I.J. Information Engineering and Electronic Business, 2013, 4, 1-7
Published Online October 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2013.04.01

A Proposal of Expert System to Select Components for the Product Line Software Engineering

M. Rizwan Jameel Qureshi
Department of Information Technology, Faculty of Computing & Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
Email: rmuhammd@kau.edu.sa

Nora Farraj
Department of Information Technology, Faculty of Computing & Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
Email: farraj.nora@gmail.com

***Abstract*—** Programming companies in last decades have started to depend more and more on software components in developing their systems in order to save development time and increase the productivity of the company. However, this led to dramatically increase in the number of components, and selecting the appropriate component becomes a tough task. In this paper the authors propose an expert system to help developers choosing the best component fit their requirements. A survey is done to evaluate the efficiency of this proposed solution.

***Index Terms*—** Component, CBD, Selection, Retrieving, Classification.

I. INTRODUCTION

Component is a piece of independent highly tested code, which can be integrated with other components to build a complete system. Component based development (CBD) is very interested topic in software engineering. It saves time and efforts needed to develop high quality software programs. Today, there are a great number of components that have different features and functionalities. This makes the task of selecting the best component fit the developer's requirements very hard. A lot of efforts have been done by researchers to solve this problem. However, up to now companies still do not find the appropriate component selection system that eases this task. This is a great indicator that the previous solutions suffer some limitations make them not ideal for most of companies.

This paper proposes an expert system to help the software companies in component selection that overcomes the limitations in previous work. In next section, we provide related work to formulate the problem statement. In section 3 we propose a solution for this problem. After that we validate the efficiency of this solution and discuss the findings in sections 5 and 6 respectively. Then the conclusion is in section 7.

II. RELATED WORK

As programs become more complex and larger, the component-based development becomes more popular and the number of software components increased dramatically. With a lot of similarities and differences among these components, considering all available components and selecting the best one fits developer's requirements has become a difficult task. Extensive researchers' effort is made to solve this problem. One of the most cited papers by researchers discuss this problem is [1]. In this paper Lau and Wang present taxonomy for 13 component models based on the idealized component life cycle without any indication to the component's performance properties. Crnkovic et al. [2] contribute in improving the components understanding by providing a component model classification framework based on principles divided into 3 dimensions: Lifecycle, construction, and extra-functional properties.

Analyzing the requirements is essential for good component selection. An approach for managing non-technical requirement is provided in [3], in which Carvallo, Franch and Quer extended ISO/IEC 9126-1 catalog to include non-technical quality factors such as licensing and reputation. Mancilla, Astudillo and Visconti in [4] reported that non-functional requirements, such as availability and security, are not directly rely on components, and thus they are difficult to address. In their article, they propose a combination of 2 existing techniques in order to divide non-functional requirements into groups and generate candidate components for each group.

There is an urgent need for a component model selection framework, as has been proved by Aris and Salim [5], in order to help developers choose the suitable component models to be used. They also propose a framework for component model selection in [6]. Using this framework, developers are capable of determining the best component to use by specifying the criteria of the needed component model. Fahmi and Choi [7]

proposed a conceptual classification framework that put into account the previously selected component for the similar requirements. This would reduce the component selection time; and thus the development time.

There are 3 processes commonly exist in most of the selection systems and frameworks [6-7]. These are:

- Preparation
- Evaluation
- Selection

In preparation process the comparison criteria is determined that is nothing but the user requirements. The evaluation process searches for components that meet the requirements. Then the selection process chooses the component best fit the requirements.

Considering large number of components in the selection process that must meet multiple specifications makes this process computationally high complex. Greedy approach and genetic algorithm are adopted by [8] to solve this problem. Comparing the results of using these algorithms with the results of choices made by experts shows the success of the former.

