Evaluation of non-functional characteristics of web-based systems modeled and designed using aspect-oriented technology by aspectual software architecture analysis method (ASAAM)

Davood Karimzadganmoghadam [1], Davood Vahdat [2], Mohammad Pira [3], Reza Askari moghadam [4]

[1.] Department of IT, Payame Noor University, Tehran, Iran, d_karimzadgan@pnu.ac.ir
[2.] Department of IT, Payame Noor University, Tehran, Iran, vahdat@pnu.ac.ir
[3.] Department of IT, Payame Noor University, Tehran, Iran, Mohammadpira408@yahoo.com, Corresponding Author, Msc
[4.] Faculty of New Sciences and Technologies University of Tehran, Tehran, IR Iran, P.O.Box:14395-1561, r.askari@ut.ac.ir

Abstract: Despite the importance and well-known status of non-functional requirements in the success of Web applications in the field of Web Engineering, they do not receive much attention. In many cases, these requirements remain pending, unconsidered, un-analyzed and undesigned after determining in the requirements engineering phase until completing application implementation. The aim of the present study is to model and analysis the non-functional requirements in designing Web applications. This will be done to ensure providing an architecture which supports the necessary quality characteristics of these applications. To realize the nature of these requirements, we focused on a large industrial case study which is a Web-based organizational application. It was observed that the majority of non-functional requirements in the Web applications architecture are intersecting concerns which should be modeled separately. However, this issue has not received necessary attention. Finally, the proposed architecture has been presented and evaluated.

[Davood Karimzadganmoghadam, Davood Vahdat, Mohammad Pira, Reza Asgarimoghadam. **Evaluation of non-functional characteristics of web-based systems modeled and designed using aspect-oriented technology by aspectual software architecture analysis method (ASAAM).** *Life Sci J* 2012;9(4):2480-2486]. (ISSN: 1097-8135). http://www.lifesciencesite.com. 366

Keywords: Concern; Requirement; Web-pages; System Architecture; Continuity; Aspect.

1. Introduction

Keep pace with technological advances, the dependence on information technology is increased and relationships become more complex. Since rapid and safe access to information is vital where success or failure of business and industry depends on it, Web-based systems have a special place. Using this technology, experts and organizations managers can access to needed information from anywhere in the world. Also, using this technology, organizational costs reduce significantly which is considered as an important issue.

Web engineering suggests an agile but also systematic framework for constructing high quality applications and industrial Web-based systems (Pressman, 2010). In developing these systems, we deal with functional and non-functional requirements. Functional requirements represent system’s performances. In fact, functional requirements are the concrete and usable system capabilities so that the user could perform its duties in the organization. Non-functional requirements or quality characteristics are related to the system performance. These requirements are rather concerned by system developers. If these two requirements are not dependent and modeled, the costs of system development and maintenance increase significantly.

Concern is an aspect of a problem which is important for stakeholder or stakeholders (Mancona, 2003). Concerns are often confused with requirements, although they are different, basically (Jacobson and Pan-Wei, 2005). Intersecting concerns are observed when the functional and non functional concerns are intertwined and not be able to module separately. Finding the intersecting concerns is a difficult task, thus designing and developing the system becomes difficult (Francescomarino and Tonella, 2009). Object-oriented approach is the dominant method for designing the systems. But, despite all its advantages and privileges, it is not able to module non-functional concerns, properly. Finally, it is observed that the non-functional concerns are scattered in the functional concerns.

Some techniques have been proposed to resolve this drawback of object-orientation including aspect-oriented programming (AOP), combination filters (CF), multidimensional separation of concerns (MDSOC) and adaptive programming (AP) (Chitchyan and Ruzanna, 2005). In the present study, aspect-oriented programming was selected. Since the language of developing this subsystem is Java, the AspectJ programming language was used. This has added the object-orientation concepts to Java. Aspect-oriented programming was introduced by

Mr. Gregory Kyzals and his group in 1997 (Kiczales and Hilsdale, 2001). Aspect-oriented programming is a technology which supports separation of intersecting concerns (Augusto and Lemos, 2011). With this language, functional and non functional concerns can be modeled. Thus, the process considerably lowers the cost of system maintenance. It also facilitates understanding architecture and the system code for program developers and improves system performance and security.

