IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-04 E-ISSN: 2347-2693

Scenario-based Evaluation of Software Architecture Styles from the Security Viewpoint

Gholamreza Shahmohammadi

Department of Information Technology, Olum Entezami Amin University, Iran

www.ijcseonline.org

Received: Mar/16/2016 Revised: Mar/28/2016 Accepted: Apr/19/2016 Published: Apr/30/ 2016

Abstract— **By increasing the use of distributed systems and increasing software attacks, software security is considered very important and treated as an active research area. Security is usually taken into account after design and implementation of the system, whereas like other quality attributes, it must be considered from the beginning of the process of building software, such as architectural design. Considering 1) the long-term effects of design stage decisions on final software product, 2) one of the important design decisions, is selection of suitable software architecture style, and 3) the quantitative impact of software architecture style on quality attributes, especially security, has not been investigated, the aim of this research is quantification of the impact of architectural styles on security quality attribute. This study aims at evaluating the software architectural styles from the viewpoint of the security quality attribute based on scenario-based evaluation method. In this study, by presenting security scenarios, the architectural styles are evaluated from the perspective of security. Then architectural styles are ranked based on the results of the evaluation and importance of scenarios using Analytical Hierarchy Process, in terms of supporting software security. The most important contribution of this paper is to propose an approach to select the software architecture style in which security attribute plays a major role.**

Keywords- Security; Scenario-based Evaluation; Software Architecture Styles

I. INTRODUCTION

Software security is one of the software quality attributes that due to the expanded distribution in the systems and networks and potential attacks to the system, is very important. Because of the importance of software security in systems, especially web systems, we need to consider security aspects from the very beginning in the requirements engineering process. As our modern society is critically dependent on software systems, the importance of software security is constantly growing [1, 2]. Software vulnerabilities, arising from deficiencies in the design or implementation of the software (e.g., due to the increase in complexity) are one of the main reasons for security incidents [2]. Security is usually considered after design and implementation of the system, whereas like other quality attributes, it must be considered from the beginning of the process of building software, such as architectural design [3]. Decisions made in the design phase have profound impact on the final software product. One of the most important decisions of design stage is the selection of software architecture style. Functionality or software tasks may be achieved using any of a number of possible structures. Therefore, software architecture styles (SASs) are selected based on amount of their support from quality attributes [4]. Software architecture is the very first step in the software lifecycle in which the non-functional requirements (NFR) are addressed [5]. For this reason, many researchers have been done in this regard [6,7,8,9,10,11]. Regarding the expansion of disruption in systems and the widespread use of distributed software systems, one of the main considerations in choosing the software architecture style, is the amount of support from software security. Since quantitative impacts of SASs on quality attributes have not been studied yet [12], their applications are not systematic [13]. In other words, present use of styles in design is based on intuition of software developers. One of the methods of software architecture evaluation is scenario-based approach. Scenario-based techniques provide one of the most general and effective approaches for evaluating software architecture. In this method, architecture is evaluated based on the scenarios [5]. Quality attribute scenarios or briefly scenarios are a means to describe the quality attributes [5]. Scenario is sequence of actions that occur as a result of interaction with the software or in other words, is an event that may occur on the software lifecycle. With the occurrence of this event, the software should be consistent with it.

In this paper, software architecture styles are evaluated from the viewpoint of security quality attribute based on security scenarios. In other words, we use from scenario-based evaluation method, to evaluate software architecture styles. By the available information from software architecture styles, there is possibility of this evaluation. After evaluating architectural styles based on scenarios, software architecture

*Corresponding Author: Gholamreza Shahmohammadi
E-mail: shah_mohammadi@yahoo.co.uk
Department of Information Technology, Olum Entezami Amin University

styles are ranked according to the priority of scenarios, evaluation results and using analytic hierarchy process (AHP). Author is evaluated software architecture style from the viewpoint of reliability and maintainability [14, 15]. This research is a movement to quantification of the impact of software architecture styles on the quality attributes.

This paper is structured as follows: in Section 2, software architecture styles are discussed. In Section 3 the scenario-based evaluation and Section 4 scenario-based evaluation of architectural styles from the viewpoint of security attributes are offered. Section 5 presents the conclusions.

