Agent based Architecture for Modeling and Analysis of Self Adaptive Systems using Formal Methods

Natash Ali Mian[1,2]
School of Computer Science, National College of Business Administration and Economics, Lahore [1]
School of Computer and Information Technology, Beaconhouse National University, Lahore[2]

Farooq Ahmad
Department of Computer Sciences, Comsats Institute of Information Technology, Lahore[3]

***Abstract*—Self-adaptive systems (SAS) can modify their behavior during execution; this modification is done because of change in internal or external environment. The need for self-adaptive software systems has increased tremendously in last decade due to ever changing user requirements, improvement in technology and need for building software that reacts to user preferences. To build this type of software we need well establish models that have the flexibility to adjust to the new requirements and make sure that the adaptation is efficient and reliable. Feedback loop has proven to be very effective in modeling and developing SAS, these loops help the system to sense, analyze, plan, test and execute the adaptive behavior at runtime. Formal methods are well defined, rigorous and reliable mathematical techniques that can be effectively used to reason and specify behavior of SAS at design and run-time. Agents can play an important role in modeling SAS because they can work independently, with other agents and with environment as well. Using agents to perform individual steps in feedback loop and formalizing these agents using Petri nets will not only increase the reliability, but also, the adaptation can be performed efficiently for taking decisions at run time with increased confidence. In this paper, we propose a multi-agent framework to model self-adaptive systems using agent based modeling. This framework will help the researchers in implementation of SAS, which is more dependable, reliable, autonomic and flexible because of use of multi-agent based formal approach.**

Keywords—Formal methods; self-adaptive systems; agent based modeling; feedback loop; Petri nets

I. Introduction

As the complexity has increased, hence, existing approaches do not suffice the requirements of modeling, managing and developing software systems. This has motivated the research community to explore new dimensions in software engineering and integrate other fields like biology, psychology; nature inspired computing, robotics, artificial intelligence and more. The change in the way the software is used needs that it has the capability of self-adaptation [1] which is one of the hot areas of research since last decade.

SAS [2] are capable of modifying their behavior due to change in environment at run time. Modeling of these types of systems is either very difficult or not possible by the use of conventional software engineering approaches. One of the major difference in requirement engineering is that the 'shall' statements become 'may' statements when developing a SAS that has the capability to adapt in accordance with the external environment [1]. Uncertainty is one of the most certain thing in modeling and development of SAS. [2]. This aspect motivates the practitioners and researchers to use multiple existing approaches or develop new approaches [3] to handle uncertainties of the system [4].

There has been a lot of research in SAS including software engineering, requirements engineering, software architectures, middleware, component-based development and programming languages [5]. In addition to these some research has been done in other areas of Computer Science which includes fault-tolerant computing, biologically inspired computing, multi-agent systems, distributed artificial intelligence and robotics [6].

Formal methods are very effective and concrete mathematical techniques and methods in specifying, modeling, verifying and developing systems. Formal methods have been majorly applied in modeling of SAS [7]. Application of formal methods for verification, model checking and theorem proving is less for SAS [8]. To utilize the formal methods according to its strengths, there is a need to apply them in validation and verification [9] of SAS, this will consequently produce systems that are more reliable and tested [6] at an early stage of development [10].

Use of agents in modeling system [11] that have capacity and capability to adapt to a new behavior at runtime has been very effective and efficient [12]. Agents [13] help the systems to perform all the tasks autonomously and efficiently, this increases the overall productivity of the system. We use agents to perform the tasks autonomously with well-defined and concrete rules which have been developed, analyzed and tested by use of formal methods. More specifically Petri nets will be used to model all these agents.

