DPS: Overview of Design Pattern Selection Based on MAS Technology

Conference Paper · May 2013

DOI: 10.1007/978-3-319-00551-5_30

Eiman Sahly
University of Benghazi
4 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Omar M. Sallabi
University of Benghazi
17 PUBLICATIONS **49** CITATIONS
SEE PROFILE

DPS: Overview of Design Pattern Selection Based on MAS Technology

Eiman M. Salah[1], Maha T. Zabata[1], and Omar M. Sallabi[1]

[1] Dept. of Computer Science, University of Benghazi, Libya

{eiman.sahly, maha.zabata,omar.sallabi}@benghazi.edu.ly

Abstract The design patterns have attracted increasing attention in the field of software engineering, since effectively selecting the fits pattern for a given problem can seriously improve the quality of the software, on the contrary of the expert developers selecting the suitable pattern process consider to be critical phase especially for novice developers which have to be provided with mechanism to help them find a suitable pattern to a particular solution. This paper introduces a design pattern selection architecture (DPS) based on a Multi-Agent System (MAS) that aim to obtain the appropriate recommendation to reduce development efforts, facilitate and assist the developers in selecting the suitable patterns for their problems.

1 Introduction

According to [1] design patterns are a generic proven solution to recurring common design problems. It is consider as a powerful tool for a developer and important concept in software engineering, due to its ability to represent a highly effective way to improve the quality of software systems. Based on [2] [3] design patterns benefits can be summarized as following:

- Increase the flexibility of maintenance and reusability of software design.
- Contribution in extensibility of software through reducing the problem of architectural decay.
- Capture the design knowledge based on real experience of software design.
- Document the best practices in solving many different kinds of problems.
- A communication tool that improves the communication among software developers and provide a mechanism to share workable solutions between developers and organizations.

This work continues the proposed research in [4]; which is used three possible scenarios aiming to decide which pattern to select, based on four steps. A developer starts with the problem, which is formulated as a problem description; the problem description is used as a query. After retrieving patterns, the next step is to choose among these patterns, this process can be repeated several times, until the desired state (i.e. find the accepted pattern that solve the problem). Then, generate recommendations that help to applied and use the patterns. The final step is the system evaluation according to the user feedback in hand to enhance and refinement the system.

In the next section, the selection pattern problem is described. Section 3 mentions related work. In section 4, introduces our approach. Section 5, explain conclusions and future work.

2 Problem Statements

One of the main challenges in the researches is how to choose suitable patterns that can solve a particular problem, and how to apply the selected pattern during system design.

Actually expert developers who have a deep knowledge of patterns can select a fitting pattern to specific design problems [5]. Conversely, for the novice developers it is quite difficult to select the suitable pattern for a given design problem, as a result this has led to significantly reduce the use of patterns due to the following reasons:

- Novice developers don't have tangible definition or a clear understanding of the problem domain [6].
- Novice developers not common to the design pattern and do not have enough knowledgeable about it for decide whether reuse patterns or develop a special-purpose solution [5].

Obviously, the lack of knowledge about patterns and knowing only very few of them, can confronted us and affect our ability choosing which pattern to apply or choosing patterns that is not exactly fitting to the design problem within a program, the situation can be worse when applying a pattern wrongly or applying the wrong pattern for a problem.

In fact, recent active researches in field of software engineering stated that determining the suitable design patterns to a specific problem is a considerable challenge facing the software designers. However, the GoF book [1]; authors explains that it not easy selecting the pattern which addresses a particular design problem among 23 different design patterns especially with progressively increasing number of patterns due to the several researches at conferences and workshops, and many published books of the new patterns and online repositories, selecting the suitable design pattern become a critical issue.