In both [9] and [10], the work is to improve the efficiency of the component retrieval method by taking into account the user's feedback on the previous retrievals. Shao, Zhang and Xu [9] take the advantage of data mining technology to extract reuse rules which are used in building a decision tree. Zheng, An, and Zhang [10] adopt the relevance feedback technology to reduce comprehension cost and ease the burden of providing precise query formulation by the user. Table 1 is a summary for the contributions of the related work and its limitations.

Table 1: The Contributions and Limitations in the Related Work

Title of Paper	Contribution	Limitations
Software Component Models [1]	Classification of 13 component models according to component life cycle	Does not provide special support for performance predictions
A Classification Framework for Software Component Models [2]	Classification framework for component model based on lifecycle, construction, and other functional properties.	The abstract does not clarify how the efficiency of this framework is validated.
Managing Non-Technical Requirements in COTS Components Selection [3]	ISO/IEC 9126-1 catalog extension to include non-technical quality factors.	No validation
Combining COSTUME and Azimut+ to Address Functional and Non-functional Requirements in Software Component Selection [4]	Classification of components according to non-functional requirements.	Nothing is done to prove the efficiency of this combination.
State of Component Models Usage: Justifying the Need for Component Model Selection Framework [5]	Proof the urgent need for a component model selection framework in both industrial and research community.	Reviewing only 2 papers related to industrial community, and 4 for research community is not enough to justify the problem. One of research questions deals with the available components models to the date, but the newest reviewed paper in research community is older by one year than [5] acceptance date.
Framework for Component Model Selection [6]	Propose a framework for component model selection helps developers in determining the best component to use by specifying the criteria of the needed component model	Vague word in the questionnaire. In results analysis, it is written that the framework has potential to be applied by developers, although the result of this question has a mean = 2.8 and mode = 3, with 1 indicates strongly agree and 5 the strongly disagree.
A study on Software Component Selection Methods [7]	Propose a classification framework that records the previously selected components for the similar requirements to reduce the component selection time.	The proposed approach assumes that user requirements are given as a set of keyword functionalities. Does not provide guidelines on how to address problems due to fix and incomplete component descriptions supplied by providers.
Approximation Algorithms for Software Component Selection Problem [8]	Adopt Greedy approach and genetic algorithm to make the process of selecting the component meets developer requirements is less complex.	They do not consider the dependencies between the requirements.
Research on Decision Tree in Component Retrieval [9]	Take the advantage of data mining technology to extract reuse rules which are used in building a decision tree in order to improve the efficacy of the component retrieval.	It is claimed that validity and feasibility of the proposed strategy is verified, but only the validity is verified.
A Component Retrieval Method Based on Query Vector Transfer [10]	Adopt the relevance feedback technology to reduce comprehension cost and ease the burden of providing precise query formulation by the user.	Zheng, An, and Zhang experimented the prototype by themselves, and it would be better to test by users not involved in the prototype building to guarantee unbiased results.

After reviewing the previous work it is the suitable time to formulate the problems statement of this paper as following:

Software development companies are lacking a suitable component model selection system to help them find the best one fit their requirements [5-7].

III. THE PROPOSED SOLUTION

In this paper the author purposes an expert system to solve this problem and to overcome the limitations of [6-8], and consider both technical and non-technical component requirements [3] and the user's feedbacks as [9] and [10].

3.1 The System Architecture

The general architecture of any expert system [11] can be specified proposed by the author includes the following:

- Database: stores the technical and non-technical components description depending on those mentioned in [3] and also the dependencies between the components –that is if there any component affects the functionality of other components.
- Knowledge Base: stores the comparison rules.
- Inference System: the engine that is responsible for comparison the user requirements and the stored components' descriptions using the rules stored in the knowledge base.
- Explanation System: provides explanations about why a particular component is suggested by the expert system.
- Knowledge Base editor: allows the knowledge engineer to add, remove, or update rules in the knowledge base.
- Graphical user interface (GUI): eases the communication between users and the rest parts of the expert system shell. It includes controls like lists, check boxes, and radio buttons to help the user determining the needed requirements and specifying the components already used in his system.