In this paper we propose an initial architecture of the system that will use the strengths and rigor of formal methods, autonomous working of agents and the effectiveness of feedback loop to model a self-adaptive system [14]. This model will further be extended for distributed systems where most of the components will be reused with some additional components like distributed feedback loop manager, distributed agent manager and distributed application manager will be added. Fig. 1 depicts the scope of the work and the gives an idea of the proposed integration. This is an initial attempt to propose an architecture which integrates agents, formal methods and feedback loop. This paper is introduced in

Section I, literature review is given in Section II, Section III introduces SAS, Section IV gives an overview of feedback loop, Section V elaborates formal methods, followed by the proposed architecture in Section VI, and finally we conclude the paper and give pointers to future work.

II. Related Work

Life patterns have changed, consequently having a huge impact on working environment [15] and the way software is used in ever changing environment [4]. Improvement in technology, change in working environment, improvement in technology, increase in storage capacity, need of high processing, and availability of variety of data is causing fundamental change in software development, testing and performance [8]. This gives rise to systems that can adapt to the working environment; these systems are categorized as SAS. Development of SAS has increased the flexibility of software systems, however, this has also led increased the complexity of systems resulting in a lot of challenges [16] for software engineering community [4]. Methods as well as processes of engineering software systems have also evolved for development of SAS [17].

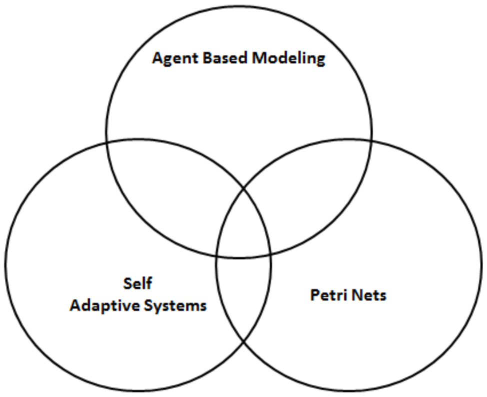

Fig. 1. Scope of work.

One of the major problems in the current approaches is that focus on quality of the output is much less [18] , this may be due to the variety of problems, uncertainty and changing environment [15] that had to be handled by SAS. Many model based approaches [19] have been proposed in literature which address the problem of requirement engineering in SAS . A context aware methodology is proposed in literature, this approach performs the adaption at run time considering the context, not only this but a complete mechanism of verification [20] and validation is also proposed, the focus of this work is done on the basis of image processing algorithms [21]. Goal based [22] and requirement driven architecture [1] for systems are proposed in literature which can adapt themselves to a better configuration by monitoring and analyzing the current actors in the system [23].

Non-functional requirements play vital role in self-adaptive approach which takes information that is not easily identifiable and overwhelmingly against the static nature of information [24]. Although, functional requirements are also important, but non-functional requirements have more infringement in software development and software quality assurance, using self-adaptive approach [20]. One of the major problems in the current approaches is that focus on quality of the output is much less, this may be due to the variety of problems, uncertainty and changing environment [15] that had to be handled by SAS. Many model based approaches [25] have been proposed in literature which address the problem of requirement engineering in SAS. A context aware methodology which performs the adaption at run time considering the context, not only this but a complete mechanism of verification [20] and validation is also proposed. A lot of work has been done in identifying the key areas of research, challenges faced, architecture problems, design techniques available, implementation issues in engineering [26] of SAS [10]. Many papers discuss the importance of adaption and propose architecture based adaptation [25], goal based adaptation [26], feature oriented adaptation [4], parameter based adaptation, requirement driven adaptation [1] and much more. Software agents have been used to model [27] and implement the adaptation process.

It has been observed that formal methods has mostly been used in modeling of SAS [7] and not in model checking and theorem proving which are major strengths of formal methods, hence, the need to apply formal methods for these is positively required to make the overall process of designing the SAS more reliable [6]. A combination of formal and semi-formal methods is also used in modeling of SAS [8] and the results have been very encouraging [6]. There have been a few studies where formal methods are used successfully in model checking [9] and domain specific languages [14] and design patterns [3] are proposed for development of SAS.