Table 1. Shows sample of published patterns

Publication	Year	Number of Pattern
Almanac book [7]	2000	1200 Software Patterns

Henninger & Corrêa Survey [8]	1994 – 2007	2241 Software Patterns
SOA book[2]	2008	60 Patterns
Bunke el. Survey [9]	1997 - mid 2012	415 Security Patterns
Neil book [10]	2012	70 Mobile Design Pattern

consequently, the growing number of discovered patterns that exceed our ability to understand and analysis the exciting design patterns call for our need of necessary tools to assist in this extremely important process. Recently, the number of documented patterns has steadily increased through the new publications, consequently the need for automated and supporting processing tool of design patterns become more important as well as supporting the learning process for the understanding of concrete patterns. However, to gain benefits from design patterns, it is necessary to solve the previously mentioned problems by supporting the developers working with them. The main problem that motivated this paper is the need for effectively supporting approach-based software development by using multi-agent system (MAS) technology .MAS architectures could be seen as a social organization of independent software (agent) that flexible, autonomous, learning, reasoning and corporate with the environments. In addition, characteristics of agent structure can assist to decide which action is appropriate and enable to recommend the suitable suggestion patterns for a particular design problem.

3 Related work

This section describes existing approaches that try to support resolving the problems discussed previously. There are many different approaches used to addressing the pattern selection problem. Table 2, shows some approaches in the literature [11- 20].

Table 2. Illustrates the different approaches of exist for design pattern selections

Approach	Summary of Purposed	Technique
ESSDP [11]	Represent a methodology for constructing expert system which can support design patterns to solve a designer's design problem through Used dialog with the designer to narrow down the choices(question-answer session), that implements the All GoF pattern.	- Expert system
ESSSDP [12]	Develop of a prototype expert system for the selection of design patterns that are used in object-oriented software for providing an automated solution regarding the design pattern application problem, and implements only five GoF patterns.	- Expert system. -Object-oriented environment
RMDP[13]	Developed a information retrieving model of two major parts, the analysis of all GoF pattern document to create search index and the calculation of index weight, this model used to retrieving correct design patterns.	- Information Retrieval (IR)
SRSDP[14]	Introduce a simple recommender system to help user in	- Recommender

	choosing among the 23 design patterns from the GoF. This approach is based on extract important words by analyzing the textual GoF patterns.	System
IC-Pattern[15]	Developed a multi-agent system for recommending patterns, which takes into account the social part of the problem, providing users with suggestions from other community members about patterns that were supports conventional IR and Case-Base Reason methods for finding design pattern suitable for the given problem.	- Recommender System - MAS - IR - CBR
RDP[16]	Proposed an initial model to solve the problem by using Case-Based Reasoning (CBR) and Formal Concept Analysis (FCA). They use FCA as the way of inductive for extracting embedded knowledge in case base and allows the flexibility to maintain index.	- CBR - FCA
DPRA [17]	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.	- Recommender System - Draw a design fragment
RSVEISIG[18]	Described a recommendation tool embedded in a visual environment for pattern-based design which aims at suggesting patterns to help novice designers to produce better designs and understand the language.	- Recommender System - collaborative filtering
DPR[19]	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 [11].	- Recommender System - Expert system

In the previous approach, we noticed that each has its properties and features. The goal of our idea is integration of these features and combines them together in new approach which it contribute the improvement of the selection process.

4 The Proposed DPS Approach

4.1 The Architecture Overview

In this section, we propose the architecture of our DPS based on MAS technology. As shown in Fig.1, it has Three-Layered services can be divided into the following distinct logical layers:

- **Interface Layer**. This layer contains the user functionalities responsible for an interaction between the user and the system. It consist two components which are Login User and Description Problem.

- **Application Services Layer**. This layer it consists of eight components used to implement the core functionality of the system.
- **Infrastructure Layer.** This layer provides access to data. It consists of different of resources such as databases, repository, and etc.