Software architecture is defined as a set of important design decisions about the system (Medvidovic and Dashofy, 2007). Software architecture plays a fundamental role in overcoming the inherent difficulties of developing large scale complex software systems (Kiczales and Lamping, 1997).

Aspectual Software Architecture Analysis Method (ASAAM) was selected for evaluating the system architecture (Pressman, 2010). Using this method, we can assess the improvement of the architecture compared to the original architecture. At the end of the study, with the help of calculations, it is shown that the system architecture has been improved significantly compared to the original architecture.

2. Case Study

"Comprehensive web-based organizational resources planning system" is a medium scale web-based application. This system consists of 11 subsystems including the financial management, production management, sales management, purchasing management, warehouse management, human resources management, management, maintenance, quality control, project management and office automation. The system has been designed based on architectural patterns of Layering, Domain Model, Data Mapper and MVC in order to exploit in organizations with 3,000 to 4,000 personnels. Fig. 1 shows the system architecture.

Due to the enormous volume of this web application, much attention has been paid to human resources management subsystem during conducting present study. The process model for web applications engineering is an agile version of the general software process model. System development process is an agile and model-oriented process. This process develops the system in terms of functional requirements systematically. However, this process of development acts occasionally for providing non-functional requirements. So that after implementation of each functional scenario, the mechanisms of providing non-functional requirements are programming in the code modules of the scenario repeatedly, non-coherently. The basic architecture of the case study is shown in Fig. 3.

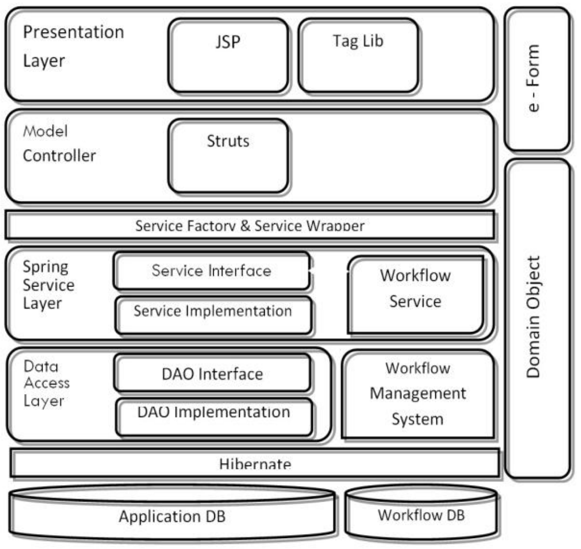


Figure 1. Layered architecture of the comprehensive web-based organizational resources planning system

3. Redesign

The case study was redesigned using an aspect-oriented process presented for developing web-based organizational applications considering the two non-functional requirements of security and response-time. These requirements are important in the current situation of the system. According to improvement process, after diagnosis of non-functional requirements with being aspect capability, their subsystem scenarios, i.e. the operational non-functional and controllable non-functional concerns must be compiled. Table 1 represents the corresponding scenario of these concerns for the case study.

After compiling these scenarios, the web-based organizational application architecture is developed by defining aspect-oriented components for moduling non-functional requirements with aspecting capabilities intersecting concerns of the application architecture. These components are described using UML 2.0.

In the next step, each modeled concern has been implemented using aspect-oriented programming independent of the core functional concerns. Fig. 3 shows a part of simplified sample code.

In the basic architecture, the components including core functional concerns, operational non-functional and controllable non-functional are intersecting i.e. the scattering and complexity of the concerns is observed throughout the architecture.

The architecture of this subsystem has been improved by aspect-oriented injection of non-functional requirements to its development process.

The obtained architecture is shown in Fig. 4 as the proposed architecture.

