile analysis are architectural rules can be applied for optimization at gross level. Mechanistic rules are applied for optimization at system level, detailed rules are applied for system at primitive level. Now this Agile design needs to be tested by phase Agile testing. Steps involved in Agile testing are unit testing, integration testing and validation testing. Appendix 1 figure 2 provides Sequence diagram for Predecessor activities of an Agile Methodology.

Design of Agile Methodologies for Security

Figure 3 provides Class diagram Design for MDA authentication using Executable UML. The significant classes here are: User and Administrator, Authenticator, Authorizer and Security. Initially for accessing information user needs to be authenticated by providing his login credentials like username and password. Authenticator authenticates the user by checking username with associated password and knows whether the user is the real claimant. Based on the check result authenticator allows the user for his available authorization. Authorizer checks for the type of user like administrator and their corresponding access rights. Authorizer has the privilege to restrict access to the information for any user based upon the situation. Username and password is encrypted so that other authorized users who had earlier authorized may not recognize this user name and password. It is the responsibility of the security class which generates a

key and chooses an appropriate algorithm for encrypting the data.

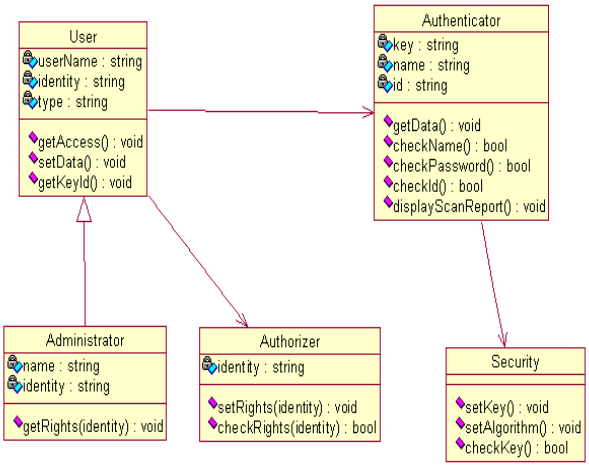


Fig Class diagram Design for MDA authentication using Executable UML

Figure 4 provides Class Diagram Design of Agile Methodologies with Security Activities. The important classes are Algorithm, Integration, SecurityAgileMethods, SecurityActivity, ART (Agility Reduction Tolerance) etc. The sequence of steps involved are extraction of security activity, calculate agility degree, integration of agility and security, activity process algorithm, ART. Recursively and iteratively proposed agile security model is applied as an algorithm as how much integration of security and agility done.

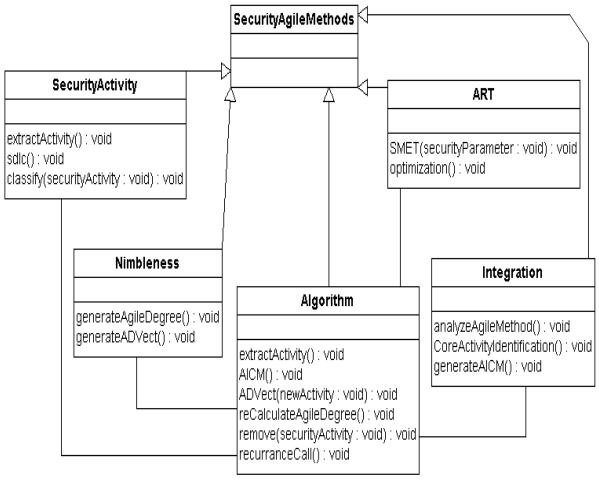


Fig 4 Class Diagram Design of Agile Methodologies with Security Activities

2 AGILE SECURITY PATTERNS

The proposed agile security model is extended by proposing the following agile security patterns like Dependency Inversion Principle, Interface Segregation Principle, Separation through Delegation, and Separation through Multiple Inheritances.

2.1 DEPENDENCY-INVERSION PRINCIPLE

This principle can be used when one class wants to send a message to another class. To illustrate this principle, as an example considered the figure 5 (Agile Security Design for Dependency Inversion) using classes like Button, ButtonServer and lamp. Here Button class and lamp class has this policy. Button class after receiving a message called poll can sense an external event. This affects the lamp class. Lamp class will respond to messages turn on and turn off appropriately. In a nutshell button class gets a message called poll and sends turn on or turn off message to lamp class. Initially there is a clear dependency between button class on lamp class. This implies that any changes to lamp affect the button. This in-turn is a violation to dependency inversion principle. Button now is holding a relationship called association with button server class, which is an interface so that the button class can turn on or turn off the lamp class. Button server interface is implemented by lamp. Now the scenario is dependency is done by the lamp instead of it being depended. This eventually gives flexibility for button for controlling any class which implements interface button server.

