ISSN:0975-9646

Software Architectures Design Patterns Mining for Security Engineering

A.V.Krishna Prasad[1], Dr.S.Ramakrishna[2]

[1] Department of MCA MIPGS Hyderabad A.P. India
[2] Department of Computer Science S.V.U Campus Tirupathi A.P. India

Abstract— **Data Mining for Software Engineering involves Architectural mining intelligence operations, which are knowledge gathering procedures. These intelligence operations go beyond basic data collection (assembling uncorrelated information) to the point of collecting fully assimilated practical knowledge – knowledge that affects important architectural decisions. Gathering knowledge is an essential element of being a software architect. Ordinary knowledge gathering for a project requires the capture of end user requirements and perhaps the evaluation of some commercial product. Proper architectural practices go well beyond these project centric traditions, which are isolationist when used exclusively. Instead, these practices should be augmented with some additional procedures that have been found to be effective, including architectural mining, architectural iteration, and architectural judgment. Architectural mining is a practice that breaches classic intelligence barriers between projects. IT can have an intelligence scope as large as entire industry or as small as one company's systems. Architectural mining is a conscious effort to eliminate the ignorance of silence that characterizes many system developments. Architectural iteration is a process focused upon a single architecture or specification. It tracks the architecture through its development and life cycle, improving quality through intelligence gathering on each project. Architectural judgment is a process of decision making, based upon intelligence gathering. Making quality decisions is at the very heart of being an architect. In today's changing world of technology, it is increasingly difficult to make long-lasting judgments without a systematic process. In this paper, we want to discuss about Architectural Mining (Design Pattern Mining) strategies for Software Security Architectures, which are validated with appropriate SAO Web Services case studies.**

Index Terms—Architecture Mining, Design Pattern Mining, Security Architectures

I. INTRODUCTION TO MINING SOFTWARE ARCHITECTURES

Software Architectural mining for intelligence operations are knowledge gathering procedures, intelligence operations go beyond basic data collection (assembling uncorrelated information) to the point of collecting fully assimilated practical knowledge – knowledge that affects important architectural decisions.

Gathering knowledge is an essential element of being a software architect. Ordinary knowledge gathering for a project requires the capture of end user requirements and perhaps the evaluation of some commercial product. Proper architectural practices go well beyond these project centric traditions, which are isolationist when used exclusively.

Instead, these practices should be augmented with some additional procedures that have been found to be effective, including architectural mining, architectural iteration, and architectural judgment.

Architectural mining is a practice that breaches classic intelligence barriers between projects. IT can have an intelligence scope as large as entire industry or as small as one company's systems. Architectural mining is a conscious effort to eliminate the ignorance of silence that characterizes many system developments.

Architectural iteration is a process focused upon a single architecture or specification. It tracks the architecture through its development and life cycle, improving quality through intelligence gathering on each project.

Architectural judgment is a process of decision making, based upon intelligence gathering. Making quality decisions is at the very heart of being an architect. In today's changing world of technology, it is increasingly difficult to make long-lasting judgments without a systematic process.

A. *.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 modelling 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 modelling level which ensures that "security at the system architecture stage will improve the requirements for that system".

B. Research Problem Definition

Mining software engineering secure software architectures using Model-driven architectures and Agile modeling for Requirements engineering, this is validated for Web Services case study and its related challenges. Our Strategies include text mining for source code and Design Patterns Mining and Graph Mining for architecture mining.

Our Research Motivation is Integrating Security and Software Engineering using mining strategies. This approach involves: Evaluating the different Software Engineering Paradigms with respect to their appropriateness to integrate security; Developing new techniques, methods, processes that consider security as part of the software development life cycle; Tool Support/define a Suitable Exemplar; Transfer of security knowledge / transit research results to mainstream system development.

Key *research questions* addressed are:

Security Analysis and Design issue:

1. "How well the system authenticates the users and protects the application and data elements access control?"
2. "How Model-driven architectures and Agile Modeling are useful for security architectures?"

Data Mining:

What are the challenges in applying data mining techniques to software engineering data?

Which Data Mining techniques are most suitable for mining design patterns?

How can agile methodology be used to generate effective security requirements? In what ways can these agile methods change the development of security requirements?

Research Methodology includes

Through extensive literature survey on data mining for software engineering, related work pertaining to Source Code, Architecture and Architecture Patterns in Security (Web Services case study) had been studied with a motivation for Good Architectural Design metrics. Drawbacks in the existing system pertaining to authorization, authentication, role based access control for Security Architecture (Web Services) are studied. Other models of software development like prototyping, formal transformations; extreme programming had been explored for secure programming practices. The present and future industry needs related to security architectures had been studied. Text mining and design pattern graph mining are applied for secure Source code and security architectures respectively. A case study on Web Services Security Architectures had been carried out to justify our approach.