Figure 1 shows an overview of the component selection expert system architecture.

3.2 Component Selection Steps

The system purposed here includes the 3 processes mentioned in the literature review as the following:

1) Preparation: The user starts to determine the needed requirements using GUI controls, which guarantee the unified form of the requirements and their completion. Also the already exist components in the user's system are specified to consider the dependencies. All this is submitted then to the inference system.

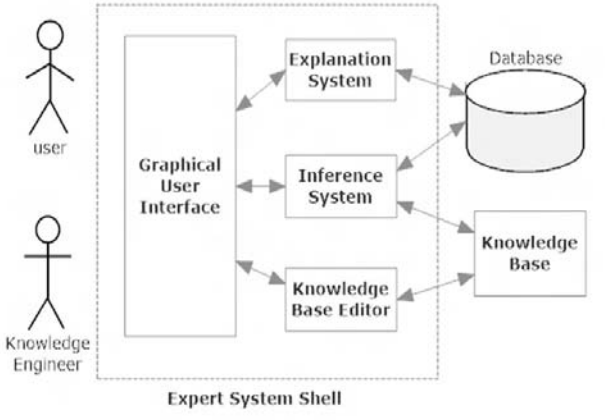

Figure 1: Component Selection Expert System Architecture

2) Evaluation: Here the inference system compares the user requirements and the stored components' descriptions using the rules stored in the knowledge base.
3) Selection: Here if the inference system found a component model that meets all the requirements, it is returned to the user. Otherwise, the inference system returns the closest component model and the explanation system clarifies the requirements met by this component and provides suggestions to the user to help him adapting this component to meet the other requirements. Then the user evaluates the selection process whether it succeed or not, and this can help in reducing the process time for similar requirements in future by storing new rules in the knowledge base.

3.3 Simple Example

The developer can specify the following as component's requirements using GUI controls:

1) Functionality: Send Report via Email
2) Security: access restricted by finger print
3) Licensing Cost: not more than $100

After submitting these requirements the interference system starts comparing these requirements with component descriptions stored in database using rules stored in knowledge base. After comparing, it finds a component that fulfills only the first and third requirements. The inference system retrieves the name of the component and the explanation system lists the features of it:

1) Functionality: Send Report via Email
2) Licensing Cost: $25
3) Availability: 90%
4) Network Cost: $500

Also the explanation system can provide an advice about how it can fulfill the missed security requirement by for example using additional component. Then user is free to provide his feedback about the selection operation.

IV. VALIDATION OF THE PROPOSED SOLUTION

In order to validate the efficiency of the proposed expert system, the authors have distributed a questionnaire. The questionnaire is divided into 3 issues based on the system stages from the user point of view:

1) Determining the component requirements: This stage takes the care of the interaction between the user and the system interface for determining the component requirements. The goal of this section in the questionnaire is to find the best and easiest way for the user select the properties and requirements of the component.
2) Retrieving the component model and advices: This stage considers the result retrieving. The result covers both component models and advises for fulfilling the missed features in the retrieved components.
3) Providing feedback about the result of the selection: Here the user can provide his opinion, whether this result is useful and the expected component is retrieved or this result is not useful. This can make future retrieves faster and more useful.

The questionnaire consists of 15 statements and allows the respondents providing their opinion about each of these statements. The questionnaire is in the appendix.

V. FINDINGS

Following are the findings divided into 3 sections, one per defined goal.

5.1 Cumulative Statistical Analysis of Issue 1. Determining the component requirements

Questions under this issue in the questionnaire test whether the proposed system facilitates the task of finding out the desired requirements in the needed component. The results are shown in table 2.