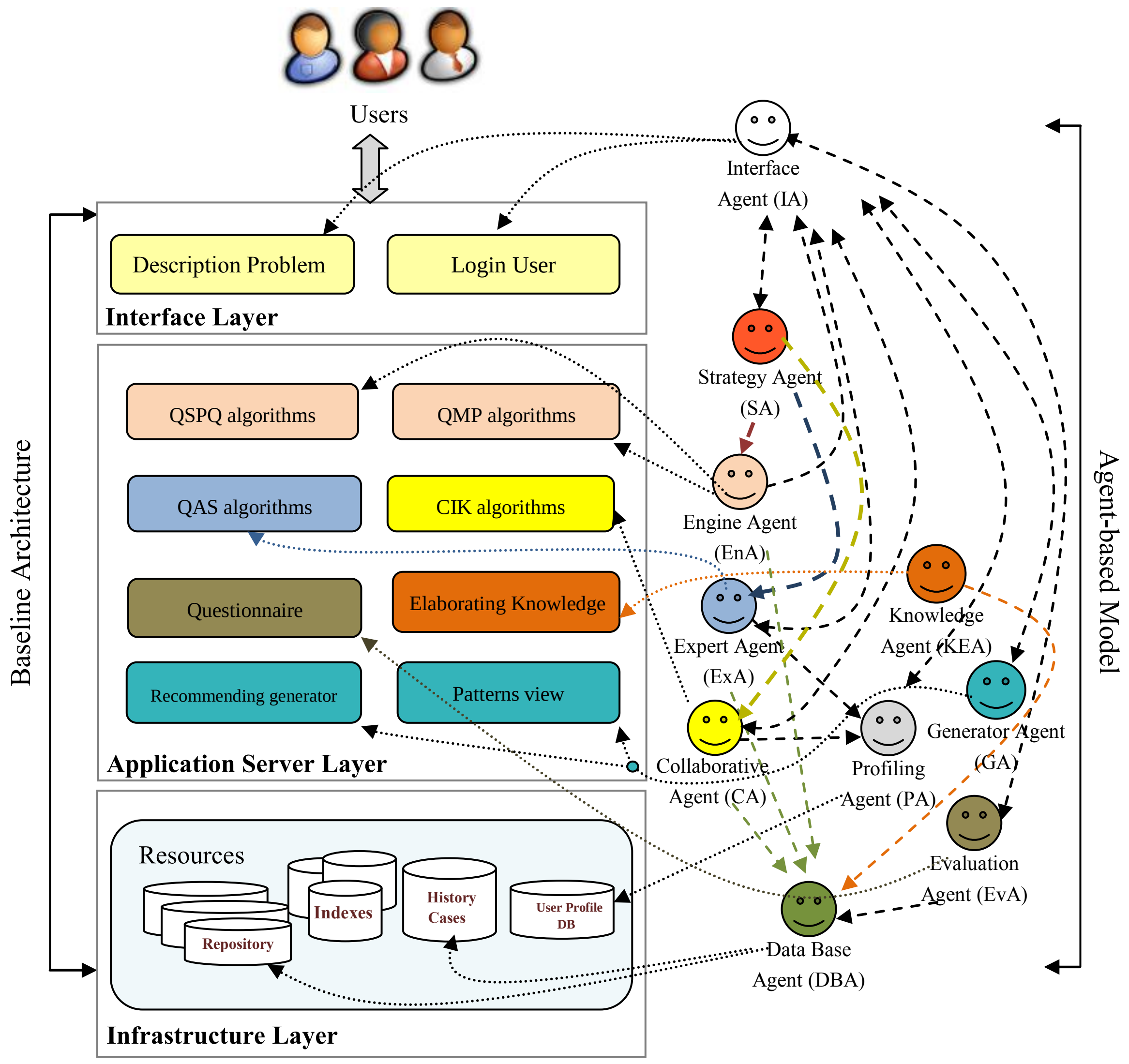

Fig. 1. DPS Conceptual Overview based on MAS.

From the previous Figure, ten agents have been used to achieve functionalities efficiently. Each agent has responsibility of performing function as follows:

- **Interface Agent (IA):** as an intermediary between the user and other Agents.