The architecture obtained from this improved process consists of two orthogonal layers. The first layer is the basic architecture of Web application. This layer is developed using conventional method of Web Engineering merely to achieve functional goals based on core functional concerns. The second layer is responsible for modeling non-functional application requirements as aspect-oriented components. Finally, the two layers are combined together by provided infrastructures in the aspect-oriented programming after independent development, and represent the final product. It is expected that the new architecture be more improved than the previous architecture version.

Table 1: A functional scenario

Scenario No	S120
Scenario Type	Core non-functional concern
Scenario Title	Calculation of group salary
Scenario Description	At first, a list of organization employees is displayed based on required filters. Then, some (or all) employees are selected from the list. Salary calculation is performed for each employee according the following procedure: 1 – a blank salary list corresponding to the current salary period is constructed for each employee. 2 - Employment certificate of the personnel which is active in the current salary period is retrieved. 3 - A list of salary elements defined in the employment certificate is retrieved. 4 – A salary item is constructed for each retrieved salary elements. This item is determined based on the specified the calculation method.

4. Evaluation Method Selection

Evaluation of the software is critical for meeting the quality requirements of the system. In software projects, as soon as a bug discovered in the project, as the costs of correcting and maintenance of the software will be reduced. Thus, assessment has a special place. Basically, architecture evaluation is performed after defining the architectural decisions before the implementation phase. Software architecture evaluation can be done in two times: early, late (Clements and Kazman, 2006). Early architecture evaluation can be carried out when architecture not still fully implemented. The late architecture is performed when the architecture has been well designed.

The basic parameters of the software architecture evaluation include: minimum coupling, maximum continuity, completeness, being understood, adaptability, realism (Garland and Anthony, 2003).

Coupling is the number of dependencies between two subsystems. If dependencies between the subsystems were less, the subsystems are independent. Continuity is the number of dependencies within components of a subsystem. If the subsystem consists of a lot of connected components, its continuity is high. If the subsystem consists of irrelevant components, its continuity is low. The software architecture must be completed to meet all functional and non-functional requirements of the system. The software architecture must be understandable for various stakeholders. The components of the software architecture must be compatible and consistent. Finally, the software architecture must be implementable.

The following items are of the advantages of evaluation: gathering the project stakeholders, prioritizing conflicting objectives, obligation to provide a clear architectural, improvement of software architecture (Clements and Kazman, 2006). There are four techniques categories for evaluating software architecture which are briefly described. These techniques are: experimental techniques, simulation, questioner and measurement techniques.

In experience-based techniques, evaluations are carried out based on knowledge and experience of developers and evaluators according to the past projects. This method is based on interviews with system stakeholders and use of their experiences (Jeong and Kim, 2006). Because of inconsistencies in stakeholders' statements and non-technicality of the results of interviews and evaluations, addressing this method is neglected in the present study.

The protype is used in the simulation technique. Prototype is a sample which provides non-performance user interface. The users may check it to ensure providing their requirements by the system. The simulation method is used to measure the performance. These methods need to implement components of the architecture and simulating other components for architecture implementation. Therefore, they need information about the underdevelopment system which is not available during the development and architectural design. Also, these methods are expensive. Moreover, many important features of the system, especially non-functional features such as storage capacity, reliability and error educability cannot be instantiated.

