GUIDE TO DESIGN PATTERN SELECTION BASED ON MAS TECHNOLOGY

Eiman M. Sahly, Rabeia N. Abdleati, Omar M. Sallabi
Department of Software Systems Engineering
Faculty of Information Technology
Benghazi University, Libya
eiman.sahly@benghazi.edu.ly, rabeia.abdleati@benghazi.edu.ly, omar.sallabi@benghazi.edu.ly

ABSTRACT
Working with design patterns made significant progress towards the goal of improving the quality of software production. One of the major issues confronting used design pattern is the appropriate selection for them. However, developers of experienced are able to choose the occasion design patterns for a given problem effectively; unluckily, are difficult task for inexperienced. Thus, there several tools for help choose of pattern. In this paper presents our attempt to use agent technology to enhance assistant the developers to selecting fits design pattern that need to solve their problem. This study focused on the improving the process of pattern selection through proposing an architecture Design Pattern Selection (DPS) based on Multi-Agent System (MAS) supports to obtain the appropriate recommendation.

KEY WORDS

1. Introduction
Over the last few decades, the field of software engineering has witness dramatically growth. Recent studies show that software evolution software systems are increasing in complexity every day [1]. Using of design pattern can build higher software quality, a reduced development cost and ease of maintenance. Design pattern [2] is a generic proven solution to a common recurring software design problem. Moreover, represent a highly effective way to improve the quality of software engineering. Due to its ability to capture the best practices and design knowledge based on real experience in field of software engineering. It is a powerful tool for the developer, thanks to the many benefits such as; communication tool, documentation the best practices, capture the best design knowledge, enrich the vocabulary, flexibility of maintenance and reusability [2] [3].

The design patterns on software quality have attracted increasing attention in the field of software engineering, for the purpose of improving and produce high quality in less time. The main objective of studying design patterns is to reduce development efforts by enabling developers to select a suitable design pattern for their need. One of the main challenges in the research is how to choose suitable patterns that can solve a particular problem, and how to apply the selected pattern during system design.

In fact, expert developers who have a deep knowledge of patterns can a select suitable pattern to specific design problems [4]. On the other hand, the novice developers it is so difficult task to find suitable design pattern for solving a given design problem, therefore, has led to significantly reduce the use of patterns the following reasons; (1) they don’t have concrete definition or a clear understanding of what the problem domain is [5]. (2) The novice developers not familiar with design pattern and do not have enough knowledgeable about it for decide whether reuse patterns or develop a special-purpose solution [4].

Over the past years of research in field of software engineering have been a growing number of design patterns [6]. This increases the degree of challenges that face the software designers in determining the suitable design patterns to solve their problems. Over time, the patterns have become a complementary part of many development approaches. However, many attempts have been made to assist in the selection pattern, which was reported in the literature [16],[17].

In this paper, we addressed the problem of design pattern selection. We describe our steps of proposed solution and DPS an architecture that based on MAS technology. In fact, the purpose of using agent technology as an aid to enhance recommendations is useful technique for selection. DPS is use combining various techniques (e.g. IR, CBR, and Expert System). The scope of this research was limited to find the right pattern from queries submitted by the user. Our aim, in this research, is to provide assistance to both novice and expert developers in the following:

- Assisting and advising: novices are helped by giving them the necessary recommendations and suggestions for a given problem.
- Guidelines: support how to implement pattern.
- Increase ability: developers learn new design patterns through navigations from one pattern to another.
- A collaborative environment: facilitating experience sharing among them, as well as support knowledge transfer to the novice.
The rest of this paper discusses the challenges of design pattern selection (Section 2), review related work (Section 3), describe our contribution in solve problem pattern selection (Section 4), and conclusions and future work (Section 5).

2. Pattern Selection

Since the last decade, emerged the problem of selecting patterns. The authors GoF book [1]; stated the difficulty of selecting suitable design pattern: “... With more than 20 design patterns in the catalog to choose from, it might be hard to find the one that addresses a particular design problem, especially if the catalog is new and unfamiliar to you”. Since then, the problem of selecting patterns still exists. Moreover, there are steadily increasing number of patterns in several researches at conferences and workshops. Additionally, there are many published books of the new patterns and online repositories. As a result, become more critical issue, Due to its difficult to search and select fitting patterns. (Figure 1 shows our current sampling for existing pattern in terms of the year they were published in the available literature review). This should be seen as example, and not as a limitation of the actual number of patterns.