II. SOFTWARE ARCHITECTURE STYLES

Software architecture styles present models for solving the problem of designing the software architecture in a way that each model describes its components, responsibilities of the components and the way they cooperate [16]. Shaw and her colleague [17] introduce seven styles. Buschmann et al [16] have also described the pattern in different levels. In the following eight architecture styles are briefly introduced.

Repository style (RPS). In this style, there are two types of components: a central storage and a set of components that store, retrieve and update information on the repository [17].

Blackboard style (BKB). The components of this style are Blackboard, experts (knowledge resources), and the control. The control component in a loop, checks the blackboard status, evaluates knowledge resource, and activates one of them for the execution [17].

Pipe and filter (P/F). This style is composed of a set of computational components. Each component acts as a filter and has a number of inputs and outputs. The output of each component is the input of the next component [17].

Layered style (LYD). In this style, the emphasis is on different abstraction level in the software. The layered style organized hierarchically. Each layer provides a service for its above layer and uses its lower layer [17].

Implicit Invocation (I/I). Implicit invocation style is an event-driven style based on broadcast concepts and announces the occurrence of the event instead of directly invoking a function. Interested components relate a function to an event. With the occurrence of an event, software invokes all registered functions [17]. Components of this style are: (1) event publishers, (2) components that are interested in events, and (3) dispatcher that invokes interested components in response to an event occurrence.

Client/Server(C/S). The components of this style are clients and servers. Clients should be aware of the name and services presented by servers [17].

Broker style (BRK). Client, servers, broker, client side proxy and server side proxy are the components of this style. Broker is responsible for coordinating the relationship between clients and servers. Servers register themselves with the broker, and make their services available to clients through method interfaces. Clients access services of servers by sending requests via the broker. Locating appropriate servers, forwarding the request to it and return the results to the client are the responsibilities of the broker [16].

Object-Oriented (OO). In this style, data presentations and the related operations encapsulated in an object. Objects are the components of this style and they interact through invoking the functions [5].

III. SCENARIO-BASED EVALUATION METHOD

In this method, software architecture is evaluated based on a number of scenarios. Scenario, a means for assessing the software quality attributes [5]. The scenario is a sequence of actions that occur as a result of interaction with software or in other words an event that may occur in the software life cycle. In the case of occurrence of this event, the software must be compatible with it. Evaluation of impact of Scenarios will show the software flexibility. Scenario-based evaluation methods such as SAAM and ATAM methods have been proposed.

IV. SCENARIO-BASED EVALUATION OF ARCHITECTURAL STYLES FROM SECURITY VIEWPOINT

In this section, the scenarios will be used to evaluate the architectural styles instead of software architecture. In this method, first security scenarios are prepared for evaluating architectural styles and then each of these scenarios are applied to architectural styles. Then style resistance against security scenarios is evaluated. Then final ranks of architectural styles are determined based on results of the evaluation and the importance of scenarios, using AHP method. Various stages of this method are shown in *"Fig. 1"*.

A. Providing sufficient number of scenarios

We use stimulus portion of security scenario to select security scenarios. In the stimulus portion of security quality attribute general scenario, attack is defined as :1) unauthorized attempts to access to system data, 2) unauthorized attempts to access to system services, 3) unauthorized attempts to change or delete data and 4) reduce availability of system services. So we use the scenarios of 1) unauthorized attempt to access system data, 2) unauthorized attempt to gain access to system services, 3) unauthorized attempt to delete or change data, and 4) attempt to reduce availability of system services, to evaluate architectural styles.

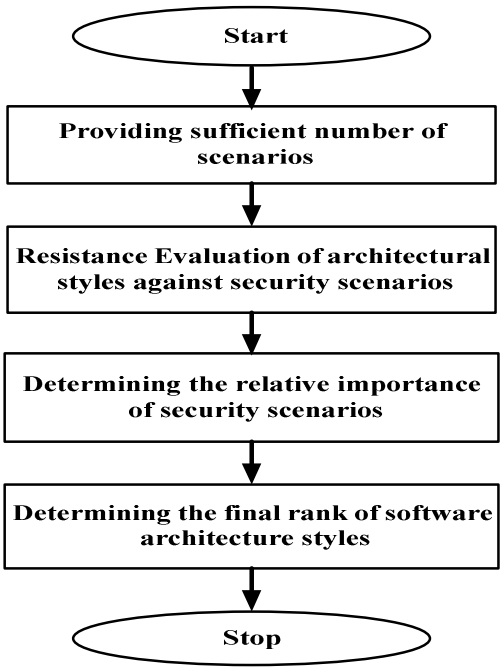

Figure 1. The process of evaluating and ranking architectural styles

B. Resistance Evaluation of architectural styles against security scenarios

At this stage, the effect of each selected security scenario on architectural styles will be evaluated.