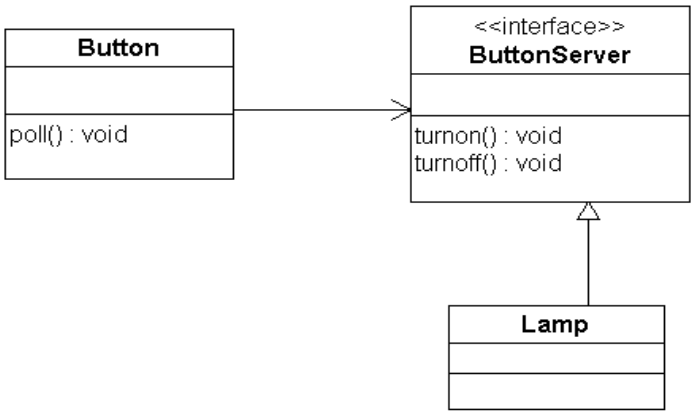


Fig 5: Agile Security Design for Dependency Inversion

2.2 INTERFACE SEGREGATION PRINCIPLE

To illustrate this principle, as an example considered the figure 6 which provides Agile Security Design for Interface Pollution. The important classes here are Timer, TimerClient (Interface), Door, TimedDoor. Door Class can be locked or unlocked there by giving information whether it is closed or opened respectively. Door is implemented as an interface, so that it supports various implementations. TimedDoor is a specialization of door which raises an alarm if the door is kept open for a long period of time, hence TimedDoor can establish a message

communication with Timer. For getting the duration of time we require to register with timer, open door forcibly, so now TimedDoor should inherit from TimerClient although its two step process. So now TimerClient receives a message called timeout. This clearly shows that door clearly depends on TimerClient. This approach reduces complexity and redundancy.

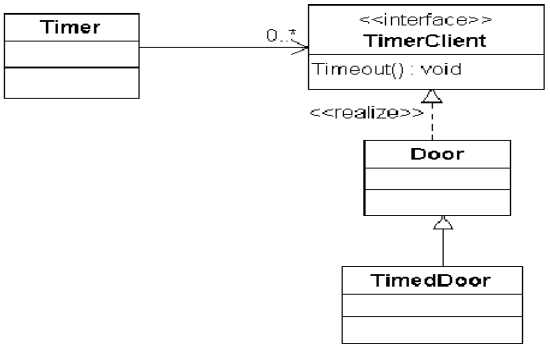


Fig 6 Agile Security Design for Interface Pollution

2.3 SEPARATION THROUGH DELEGATION

To illustrate this principle, as an example considered the figure 7 which provides Agile Security Design for Separation through delegation. This is an enhancement solution to the earlier interface pollution principle. TimerClient can create an object and there by delegates it to the TimedDoor. Whenever TimeoutRequest has to be registered a DoorTimerAdapter is created by TimedDoor and it is registered by the Timer. This solution is very general purpose but because of delegation it requires small quantity of run time dynamic memory.

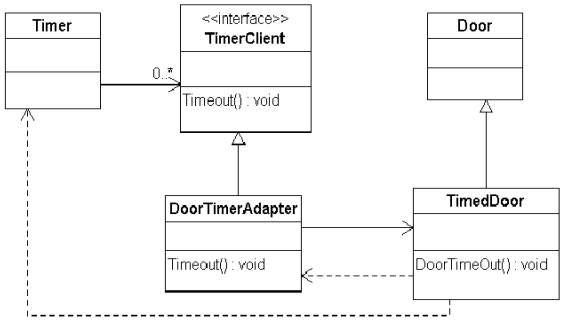
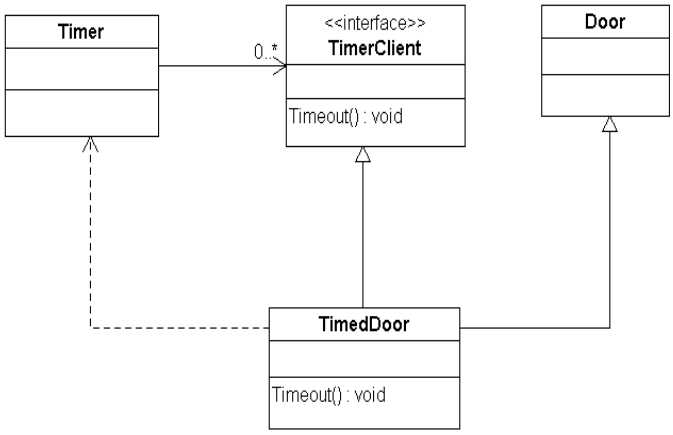


Fig 7 Agile Security Design for Separation through delegation

2.4 SEPARATION THROUGH MULTIPLE INHERITANCES

To illustrate this principle, as an example considered the figure 8 which provides Agile Security Design for separation through Multiple Inheritance. TimedDoor gets inheritance by both TimerClient and TimedDoor. Here parent class does not depend on child classes. Using separate interfaces and multiple inheritance this solution provides usage of a specific object using separate interfaces.