- **Profiling Agent (PA):** is responsible for the determine level of user (See in [4, table 1]) and updating an active user profile.
- **Engine Agent (EnA):** It Applies two algorithms parallel *QMP*, *QSPQ* (See [4, Figure 3, 5]), filter process by intersect result two algorithms, send this result to *IA*. If *IA* informs *EnA* Accept then inform *SA* for accept and inform *DBA* for storage. Else inform SA for failure.
- **Expert Agent (ExA):** It Applies *QAS* algorithms which performing the question and answer session with *IA*. If result *IA* is Accept then inform *SA* for accept and inform *DBA* for storage. Else invoke *CA* to begin.
- **Collaborative Agent (CA):** It uses *CIK* algorithm in [4, Figure 8], (i.e. This work similar as; Yahoo! Answers[1], stack overflow[2]).
- **Strategy Agent (SA):** Incorporates situation action rules to switch between three Scenarios as illustrated in Fig. 2. By following these rules will be avoid problems associated with: new pattern, keyword-search problem, and learn uses the patterns.

```
Scenarios 1:
    IF SA received message from IA for a user's query description of problem Then Send
      message to EnA for work.
        - QMP:  Analyzed query to find a match in stored pattern intents,
        - QSPQ: used the query to search for similarities to previous users' queries and
            reusing the previous solution for the new case.
        -Filter Process (FP):  result QMP ∩ result QSPQ
     Send message to IA for present the result of FP to user.
     IF EnA received message from IA inform user's accept the patterns suggested Then
        inform GA, DBA, PA
    Else inform SA is not accept the Apply Scenarios2
Scenarios 2:
   Send message to PA for determine level of user.
   IF the received message ∈ Expert Then executes Scenario3
   Else Send message to ExA for work // question and answer session with IA.
   IF ExA received message from IA inform user's accept the patterns suggested Then in-
       form GA, DBA, PA for work.
   Else Send message to CA for work.
Scenarios 3:
  CA Send message to PA for find $\bigcap_{k=1..n} E(D, P_i)$ // for more details see in [4]
  Send message to IA to inform user Suggest $\bigcap_{k=1..n} E(D, P_i)$
  IF CA received message from IA inform user's accept the patterns suggested Then in-
  form GA, DBA, PA for work.
```

Fig. 2. Strategy Agent action rules.

[1] http://answers.yahoo.com/

[2] http://stackoverflow.com/

- **Generator Agent (GA):** After the user fined the appropriate pattern, SA sends massage to *GA* for generate three recommendations to the user.
  - Recommending pattern sequences: Showing the recommendations consist of sequences of patterns usually apply together that depending on relationship between patterns (i.e. 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).
- **Data Base Agent (DBA):** It's responsible for the access to databases, data documentation, and informs the *KEA* about update the database.
- **Knowledge Agent (KEA):** Is an autonomous agent which works to update the knowledge base (for more detail see in [4]). It's depends at two equation as following:

$$R(D,P_i) \notin KB \rightarrow \text{Build}[\textstyle\bigcap_{k=1..n} \mathrm{E}(D,P_i)]: i \in D \qquad (1)$$

$$R(D,P_i) \in KB \rightarrow \text{Evaluation}[\textstyle\bigcap_{k=1..n} \mathrm{E}(D,P)]: i \in D \qquad (2)$$

- **Evaluation Agent (EvA):** Evaluate the effectiveness of the system, from the user feedback every time you using the system (See in [4, table 2].

5 Conclusion and future work

The selection a suitable design pattern is a worthwhile topic, important to the design community and it is an area that needs addressing design pattern selection problem and improving this process. In this paper, we presents DPS approach based MAS technology for generate advice for using design patterns and assists developers to learning a design patterns.

As future work we would like to find a solution for searching patterns problems [20] via the following points:

- Documentation techniques to:
  - Summaries patterns descriptions in a standard form to allow easy comparing of the patterns and allow the search using patterns forms.
  - According to researchers conducted in [21]; we thing of applying a patterns classification and clustering techniques to enhance the indexing, searching and selection of the patterns.
- Aggregation algorithms to collect patterns from different sources (books, proceedings, journals, etc), to single patterns repositories.