![Figure 1. Number of patterns existed between in 1995 - 2012](image-url)

From the above Figure, we have found that long list of pattern in three surveys [9], [10], [12]. Henninger and Corrêa [10] counted 2241 software patterns were created in 1994 – 2007, and Bunke et al [12] collected the published security patterns in the period of 1997 to mid 2012. They proposed a new classification scheme that summarizes 415 Security patterns. Moreover, in recent years, published several books of the new patterns; we present three books [7],[8],[11],[13],[15], and the most popular SOA Patterns book by Thomas [2].

In fact, do not stop number of patterns at this limit, but in an ongoing increase. However, the process of selection a suitable pattern to the particular problem becomes more complicated issue, and existence of necessary tools to assist in this process is extremely important. Therefore, there are many different approaches exist to addressing to pattern selection.

3. Related Work

In this section, we review some existing approaches that attempted to selecting a suitable design pattern.

Nadia et al [18] presented an interactive tool to assists the designers choosing their suitable design patterns; through gives a bitty to draw a design fragment, present the problem, re-phrases the problem in order to obtain the intention of a certain pattern via use of WordNet which lexical dictionaries. Then, it explores the candidate solutions by filtering patterns that meet the intentions through the use of recommendation rules. They didn't feedback from the users and limited search queries.

Birukou et al [19] presented an approach use of a multi-agent system (MAS) based on the Implicit Culture (IC) framework to help designer in selecting patterns by getting suggestions from the group. Repository built only on 23 design patterns from GoF book. Depending on the match between problems of design pattern selection with problem of selecting web links relevant to keyword. Therefore, they used the simulator developed for the application of the IC Multi-Agent Platform to web search. The results have shown that an increase in the number of users causes an increase in the recall of the suggestions produced by the system.

Guéhéneuc et al [20] presented a first prototype as a simple recommender system for design patterns. This approach is based on extract important words by analyzing the textual GoF patterns. In their application they didn’t use collaborative filtering and feedback from the users, rather they just choose important words.

Díaz et al [21] proposed a module for a recommendation tool embedded in a visual environment, according to a preliminary study that recommends pattern based on the selection designers, used collaborative filtering techniques. With a view to supporting designers in making the correct design decision. It supports a novice user to understad a design patterns language.

Hasheminejad et al [22] proposed a method based on the text classification approach. It has simple automation of pattern selection process, for suggest the suitable design patterns according to the degree extent of similarity between the problem definitions of the retrieved design pattern. The proposed method has four phases, preprocessing, learning design patterns classifiers, determination of a design pattern classes, and suggestion of design pattern(s).

Suresh et al [23] proposes a prototype methodology to find a suitable pattern to the user. The methodology has two search scenarios and the three algorithms for finding the suitable pattern. They used information retrieval techniques and social recommendations to recommends pattern.

Palma et al [24] proposes a DPR prototype for suggesting design patterns, based on a simple Goal-Question-Metric (GQM) approach, and knowledge-base (KB) for pattern details and relative information. They presented a sample interactive session with the designer. DPR prototype was inspired from a previous work [25].

Kung et al [25] presented a prototype of an expert system for suggesting design patterns, which is selection based on ranking through interactive session (asking
questions) with the users that helps narrow down the selection process. They used a rule base as the knowledge base implemented the 23 GoF design patterns.

We noticed in the previous approach, that each has its properties and features. The essence of our idea is integration of these features and combines it together in one method. As a result, improve the selection process.

The closest work is presented in [23]; we try to propose a similar approach with addition of expert system, agent technology for improvement and development.

4. The Approach for Design Patterns Selecting (DPS)

In this section we present the steps of proposed solution of approach. Algorithms proposed, architecture of the DPS based on MAS technology, and example.

4.1 Steps of Proposed Solution Approach

We proposed four steps to solve the problem of selecting appropriate design pattern; (Figure 2 illustrates a conceptual overview of it).

Step 1: Identify Design Problem

<table>
<thead>
<tr>
<th>QDP</th>
<th>DLU</th>
</tr>
</thead>
</table>

Step 2: Retrieving Patterns

| QSPQ algorithms | QMP algorithms |
| QAS algorithms | CIK algorithms |

Step 3: Recommendations

| Recommending patterns | Recommending pattern sequences |
| Recommending practical | Recommending apply patterns |

Step 4: Evaluation

| Questionnaire |

Figure 2. Illustrates the steps of solution strategy method

Step 1: Identify design problem

Identify two combined in this step, Determine the Level of User (DLU), and receives Queries contain the Description Problem (QDP). In fact, the users identify the descriptions of their problem through submit queries to obtain the suitable patterns, and the proposed approach depends on the three user’s level that has (Novice, Advanced beginner and Expert). In order to determine the user's level, three questions will help us to determine his/her level. (Table 1 summarizes these questions), and the list of answers that used to identify user’s level.