C. Objectives of the Research Work

The objective of the research work to propose "Mining Security Architectures using Model-Driven & Agile Modeling for Requirements Engineering – Web Services Case Study" is to:

Overview of Research

Software Security Engineering is about building systems to remain dependable in the face of malice, error, or mischance. Most attacks to software systems are based on vulnerabilities caused by poorly designed and developed software. The enforcement of Security at the Design phase can reduce the cost and effort associated with the introduction of security during implementation. *Security Architectures* are architectures which enable implementations that are resilient to an appropriate and broad-based spectrum of threats. Issues are: Complexity is the source of security holes; Security is the matter of the weakest link. Tradeoffs need to be based for complexity vs. protection, performance, usability and flexibility.

Security analysis and design issue: "How well the system authenticates the users and protects the application and data elements?"

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 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 deign decisions consists of architectural design rules. In a 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 such as eXtreme Programming (XP), Scrum, Feature Driven Development (FDD) and Test Driven Development (TDD) etc. Agile processing includes the phases like 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".

The most important ten things a software architect does are; inquire, integrate, analyze, conceptualize, abstract, visualize, formalize, communicate, enable and assist. Software architecture captures the broad-stroke strategic design decisions of a particular system. Architectural design fits into the overall development process with security encompassing all the phases. Successful software and software systems are directly attribute to elegant and efficient modeling and design. Models let users, architects and developers create readily understandable representations of complex object-oriented systems, before development begins. Over the past few decades the IT industry has developed a range of approaches for modeling and documentation. Structured approaches, Data-driven approaches, Process-driven approaches, Object-oriented approaches, Unified approaches, Domain-specific approaches, Serial approaches, and agile approaches. Web

services security architectures includes core standards, implementations on various technologies and platforms.

Financially driven attackers and high profile breaches have changed the economics of security. Software developers and designers needs to rethink the motivations of attackers the new attacker economy had given a growing stolen identity information trade the raise of organized electronic crime. "*Hackernomics"* is a social science concerned with description and analysis of attacker motivations, economics and businesses.

II. Mining Design Patterns for Security Architectures

Design patterns are micro-architectures that have proved to be reliable, easy to implement and robust. There is a need in science and industry for recognizing these patterns. We present a new method for discovering design patterns in the source code. This method provides a precise specification of how the patterns work by describing basic structural information like inheritance, composition, aggregation and association, and as an indispensable part, by defining call delegation, object creation and operation overriding. We introduce a new XML–based language, the Design Pattern Mark up Language(DPML),which provides an easy way for The users to modify pattern descriptions to suit their needs, or even to define their own patterns or just classes in certain relations they wish to find.

Graph Mining:

Data mining is the extraction of novel and useful knowledge from data. A graph is a set of nodes and links (or vertices and edges), where the nodes AND/OR links can have arbitrary labels, and the links can be directed or undirected (implying an ordered or unordered relation). Therefore, mining graph data, sometimes called graph-based data mining, is the extraction of novel and useful knowledge from a graph representation of data. In general, the data can take many forms from a single, time-varying real number to a complex interconnection of entities and relationships. While graphs can represent this entire spectrum of data, they are typically used only when relationships are crucial to the domain. The most natural form of knowledge that can be extracted from graphs is also a graph. Therefore, the knowledge sometimes referred to as patterns, mined from the data is typically expressed as graphs, which may be subgraphs of the graphical data, or more abstract expressions of the trends in the data. Graph visualization is the rendering of the nodes, links, and labels of a graph in a way that promotes easier understanding by humans of the concepts represented by the graph.

A. Architecture and Design Pattern Discovery Techniques – A Review

Architecture and design patterns, as demonstrated solutions to recurring problems, have proved practically important and useful in the process of software development. They have been extensively applied in industry. Discovering the instances of architecture and design patterns from the source code of software systems can assist the understanding of the systems and the process of re-engineering. More importantly, it also helps to trace back to the original architecture and design decisions, which are typically missing for legacy systems. This approach presents a review on current techniques and tools for discovering architecture and design patterns from object-oriented systems. We classify different approaches and analyze their results. We also discuss the disparity of the discovery results from different approaches and analyze possible reasons with some insight.