TABLE 2: Cumulative Statistical Analysis of Issue 1

Q. No.	Str. Disagree	Disagree	Neutral	Agree	Str. Agree
1	0	5	4	12	10
2	3	1	3	7	17
3	0	0	8	11	12
4	0	0	7	9	15
5	0	4	7	12	8
Total	3	10	29	51	62
Average	0.6	2	5.8	10.2	12.4
Percent	2%	6%	19%	33%	40%

As it is clear from the table 2, 40% and 33% of the sample strongly agree and agree respectively that the proposed system would help them determining the requirements and facilitate this task for them. While 6% and 2% of the sample disagree and strongly disagree respectively about this issue. The remaining of the sample, 19% chose to be neutral. Figure 2 shows these percent.

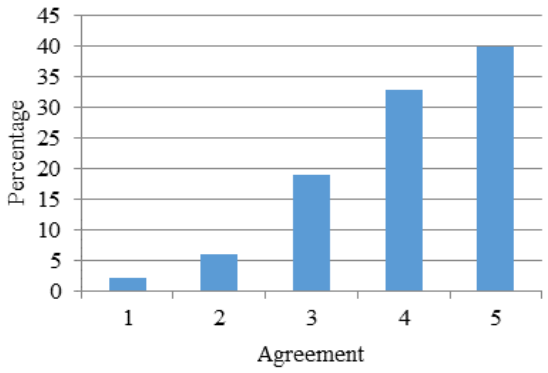

Figure 2: Graph shows the cumulative results of issue 1

This result shows that the developer takes time to determine the required features in the needed component, and it is difficult to determine all possible technical and nontechnical requirements. Using lists of these requirements classified into groups would eases this task for the developer. Also, because it is difficult for the developer to determine whether the required functionalities affect the ones already exists in the system, the user may mention the components already exists in the system that he afraid to negatively affected by the selected functionalities.

5.2 Cumulative Statistical Analysis of Issue 2. Retrieving the component model and advices

Questions under this issue in the questionnaire test the features provided by the proposed system when retrieving the component and find out whether these features are useful for the user or not. The results are shown in table 3.

TABLE 3: Cumulative Statistical Analysis of Issue 2

Q. No.	Str. Disagree	Disagree	Neutral	Agree	Str. Agree
1	0	0	6	10	15
2	0	0	6	11	14
3	0	6	6	10	9
4	0	0	9	8	14
5	0	5	6	12	8
6	0	6	3	9	13
7	0	6	6	9	10
Total	0	23	42	69	83
Average	0	3.29	6	9.86	11.86
Percent	0%	11%	19%	32%	38%

As it is clear from the table III, 38% and 32% of the sample strongly agree and agree respectively that the proposed system provides useful features for the user when retrieving the component. While 11% and 0% of the sample disagree and strongly disagree respectively about this issue. The remaining of the sample, 19% chose to be neutral. Figure 3 shows these percent.

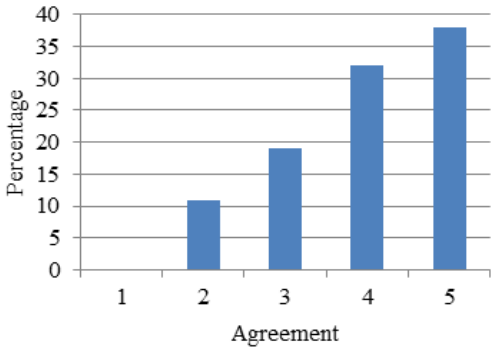

Figure 3: Graph shows the cumulative results of issue 2

This result shows that it would be helpful for the developer to know and understand why a specific component is retrieved, and which of the selected features and requirements is fulfilled by the selected component. If there is no component has all the needed requirements, the system would return the components that fulfill most of the requirements and would show the user a list of the missed requirement in order to help him in updating this component to fulfill these missed requirements. The system could make this task – updating the component- easier providing advices. Also a notification is raised if there is any negative effect between the functionalities of the retrieved component and the other components exist in the developer's system, and additional advices are provided to help developer remove this effect.