III. Self-Adaptive systems

SAS can alter their behavior during operation [4]. These systems fall under the category of context aware systems [28]. Adjusting as per needs of the user at run-time is one of the major strength of these systems [29]. The adaptations that these systems perform during executions are not included in the requirements for which these systems are developed [24]. This variability makes the development of these systems challenging as the development team has to plan for the uncertainties that may arise at run time [30], [15]. Hence, major part of the requirement engineering has to completed at run time [31]. Not only the requirement engineering, but testing is also done at run time, all this is done by use of feedback loops [32]. To enhance the efficiency and reliability of these systems, all these steps are performed autonomously by agents [33].

Almost all major systems that exist today have the capability of adaptation; however, in some systems the adaptations are planned at design time and in other it is done at run time. In case, the adaptations are implemented at design time, the systems are categorized as simple adaptive systems and when the adaptation is done at rum time, the systems are classified as self-adaptive [34].

IV. Formal Methods

Formal methods are mathematical tools and techniques that are used in analysis and modeling of different hardware and software systems [35]. Additionally they help us in validation and verification of systems at an early stage of development [9]. These methods are reliable and help us in analyzing, modeling, reasoning and testing the systems. As these methods are based on concrete mathematical principles, hence the reliability of systems that are developed using formal methods is increased many folds [36]. Strength of these methods is that they can be used in combination with existing software development methods. Most of these methods have specification languages that are based on first and second order predicate calculus, temporal logic, algebraic theory and graph theory. Sets, sequences, relations, functions, mappings and state machines are the foundation of formal modeling techniques. Due to the use of precise mathematical symbols the effective ness of these methods is much more than conventional methods.

Formal methods are supported by a variety of case tools which help in model development, model checking and simulating the overall scenario. We have successfully modeled a small self-adaptive system by use of Petri Net which is a formal method and is based on graph theory. The tools help in development of concrete model efficiently and reliably. Hence, formal methods are very effective in modeling of complex system like self-adaptive systems. These methods have already been successfully applied in development of many complex industrial systems and many safety critical systems.

V. Feedback Loop (MAPE-K Loop)

Feedback loop comprises of four major steps which are monitor, analyze, plan and execute, this loop is also referred as MAPE-K Loop [6]. Each phase is further divided in to further sub-phases and multiple strategies are used for design and implementation of each phase. Formal methods have the capability to model all phases with success and reliability.

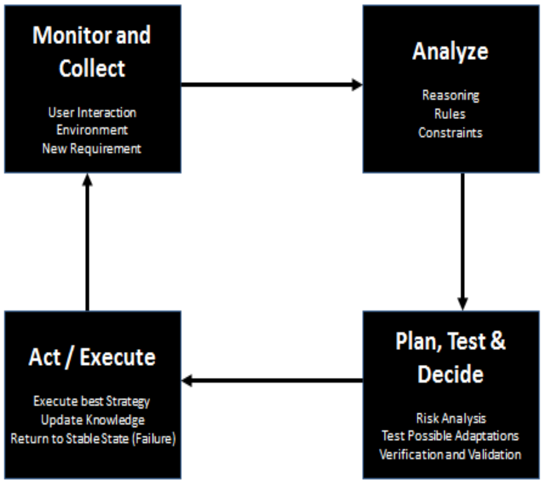

Fig. 2. MAPE-K feedback loop.

In the monitor phase the input is received from the external environment, and after performing the initial transformation of inputs it is checked against the existing requirements. In case a match is found, no adaptations are performed and the requirement is executed. However, if the set of input are new then analysis of inputs is performed which is followed by the planning and testing phase. It is to be noted that the possible adaptations, testing and execution is done at run time. The execution is executed by the system effectors. In a situation where the proposed adaptation is not successful, the loop starts again and this process continues iteratively [37] till a final adaptation is executed [14]. A simplified version of MAPE-K feedback loop is shown in Fig. 2.