B. Software Metrics by Architectural Pattern Mining

A software architecture is the key artifact in software design, describing the main elements of a software system and their interrelationships. We present a method for automatically analyzing the quality of an architecture by searching for architectural and design patterns from it. In addition to approximating the quality of the design, the extracted patterns can also be used for predicting the quality of the actual system. The method is demonstrated by an industrial case over a complex telephone exchange software.

C. Mining Design Patterns form (C++) Source Code

Design patterns are micro architectures that have proved to be reliable, easy to implement and robust. There is a need in science and industry for recognizing these patterns. We present a new method for discovering design patterns in the source code. This method provides a precise specification of how the patterns work by describing basic structural information like inheritance, composition, aggregation and association, and as an indispensable part, by defining call delegation, object creation and operation overriding. We introduce a new XML–based language, the Design Pattern Mark up Language(DPML),which provides an easy way for the users to modify pattern descriptions to suit their needs, or even to define their own patterns or just classes in certain relations they wish to find. We tested our method on four open-source systems, and found it effective in discovering design pattern instances.

D. A review of Design Pattern Mining Technique

The quality of a software system highly depends on its architectural design. High quality software systems typically apply expert design experience which has been captured as design patterns. As demonstrated solutions to recurring problems, design patterns help to reuse expert experience in software system design. They have been extensively applied in industry. Mining the instances of design patterns from the source code of software systems can assist the understanding of the systems and the process of re-engineering them. More importantly, it also helps to trace back to the original design decisions, which are typically missing in legacy systems. This approach presents a review on current techniques and tools for mining design patterns from source code or design of software systems. We classify different approaches and analyze their results in a comparative study. We also examine the disparity of the discovery results from different approaches and analyze possible reasons with some insight.

E. Architectural Risk Analysis of Software Systems based on Security Patterns

The importance of software security has been profound, since most attacks to software systems are based on vulnerabilities caused by poorly designed and developed software. Further more, the enforcement of security in software systems at the design phase can reduce the high cost and effort associated with the introduction of security during implementation. For this purpose, security patterns that offer security at the architectural level have been proposed in analogy to the well-known design patterns. The main goal of this approach is to perform risk analysis of software systems based on the security patterns that they contain. The first step is to determine to what extent specific security patterns shield from known attacks. This information is fed to a mathematical model based on the fuzzy-set theory and fuzzy fault trees in order to compute the risk for each category of attacks. The whole process has been automated using a methodology that extracts the risk of a software system by reading the class diagram of the system under study.

F. Design Pattern detection using Similarity Scoring

The identification of design patterns as part of the reengineering process can convey important information to the designer. However, existing pattern detection methodologies generally have problems in dealing with one or more of the following issues: Identification of modified pattern versions, search space explosion for large systems and extensibility to novel patterns. In this appraoch, a design pattern detection methodology is proposed that is based on similarity scoring between graph vertices. Due to the nature of the underlying graph algorithm, this approach has the ability to also recognize patterns that a remodified from their standard representation. More over, the approach exploits the fact that patterns reside in one or more inheritance hierarchies, reducing the size of the graphs to which the algorithm is applied. Finally, the algorithm does not rely on any pattern-specific heuristic, facilitating the extension to novel design structures. Evaluation on three open-source projects demonstrated the accuracy and the efficiency of the proposed method.

III. IMPLEMENTATIONS AND VALIDATIONS

A. Design and Mining of Web Services Security Patterns

Service-Oriented Architectures (SOA) represents a new evolving model for building distributed applications. Services are distributed components that provide well-defines interfaces that process and deliver XML messages. A service-based approach makes sense for building solutions that cross organizational, departmental, and corporate domain boundaries. A business with multiple systems and applications on different platforms can use SOA to build a loosely coupled integration solution that implements unified workflows. Security in an SOA environment involves verifying several elements and maintaining confidence as the environment evolves. Organizations deploying SOA implementations should identify practical strategies for security verification of individual elements, but should be aware that establishing the security characteristics of composites and applications using services is an active research. Organizations should also identify the deployment strategies for the SOA infrastructure, services, composites, and applications because different deployment strategies can entail different security verification practices. Finally, all elements should be verified in their operational contexts.