5.3 Cumulative Statistical Analysis of Issue 3. Providing feedback about the result of the selection.

Questions under this issue in the questionnaire test whether these users would be willing to provide feedback to speed up the future retrieves. The results are shown in table 4.

TABLE 4: Cumulative Statistical Analysis of Issue 3

Q. No.	Str. Disagree	Disagree	Neutral	Agree	Str. Agree
1	0	3	6	12	10
2	0	4	8	9	10
3	0	0	8	11	12
Total	0	7	22	32	32
Average	0	2.33	7.33	10.67	10.67
Percent	0%	8%	24%	34%	34%

As it is clear from the table 4, 2 equal percentages, 34%, of the sample strongly agree and agree that they are willing to provide feedback about the retrieved component for faster and better future results. While 8% and 0% of the sample disagree and strongly disagree respectively about this issue. The remaining of the sample, 24% chose to be neutral. Figure 4 shows these percent.

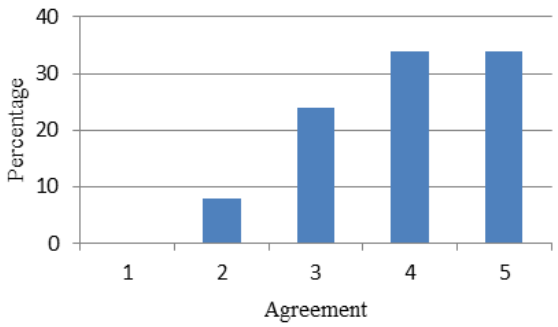

Figure 4: Graph shows the cumulative results of issue 3

The result shows that it would be great to take the advantage of the previous results for speeding up the retrieve operation for similar requirements. If the suggested component is not suitable for any reason, it would be useful to tell the system so it tries to search for another one.

5.4 The Final Cumulative Evaluation of all Issues

The result of cumulative statistical analysis of the three previous issues is shown in figure 5.

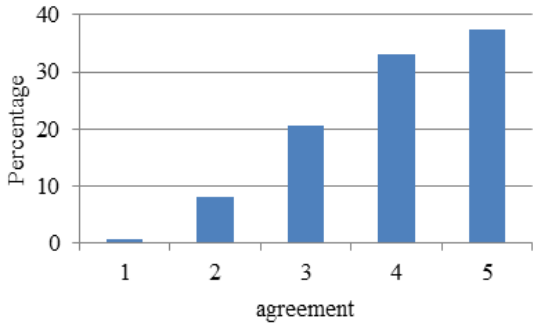

Figure 5: Graph shows the cumulative results of all issues

The results show the support of around 70% (33% agree and 37% strongly agree) of respondents for the component selection expert system.

VI. CONCLUSION

Large numbers of components spread recently, and selecting the best component fits the developer's needs has become a tough job. Many researches are done to make this job easier but unfortunately there is no ideal

solution up to now. In this paper, an expert system is proposed to solve the problem and help developers select the needed component. This system takes into account limitations in previous component selection frameworks and takes the advantage of the previous classification of the functional and non-functional requirements. The efficiency of every stage in the system is validated by a questionnaire. The cumulative results of the 3 stages shows the support of 70% of respondents and this result shows a great potential success of the proposed system. This system is going to be built as future work.

REFERENCES

[1] Kung-Kiu Lau; Zheng Wang; "Software Component Models," Software Engineering, IEEE Transactions on, vol.33, no.10, pp.709-724, Oct. 2007.

[2] Crnkovic, I.; Sentilles, S.; Vulgarakis, A.; Chaudron, M.R.V.;, "A Classification Framework for Software Component Models," Software Engineering, IEEE Transactions on , vol.37, no.5, pp.593-615, Sept.-Oct. 2011.

[3] Carvallo, J.P.; Franch, X.; Quer, C.; "Managing Non-Technical Requirements in COTS Components Selection," Requirements Engineering, 14th IEEE International Conference, vol., no., pp.323-326, 11-15 Sept. 2006.