<table>
<thead>
<tr>
<th>Q1</th>
<th>How many design patterns do you know?</th>
</tr>
</thead>
<tbody>
<tr>
<td>None – less 3 - 23GoF – all 23GoF and more.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q2</th>
<th>Do you work with design pattern?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not at all - Sometimes – Always</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q3</th>
<th>Are you write and publish your design patterns?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No – yes</td>
<td></td>
</tr>
</tbody>
</table>

From the previous table, Q1 focuses on identifying the user’s level (i.e. user is a novice user if the choice none or less than 3, then there is no need to answer the further questions Q2 and Q3). If a user selects (23GoF or all 23GoF and more), then the user is not a novice user but, answer of Q2 is required to identify his/her other level. Furthermore, Q2 is used to distinguish between advanced beginner and expert. The (not at all) option means novice group and the option (Sometimes) is advanced beginner group but the option (Always) is not certain that an expert group. The last question, Q3 confirm that the user in the expert group. (In Figure 3 illustrates the algorithm to determine the level of user).

Figure 3. Illustrate the algorithm to determine the level of user (DLU).

Step 2: Retrieving patterns

In our current work, finding suitable pattern is based on many strategies, and it depends on three scenarios situations (See Figure 4). Our approach proposed use four algorithms (QMP, QSPQ, QAS and CIK); these algorithms will be explained in (section 4.2).

In fact, the system starts scenario1 when submitting a query via (novice or advanced beginner or expert) user to search pattern needed. This scenario applies two algorithms parallel (i.e. QMP, QSPQ). In fact, sometimes users’ queries fail to describe completely their requirements clearly. Hence, apply parallel matching query intent with pattern intent and search similar query with previous user’s queries. However, filter process via
intersect result two algorithms, if filter process is found and satisfies minimum thresholds of confidence then suggest pattern and storage. Else if not found satisfactory patterns then checking in user level; if the user is an expert then continue with scenario3 (i.e. CIK). Else, continue with scenario2 (i.e. QAS); question and answer session would begin. May not find the appropriate pattern for several reasons, some of patterns maybe don’t have the knowledge base in repository or maybe the users do not specify correct domain of the problem. Therefore, continue with scenario3 with interaction collaborative which present the problems on the group of experts to find a solution for design problem (See Figure 5).

<table>
<thead>
<tr>
<th>Q1</th>
<th>Does the recommendation help you to solve the problem?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2</td>
<td>Is the use of our system easy?</td>
</tr>
<tr>
<td>Q3</td>
<td>Are the suggested solutions satisfactory for you?</td>
</tr>
<tr>
<td>Q4</td>
<td>What is your assessment of the recommendation?</td>
</tr>
<tr>
<td>Q5</td>
<td>How useful was the system to understand the patterns?</td>
</tr>
<tr>
<td>Q6</td>
<td>How would you improve the quality of the solutions?</td>
</tr>
</tbody>
</table>

Figure 4. Illustrates three scenario of potential to find the suitable pattern

**Step 3: Recommendations**

The recommendations contribute significantly to raising the efficiency of users through influencing their selection. There are four types of recommendations supported by the proposed method:

- Recommending patterns: provide suitable patterns for solving a particular problem.
- Recommending pattern sequences: Showing the recommendations consist of sequences of patterns usually apply together that depending on relationship between patterns (patterns language).
- Recommending apply patterns: provide recommendations how to implement the pattern(s) (i.e. by giving guidance how to implement pattern).
- Recommending practical: provide a list of users who have used this pattern(s) before (i.e. Suggests users who have previous experience in the application of the same pattern which proposed).

**Step 4: Evaluation**

In the last step, we evaluate process the effectiveness of this method. The main purpose of the evaluation is to assist of identifying necessary improvements for the proposed method. Using a questionnaire to ask questions to the users to find out whether the system helps choose the appropriate patterns as well as suggestions for improvement. The following table provides a few these questions:

**Table 2**

<table>
<thead>
<tr>
<th>Questions about evaluate the recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
</tr>
<tr>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
</tr>
<tr>
<td>Q4</td>
</tr>
<tr>
<td>Q5</td>
</tr>
<tr>
<td>Q6</td>
</tr>
</tbody>
</table>

From the previous table, Q1, Q3 and Q6 to give assessment about solutions offered of the system how useful for the users. Q5, evaluate the contribution of the system to assist in the understanding the patterns.Q2, gather their impression on whether the system was easy to use, and Q4 evaluate about the effectiveness of the recommendations.

