I J C T A, 9(17) 2016, pp. 8455-8462

Intelligent Agent Architecture for Runtime Software Evolution

Jaya Sinha[*] and Kiran K Ravulakollu[**]

ABSTRACT

Making software adaptable to evolving requirements is a challenge for the software industry. The reason behind it is that adaptation compels software developers to follow almost all software developmental stages, from code development to software deployment. Due to this, any change in requirements, the iteration should be re-initiated. Thereby, adaptation requires software reconfiguration and redeployment. However, adaptation can be automated by giving the ability for software to evolve dynamically with evolving requirements. Hence, during the initial stages of software development, requirements adaptation has to be incorporated. This paper presents a solution as a novel intelligent agent-based architecture as a concept to support runtime adaptation. The focus lies in determining the feasibility and potential of having an intelligent agent facilitating acquisition and capturing of dynamic requirements within the architecture to obtain evolutionary mechanism in software. Fuzzy reasoning approach shows the usefulness of requirement gathering agent in making decisions regarding dynamic adaptations during runtime.

Keywords: Runtime software evolution, Intelligent agent, Agent-based software development, MAS

I. INTRODUCTION

An analysis of research articles published from 1990-2010 on factors affecting software project success or failure reveals that clear and frozen requirement is the topmost factor that affects the success of software projects [1], while the new and changing requirements are one of the root causes of software project failure [2]. Over the last several decades of experiences in software development also indicate that inadequacy in software requirements contribute the most to software project failures [3]. Such inadequacies are inevitable and they cannot be eliminated completely because of the inherent problems in predicting the full functional and non-functional requirements at initial, intermediate or at any phase of software development process. Also, other external factors as changes in the operating environment, occurrences of unanticipated errors, improvements in the technical environment, enhancing original functionality to increase reliability & quality of software, expansion of the developer's knowledge-base in a particular application domain and innovations in almost every domain such as business, industry and research, have also contributed to the never ending updates in the legacy software [4]. Thus, there is a need to apply sound engineering principles to create a continuous process of updates.

The section II of the paper highlights the features of software evolution as well as describes one of the most promising approaches for runtime software evolution, the agent-based software development. Section III details the specifications of the architecture. It also defines and describes the role, responsibilities and goal of Requirement Gathering Agent. In Section IV fuzzy rule-based reasoning of the architecture has been done. Section V specifies experimental result and section VI contains conclusion.

[*] Department of Computer Applications, Galgotia Institute of Management and Technology, Greater Noida, U.P, India, *E-mail: jsjoysinha@gmail.com*

[**] Department of Computer Science and Engineering, Sharda University, Greater Noida, U.P, India, *E-mail- Kiran.ravulakollu@sharda.ac.in*

II. AGENT-BASED SOFTWARE DEVELOPMENT APPROACH FOR SOFTWARE EVOLUTION

2.1. Runtime Software Evolution

Software evolution during runtime enables continuous updating and maintenance of application software overtime to adapt to required changes in the software dynamically. This evolutionary characteristic allows some functionality to be embedded in every software system through which it can identify and capture the evolving requirements as well as reconfigure itself accordingly to adapt to these changes. Such a system would benefit if the maintenance process becomes continuous and performed during execution without offloading the software, decreasing the cost and risk of shutting down the system and restarting it again.

2.2. Agent-based Software Development Approach

Recent research have shown that now the focus has been shifted from object oriented software engineering, to agent-oriented software engineering because an intelligent agent continuously perceives its environment through sensors and thinks rationally to act autonomously with the help of effectors [5] [6] [7]. Intelligent agents thus exhibits some orthogonal features that are autonomous, flexible, proactive, reactive, adaptability and socially able to interact, negotiate and cooperate with others in and around the environment [8][9]. Thus, intelligent agents accordingly adapt to the changes dynamically during runtime by reconfiguring itself making the system more intelligent and mature. They can intelligently perform domain- oriented reasoning to analyze conditions affecting the system and accordingly take actions independently without human guidance to adapt to these changing conditions during runtime, thus supporting dynamic maintenance [10]. Agent-based methodology thus models the real world complex systems more cleanly where various entities interact and work collaboratively to achieve a specific goal. The agent-oriented programming framework thus has an upper hand specialization over the object-oriented programming (OOP) paradigm as its architecture can integrate mental and social behaviour of agents.