1)*Scenario of Unauthorized Attempt to Access the System Data*

In applying this scenario to architectural styles, the styles are checked based on the amount of their protection of the data. In other words, the effective components in access to the repository data in style are checked. Components through which requests of create / delete / modify / read information from their canal passes and receiving requests regarding the legality request are checked [18]. Considering the foregoing, architectural styles are evaluated based on this scenario.

Repository style. In this style, all system data are in repository components. Repository component is solely responsible for the protection and maintenance of data and preventing unauthorized access to data and provides access to data according to the user level.

Blackboard Style. In this style, all of system data are maintained in the Blackboard component and any access to the data in blackboard is licensed by control component.

Pipe and filter style. In this style, there is not repository certain component and all of the system data is distributed among components of style. So, in this style, there is no control for the system data.

Layered style. In this style, all of system data is located in repository component. Repository component is at the lowest level. To achieve repository the total layers should be passed. So through various layers there is the possibility to control access to repository data.

Implicit invocation style. In this style, by occurrence of event, distributer component calls interested in the event component. So if there is a repository component, interested in the event component can access data in the repository only by licensed distributor component.

Client/server style. In this style, all of system data is located in repository component, and any access request to the repository data is checked by server and is licensed by this component. Since the server is includes n / 2 component in average, in practice, there are the possibility of n / 2 controls.

Broker style. In this style, all of data are contained in repository component and repository is accessed through the server and broker components. But all requests for access to data repository are checked and the access authorization is issued by the server. Since the server contains n / 2 component, in practice, there is the possibility of n / 2 controls.

Object-Oriented style. In this style, objects are divided into three categories, borderline, entities, and control. In sequence diagram, control objects control access to the data repository. Thus, in any use case, the number of effective components for accessing to the data repository is 1**.**

Table 1 shows the evaluation results of software architecture styles in terms of resistance to unauthorized achieve data repository, or the number of components that prevents unauthorized access to the data repository is of style.

2)Scenario of Unauthorized Attempt to Access System Services

In applying this scenario to architectural styles, styles are evaluated from a perspective. The perspective clarifies that in the case of an unauthorized attempt to gain access to system services, how many defense layers each style has to prevent unauthorized attempt. The architectural styles are evaluated according to this scenario.

TABLE 1. STYLE FROM VIEWPOINT OF ACCESSING REPOSITORY DATA

Row	Style	Symbol	Number of Components
1	Repository	RPS	1
2	Blackboard	BKB	1
3	Pipe and filter	P/F	0
4	Layered	LYD	n-2
5	Implicit invocation	I/I	1
6	Client/server	C/S	n/2
7	Broker	BRK	n/2
8	Object-oriented	OO	1

Repository style. In this style, each independent component offers specific service using available data in the repository. Thus accessing to system services is possible through independent components and there is a layer to protect the services that is the independent component.

Blackboard style. The services of this style are problem-solving and for solving the problem knowledge resources are activated by the control components and access to Blackboard. So only control components to prevent any access to the system services.

Pipe and filter style. In this style, there is no control of checking request of system services and by any service request; system services can be easily achieved.

Layered style. In this style, all the layers except the repository are involved in providing services and each service request system can be controlled by all the layers that their number will be n-1.

Implicit invocation style. In this style, after the occurrence of event and activation of component interested in the event, a special service will be provided. So, to have access to the service, there must be access to component interested in the current event.

Client/server style. In this style, each request is checked by server component and any unauthorized request is rejected. So, there are n / 2 components that control access to system services.

Broker style. In this style, any service request is checked by server and unauthorized request is rejected. As a result, there are n / 2 components that control access to system services.

Object-oriented style. In this style, in each use case just an object is the role of the server and checks service requests and rejects illegal requests.

