A Method to Challenge XP Agile Method through Software Architecture

Zafar Karimi¹, Hassan Rashidi², Ali Broumandnia³
¹Islamic Azad University at Qazvin, Graduate Student of Software engineering.
²Assistant Professor of Computer Engineering Faculty, Islamic Azad University at Qazvin
³Assistant Professor of Computer Engineering Faculty, Islamic Azad University at Qazvin
zk_karimi2006@yahoo.com, hrashi@gmail.com, broumandnia@gmail.com

Abstract: One challenge of XP agile method in software development is this method’s underestimation over software quality attribution; these attributes are of main indices of software architecture. In this study, a method is represented for responding this challenge on the bases of probability theory. In this method, firstly, the rating matrix is structured on the bases of quality attributions architecture solutions. Each element of this matrix shows a rating for every solution and the ratings will be initialized through analytic hierarchy process (AHP). Via quality weight implement on above matrix the rating vector is created whose each element represents the rate of each solution in reaching those quality attributes in rating matrix. Because rating vector follows normal probable distribution, its elements' rating probable density is mostly gathered around the mean.

In this study, the probable density of architecture will be defined, the appropriateness of a solution in comparison to other solutions will be evaluated or and the respond to XP challenge will be done easily in order to reach architecture solutions in the frame of rating vector. A complex of data is gathered and the results are compared in an experimental method in order to investigate the represented method of this paper.

Keywords: Quality Attributes, Architecture Solutions, Normal Distribution and Analytic Hierarchy Process AHP.

1. Introduction

In XP¹ process, software development is carried out in several release and replicate and at the end of each release a complex of system demands are performed. Along each replicate, programmer pairs are busy in performing new responsibilities which in previous are stated in the form of applicant statements by customers. Usually every programmer pair receives a duty as an entry which states a responsibility of system. Then the pair plans and creates unit tests for related duty. In continuous the development is performed and it is possible of code reformation to be done by programmers. When duty performance is finished, it is integrated with caused codes and certainty of final code resistance is made. This process is called duty completion. Since this process is done in parallel to different pairs and without any quality observation, the possibility of architectural deficient structures formation increases and under the effect of quality attributes will cause serious problems for software architecture. The gradual increasing of these deficiencies will make system architecture non-manageable and incorrigible. In conclusion, it can be evolved that as programmers control the integration of software code through continuous integration activities and do tests on codes so, evaluations can be done through software architecture and modern developments on system structure. It has been attempted to reach two aims beside work review and the activities of XP process: 1) Classifying the qualitative attributions, architectural solutions and their rating estimation in order to matrix represent of system responsibility 2) Creating mathematical-possibility framework for restructuring architecture. In that case, the architecture will find software quality attributes in each period according to his own experiment and knowledge and he will represent suitable solutions with the aid of other specialists.

The following parts of study are structured in this method. Part 2 considers basic expressions and related activities. Part 3 shows qualitative attributes, architectural solutions and the way of relating them to each other. Part 4 discusses a problem with proposed method. And finally in part 5,6 discussion, conclusion and propositions will be continued..

2. Basic Expressions and Related Activities

In the context of software architecture some valuable activities are carried out. In some studies, the manners of finding suitable and effective qualitative attributions of software systems are discussed. And in others, different kinds of methods and techniques related to agile methods are studied. In [25], the complexes of all non-responsibility or software qualitative demands are considered and along stating their finding manner, choice mechanism and quality attribution priority are discussed. In [26], [20], some methods are represented in order to implement quality attributions on the bases of which software architecture can perform quality attributions on the software. In [18],[19], [21] their effects are described too. Along recent years, software architecture has been developed beside other development methods of software[30]. Software architecture leads to the increase of beneficiaries' knowledge related to system and will result in a high quality product. As agile methods came into existence,
the attention to software architecture decreased which was under the negative effect of process and plan focus. Some believe that agile methods should consider software architecture more than the past era [21]. Therefore, representing a method in order to intensify agile methods from software architecture point seems to be positive and valuable. Two methods of this context are described as following:

- Consolidating software architecture methods via XP: Paper [10] attempts to reach software architecture advantages through adding modern methods. These advantages include reaching quality attributions, software conceptional model and right relation among beneficiaries. In this method, QAW [32], ADD [31] and ATAM/ CBAM[33] methods are added separately to XP. QAW is a workshop which is held in order to reach to a better knowledge over quality attributions. ADD is proposed for architecture planning and ATTAM and CBAM will estimate architecture. These methods will be performed in specific periods of project.

- Utilizing architecture activities of developing quotes: This method [21] disagrees utilizing software architecture techniques in agile environments from a point that modern architecture activities related to XP method should be planned and utilized by a team on the bases of XP development process.