Agent-based approach is suitable for decision making under uncertainty as intelligent agent may take next decision independently and autonomously using domain-oriented reasoning. This approach is suitable for complex dynamic environment where agents have to change their behavior with respect to changing goals. MAS are suitable within the environment where multiple agents should cooperate and work collaboratively to achieve a common goal. [11][12].

Still there are some pitfalls in agent-oriented development of an application [13]. So, developing an agent-architecture and handling agents becomes more complex and difficult in terms of development time and effort. Moreover, agent-based systems still limits solution for a specific domain as they collectively and collaboratively work for achieving a specific goal. Arriving at generic multi agent architecture remains a major challenge for all the researchers. FIPA ACL and KQML are the agent communication languages which are commonly used. JADE, JACK, Aglet, Cougar are some of the common platforms for developing agent-based system.

III. ARCHITECTURAL SPECIFICATION

Many frameworks have been designed to support developing MAS. Our architecture has been inspired by open agent architecture [14]. This section represents a simple agent architecture containing the modules based on the objectives of supporting dynamic adaptation. The modules of the proposed architecture are composed of a set of intelligent agents where requirement gathering plays a key role as shown in Figure. 1. It also specifies the role and responsibilities of intelligent RGAgent as a requirement gathering tool, mainly in identifying specific requirements evolving during runtime as well. This paper mainly focuses on the overall behavior of such a system that has an RGAgent.

3.1. Architectural Components

- **Interface Module** within the application domain contains interface agent which interacts with the user interface.
- **Dynamic Module** is composed of RGAgent, Service Provider Agent and Runtime Execution frame**.**

RGAgent is an intelligent agent which continuously senses and monitors the application domain and the runtime data with the help of dynamic runtime data monitoring mechanism (DRMM). Any input from DRMM representing a need to be dynamically adapted is gathered, captured and analyzed. This agent also acts like an information provider agent as can directly access the knowledge base. It performs task classification. Also, new data or rule can be added to or stored data can be retrieved from knowledge base thru RGAgent.

Dynamic Runtime data Monitoring Mechanism (DRMM) continuously monitors the designated system / modules. In case of any runtime anomalies during execution it notifies and communicates with RGAgent. Reflection API's or other runtime data capturing techniques can be used to read the metadata.

Service Provider Agent (SPAgent) acts as a coordinator to manage the composite task. After thorough analysis, RGAgent notifies the SPAgent that further delegate the task of runtime adaptation to Task Agents (TA). It also manages coordination and communication among various task agents to reach a specific goal.

Task Agents carry out the atomic problem solving task. The intelligent Task Agent communicates with SPAgent and collaborates with other Task Agents to implement dynamic adaptation. During this process implementation code may cause reconfiguration of the system architecture. A component may be added, deleted or may be replaced that may require modifying an existing rule. During any modifications the links among various modules of the system must be maintained to ensure system integrity.

- **Knowledge Management Module** contains detailed description about all the classes, domain specific historic data, rules and an inference engine. The paper does not specify how the rules are to be stored.