Table 2 shows the results of the styles evaluation based on the scenario of unauthorized attempts to access system services or in other words, the number of effective components in the access to system services**.**

TABLE 2. STYLES IN TERMS OF EFFECTIVE COMPONENTS IN THE ACCESS TO SYSTEM SERVICES

Row	Style	Symbol	Number of Components
1	Repository	RPS	1
2	Blackboard	BKB	1
3	Pipe and filter	P/F	0
4	Layered	LYD	n-1
5	Implicit invocation	I/I	1
6	Client/server	C/S	n/2
7	Broker	BRK	n/2
8	Object-oriented	OO	1

3)Scenario of Unauthorized Attempt to Delete or Modify system data

Results of applying this scenario are according to applying scenario of unauthorized attempts to access system data, accordingly to Table 1.

4) Scenario of Attempt to Reduce Availability of System Services

Availability (ability of systems to provide service to authorized users), is one of the aspects of security. Software system availability has reverse relationship to critical components of the system architectural style, the components whose failure prevents them from service delivery to authorized users. As the number of critical components of architectural style increases, the availability of the outcome system software reduces.

By applying this scenario to architectural styles, styles are evaluated in terms of the number of critical components.

Repository style. In this style, repository components and independent components interacting with the repository in a use case are critical components. So the number of critical component of this style is two.

Blackboard style. In this style, the blackboard component, knowledge resource and control component interacting with the blackboard in each use case are critical components. So number of critical components of this style is 3.

Pipe and filters style. In this style, the entire components are critical. So critical component of the style is equal to n.

Layered style. In this style, all components are critical and the number of critical component of style is equal to n.

Implicit invocation style. In this style, distributor component and independent component affecting in use case are crucial. So number of critical components of this style is two.

Client/ server style. In this style, n / 2 server component and repository are critical. So the number of critical component of this style is (n / 2) +1.

Broker style. In this style, in addition to n / 2 components of the server, the client proxy, broker, server side proxy, the server and repository components are critical components. So the number of critical component of this style is equal to (n / 2) +4.

Object-oriented style. All objects in the sequence diagram are critical. But since there are multiple use case per system. On average, failure of a use case, the system works with lower throughput. Regarding the reliability evaluation of architectural styles [19], the number of classes in use cases is common and the 20 percentage of classes was common in use cases based on experience in the production of software systems.

Table 3 shows evaluation results of architectural styles based

on the scenario of effort to reduce availability of system services. In other words, it shows the number of critical components of each style.

C. *Determining the relative importance of security scenarios*

Frequency scenario of unauthorized attempts to access system data is equal to frequency of scenario of unauthorized attempts to gain access to system services and the frequency of occurrence of these two scenarios are more than the scenario of unauthorized attempts to delete or change the system data. Frequency of scenario of unauthorized attempts for deleting or modifying system data as well as more frequent scenario of an attempt to reduce availability of system services. Considering these cases and forming the scenarios paired comparison and the calculation of AHP method, the relative importance of each scenario is obtained and shown in Table 4.

TABLE 3. NUMBER OF CRITICAL COMPONENTS OF ARCHITECTURAL STYLES

Symbol	Critical components	Number of Critical Components
RPS	Repository component and a component	2
BKB	Control and blackboard components	3
P/F	All filters	N
LYD	All Layers	N
I/I	Component distributor and independent component	2
C/S	n / 2 server components and Repository component	(n/2)+1
BRK	Client Side Proxy, Broker, Server Side Proxy, n / 2 server components and repository component	(n/2)+4
OO	Objects in Sequence Diagram	$20\%n_t$

TABLE 4. THE RELATIVE IMPORTANCE OF SCENARIOS

Scenario	Relative Importance
Unauthorized attempt to access the system data(S1)	0.39
Unauthorized attempt to access system services(S2)	0.39
Unauthorized attempt to delete or modify system data(S3)	0.13
Attempt to reduce availability of system services(S4)	0.08

D. *Determining final rank of software architecture styles*

"Fig. 2" shows Hierarchical structure of *SASs* ranking. The objective of styles ranking, which appeared to be at the top, is to choose the best style from the perspective of security. The lowest level is architectural styles and there are security scenarios in the middle.

To determine the final rank of architectural styles, the relative rank of architectural styles must be calculated in terms of security scenarios. For this purpose, according to scenario-based evaluations styles, paired comparison table of styles are formed based on four security scenarios and styles relative rank are calculated by AHP method using Expert choice Software.

The effect of software size on styles ranking is taken into account in the computations. In object-oriented style, the number of objects (N_O) and in other styles the number of components (N) correspond the software size. So in the evaluation done in this section, the number of styles components are considered as 3, 4, 5, 6, 7, 8 and 9 and the number of classes in object-oriented style is considered accordingly as 21, 28, 35 ,42 ,49, 56 and 63.