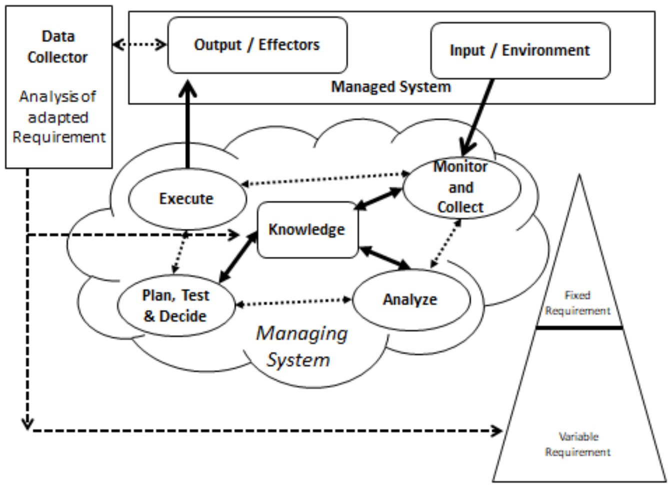

Fig. 3. Proposed architecture for modelling SAS.

VI. Proposed Architecture

We have proposed an architecture which not only focuses on performing the adaptation at run time but it also keeps a track of the results of adaptation when it is executed. As given in Fig. 3, the overall system is divided in to four major parts namely managed system, managing system, data collector and requirement pyramid. The top most part covers the input and output channels of the system and is classified as managed system in our architecture. Generally, input is given by sensors and is received by monitor agent. Monitor agent performs the transformation and converts the inputs in to a form that can be further analyzed. Another step that is performed by the monitor phase is to check the set of inputs against the existing knowledge. In case, the input from the environment matches any existing requirement, no adaptation is performed and the system acts according to the existing requirement. If the inputs do not match any of the existing requirements, then it is passed to the next phase, which analyzes the input according to the system goals, existing requirements, system objectives and overall preferences of the user. All this is available in the knowledge repository. Once the analysis is completed, all the data is forwarded to the next agent which is the plan, test and decide agent. Here the requirements are mapped to the nearest match, fuzzy rules are applied and possible adaptations are proposed. After formulating a few possible adaptations, these are tested on the criterion given in knowledge base, additionally; the capability of system effectors is also checked. For instance, an adaptation to take an aerial route to destination will fail the test for a car. Once the planning and testing is completed, one proposed adaptation is finalized and sent to the next agent which transforms the proposed execution in to the form that can be understood by the output channels. The process does not end here, the data collector agent continuously monitors the managed system during its execution and results of execution are recorded. We may have two possible scenarios here, either the proposed adaptation has been successful or it has ended up in failure. In both cases the data is recorded and knowledge base is updated with a flag of success or failure, the successful adaptation is also recorded in the variable requirement part. In case of failure, the process is repeated iteratively till a final goal is achieved. This is kept for future enhancement in system and to make sure that all capabilities of system are available in the requirement set of the system.

An important contribution of this work that the system is designed in way that performs the adaptation at run-time, monitors the quality of output produced by the proposed adaptation and regular update of the knowledge and requirement base. All these modules will be analyzed, modeled and verified using formal methods.

VII. Conclusion and Future Work

This research has two major contributions, firstly we have proposed an integration of formal methods, agent based modeling and SAS for successfully analyzing, testing and implementing the systems that have the capability to adapt at run time. Secondly, an overall architecture of the complete system is given, which includes four major components. It is to be noted that we have successfully modeled the first phase using Petri Nets, the results have been very encouraging and the complete system will be analyzed, modeled, simulated, verified and tested by using formal methods. The given architecture gives a concrete base for the researchers and practitioners to implement systems that have the capability to adapt during execution. This is the first step toward development of a multi-agent autonomous formal model for self-adaptive system. We have successfully applied Petri nets to model feedback loop [12] and the work will be extend for a complete model for SAS using formal methods.