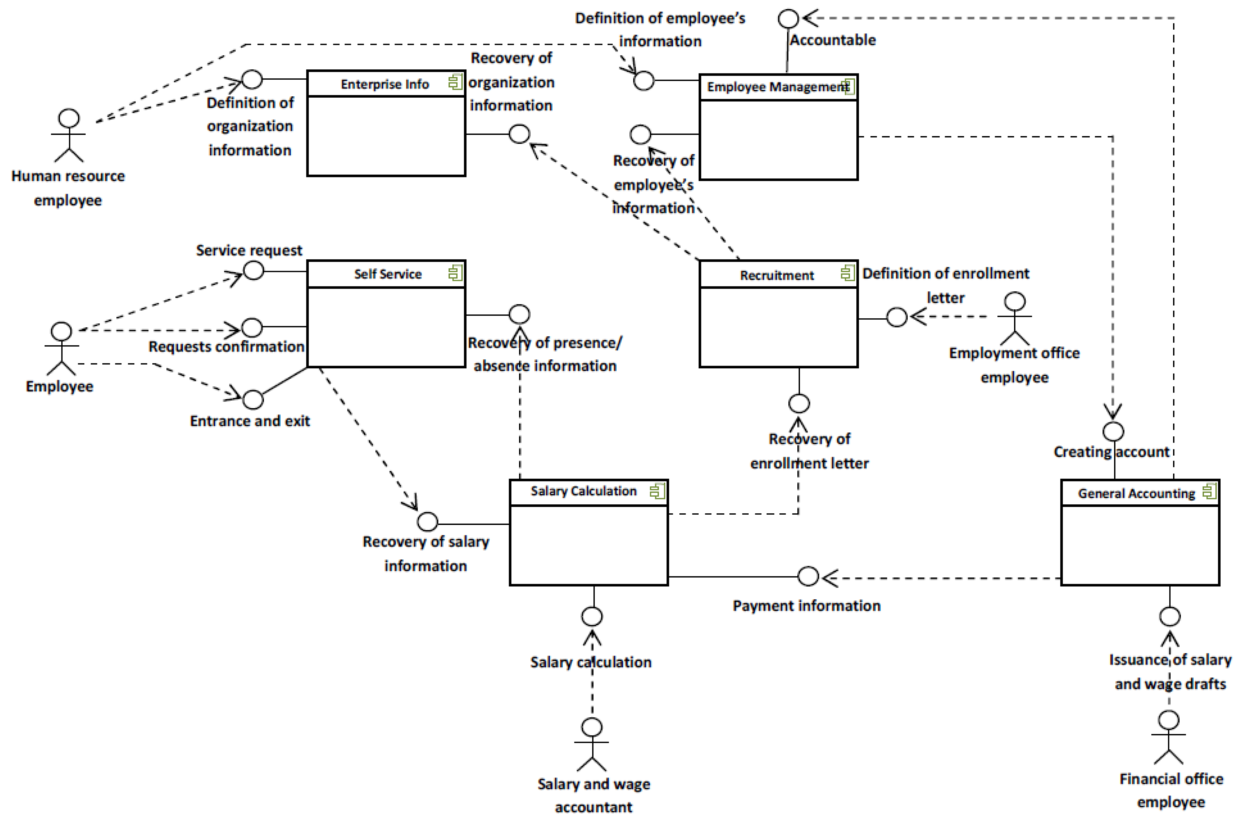
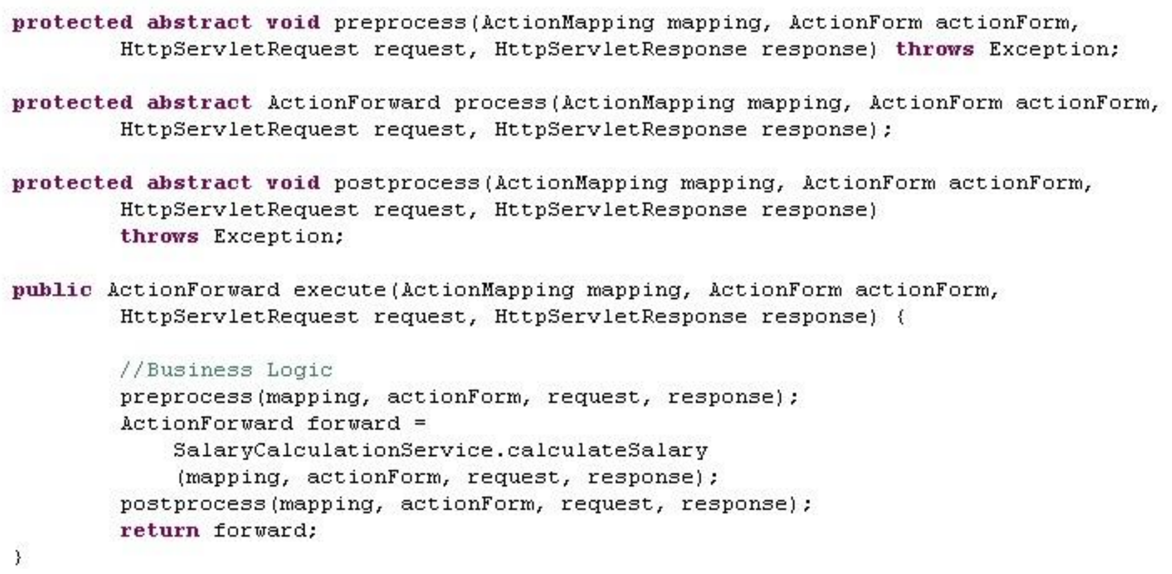