. Fig 8 Agile Security Design for separation through Multiple Inheritance

3 WEB 2.0 SERVICES SECURITY DESIGN FOR NGSWEA USING AGILE MODELING

This implementation is for designing secured web 2.0 AJAX Web Services. For web 2.0 AJAX architecture access to the web server is minimized because the AJAX engine in the web 2.0 supporting browser will handle the request to and response from web server. Regarding security issues first the web application needs to be secured for web services interface design. Next restriction of access to web services for specific parties needs to be designed. Figure 9 provides Initial Page of the mygoogle_search.html. This implementation is about creating a search control using web 2.0 third party services. Java Script provides the browser security with usage of features from Google Search API. Initially OnLoad function loads the Google Search. To start with a variable is assigned to search control.

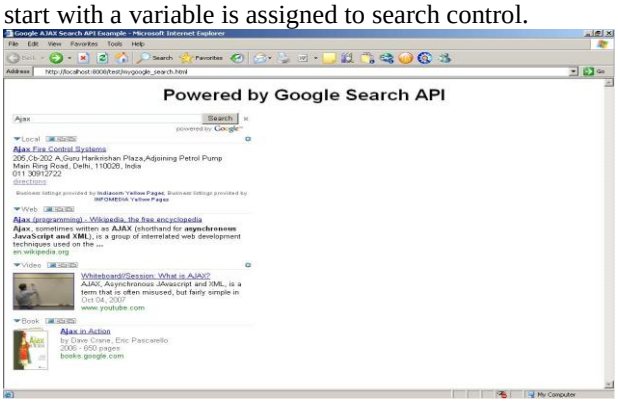


Fig 9 Initial Page of the mygoogle_search.html

Figure 10 provides Encryption of data using JavaScript (MD5 algorithm). This implementation creates MD5 Hashing and Encryption. This is useful for protecting the data and restricting unauthorized access.

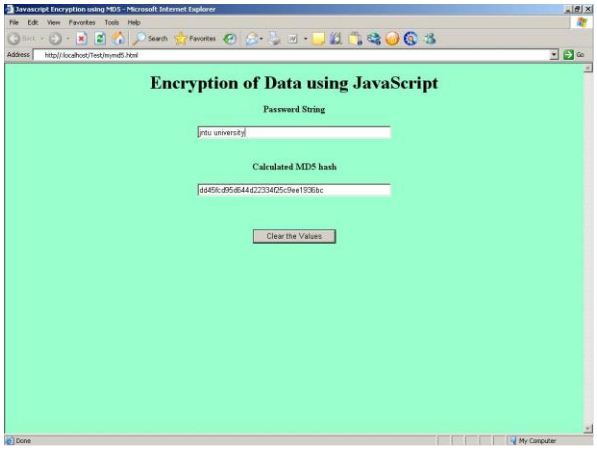


Fig 10.Encryption of data using JavaScript (MD5 algorithm)

Figure 11 provides design of Web 2.0 services SSL using Agile Modeling. The modules used for Web 2.0 services authentication using Visual Studio are Add Security Policy, include customized authentication (using X.539 certificates), create a client etc. Secure Socket layer (SSL) provides a mechanism for sending SOAP messages through secure tunnel. WS-Security provides SOAP message protection. The advantages here are simple agile security design and ease of reusability of code, correctness in operation and also provide security against sniffing and confidentiality. Testing is done for correctness of the proposed agile security model. The layered approach components used at message layer are WS-Security, SOAP, and SSL/TLS. Implementation is done as additional four classes: Factory classes (SSLServerSocketFactory and SSLFactory), Socket classes (SSLServerSocket and SSLSocket). Based on SSL client gets and sends a chain of certificates with a signature. Client needs to do additional configuration to the specified four classes. Figure 12 provides execution screenshot for the key generation of Web 2.0 services SSL implementation. Figure 13 provides the package diagram of Web 2.0 services SSL. Figure 14 provides execution screen shot of this application where Users Gives Username and password for their authentication. Figure 15 shows execution screen shot of, on entering valid username and password the company quote will be displayed. Figure 16 shows execution screenshot of, on entering wrong username or password the error is displayed violating authentication.