This architecture will be further be extended for distributed systems where the variability of inputs in more and there are multiple feedback loops at each node. Further, each agent will be implemented and the task will be further sub divided in to multiple agents where each agent will be designed to perform an atomic task.

References

[1] G. Tallabaci and V. E. Silva Souza, “Engineering adaptation with Zanshin: An experience report,” in ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2013, pp. 93–102.

[2] F. Kneer and E. Kamsties, “A framework for prototyping and evaluating self-adaptive systems - A research preview,” in CEUR Workshop Proceedings, 2016, vol. 1564.

[3] Y. Abuseta and K. Swesi, “Design Patterns for Self Adaptive Systems Engineering,” Int. J. Softw. Eng. Appl., vol. 6, no. 4, pp. 11–28, 2015.

[4] N. Esfahani and S. Malek, “Uncertainty in Self-Adaptive Software Systems,” in Lecture Notes in Computer Science, 2013, pp. 214–238.

[5] C. Krupitzer, F. M. Roth, S. Vansyckel, G. Schiele, and C. Becker, “A survey on engineering approaches for self-adaptive systems,” Pervasive Mob. Comput., vol. 17, no. PB, pp. 184–206, 2015.

[6] D. G. D. La Iglesia and D. Weyns, “MAPE-K Formal Templates to Rigorously Design Behaviors for Self-Adaptive Systems,” ACM Trans. Auton. Adapt. Syst., vol. 10, no. 3, pp. 1–31, 2015.

[7] N. Khakpour, S. Jalili, C. Talcott, M. Sirjani, and M. Mousavi, “Formal modeling of evolving self-adaptive systems,” in Science of Computer Programming, 2012, vol. 78, no. 1, pp. 3–26.

[8] M. Luckey and G. Engels, “High-quality specification of self-adaptive software systems,” in ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2013, pp. 143–152.

[9] P. Arcaini, E. Riccobene, and P. Scandurra, “Formal Design and Verification of Self-Adaptive Systems with Decentralized Control,” ACM Trans. Auton. Adapt. Syst., vol. 11, no. 4, pp. 1–35, 2017.

[10] R. De Lemos et al., “Software engineering for self-adaptive systems: A second research roadmap,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2013, vol. 7475 LNCS, pp. 1–32.

[11] C. Macal and M. North, “Introductory tutorial: Agent-based modeling and simulation,” in Proceedings - Winter Simulation Conference, 2015, vol. 2015–Janua, pp. 6–20.

[12] N. A. Mian and F. Ahmad, “Modeling and Analysis of MAPE-K loop in Self Adaptive Systems using Petri Nets,” vol. 17, no. 12, pp. 158–163, 2017.

[13] M. I. Tariq, S. Tayyaba, M. U. Hashmi, M. W. Ashraf, and N. A. Mian, “Agent Based Information Security Threat Management Framework for Hybrid Cloud Computing,” vol. 17, no. 12, pp. 57–66, 2017.

[14] F. Krikava and P. Collet, “A Reflective Model for Architecting Feedback Control Systems,” in Proceeding of the 2011 International Conference on Software Engineering and Knowledge Engineering, 2011, p. 7.

[15] F. D. Macías-Escrivá, R. Haber, R. Del Toro, and V. Hernandez, “Self-adaptive systems: A survey of current approaches, research challenges and applications,” Expert Systems with Applications, vol. 40, no. 18. pp. 7267–7279, 2013.

[16] B. H. C. Cheng et al., “Software Engineering for Self-Adaptive Systems: A Research Roadmap,” Softw. Eng. Self-Adaptive Syst., pp. 1–26, 2009.

[17] M. Amoui, M. Derakhshanmanesh, J. Ebert, and L. Tahvildari, “Achieving dynamic adaptation via management and interpretation of runtime models,” J. Syst. Softw., vol. 85, no. 12, pp. 2720–2737, 2012.