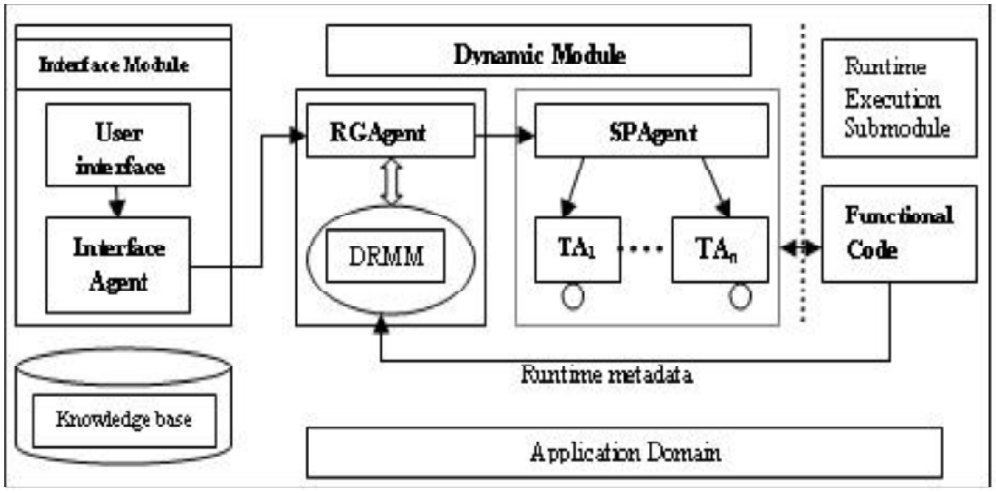

Figure 1: Multi-agent architecture of the system

Use case diagram of the overall system showing notification navigation among existing agents has been depicted in Figure 2.

3.3. Role and responsibilities of RGAgent

- Supporting Self-Management**:** If the RGAgent is able to gather and capture requirement changes during runtime , it can lead to a software system which can support self-managing the software during runtime.

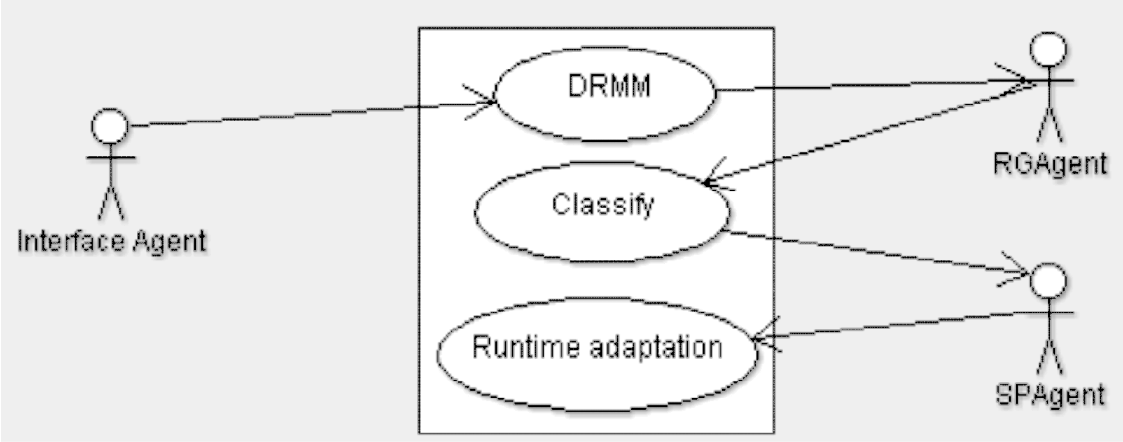

Figure 2: Use case diagram of overall system

- Continuous Sensing and Monitoring: It should be able to monitor state and behaviour of each designated components of the system at frequent intervals to gather any new requirement using DRMM.
- Efficient Requirement Capturing: One among various techniques of reading runtime metadata can be selected and implemented accordingly in DRMM.
- Classification: RGAgent should be able to classify and categorize the evolving requirements as rules.
- Embedded Learning: After classification, knowledge base is updated accordingly with newer rules which may lead to implementing embedded learning in the software system.
- Delegation: RGAgent should delegate its responsibility to further levels such as to SPAgent to carry out the actual process of executing runtime adaptation of captured requirement.
- Decision Maker: RGAgent is the key decision maker of the whole model. Autonomy allows it to independently decide on next course of action.

3.3. Working of RGAgent

The main task of RGAgent is to take decision about dynamic adaptation. Table 1 shows the use case of RGAgent specifying its tasks.

Table 1
Requirement Gathering Agent Tasks

Role	Requirement Gathering Agent
Usecase	Gathering and analyzing evolving requirements
Goal	Capturing evolving requirements from Interface Agent/DRMM as well as classification.
Tasks	1. Waits to receive notification from Interface Agent and simultaneously monitors DRMM. 2. Captures input requirements from Interface Agent and DRMM. 3. Performs rule-based classification of captured requirements. 4. Takes decision to modify or retain the system state during runtime. 5. Depending on decision it delegates the task of adaptation to Service Agent. 6. Saves the state in the knowledge base.