[4] Mancilla, F.; Astudillo, H.; Visconti, M.; , "Combining COSTUME and Azimut+ to Address Functional and Non-functional Requirements in Software Component Selection," Chilean Computer Science Society (SCCC), 2010 XXIX International Conference of the , vol., no., pp.102-109, 15-19 Nov. 2010.

[5] Aris, H., Salim, S.S.: "State of component models usage: justifying the need for component model selection framework", Int. Arab J. Inf. Technol., 2011, 8, (3), pp. 313–320.

[6] Aris, H., and S. S. Salim. "Framework for component model selection." Software, IET 5.5 (2011): 474-486.

[7] Fahmi, S.A.; Ho-Jin Choi; "A study on software component selection methods," Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on, vol.01, no., pp.288-292, 15-18 Feb. 2009.

[8] Haghpanah, N.; Moaven, S.; Habibi, J.; Kargar, M.; Yeganeh, S.H.;, "Approximation Algorithms for Software Component Selection Problem," Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific, vol., no., pp.159-166, 4-7 Dec. 2007.

[9] Yanhua Shao; Mingsheng Zhang; Shengnan Xu; , "Research on decision tree in component retrieval," Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on , vol.5, no., pp.2290-2293, 10-12 Aug. 2010.

[10] Liyin Zheng; Lu An; Li Zhang; , "A Component Retrieval Method Based on Query Vector Transfer," Computer Science and Software Engineering, 2008 International Conference on , vol.2, no., pp.116-118, 12-14 Dec. 2008.

[11] Luger, George F. "Artificial intelligence: structures and strategies for complex problem solving." (2009).

Authors' Profiles

M. Rizwan Jameel Qureshi: Assistant Professor of Information Technology, Faculty of Computing & Information Technology, King Abdulaziz University, major in software engineering and database management systems

Nora Farraj: Post-graduate student for master degree for Information Technology in King Abdul-Aziz University, major in Internet Technology. She received a Bachelor in Computer Science in 2012 and worked in artificial intelligence projects and expert systems in undergraduate.

Appendix

In this appendix we list the questions of the questionnaire used to validate the efficiency of our proposed solution. The questionnaire is divided into 3 sections based on the system stages from the user point of view:

1) Determining the component requirements.
2) Retrieving the component model and advices.
3) Providing your feedback about the result of the selection.

Under every section there are from 3 to 7 statements and the respondents were asked to determine their opinion about each statement, whether they strongly disagree, disagree, neutral, agree or strongly agree. These statements are as follow.

A) Determining the component requirements.

- It takes time to specify the requirements in the needed component.
- It is difficult to consider both technical and non-technical requirements of the component.
- It is better to see all possible requirements and choose from them rather than recalling the requirements and write them down.
- It is better to see requirements grouped into different categories, and specify which groups you want to select requirements from them rather than select from a list of all possible requirements.

- It is difficult to determine whether the component would affect the functionality of other components in your system before trying it.
- Retrieving the component model and advices.
- It would be useful if the system explain why a component is chosen to be the best component fit your system.
- Listing the features and functions of the selected component would be useful.
- In case there is no component fulfills all the selected requirements, it would be useful to return the components that fulfill most of the requirements.
- It would be useful to notify you with the requirements that are not fulfilled by the retrieved component, so you try to update the component to do.
- It would be useful to provide you with advises about how to update the retrieved component to fulfill the missed requirements.
- It would be useful to notify you whether the retrieved component would affect the functionality of the other components exist in your developed system.
- It would be useful to provide you with advises about how to update the retrieved component to not affect the functionality of the other components exist in your developed system.

B) Providing your feedback about the result of the selection

- It is common to reuse the same component for the same requirements.
- If the suggested component is not suitable for your system for any reason, it would be useful to tell the system so it tries to search for another one.
- It would be useful to provide your feedback about the result of the selection operation to speed up the future selection operations.