Definition of employee's information
Accountable
Recovery of organization information
Enterprise Info
Employee Management
Definition of organization information
Human resource employee
Recovery of employee's information
Service request
Self Service
Recruitment
Definition of enrollment letter
Employee
Requests confirmation
Recovery of presence/ absence information
Employment office employee
Entrance and exit
Recovery of enrollment letter
Creating account
Salary Calculation
General Accounting
Recovery of salary information
Payment information
Salary calculation
Issuance of salary and wage drafts
Salary and wage accountant
Financial office employee

Figure 2. Basic architecture of human resources management subsystem



```
protected abstract void preprocess(ActionMapping mapping, ActionForm actionForm,
        HttpServletRequest request, HttpServletResponse response) throws Exception;

protected abstract ActionForward process(ActionMapping mapping, ActionForm actionForm,
        HttpServletRequest request, HttpServletResponse response);

protected abstract void postprocess(ActionMapping mapping, ActionForm actionForm,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception;

public ActionForward execute(ActionMapping mapping, ActionForm actionForm,
        HttpServletRequest request, HttpServletResponse response) {

        //Business Logic
        preprocess(mapping, actionForm, request, response);
        ActionForward forward =
            SalaryCalculationService.calculateSalary
            (mapping, actionForm, request, response);
        postprocess(mapping, actionForm, request, response);
        return forward;
}
```

Figure 3. Part of the applied code for salary calculation in Model Controller layer after applying the proposed aspect-oriented development precess

Measurement methods use mathematical equations and expressions. These methods evaluate the software architecture based on measurable quality characteristics. Measurement techniques express questions with measurable answers. However, because measuring the questioned values is a difficult task in the software design stage and it is limited to some small qualitative characteristics, these techniques are not widely used as questioning techniques.

In questioning techniques, questions are expressed about the quality of the architecture (These questions can be expressed in the form of checklists or scenarios). Because these questions are qualitative, their answers cannot be accurately determined. But, they can be used for relative comparison of several items. In this study, the questioning technique was used for evaluation.

Questioning techniques consist of three techniques including scenarios, questionnaires and checklists. Scenario is a technique for determining the requested quality of the architecture. Six components are available for consistenting and normalizing different scenarios to the standard scenario. This will facilitate the evaluation processes i.e. stimulus source, stimulus, environment, product, response, response measurement (Clements and Kazman, 2006).

One of the advantages of the scenarios is that they are specific to a particular system. Each software system needs a certain degree of quality with respect to their duties. Therefore, the expected quality level of the system should be determined according to the type of system tasks. Architecture evaluation by scenarios is such that it is determined whether the architecture can meet the desired scenarios or not.

The SAAM can be considered as the first scenario-based software architecture evaluation method. This architecture evaluation method is used in terms of non-functional requirements. The goal of SAAM is to provide a method for evaluating the quality features of the architecture versus the available documentation of the system requirements. If the SAAM is employed for the architecture, the strengths and weaknesses of the architecture, and the failed points in the terms of change capability will be determined. If the method is used for two or more architectures, it will compare the architectures in terms of change capability.