References

1. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of ReusableSoftware*. Addison-Wesley, Reading.
2. Erl, T. (2008). *SOA Design Patterns*. Prentice Hall, New York.
3. Knox, J. (Spring 2011). Adopting Software Design Patterns in an IT Organization: An Enterprise Ap- proach to Add Operational Efficiencies and Strategic Benefits. *M.S. thesis, AIM program, Dept of Computer and Information Science,* University of Oregon.
4. Saleh, E.M., Sallabi, O. (2012). Design Pattern Selection: A Solution Strategy Method. *In: 2012 IEEE International Conference on Computer Systems and Industrial Informatics* (ICCSII'12). Sharjah,UAE.
5. Sommerville, I. (2004). *Software Engineering*. Addison-Wesley, Boston.
6. Kim, D.-K., Khawand, C. El. (2007). An approach to precisely specifying the problem domain of design patterns. *Journal of Visual Languages and Computing*, 18 (6), 560-591.
7. Rising, L. (2000). *The Pattern Almanac*. Addison-Wesley, Boston, MA, USA.
8. Henninger, S., &Corrêa V. (2007). Software pattern communities: current practices and challenges. *In: PLOP '07: Proceedings of the 14th Conference on Pattern Languages of Programs*, ACM, NY, USA.
9. Bunke, M., Koschke, R., & Sohr, K. (2012). Organizing Security Patterns Related to Security and Pattern Recognition Requirements. *International Journal on Advances in Security*, 5:46–67
10. Neil, T. (March 2012). *Mobile Design Pattern Gallery UI Patterns for Mobile Applications*. O'Reilly Media.
11. Kung, D. C., Bhambhani, H., Shah, R. & Pancholi, G. (2003). An expert system for suggesting design patterns: a methodology and a prototype. *In: Software Engineering with Computational Intelligence.*
12. Moynihan, G.P., Suki, A., Fonseca, and D.J. (2006). An expert system for the selection of software design patterns: *Expert System Journal*, **23**(1).
13. Sarun, I., & Weenawadee, M. (2007). Retrieving Model for Design Patterns. *In.ECTI Transactions on computer and information technology,* 51-55.
14. Guéhéneuc, Y. and Mustapha, R. (2007). A simple Recommender System for Design patterns. *In: the 1st EuroPLoP Focus Group on Pattern Repositories.*
15. Birukou, A. & Weiss, M. (2009). Patterns 2.0: a Service for Searching Patterns. *EuroPLoP2009*, Irsee Monastery, Germany.
16. Sarun, I., & Weenawadee, M. (2009) .Retrieving Design Patterns by Case-Based Reasoning and Formal Concept Analysis. *ICCSIT International Conference on Computer Science and Information Technology*, 424-428.
17. Nadia, B., Kouas, A., Ben-Abdallah, H. (2011). A design pattern recommendation approach. *In:CORD Conference Proceedings*, 590-593.
18. Díaz, P. Malizia, A. Navarro, I. & Aedo, I. (2011). Using Recommendations to Help Novices to Reuse Design Knowledge. *In: IS-EUD*, 331-336. dio.10.1007/978-3-642-21530-8_35.
19. Palma, F., Farzin, H. Guéhéneuc, Y.G. & Moha, N. (2012).Recommendation System for Design Patterns in Software Development: An DPR Overview, In: Recommendation *Systems for Software Engineering (RSSE), 2012 Third International Workshop*.
20. Birukou, A.(2010). A Survey of Existing Approaches for Pattern Search and Selection. *In Proceedings of the 15th European Conference on Pattern Languages of Programs (EuroPLoP'2010).* Irsee Monastery, Germany.
21. Hasheminejad, S.M.H. and Jalili, S. (2012). Design patterns selection: An automatic two-phase method. *Journal of Systems and Software*, 2012. **85**(2). 408–424.