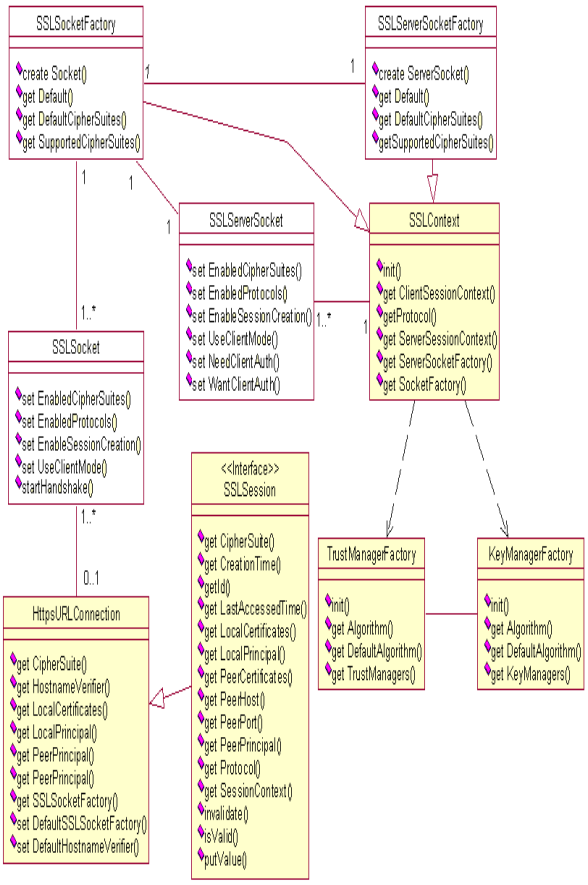


Fig 11 Class Diagram of Web 2.0 services SSL by Agile Modeling

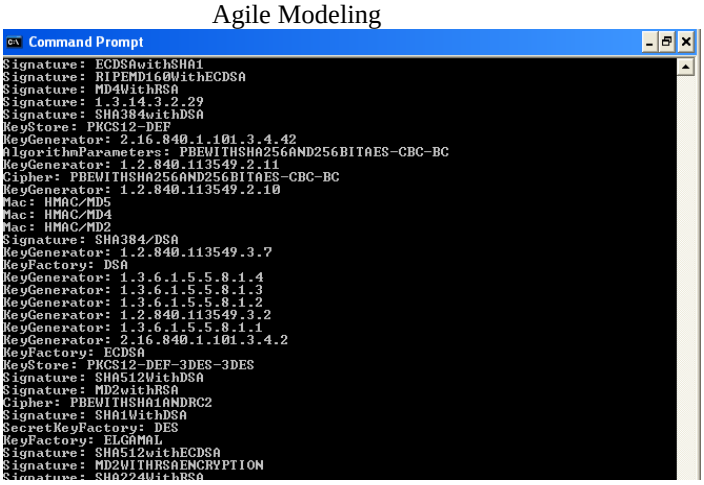


Fig 12: Key generation of Web 2.0 Services SSL application

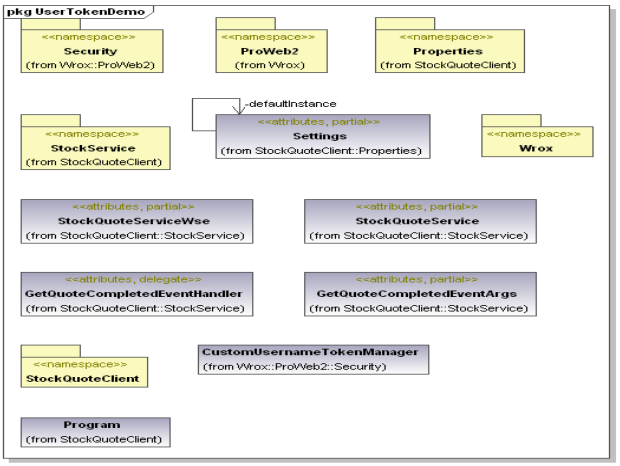


Fig 13: Package diagram of the web 2.0 services authentication by Agile Modeling

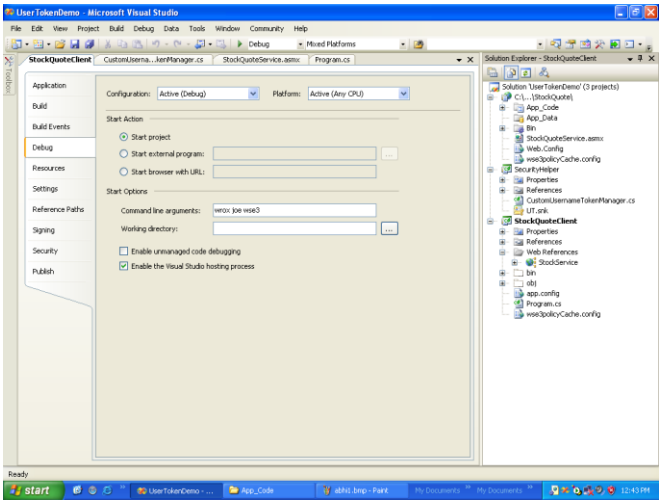


Fig 14: Users Giving Username and password

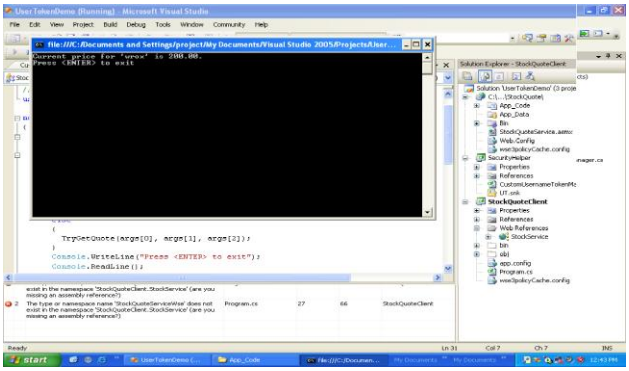


Fig 15: On entering valid username and password the company quote will be displayed

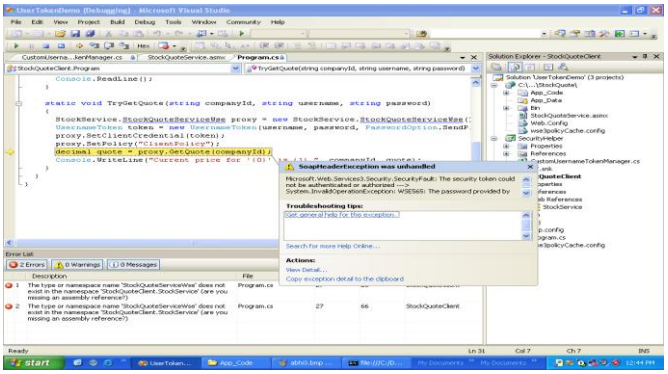


Fig 16 on entering wrong username or password the error is displayed violating authentication

4 SUMMARY AND CONCLUSION

In this paper, based on the earlier paper of theoretical analysis, an agile security model is proposed which is a layered one having phases like agile analysis, agile design and agile testing with propose of four security patterns. This model is initially validated based on earlier paper basic web services secure agile design for Web 2.0 services authentication using proposed agile security model. Next paper provides agile security privacy requirements, web 2.0 services privacy design and application of proposed agile security model for secure web engineering.

REFERENCES

1. 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, PP. 101-114.