### 2.1 Agility

Agility is a near connection to change. The change is routed from professional environmental changes and forces the organizations to confront it. An organization which can not or do not want to react in front of changes should wait to accept severe damages and finally catastrophe and death. The organizations that can accept necessary ability and agility in front of changes can narrow the market for their opponents and will benefit more. In changing environments, the plans are usually mistaken which are because of human factors in software environments [11]. In this kind of environments these changes are normal and usual. Agility emphasize is on individual ability for responding changes and understimating programming and preparing large plans. This does not means that agility methods will not plan. Their programming is in a way that have demanded flexibility in front of changes and benefitting. In the case that the environmental changes are much more than usual, the agility methods will be the only achievement methods. In the presence of software architecture challenge despite agile methods [26],[13],[14], these methods are in focus and their advantages are of a high value that makes them as affective and beneficial methods.

### 3. Matrix Representation of Quality Attributions and Architectural Solutions

In this paper the relation between quality attribution and architecture solutions are shown in matrix form. The rows show the quality attributions and the columns show the solutions. The amount of each matrix entry is a number between +9 to -9 that on the basis of AHP\(^5\) method the effect of solution in providing quality attribution is shown by a number from +1 to +9 (table.1)[14],[24]. 1 shows the least positive effect and 9 the most positive effect in providing quality attribute. As in [22] attentions are paid to agreement and disagreement of some quality attributes e.g. efficiency increase will decrease reformation. So in this paper -1 to -9 is introduced for negative effect of solution on attribution. -1 shows the least negative effect and -9 the most negative effect of a solution on an attribution.

<table>
<thead>
<tr>
<th>Numerical Effect</th>
<th>Verbal Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1 to +9</td>
<td>Attribution provision from least to most with positive effect</td>
</tr>
<tr>
<td>-1 to -9</td>
<td>Attribution provision from least to most with negative effect</td>
</tr>
</tbody>
</table>

Every software attribution is of weight of 0 to 1 and indicates its importance in software. Also completion weight of all software attribution equals 1. (Attribution weight is normalized).

### 4. Problem Description

In XP process if it is assumed that each system responsibility demanded quality attribution \(n\) and \(m\) proposed solution (decision) for providing attributions that each solution could affect positively or negatively one or several solutions, we will represent a criterion for choosing a solution among \(m\) proposed solutions for architecture reformation and planning in XP process.

#### 4.1 Introducing Utilized Symbols

At first, utilized symbols in this paper are introduced here.

<table>
<thead>
<tr>
<th>(n)</th>
<th>The number of quality attribution for responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M)</td>
<td>The number of architecture solutions (decisions) for system responsibility</td>
</tr>
<tr>
<td>(W)</td>
<td>Matrix (n \times 1) weight of software quality attributions</td>
</tr>
<tr>
<td>(W^T)</td>
<td>Transposed matrix of (W)</td>
</tr>
<tr>
<td>(X_i)</td>
<td>Matrix (m \times n) the amount of quality attribution via responsibility solutions of (i)</td>
</tr>
<tr>
<td>(X_{ij})</td>
<td>(j_{th}) vector of (X_i) (matrix (shows the rate of (j_{th}) solution on quality attribution without counting attribution weight)</td>
</tr>
<tr>
<td>(G_i)</td>
<td>Matrix (m \times 1) shows the importance of responsibility solutions (i) system on software</td>
</tr>
<tr>
<td>(G_{xi})</td>
<td>(X) entry of matrix (G_i)</td>
</tr>
</tbody>
</table>
### 4.2 Proposed Solution

According to the fact that each solution is of a rating in providing quality attribution, the product of solution ratings in attributions’ weight results in matrix \( G_i \):

\[
G_{i}^{T} X_i
\]  

(1)

Matrix \( G_i \) shows the importance or rating of responsibility solutions I and is called rating vector. Each one of its entry indicates a solution in providing all quality attributions of responsibility and the more the amount is, the more the rating density is. If matrix \( G_i \) is calculated for several different solutions and same software, through their propriety the best solution is gained for each software responsibility.

If the mean of matrix \( G_i \) is called \( \mu_i \), the mean is calculated through formula (2). Covariance or difference of mean for \( G_i \) is showed by \( \delta_i \).

\[
\mu_i = E(G_i)
\]  

(2)

Possible density of \( G_{i\alpha} \) is defined propriety of matrix \( G_i \) and calculated through formula (5).

\[
(G_{i\alpha}) = \frac{-(G_{i\alpha} - \mu_i)^2}{2\delta_i} \exp\left(-\frac{1}{2}\right)
\]  

(3)

In formula (3), \( f(G_{i\alpha}) \) is a numeric which shows the solution rating on the responsibility of \( X_i \). In other words, \( f(G_{i\alpha}) \) shows the suitability of solution which is called solution propriety and the bigger this numeric is, the more density is the solution.