For evaluation by the SAAM, the non-functional scenarios that need to be evaluated must be identified and numbered. The software architecture should be briefly outlined. If more details of the architecture are needed during the evaluation, the architecture will be developed.

At the end of the SAAM evaluation, mapping between scenarios and architecture and its changing costs will be presented in order to identify sensitive areas of the architecture which have the potential to change. Also, understanding of the system performance is completely done. Furthermore, a comparison between the architectures and the level of their support from the system performance is presented at the end of evaluation.

Assessment Team of SAAM: Assessment Team is composed of three groups:

i - External stakeholders are the owners and users of the system who are involved in providing commercial purposes.

ii - Internal stakeholders: system evelopers including the architect and the architectural team who have direct role in software architecture providing and analyzing.

iii – SAAM Team: the architecture evaluation team which do the evaluation task.

Before implementing SAAM, a brief explanation is given about the system functionality and the main purpose of the system. Then evaluation process is started. SAAM consists of 6 stages i.e. scenario development, architecture description, classification and prioritization of scenarios, evaluating scenarios, obtaining communication between the scenarios and providing the overall assessment.

ASAAM has been developed by expanding the SAAM method in order to identify architectural features using the scenarios. Fig. 5 shows the ASAAM activities.

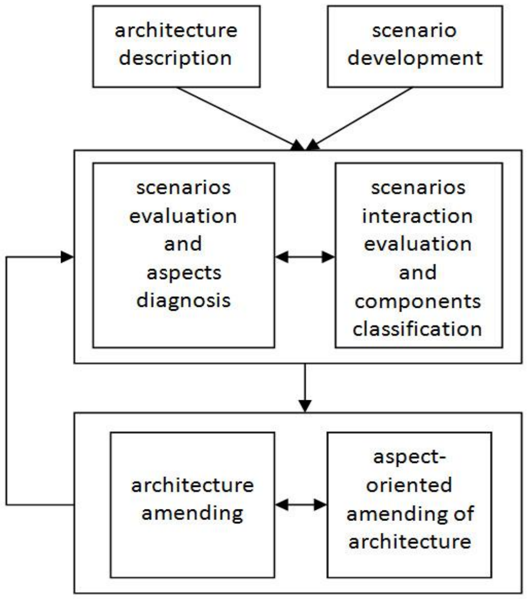

architecture description
scenario development
scenarios evaluation and aspects diagnosis
scenarios interaction evaluation and components classification
architecture amending
aspect-oriented amending of architecture

Figure 5. Activity of ASSAM (Tekinerdogan, 2004)