The architecture using an RGAgent has been designed to currently achieve the following key goals: Dynamic adaptation and self management

3.2. Goal Hierarchy

As the first step towards analysis of the system, the macro level goal and sub-goals of the overall system are identified. The macro analysis of the system is based on MaSE Methodology [11] [15]. The overall goals and the sub-goals are structured with the help of goal hierarchy diagram as shown in Figure. 3. MaSE agentTool III has been used to create goal hierarchy of the system.

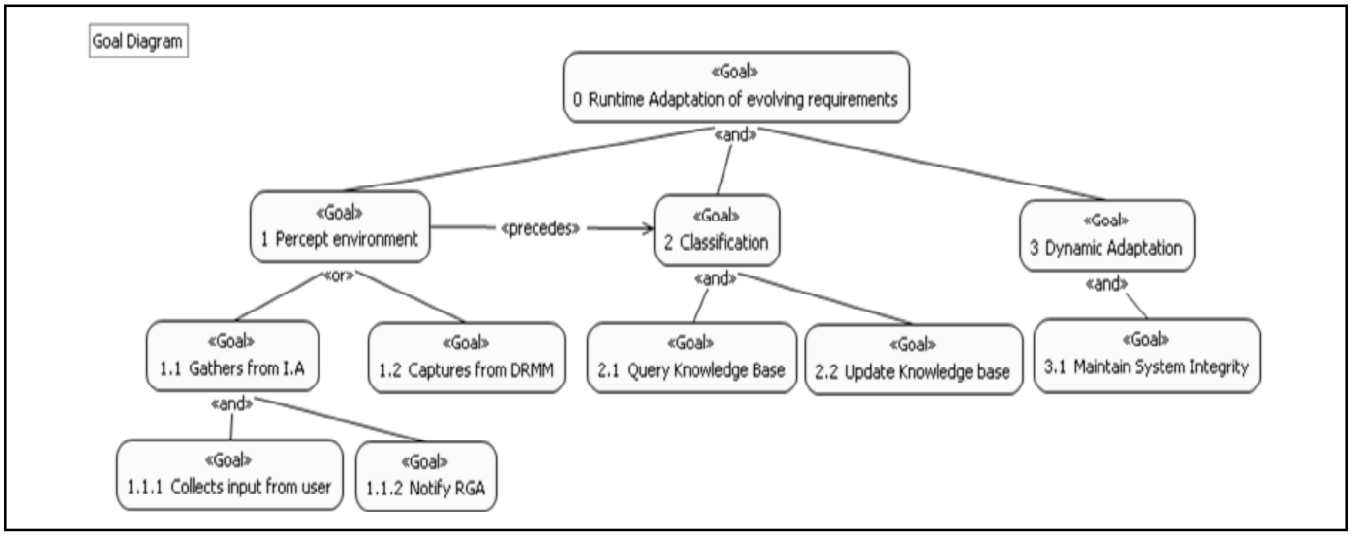

Figure 3: Goal hierarchy diagram

IV. FUZZY REASONING IN RGAGENT

Fuzzy logic framework by Zadeh has an advantage of efficiently representing the imprecise knowledge in an uncertain environment [16]. Many complex and dynamic applications have been successfully modeled using fuzzy approximate reasoning [17] [18]. Since the system for software evolution during runtime is complex and dynamic, fuzzy logic has motivated us to incorporate fuzzy reasoning for decision maker RGAgent in the proposed MAS architecture.

4.1. Case Study

To illustrate our approach, we have applied fuzzy reasoning in making decision regarding changes in the refund policy of the system for the purchased items in a super market management system. The current scenario of tough market competition compels strategy makers to satisfy and attract customers towards them. In order to achieve their target, they tend to decide on throwing attractive promotional offers or may be some changes in their current sales and marketing policies. We have taken change in refund policy as an example for fuzzy simulation. The decision regarding the form of changes depends on two key parameters: customer satisfaction frequency and internal management decision. The customer satisfaction frequency is calculated by finding average of customer feedback over a period of time. Whereas, the level of internal managerial decision depends on various factors, among them influence of current market conditions is one of the prime factor.