2. Orit Hazzan, Yeal Dubinsky, "Agile Software Engineering", Springer London, ISBN 978-1-84800-198-5, DOI 10.1007/978-1-84800-199-2, 2008, PP. 6.
3. Tore Dyba, Torgeir Dingsoyr, "Empirical studies of agile software development: A systematic review", Elsevier, Science Direct, 2008, PP. 1-27.
4. Jeff Younker,"Foundations of Agile Python Development", Apress Publishers, 2008, PP. 12
5. Barbara Russo, Maro Scotto, Alberto Silliti, "Agile Technologies in Open Source Development" IGI Global publishers USA 2010, PP. 11- 24
6. Staurt Wray,"How Pair Programming Really Works", IEEE Software, January/February 2010 PP. 50-55.
7. Jo E.Hannay, Erik Arisholm, Harald Engvik, and Dag I.K.Sjoberg, "Effects of Personality on Pair Programming", IEEE Transactions on Software Engineering, Vol 36, No.1, Jan/Feb 2010, PP. 61 – 80.
8. Laurie Williams "What Agile Teams Think of Agile Principles" Communications of the ACM Vol.55 No:4, April,2012. PP. 71-76.
9. Bartlomies Gamel and Iwona Skalne," Model driven architectures and classification of business rules modeling languages", IEEE International conference, ISBN 978-83-608-51-4, PP. 949 – 953.
10. Emrah Asan, "Agile and Collaborative Systems Engineering", Ph.D. Thesis, School of Informatics, Middle East Technical University, January 2014, PP. 1-126.