**4.2 Algorithms**

The purpose of the following algorithms is to find a suitable pattern based on the user's query to solve a specific design problem (see Figure 5).

- **Algorithm 1: Query-Matching-Pattern (QMP)**
  
  **Purpose**: The purpose of this algorithm is to parse and analyze the text user's query to find matching between patterns intents and the given query. (See Figure 6) shows the pattern of recovery processes using the QMP algorithm.

  We propose applies approach in [27]; allows the retrieval of design patterns from the specific design problems. They use of Information Retrieval (IR) techniques [26]; the most common model in IR is Vector Space Model (VSM) that allows users to search for information based on their needs.

- **Algorithm 2: Query-Similarity-Previous-Query (QSPQ)**
  
  **Purpose**: This algorithm works on search similarity between the requested query and previous users' queries to find recommends pattern (See Finger 7).

  We propose applies approach of Formal Concept Analysis (FCA) support Case-Based Reasoning (CBR) in [28-30].
• Algorithm 3: Question-Answer-Session (QAS)

**Purpose:** This algorithm focuses on finding the pattern based on question-answer session that helps narrow down the selection process, questions will vary according to the previous answers of the user and calculates weights obtained for to recommendations suitable as shown in (Finger 8).

However, we propose applies approach partition the questions into different levels in [25].

In particular, we propose to develop the knowledge base by experts’ users which enrolled in the system and not just rely on the system administrator. More specifically, the development process rule-bases are done by a group of experts who intersect experience in the same domain as shown in the (Finger 9), for develop the rule-base of pattern.
There is a relation between design pattern and application domains. That can be defined in formal logic as \((\exists P)(P \in D)\) where \(P\) is a design pattern, and \(D\) is application domains in different fields such as (software architecture, communications, user interface, mobile application, real-time system, etc.). Also, we can describe relation between expert users \(E\) with \(D\) as \((\exists D)E(D)\). Consequently, each pattern has rules \((R)\) in the knowledge-base (KB). Also, \(E\) intersects experience in same \(D\), which can be defined as \(E_i(D, P_i) \cap E_2(D, P_i) \ldots \cap E_n(D, P_i)\). Therefore, it can be written as:

\[
\cap_{k=1,n} E(D, P_i) = 1, 2, 3, \ldots m, i \in D
\]

Where,

\(P_i; i = 1, 2, 3, \ldots m, i \in D\).

Therefore, a simple measures rules are defined that can drive development or evaluate of the knowledge base automatically. The rule can be in one of the following two forms:

\[
R(D, P_i) \notin KB \rightarrow Build[\cap_{k=1,n} E(D, P_i)]; i \in D
\]  

\[
R(D, P_i) \in KB \rightarrow Evaluate[\cap_{k=1,n} E(D, P_i)]; i \in D
\]

Note that the value of \(R(D, P_i)\) is building \(R\) of \(P_i\) by users defined in the form (1), if \(P_i\) is don’t have rule in KB as in the form (2). Alternatively, in the form (3), \(R(D, P_i)\) is evaluated by users defined in the form (1), when \(P_i\) have \(R\) in KB.

- Algorithm 4: Collaborative-Implicit-knowledge (CIK)

**Purpose:** The purpose of this algorithm is the transfer of implicit knowledge through collaboration with communities of users to find the pattern as shown in (Figure 10).

- Blackboard: Showing problem and solutions proposed by experts in a given domain formerly with problem, and the possibility of a vote for each a solution (i.e. this work similar as; Yahoo! Answers\(^1\), stackoverflow\(^2\)).

- Observer: Has several functions; receive domain of the problem displayed on (Blackboard), find a group of experts in this domain, send a group of experts to (User Sponsor), receiving suggestions from (User Sponsor), display in (Blackboard), inform the user about changes in (Blackboard), receive (feedback) from the user and storage.

- User Sponsor: Receives a group of expert from (Observer), Informing of the expert group for the problem displayed on (Blackboard), and send suggestions solutions to (Observer).