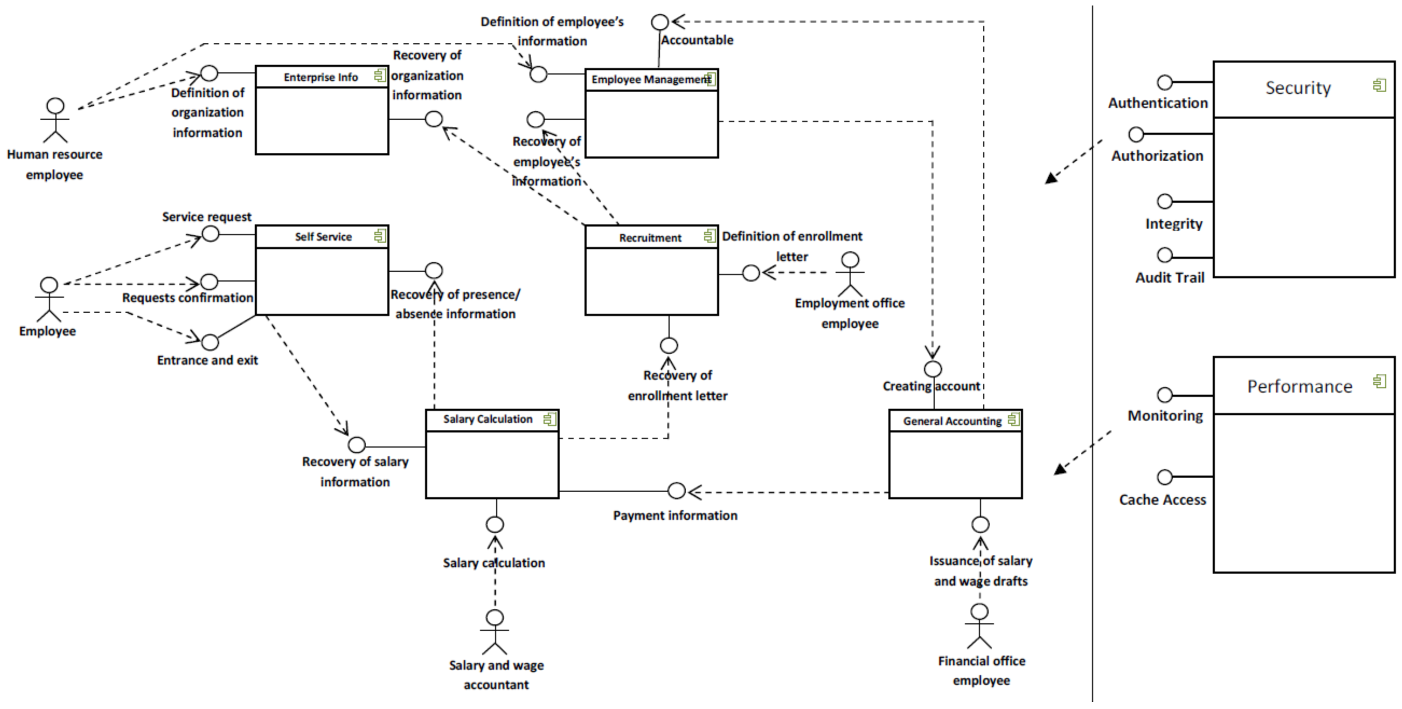

Definition of employee's information
Accountable
Recovery of organization information
Enterprise Info
Employee Management
Definition of organization information
Human resource employee
Recovery of employee's information
Service request
Self Service
Recruitment
Definition of enrollment letter
Employee
Requests confirmation
Recovery of presence/ absence information
Employment office employee
Entrance and exit
Recovery of enrollment letter
Creating account
Salary Calculation
General Accounting
Recovery of salary information
Payment information
Salary calculation
Issuance of salary and wage drafts
Salary and wage accountant
Financial office employee
Security
Authentication
Authorization
Integrity
Audit Trail
Performance
Monitoring
Cache Access

Figure 4. The proposed architecture

Table 2: A corrective maintenance scenario

Scenario No	**S300**
Scenario Type	Corrective maintenance scenario
Relevant Scenario	NFR-1 (Security)
Scenario Description	The has Permission On (username, action) method does not consider the user role in the permission scenario, while the user may have several roles in the system. Neglecting this may cause problem in the system. Because if the user has multiple roles, this method considers only his first role. Determination of the Right of access should be done separately for each user role.
Response	This method should be corrected as has Permission On (username, action and role).

5. Evaluation of the basic and proposed architectures

Two architecture design approaches of the case study were evaluated using ASAAM in terms of maintenance capability.

Four scenarios categories including scenarios for corrective maintenance, scenarios for perfective maintenance, scenarios for adaptive maintenance, and scenarios for preventive maintenance are defined in order to evaluate the maintenance capability. A typical scenario is shown in Table 2.

In order to compare the basic and proposed architectures, each maintenance capability evaluation scenario was weighted considering the cost of the changes. The weight of each scenario is calculated using the following equation. The results are listed in Table 3. The average cost of each maintenance capability evaluation scenario was calculated. The results are listed in Table 4 as the costs of maintenance capability.