The final Rank of architectural styles, were calculated regarding 1) the relative importance of security scenarios, 2) the relative rank of architectural styles for security scenarios and 3) different sizes of software (values N and N_O) by AHP method using Expert choice software. Table 5 shows the final rank of software architectural styles for different sizes of software.

The results show that considering security scenarios, layered(LYD) style have maximum support and pipe and filter(P/F) style have minimum support for security quality attribute. Client/server(C/S), broker(BRK), repository(RPS), implicit invocation(I/I), blackboard(BKB) and object-oriented(OO) styles respectively are second to seventh positions in this evaluation.

With the increasing of software size, the rank of some styles such as Pipe and Filter (P/F) and blackboard (I/I) are decreased, and the rank of some styles such as layered (LYD) are increased.

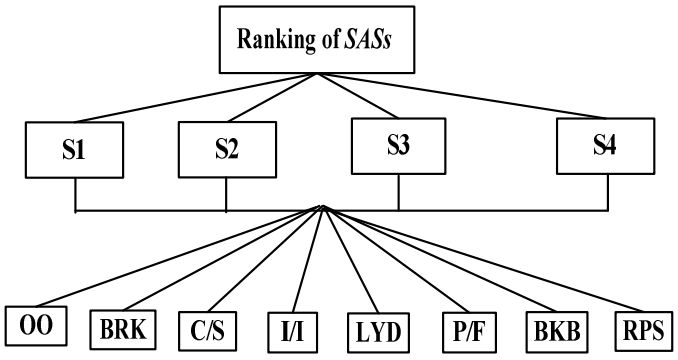

Figure 2. Hierarchical structure of *SASs* ranking from the viewpoint of security software

TABLE 5. SOFTWARE ARCHITECTURE STYLE RANKING

Symbol	N=3	N=4	N=5	N=6	N=7	N=8	N=9
	N_O=21	N_O=28	N_O=35	N_O=42	N_O=49	N_O=56	N_O=63
RPS	125	108	99	91	86	81	78
BKB	119	105	92	84	78	74	70
P/F	23	22	22	21	20	19	19
LYD	153	203	238	262	280	295	307
I/I	125	109	99	91	86	82	78
C/S	170	180	186	19	193	196	198
BRK	167	174	181	185	189	192	194
OO	117	98	84	75	67	61	57

\

Related Research

- Software architecture styles were evaluated from the viewpoint of reliability [14]. In this research, using reliability block diagram (RBD), reliability formulas of software architectural styles were extracted. To evaluate the effect of software size (number of components) on the reliability of software architecture styles, by changing the size software and based on architecture styles relation, the reliability values of architectural styles are computed.
- Architectural styles were evaluated from the viewpoint of maintainability [15]. In this research, architectural styles were evaluated from the viewpoint of maintainability, according to metrics of coupling, complexity and cohesion, and were ranked using analytic hierarchy process. In calculations of this method, the effect of the software size is considered in ranking of architectural styles.

But so far, Evaluation of architectural styles from the perspective of security has not been conducted.

V. CONCLUSIONS

In this paper, after reviewing software architecture styles and software architecture scenario-based evaluation method, due to the lack of studies on quantitative impacts of architectural styles on quality attributes specifically security, architectural styles effect on the software security were evaluated based on scenario. Finally, the final ranks of architectural styles were calculated regarding 1) the relative importance of security scenarios and 2) the relative rank of architectural styles for security scenarios, using AHP method. The results show that considering security scenarios, layered (LYD) style have maximum support and pipe and filter (P/F) style have minimum support for security quality attribute. Client/server(C/S), broker (BRK), repository (RPS), implicit invocation(I/I), blackboard(BKB) and object-oriented(OO) styles respectively are second to seventh positions in this evaluation.

The effect of software size on SASs ranking is taken into account in the computations. With the increasing of software size, the rank of some styles such as Pipe and Filter (P/F) and blackboard (I/I) are decreased, and the rank of some styles such as layered (LYD) are increased.

The most important outcome of this research is to quantify the impact of software architecture styles on security quality attribute based on scenarios.