[18] J. C. Muñoz-Fernández et al., “Capturing ambiguity in artifacts to support requirements engineering for self-adaptive systems,” in CEUR Workshop Proceedings, 2017, vol. 1796.

[19] S. Kounev et al., “The Notion of Self-aware Computing,” in Self-Aware Computing Systems, 2017, pp. 3–16.

[20] M. Ahmad, N. Belloir, and J. M. Bruel, “Modeling and verification of Functional and Non-Functional Requirements of ambient Self-Adaptive Systems,” J. Syst. Softw., vol. 107, pp. 50–70, 2015.

[21] Y. Brun et al., “A Design Space for Self-Adaptive Systems,” Softw. Eng. Self-Adaptive Syst. II SE - 2, vol. 7475, pp. 33–50, 2013.

[22] B. H. C. Cheng, P. Sawyer, N. Bencomo, and J. Whittle, “A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2009, vol. 5795 LNCS, pp. 468–483.

[23] F. Dalpiaz, P. Giorgini, and J. Mylopoulos, “Adaptive socio-technical systems: A requirements-based approach,” Requir. Eng., vol. 18, no. 1, pp. 1–24, 2013.

[24] N. Bencomo, K. Welsh, P. Sawyer, and J. Whittle, “Self-explanation in adaptive systems,” in Proceedings - 2012 IEEE 17th International Conference on Engineering of Complex Computer Systems, ICECCS 2012, 2012, pp. 157–166.

[25] J. Cámara et al., Self-aware computing systems: Related concepts and research areas. 2017.

[26] N. A. Qureshi, A. Perini, N. A. Ernst, and J. Mylopoulos, “Towards a continuous requirements engineering framework for self-adaptive systems,” in 2010 First International Workshop on Requirements@Run.Time, 2010, pp. 9–16.

[27] D. B. Abeywickrama, N. Bicocchi, and F. Zambonelli, “SOTA: Towards a general model for self-adaptive systems,” in Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, 2012, pp. 48–53.

[28] Q. Liu, S. Wu, D. Wang, Z. Li, and L. Wang, "Context-Aware sequential recommendation," in Proceedings - IEEE International Conference on Data Mining, ICDM, 2017, pp. 1053–1058.

[29] B. Ciloglugil and M. M. Inceoglu, "User Modeling for Adaptive E-Learning Systems," in ICCSA, 2012, pp. 550–561.

[30] J. Andersson, R. De Lemos, S. Malek, and D. Weyns, "Modeling dimensions of self-adaptive software systems," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2009, vol. 5525 LNCS, pp. 27–47.

[31] L. Gherardi and N. Hochgeschwender, "Poster: Model-based Run-time Variability Resolution for Robotic Applications," in Proceedings - International Conference on Software Engineering, 2015, vol. 2, pp. 829–830.

[32] Y. Brun et al., "Engineering self-adaptive systems through feedback loops," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2009, vol. 5525 LNCS, pp. 48–70.

[33] J. Levinson et al., "Towards fully autonomous driving: Systems and algorithms," in IEEE Intelligent Vehicles Symposium, Proceedings, 2011, pp. 163–168.

[34] S. Sucipto and R. S. Wahono, "A Systematic Literature Review of Requirements Engineering for Self-Adaptative Systems," in Journal of Software Engineering, vol. 1, no. 1, 2015, pp. 55–71.

[35] Y. Zhao, Z. Yang, and D. Ma, "A survey on formal specification and verification of separation kernels," Frontiers of Computer Science, vol. 11, no. 4. pp. 585–607, 2017.

[36] S. M. Edgar and S. A. Alexei, "Power and limitations of formal methods for software fabrication: Thirty years later," Informatica (Slovenia), vol. 41, no. 3. pp. 275–282, 2017.

[37] G. Su, T. Chen, Y. Feng, D. S. Rosenblum, and P. S. Thiagarajan, "An iterative decision-making scheme for markov decision processes and its application to self-adaptive systems," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, vol. 9633, pp. 269–286.