In order to solve problem, there needs to be an architecture role in software development. The activities are performed in this way that at the end of duty development, each programmer pair should create simple sensible models and send them to the architecture. The architecture will gather these models and by juxtaposing them will reach a current system structure. Also, he will find quality attributions of each system responsibility through investigating system structure and will represent them in the form of quality scenarios for some programmers and a minimum of customers. The meeting members will propose several solutions to reach quality attributions of each responsibility in software production through quality scenario review. Then the architecture will choose and represent the most suitable rating vector solution of each responsibility according to table 2 of proposed solutions and quality attributions of every responsibility of system through AHP method in the form of an inter-connected matrix via formula 3. Through these activities the system architecture restructuring and improvement process will be completed in XP method.

#### 4.3 Gi Normal Distribution Confirmations

If it is assumed that \( X_i \) or every vector of \( j \) is a random sample, every entry of this vector will be of monotonous distribution in \((+9, -9)\). So, \( X_{ij} \) are of monotonous distribution as random variables of \( n \) variables. On the bases of centric matter [28], each entry of \( G_i \) which is caused through formula (1) is a random variable that is produced from dependent and distributed variables so they will be of normal distribution. According to normal vector definition, \( G_i \) is normal probable distributed, of \( \mu_i \) mean and is calculated through formula (4):

\[
\mu_i = \frac{1}{m} (G_{i1} + G_{i2} + \ldots + G_{im})
\]  

(4)

So, it can be said that \( \mu_i \) is an assessing \( \mu \) i.e. \( \mu \) is not clear and if the rating vector \( G_i \) is calculated on the bases of each software and through formula (4) which can be used in formula (3). As calculating \( \delta_i \) through formula (5) and putting it in formula (3), the solution proprieties will be accessed for quality attributions of variant new duties of every \( f(G_{i\alpha}) \) rated pair.

\[
\delta_i = \frac{1}{m - 1} \sum_{i=1}^{n} (G_{ij} - \mu_i)^2
\]  

(5)

It should be considered that the condition of using formula (3) is having \( G_i \) normal distribution. On the attribution of normal distribution [28], the possible density of random variables which are near mean, is higher. So, it can be said that the solution is of high possible density rating that is nearer to mean. Therefore, in solution propriety the solution is of high density which is nearer to mean and it would be the problem answer.

### 5. A Case Study of a Problem

An experimental problem is studied for proposed make propriety of responsibility solution and architecture structure through XP process in order to evaluate different aspects of represented method. This matter is related to a software production company which is producing trade-industrial software [29]. Since there is the experience of this software production it is going to have analyses to review on its production method.

#### 5.1 Quality Attribution of Software

For software [18] a series of under mentioned qualities are demanded:

<table>
<thead>
<tr>
<th>( \mu_i )</th>
<th>The mean of rating responsibility solutions I system</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta_i )</td>
<td>Covariance of ( G_i ) vector</td>
</tr>
<tr>
<td>( F(G_{i\alpha}) )</td>
<td>Rating density implement on ( x ) responsibility solution I process on software</td>
</tr>
</tbody>
</table>
• **Flexibility from Performance Points:** This quality attribution is called \( A_1 \), the architecture solution which can be attributed to this quality. Separation is representing service from data and is named \( D_1 \).

• **Applicability from Performance Velocity Point:** This attribution is called \( A_2 \). The solution that can be proposed to provide this attribution is replicate of data accessibility factors in different places (\( D_2 \)) or utilizing for parallel processing of commands (\( D_3 \)).

• **Applicant Accessibility Safety:** Safety is a basic factor in applicant confidence in trade software systems and factors that try to represent accessibility to applicants by hiding some data should be searched in wide level of architecture. This attribution is called \( A_3 \) and creating factors in order to control accessibility to information banks and proposed solution system is named \( D_4 \).

• **Time Cost:** This quality attribution is called \( A_4 \) and utilizing packs and completions of prepared codes as proposed solution is \( D_5 \).

### 5.2 Representing Several System Responsibilities and the Process of Architecture Formation

Two different responsibilities are assumed on the bases of quality attributions. Each one includes 2 quality attributions and 2 solutions (assuming that necessary stages for matrix formation for each responsibility is carried out i.e. all transmits and receives between architecture and pairs). Rating matrix is shown in table 4 on the bases of authors’ and experienced people experiment for every responsibility. Each responsibility is on the bases of VB.NET capabilities. Also, for stated attributions \( W \) matrix is defining attribution weight. These amounts are chosen according to rightness and importance of quality attributions.

\[
W^T = \begin{bmatrix} 0.6 & 0.4 \end{bmatrix}
\]