Receives a request from the user includes a description of the problem and which domain. The problem is displayed on the Blackboard and domain sends to the Observer, the Observer sends queries to the database for users interested in the same domain. Subsequently, it sends a list of those users to the User Sponsor for interact with users and send all suggestions solution of the problem presented in the Blackboard to Observer.

The observer used rules to store documentation of new knowledge. It can be represented in pseudo-code as follows:

1. **If request** \((U_a; problem = a; domain = b)\) then 
   \[\text{Suggest} (U_a; n \times \text{pattern})\] // meaning (*), any suggestions from other users.

2. Compute a score of weighted pattern for each suggestion submitted.

3. **If Accept** \((U_a; \text{Suggest} (U_b; n \times \text{pattern}))\) then

   - Strong \((U_a; \text{problem = a; domain = b; Suggest} (U_b; n \times \text{pattern})\) \(; \text{PWS}\)) // PWS = pattern weighted score.

- Observer: Has several functions; receive domain of the problem displayed on (Blackboard), find a group of experts in this domain, send a group of experts to (User Sponsor), receiving suggestions from (User Sponsor), display in (Blackboard), inform the user about changes in (Blackboard), receive (feedback) from the user and storage.

- User Sponsor: Receives a group of expert from (Observer), Informing of the expert group for the problem displayed on (Blackboard), and send suggestions solutions to (Observer).

Receives a request from the user includes a description of the problem and which domain. The problem is displayed on the Blackboard and domain sends to the Observer, the Observer sends queries to the database for users interested in the same domain. Subsequently, it sends a list of those users to the User Sponsor for interact with users and send all suggestions solution of the problem presented in the Blackboard to Observer.

The observer used rules to store documentation of new knowledge. It can be represented in pseudo-code as follows:

1. **If request** \((U_a; problem = a; domain = b)\) then 
   \[\text{Suggest} (U_a; n \times \text{pattern})\] // meaning (*), any suggestions from other users.

2. Compute a score of weighted pattern for each suggestion submitted.

3. **If Accept** \((U_a; \text{Suggest} (U_b; n \times \text{pattern}))\) then

   - Strong \((U_a; \text{problem = a; domain = b; Suggest} (U_b; n \times \text{pattern})\) \(; \text{PWS}\)) // PWS = pattern weighted score.

- Observer: Has several functions; receive domain of the problem displayed on (Blackboard), find a group of experts in this domain, send a group of experts to (User Sponsor), receiving suggestions from (User Sponsor), display in (Blackboard), inform the user about changes in (Blackboard), receive (feedback) from the user and storage.

- User Sponsor: Receives a group of expert from (Observer), Informing of the expert group for the problem displayed on (Blackboard), and send suggestions solutions to (Observer).

Receives a request from the user includes a description of the problem and which domain. The problem is displayed on the Blackboard and domain sends to the Observer, the Observer sends queries to the database for users interested in the same domain. Subsequently, it sends a list of those users to the User Sponsor for interact with users and send all suggestions solution of the problem presented in the Blackboard to Observer.

The observer used rules to store documentation of new knowledge. It can be represented in pseudo-code as follows:

1. **If request** \((U_a; problem = a; domain = b)\) then 
   \[\text{Suggest} (U_a; n \times \text{pattern})\] // meaning (*), any suggestions from other users.

2. Compute a score of weighted pattern for each suggestion submitted.

3. **If Accept** \((U_a; \text{Suggest} (U_b; n \times \text{pattern}))\) then

   - Strong \((U_a; \text{problem = a; domain = b; Suggest} (U_b; n \times \text{pattern})\) \(; \text{PWS}\)) // PWS = pattern weighted score.

- Observer: Has several functions; receive domain of the problem displayed on (Blackboard), find a group of experts in this domain, send a group of experts to (User Sponsor), receiving suggestions from (User Sponsor), display in (Blackboard), inform the user about changes in (Blackboard), receive (feedback) from the user and storage.

- User Sponsor: Receives a group of expert from (Observer), Informing of the expert group for the problem displayed on (Blackboard), and send suggestions solutions to (Observer).

Receives a request from the user includes a description of the problem and which domain. The problem is displayed on the Blackboard and domain sends to the Observer, the Observer sends queries to the database for users interested in the same domain. Subsequently, it sends a list of those users to the User Sponsor for interact with users and send all suggestions solution of the problem presented in the Blackboard to Observer.

The observer used rules to store documentation of new knowledge. It can be represented in pseudo-code as follows:

1. **If request** \((U_a; problem = a; domain = b)\) then 
   \[\text{Suggest} (U_a; n \times \text{pattern})\] // meaning (*), any suggestions from other users.