REFERENCES

[1] P. T. Devanabu and S. Stubblebine, Software engineering for security: a roadmap", in ICSE '00: Proceedings of the Conference on the Future of Software Engineering. ACM, 2000, pp. 227–239.

[2] J. Juerjens, Secure Systems Development with UML, Springer, 2005.

[3] D.G. Rosado, E. Fernández-Medina, and M. Piattini, Comparison of Security Patterns, International Journal of Computer Science and Network Security, Vol. 6. Issue 2B. pp. 139-146, 2006.

[4] Len. Bass, Paul. Clements P, Rick. Kazman, 2003, "Software Architecture in Practice", Addison-Wesley Professional Publisher ,Second Edition, 2003, Addison-Wesley, ISBN: 0321154959

[5] Len. Bass, Paul. Clements, and Rick. Kazman, "Software Architecture in Practice", Pearson Education Publisher**,** Third Edition, 2013, ISBN**:** 9332502307

[6] F.Losavio, et al, ISO Quality Standards for Measuring Architectures, The Journal of System and Software 72, Page No. 209-223, Elsevier, 2004.

[7] M. AlSharif, et al, "ding the Complexity of Software Architecture", ACM Southeast Regional Conference, Proc. of the 42nd annual southeast regional conference, Huntsville, Alabama, 2004.

[8] S. Jingqiu and H. Behrouz, "Development of an Intelligent System for Architecture Design and Analysis", Canadian Electrical and Computer Engineering, Conference, pp. 539-542, 2004.

[9] H. Koh, S. Kung, and J.Park, "The Method to Choose Architectural Approaches in the Software Architecture Design Phase", ICITA (1), pp. 103-106, 2005.

[10] H. Reza, D. Jurgens, J. White, J. Anderson, and J. Peterson, "An architectural design selection tool based on design tactics, scenarios and nonfunctional requirements", Electro Information Technology, 2005 IEEE Int. Conf. , pp: , 2005.

[11] G. Zayaraz, and P. Thambidurai, "Software Architecture Selection Framework Based on Quality Attributes", Annual IEEE INDICON, pp. 167– 170, December 11-13, 2005, ISBN:0-7803-9503-4

[12] N.B. Harrison and P. Avgerinos, "Leveraging Architecture Patterns to Satisfy Quality Attributes", 1th European Conf. on Software Architecture Springer, pp (263-270), ECSA 2007 Madrid, Spain, September 24-26,

[13] P. Avgeriou, U. Zdun, "Architectural patterns revisited: a pattern language", Proc. of 10th European Conf. on Pattern Languages of Programs, pp.1-39, 2005 , Butterworth-Heinemann,

[14] G.R.Shahmohammadi, "Reliability Evaluation of Software Architecture Styles", 5-th International Conference on Parallel, Distributed Computing Technologies and Applications (PDCTA-2016), pp. 117-129, January 2th-3th,,2016, ISBN: 978-1-921987-45-8.

[15] G.R.Shahmohammadi, 2014, "Evaluation of the Software Architecture Styles from Maintainability Viewpoint", International Conference on Foundations of Computer Science & Technology, Zurich, Switzerland, pp. 183-197, January 2th-4th , 2014.

[16] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, "Pattern-Oriented Software Architecture: A System of Patterns", Vol. 1, Wiley, 1996, ISBN: 978-0-471-95869-7.

[17] M. Shaw, D. Garlan, 1996, "Software Architecture: Perspectives Discipline on an Emerging", Prentice Hall.

[18] K. Jiwnani and M. Zelkowitz, 2002, "Maintaining Software with a Security Perspective", Proc. of the International Conference on Software Maintenance, pp. 194 – 203, ISBN: 0-7695-1819-2

[19] Gholamreza. Shahmohammadi, and Saeed. Jalili, "Scenario-Based Quantitative Evaluation of Software Architecture Style from Maintainability Viewpoint", 14st Annual of CSI Computer Conference (CSICC 2009), Iran, Amirkabir University, 2009.

Author Profile

Mr Gholamreza Shahmohammadi received his Ph.D. degree from Tarbiat Modares University (TMU, Tehran, Iran) in 2009 and his M.Sc. degree in Computer Engineering from TMU in 2001. Since 2010, He has been Assistant Professor at the Department of Information Technology, Olum Entezami Amin University (Tehran, Iran). His main research interests are Software Engineering, Software Architecture, Software Metrics, Software Cost Estimation and Software Security .