A. Web 2.0 Services Security Mining

“If we think implementing security is expensive, see how much a breach it will cost us”. The web services standards are a group of agreements designed to facilitate interaction and provide common protocols in all areas of web services. They are produced by the OASIS organization, www.oasis-open.org, which has a large number of participants including many of the large software companies such as IBM and MICROSOFT.
In this mini project case study, we take a look at WS-security agreements and use the Microsoft implementation, known as WSE 3.0, in conjunction with visual studio 2005 to produce a web service that requires authentication. There are a number of ways to secure a web service. A simple way might be to use SSL together with client side certificates to ensure that the potential user is some one allowed to call the services method. The problem with this approach would be to pass the user credentials as part of the method call. This can be overcome by using SOAP protocol along with Secure Socket Layer (SSL). In this mini project we want to implement web 2.0 services security using visual studio 2005/2008/2010. Refer to Figure 1, Figure 2 and Figure 3
which consists of Class diagram, sequence diagram and execution screen shot of the application respectively.

Figure 1. Class Diagram of the application

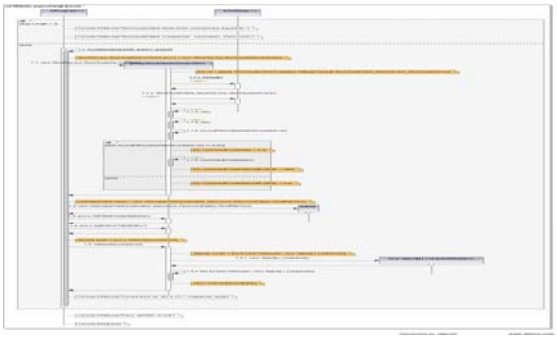

Figure 2. Sequence Diagram of the application

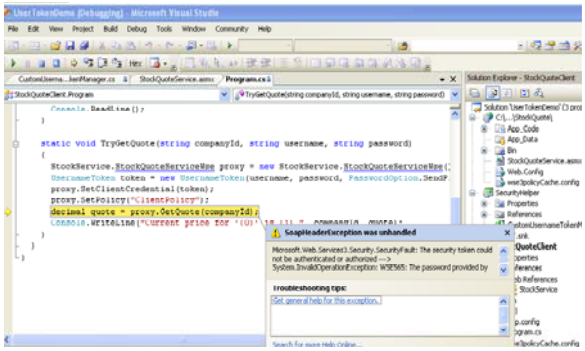

Figure 3. Execution screen shot of the application

B. *SSL Crypto Implementation for Web Services Mining*

The web is everywhere. "Web services" is an effort to build a distributed computing platform for the web. One problem when we administer a network is securing the data across the wide web. While many of web services have been met wit the existing standards, there are still a number of challenges that have yet to be considered.

In this mini project case study, we take a look at web services security architecture's modification with the existing standard in order to overcome with the authentication attack. Authentication process deals with the SSL/TLS layer. In this mini project we want to implement with the web 2.0 services security in the SSL version 3.0(TLS) using java as technology. Initially, JCA and JCE will be studied. Next the provider bouncy castle is dealt with. Later this Web Services will be secured by adding policy, custom authentication, certificate and key generation and storage etc. Refer to Figure 4, Figure 5 and Figure 6 which consists of Class diagram, sequence diagram and execution screen shot of the application respectively.