2. Compute a score of weighted pattern for each suggestion submitted.

3. **If Accept** \((U_a; \text{Suggest} (U_b; n \times \text{pattern}))\) then

   - Strong \((U_a; \text{problem = a; domain = b; Suggest} (U_b; n \times \text{pattern})\) \(; \text{PWS}\)) // PWS = pattern weighted score.
After the personal agent receives a query, then, forwards it to retrieval agent.

Retrieval agent uses four algorithms works according to what has been expounded in (section 4.2). Filtering agent receives a message from agent retrieval and send suggest patterns to the personal agent, then displayed to the user.

User apply these patterns proposed, and must given which the patterns were suitable. In contrast, in the case of rejection of the proposed patterns is switch algorithm as shown in (Figure 5)

4.4 Example of Our Approach

Based on the problem illustrated on [19], in this section, we use an example to illustrate DPS how to select the suitable design pattern. Let us suppose that the DPS uses a repository of security patterns. Furthermore, let us assume have four users {X, Y, Z, W}, Where X, Y have earlier registered in the system (X, Y is from the group expert). Z, W has not been registered. In practice, when Z or W needs to access the system to solve their problem; they need to register before using the system to determine the level (See table 1 and Figure 3 in section 4.1).

Let us now consider Z answered Q1 is (less 3); and W answered Q1 is (all 23GoF and more), and continue with Q2 (Always) is and Q3 is (yes). Other word, Z is level novice group and W is expert group (See Table3). However, the system continues with Z and W to complete the collection of information (e.g. names of patterns used by field of interest, etc.)

Table3

<table>
<thead>
<tr>
<th>User</th>
<th>Level of user</th>
<th>Interested Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Expert group</td>
<td>Real Time</td>
</tr>
<tr>
<td>Y</td>
<td>Expert group</td>
<td>security, software architecture</td>
</tr>
<tr>
<td>Z</td>
<td>Novice group</td>
<td>Web application</td>
</tr>
<tr>
<td>W</td>
<td>Expert group</td>
<td>Security</td>
</tr>
</tbody>
</table>

Assuming that user Z, who need to solve problem in a system. Where:

- X send Quire (Keyword="access control in a system that offers multiple services", Domain="Security") to the system
- Z send Quire (Keyword="I need to improve access control in a system that offers multiple", Domain="Security")

Suppose the user X uses the system and got the need pattern (Single Access Point pattern).

Now we explain how the user Z uses the system that offers three scenarios with a potential to find the suitable pattern as following:

4.4.1 Scenario 1

The system starts scenario1 regardless of the level of user. This scenario applies two algorithms (QMP, QSPQ) on parallel.

1. Result from QMP (Keyword, Domain) =
2. {Single Access Point, Policy Enforcement Point}  
3. Result from QSPQ (Keyword, Domain) =
4. {Single Access Point, Policy Enforcement Point, Role Based Access Control}
5. Filter Process = {Single Access Point, Policy Enforcement Point} ∩ {Single Access Point, Policy Enforcement Point, Role Based Access Control}
6. Recommending (Single Access Point, Enforcement Point)
7. Feedback (Accept, Single Access Point)
8. Storage new case (See [29] for more details on CBR)
9. Present recommending of pattern sequences and recommendations how to implement the pattern, and a list of users who have used this pattern(s) before.

4.4.2 Scenario 2

The second scenario will be executed if the user are novice and he/she has no experience in choosing the right suitable key words to explain the problem, in case this scenario failed to give the desired patterns because the required patterns has no predefined knowledge base

4.4.3 Scenario 3

The second scenario will be executed in two cases:

- The user is expert because the scenario 1 failed to give the patterns since they are new and not found in the repositories.
- The user is novice and he/she can’t get the desired patterns because the required patterns have no predefined knowledge base.

Idea this scenario display problem on (Blackboard) by (Observer) Receive domain and retrieved a list of experts (e.g. Y,W in domain Security but X not same) informed expert users in this system for this problem on (Blackboard)

5. Conclusion

In this paper we discussed about our an approach to help selecting design patterns and assists developers to learn a design patterns that it might produce produce better solutions or more complete solutions. The proposed approach can be serving as a guide for new developers to find their needed design pattern according to recommendations given to them. In the paper, we have several techniques was used such as; information retrieve techniques, expert system, Retrieve a previous case (i.e.
References