The decision regarding updating the legacy system is as follows:

- Adding a new refund policy.
- Deleting an old refund policy
- Replacing a part within refund policy.

4.2. Reasoning Methodology

For the proposed architecture, RGAgent act as a fuzzy agent. RGAgent uses fuzzy knowledge representation in the form fuzzy sets, fuzzy rules and fuzzy membership functions to show the degree of membership to a fuzzy set. In Mamdani [19] fuzzy rule-based reasoning model inputs are in the form of fuzzy linguistic variables. This is the reason which motivated our research to use Mamdani model of reasoning to guide the decision making process.

The Main modules of Mamdani system are fuzzification interface, inference engine and defuzzification interface. Along with them there is a strong fuzzy rule-set knowledge base. The reasoning methodology is as follows:

1. Universe of discourse determination: The input parameters customer satisfaction frequency (c1), management factor (c2) and output system parameter decision class (op) are determined.
2. Quantization: The input parameter c1 has been quantized to four linguistic variables LOW(L), MED(M), HIGH(H) and VERYHIGH(VH) whereas input parameter c2 into three LOW(L), MED(M) and HIGH(H) ranging over 0 to 1. The output variable op is quantized as NOCHANGE, REPLACE, ADDNEW based on adaptation decision.
3. Fuzzy rule set specification: Fuzzy logic uses IF (condition) –THEN (action) form of rules to depict mapping between input and output parameters. Some of the heuristic rules formed for the present fuzzy system are specified below and stored in knowledge base.

IF (*customer satisfaction is* LOW) and (*management factor is* HIGH*)* THEN (*adaptation decision is* ADDNEW).

IF (*customer satisfaction is VERYHIGH)*) and (*management factor is MED)* THEN (*adaptation decision is* NOCHANGE).

The fuzzy agent RGAgent activates the framed rules and the decision output is obtained and defuzzified using triangular membership function.

V. EXPERIMENT RESULT

The user input is captured by the proposed agent system. We have implemented decision making using Fuzzy Logic Toolbox in MATLAB®. Figure 4 clearly shows the impact of changes in customer satisfaction

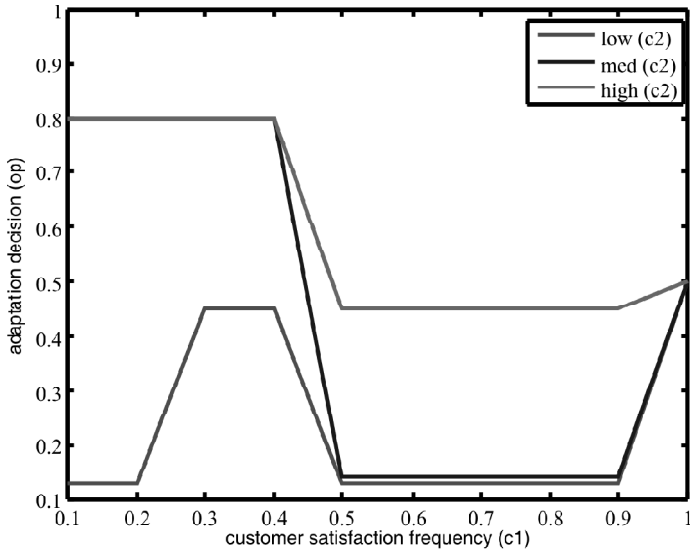

Figure 4: Customer satisfaction frequency v/s adaptation decision

frequency on adaptation decision taken by RGAgent. It shows that when the customer satisfaction frequency is low or medium as well as management support is high the dynamic adaptaion such as adding a new component or replacing a component is the recommended decision by the RGAgent. If customer satisfaction is high then recommendation for changes will be less or no change will be recommended.