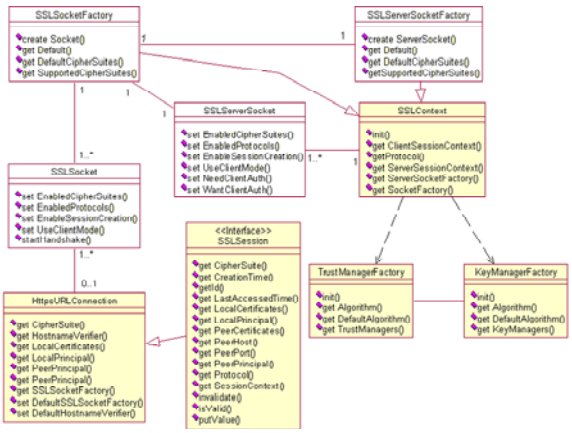

Figure 4. Class Diagram of the application

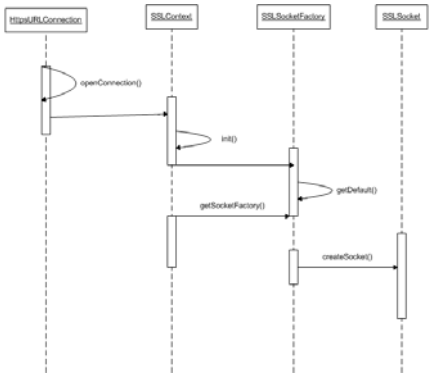

Figure 5. Sequence Diagram of the Application

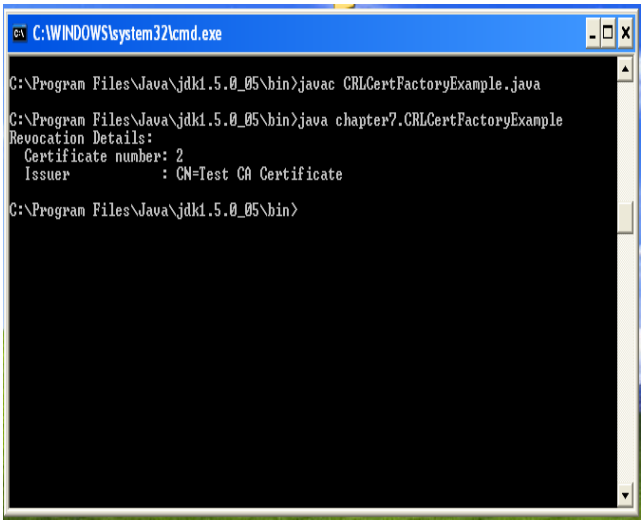

Figure 6. Execution Screen Shot of the application

IV. Conclusions and Suture Work

In this paper, we proposed strategies for Software Architectures Mining for Software Security Engineering, with SOA Web Services Case Study.

Further Work includes: In modern integrated software engineering environments, software engineers must be able to collect and mine software engineering data on the fly to provide rapid just-in-time feed back. SE researchers usually conduct offline mining of data already collected and stored. Stream data mining algorithms and tools could be adopted or developed to satisfy such challenging mining requirements. For Web Services Security Architecture Mining, we can extend this work for (Spatial) Virtualization and Cloud Computing.

V. 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

VI. REFERENCES

[1]. Tao Xie, Suresh Thummalapenta, David lo, Chao Liu, ”Data Mining for Software Engineering”, IEEE Computer, Auguat 2009, pp. 55-62.
[2]. Tao Xie, Jain Pei, Ahmed E Hassen, “Mining Software Engineering Data”, IEEE 29 th International Conference on Software Engineering ICSE 07.
[3]. Gang Kou Yipeng, “ A Standard for Data Mining based Software Debugging”, IEEE 4 th International Conference on Networked Computing and advanced Information Management, pp. 149 – 152.
[4]. Ray-Yaung Chang, Andy Podgurski, Jiong Yang, “Discovering Neglected Condition in Software by Mining Dependency Graphs”, ”, IEEE Transactions on Software Engineering, Vol. 34, No. 5, September/October 2008, pp. 579-596.
[5]. Gunnar Peterson, “Security Architecture Blueprint”, Arctec Group LLC, 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]. Mouratidis and Giorgini, *Integrating Security and Software Engineering: Advances and Future Vision.* Idea Group Publishing Inc., 2007.
[8]. John Paul Mueller, *Mining Google Web Services – Building Applications with the Google API* SYBEX Publishing Inc., 2004.
[9]. Hossein keramati, Seyed-Hassan Mirian-Hosseinabadi, “Integrating Software Development Security Activities with Agile Methodologies”, 2008, IEEE.
[10]. 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.
[11].. Yann-Gael Gueheneuc, Giuliano Antoniol, “DeMIMA: A Multilayered Approach for Design Pattern Identification”, 2008, IEEE Transactions on Software Engineering, vol. 34, no. 5.
[12]. 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.
[13]. Erich Gamma, “Design Patterns”.
[14]. M. Siponen, R. Baserville, T. Kuivalainen, “Extending Security in Agile Software Development Methods”, pp 143-157.
[15]. Johan Peeters, “Agile Security Requirements Engineering”.
[16]. Alexander Chatzigeorgiou, Nikolaos Tsantalis, George Stephanides, "Application of Graph Theory to OO Software Engineering"
[17]. Ladislav Burita–Vojtech Ondryhal, "Extending Uml For Modelling
Of Data Mining Cases"
[18]. Jing Dong, Yajing Zhao, Tu Peng , "A Review of Design Pattern Mining Techniques ".
[19]. Jing Dong, Yajing Zhao, Tu Peng , " Architecture and Design Pattern Discovery Techniques – A Review”
[20]. Zsolt Balanyi, Rudolf Ferenc, "Mining Design Patterns from C++ Source Code"
[21]. Jing Dong, Yajing Zhao, Tu Peng "A Review of Design Pattern Mining Techniques "
[22]. Nikolaos T santalis, Alexander Chatzi , "Design Pattern Detection Using Similarity Scoring"