#### Table 4, The Density of Responsibility Solutions

<table>
<thead>
<tr>
<th>Responsibility 1</th>
<th>Responsibility 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D_1 )</td>
<td>( D_1 )</td>
</tr>
<tr>
<td>( D_2 )</td>
<td>( D_2 )</td>
</tr>
<tr>
<td>( D_3 )</td>
<td>( D_3 )</td>
</tr>
<tr>
<td>( D_4 )</td>
<td>( D_4 )</td>
</tr>
<tr>
<td>( D_5 )</td>
<td>( D_5 )</td>
</tr>
</tbody>
</table>

| \( A_1 \) | \( -5 \) | \( 3 \) | \( 2 \) | \( 1 \) | \( 2 \) |
| \( A_2 \) | \( 5 \) | \( 2 \) | \( 1 \) | \( 1 \) | \( 2 \) |

| \( A_1 \) | \( 2 \) | \( -5 \) | \( 5 \) | \( 1 \) | \( 0 \) |
| \( A_2 \) | \( -6 \) | \( 3 \) | \( 1 \) | \( 2 \) | \( 2 \) |

### 5.3 Solving Case Problem

First, for solving case problem \( G_i \) should be calculated for each responsibility through formula (1) and \( \mu_i \) through formula (2).

\[
G_1 = [1 \ 2.4 \ 1.4 \ 1.8 \ 1.2] \quad \mu_{i_1}=1.56 \\
G_2 = [-2.8 \ -0.2 \ 2.6 \ 1.6 \ 1.2] \quad \mu_{i_2}=0.48
\]

According to calculations, \( G_{i_1} \) graph or rates of rating vector solutions of each responsibility is drawn with \( \mu_i \) mean in figures (1) and (2). As it is shown solution 2 responsibility 1 has the most positive difference toward mean. Also, solution 3 has the nearest amounts toward mean. In responsibility 2 solution 1 has the most negative difference toward its mean.

#### Table 5, The Density of Responsibility Solutions

<table>
<thead>
<tr>
<th>Responsibility 1</th>
<th>Responsibility 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(G_{i_1}) ) = 0.248</td>
<td>( f(G_{i_2}) ) = 0.119</td>
</tr>
<tr>
<td>( f(G_{i_2}) ) = 0.131</td>
<td>( f(G_{i_2}) ) = 0.397</td>
</tr>
<tr>
<td>( f(G_{i_2}) ) = 0.392</td>
<td>( f(G_{i_2}) ) = 0.244</td>
</tr>
<tr>
<td>( f(G_{i_2}) ) = 0.374</td>
<td>( f(G_{i_2}) ) = 0.335</td>
</tr>
<tr>
<td>( f(G_{i_2}) ) = 0.331</td>
<td>( f(G_{i_2}) ) = 0.384</td>
</tr>
</tbody>
</table>

As it is observed solution 3 in responsibility 1 has the most rating density 0.392 and solution 2 the least density. Also, solution 2 in responsibility 2 has the most rating density 0.397 and solution 1 the least density. Therefore, solution 2 in responsibility 2 and solution 3 in responsibility 1 are the best choice among proposed solutions.

So, the result architecture is in this form for 2 responsibilities:

#### Table 6, Chosen Architecture of System Represented Responsibilities

<table>
<thead>
<tr>
<th>Responsibility 1</th>
<th>Responsibility 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D_1 )</td>
<td>( D_2 )</td>
</tr>
<tr>
<td>( A_1 )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>( A_3 )</td>
<td>( 2 )</td>
</tr>
</tbody>
</table>
As it is shown in table 5, according to rating density amounts solution 3 of responsibility 1 and solution 2 of responsibility 2 are as the best solutions and form table 6 system architecture.

For all responsibilities of software system, the architecture is created in this form i.e. if k responsibilities are defined for system, k suitable solution can be chosen for developing qualitative attributes of each responsibility, selection and system architecture. Finally, after architecture structure, a meeting composed of minimum numbers of customers, architecture, some of programmers and performing system is hold. The customer will represent his ideas over the estimated attributes and in the case of his disapproval, the participants will come to a tactic in order to reach it.

6. Conclusions and Future Plans

On the bases of following achievement future plans are stated:

• Preparing smart software from possible responsibilities including the most quality attributions and on the bases of represented method of solution propriety and considered responsibilities are demanded for identification of a basic architecture.

• The amount of rating matrix of passive responsibilities quality attributions are calculated through this method. For this purpose \( f(G_x) \) is calculated and rating matrix produced from numeric methods. On this basis architecture can be proposed which is of less calculated responsibility ratings.

• Exploring and omitting weaknesses of product architecture are gradually and easily performed in XP process.

References


Appendix

- Extreme Programming
- Quality Attribute Workshop
- Attribute Driven Design
- Analytic Hierarchy Process