VI. CONCLUSION

Software evolution is the key to change the future of software engineering. It drives to create such mature software that will be subject to changes over time and thus evolves. This paper aims at proposing a simple conceptual intelligent agent-based architecture to support runtime adaptation. Agent-based software development approach is selected to develop the conceptual model. It appears that runtime adaptation can be enhanced to a greater extent. However, the expected result demonstrate that at requirement gathering stage the proposed architecture has significant scope to evolve. RGAgent focus more on to evolution of the existing software by capturing the continuous changing requirements at any developmental stage. This can enhance the capability of system enabling it to be self-managed and adaptive in nature. The work is also supported by fuzzy approach in reasoning and decision making to show the usefulness in having a requirement gathering agent for runtime evolution.

REFERENCES

[1] H. N. Mohd Nasir and S. Sahibuddin , "Critical success factors for software projects: A comparative study", *Academic Journals*, vol. 6, pp. 2174-2186, 2011.

[2] C.Jones, "The Economics of Software Maintenance in the Twenty First Century", 2006. *http://wwwcompaid.com/caiinternet/ezine/capersjones-maintenance.pdf*

[3] Alford M. and Lawson J., "Software Requirements Engineering Methodology (Development)", U.S. Air Force Rome Air Development Center RADC-TR- 79- 168, 1979.

[4] Ian Sommerville, *Software Engineering*, Addison- Wesley, 2004.

[5] Amund Tveit, "A survey of Agent-Oriented Software Engineering," NTNU Computer Science Graduate Student Conf, 2001.

[6] O. Zohreh Akbari, "A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance," *Journal of Computer Engineering Research*, vol. 1, no.2, pp. 14-28, Apr., 2010.

[7] Shoham Y., "An Overview of Agent-oriented Programming", Software Agents, AAAI Press, 1997

[8] Nicholas R. Jennings and Michael Wooldridge, "Agent-Oriented Software Engineering", Artificial Intelligence, vol. 117, pp. 277-296, 2000.

[9] Gopal Sakarkar and Sachin Upadhye, "A Survey of Software Agent and Ontology," *International Journal of Computer Applications*, vol. 1, no. 7, pp. 0975 – 8887, 2010.

[10] Patricia Paderewski-Rodrý´guez, Ma. Jose´ Rodrý´guez-Fortiz, Jose´ Parets lorca, "An architecture for dynamic and evolving cooperative software agents," *Journal Computer Standards and Interfaces- special issue: Adaptable Software Architectures*, vol. 25, no.3, pp. 261-269, Jun., 2003.

[11] N.R.Genza and E.S.Mighele, "Review on Multi-Agent Oriented Software Engineering Implementation," *International Journal of Computer and Information Technology*, vol.2, no. 3, pp. 511-520, May., 2013.

[12] Chouarfia Abdallah, Hafida Bouziane, "Dynamic Maintenance and Evolution of Critical Components-Based Software Using Multi Agent Systems", Computer and Information Science, vol. 4, no. 5, pp. 78-91, Sep. 2011.

[13] M.J. Wooldridge and N.R. Jennings, "Pitfalls of agent-oriented development", In Proc. of Second Int. Conf on Autonomous Agents, pp. 385-391, 1998.

[14] Adam Cheyer and David Martin," The Open Agent Architecture," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 4, no. 1, pp. 143-148, Mar, 2001.

[15] Scott A.DeLoach and Juan Carlos Garcia-Ojeda, " O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems," *Int. J. Agent-Oriented Software Engineering*, vol.4, no.3, 2010.

[16] L.A.Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes", *IEEE Transactions on Systems, Man and Cybernatics*, vol. SMC-3, pp. 28-44,1973.

[17] Hanli Wang, S. Kwong, Yaochu Jin, Wei Wei and Kim-Fung Man, "Agent-based evolutionary approach for interpretable rule-based knowledge extraction", *IEEE Transactions on Systems, Man and Cybernatics*, vol. 35, no. 2, pp. 143-155, 2005.

[18] Jong Yih Kuo, Shin Jie Lee, Chia Ling Wu, Nien Lin Hsueh and Jonathan Lee, "Evolutionary agents for intelligent transport systems," *International Journal of Fuzzy Systems*, vol. 7, no. 2, Jun., 2005.

[19] Alain-Jerome Fougeres, "A modelling approach based on fuzzy agents", *International Journal of Computer Science Issues*, vol. 9, 2012.