$$\text{Cost} = [(L * M * C) / T] * 100$$

Cost: the average cost of change

L: the number of affected layers of the architecture

M: the average number of the affected modules of each component in each layer

C: the number of affected components

T: total number of the subsystem modules

6. Conclusion and Summary

In the proposed approach, using the concept of aspect, the core functional concerns, operational non-functional concerns and controllable non-functional concerns were separated in the process of developing Web applications. The obtained architecture firstly fulfills the principle of separation of concerns better than before. Secondly, non-functional requirements with aspectual capability are clearly described. Therefore, it is expected the new architecture has higher maintenance capability than the old architecture. The results of evaluation and testing performed on the case study as a web-based organizational application shows that the system maintenance capability wille grow significantly with increasing adhesion of architectural components and removal of dependence of hundred system modulus to non-functional requirements as intersecting concerns through the use of aspects. Thus, according to evaluation results and by comparing these two architectures, it can be concluded that the use of aspectual approach in modeling non-functional requirements in designing web applications result in increasing the maintenance capability of the application.

Table 3. Comparison of basic and aspect-oriented architectures in terms of maintenance costs

Scenario No	Average maintenance cost of each module in basic architecture	Average maintenance cost of each module in aspect-oriented architecture
S300	[(4*4*6)/96]*100	[(4*1*1)/104]*100
S301	[(3*4*6)/96]*100	[(3*1*1)/104]*100
S302	[(1*2*6)/96]*100	[(1*1*1)/104]*100
S303	[(3*4*6)/96]*100	[(3*1*1)/104]*100
S304	[(3*4*6)/96]*100	[(3*1*1)/104]*100

Table 4. Average cost of each maintenance capability scenarios categories

Maintenance activity	Average cost in basic architecture (scale: 100)	Average cost in aspect-oriented architecture (scale: 100)
Corrective maintenance	87.5	3.4
Perfective maintenance	43.7	1.9
Adaptive maintenance	87.5	3.4
Preventive maintenance	25	1

Corresponding Author:
Msc. Mohammad Pira
Department of IT
Payame Noor University
Tehran, Iran
E-mail: Mohammadpira408@yahoo.com

References

1. Roger S. Pressman. “Software Engineering: A Practioner's Approach”, Seventh Edition, McGraw-Hill Higher Education, 2010.
2. Otávio Augusto, Lazzarini Lemos. “A pointcut-based coverage analysis approach for aspect-oriented programs”, Information Sciences, Volume 181, Issue 13, 1 July 2011.
3. Nenad Medvidovic, Eric M. Dashofy, “Moving architectural description from under the technology lamppost, Information and Software Technology, Vol. 49, No. 1, pp. 12-31, Jan 2007.
4. Chitchyan, Ruzanna, “Survey of Analysis and Design Approaches”, AOSD-Europe., Deliverable D11, 2005.
5. Chiara Di Francescomarino, Paolo Tonella, “Crosscutting Concern Documentation by Visual Query of Business Processes”, Lecture Notes in Business Information Processing, FBK-IRST, Trento, Italy, 2009.
6. Kandé, Mohamed Mancona. “A Concern-Oriented Approach to Software Architecture”, PhD Thesis, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, 2003.
7. Ivar Jacobson, Pan-Wei NG. “Aspect-Oriented Software Development with Use Cases”, Addison Wesley Professional, 2005.
8. Gregor Kiczales, John Lamping, “Aspect-Oriented Programming”, In Proceedings of the European Conference on Object-Oriented Programming (ECOOP), 1997.
9. Gregor Kiczales, Erik Hilsdale, “An Overview of AspectJ”, In Proceedings of the European Conference on Object-Oriented Programming (ECOOP), Pages 327-353, 2001.
10. Paul Clements, Rick Kazman, “Evaluating software architectures: methods and case studies”, Addison-Wesley, 2006.
11. Jeff Garland, Richard Anthony, “Large-Scale Software Architecture, A Practical Guide using UML”, John Wiley & Sons ltd, 2003.
12. Gu-Beom Jeong, Guk-Boh Kim, “A Study on Software Architecture Evaluation”, Springer Berlin / Heidelberg, 2006.
13. Bedir Tekinerdogan, “ASAAM: Aspectual Software Architecture Analysis Method”, In Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA), 2004. pp. 5-14.

10/5/2012