Modeling and Analysis of MAPE-K loop in Self Adaptive Systems using Petri Nets

Natash Ali Mian[1,2], Farooq Ahmad

School of Computer Science, National College of Business Administration and Economics, Lahore[1]
School of Computer and Information Technology, Beaconhouse National University, Lahore[2]
Department of Computer Sciences, Comsats Institute of Information Technology, Lahore[3]

Abstract
Feedback loop plays a pivotal role in modeling of systems that have capability to adapt to new requirements during execution. These systems are categorized as self-adaptive systems (SAS) which can alter their working according to the inputs received from the environment. At present, we are surrounded by software systems that are either adaptive or self-adaptive. In both cases feedback loop has a major role in the adaptation process. Hence, reliable and efficient working of this loop is critical towards successful development of software systems that have ability to work with requirements that were not known at the time of development. Formal methods are mathematics of software and hardware systems, these methods include modeling languages and tools to model and analyze systems with use of concrete mathematical principles. Petri-Nets is a formal specification language that is used for analysis, modeling and testing of complex systems. In this paper we have presented an initial model of feedback loop using Petri-Nets. It has been observed that modeling and analysis of feedback loop using Petri-nets has been useful in verification of system at an abstract level and the generated model is free from deadlock and has capability to expand in future. This is an abstract model with limited inputs, invariants and constraints, this model will be enhanced for a complete SAS in future research.

Keywords
Feedback loop, Formal Methods, MAPE-K, Petri nets, Self-adaptive systems

1. Introduction

Self-Adaptive Systems are much more complex than conventional systems, hence modeling of these system using existing approaches of software development is either extremely difficult or not possible. One of major problems in modeling of SAS is managing uncertainty. In case of conventional systems we have to be sure about the requirements of the software before its development and all efforts are put in to minimize uncertainty during the requirement engineering phase [1]. In contract, we actually plan, model and develop SAS for handling uncertainty, which means that we are developing the software to handle uncertain situations during execution [2]. This aspect motivates the practitioners and researchers to use multiple existing approaches or develop new approaches to handle uncertainties of the system [2].

Manuscript received December 5, 2017
Manuscript revised December 20, 2017

SAS is one of the emerging areas of computer science and there is an increasing trend in research outcomes in software engineering, software architectures, middleware, component-based development, requirements engineering and programming languages [3]. Additionally a lot of work been done in other areas including fault-tolerant computing, biologically inspired computing, multi-agent systems, distributed AI and robotics[4].

Control engineering methodology enables the integration of Control Management system and Feedback loop system to gain operational goals and reducing cost [5]. Run-time performance objectives are used to liaison unpredictable demand and prompt change with control engineering methodology. External environment of system is also considered for the development of software which is an integral part of systems that can adapt [6]. In [7] rainbow model is used to design SAS. According to their point of view this approach based on control and utility theories, and main advantages are cost effectiveness and self-adaptation [8].

It has been observed that formal methods has mostly been used in modeling of SAS [9] and not in model checking and theorem proving which are major strengths of formal methods. Hence, the need to apply formal methods for these aspects is positively required to make the overall process of designing the SAS more reliable [10]. A combination of formal and semi-formal methods is also used in modeling of SAS [11] and the results have been very encouraging [4]. There have been a few studies where formal methods are used successfully in model checking [12]. A few domain specific languages [13] and design patterns [14] are also proposed for development of SAS.

This paper presents a formal model of MAPE-K loop; we have used Petri nets to model, analyze and verify the working of feedback loop. In this model the inputs, constraints, invariants and process is kept simple for easy of understanding and analysis. This model will serve as the basis of complete adaptive system for a real life scenario. It has been observed that use of Petri nets has been useful in understanding, analyzing and verifying the

feedback loop and the results can be used for further development of a complete system. The paper is organized in to 6 sections; first section gives an introduction of this work, followed by an overview of self-adaptive systems and formal methods in section 2 and 3 respectively. Section 4 gives a detailed description of feedback loop and section 5 presents a formal Petri net model of MAPE-K loop with the description of complete process, analysis and findings. Finally the paper is concluded and pointers to future work are given in section 6.

2. Self-Adaptive systems

Systems that have the capability to alter their behavior during execution are classified as self-adaptive systems [15]. Another classification of these systems is that they fall under the umbrella of context aware systems [16]. Strength of these systems is that they can adjust to the external inputs and work as per the needs of the user [17]. The interesting aspect of these systems is that they are able to perform adaptations for which they were not initially developed [18]. Though, this is one of the major strengths of these types of systems, but this makes it extremely hard for the software engineering professionals to develop systems for incomplete or uncertain requirements [19][20]. System 'shall' statements are converted in to 'may' statements while engineering requirements for these systems. This means that part of requirement engineering has to be conducted at run time [21]. In addition to requirement engineering, testing is also conducted at run time. This process of adaptation is generally carried out using feedback loops [22]. To make these systems efficient all these steps have to be performed autonomously [23] and reliably.

Today, we are surrounded by systems that have capability to adapt or to self-adapt, the difference is that adaptive systems only change their behavior according to the pre-defined requirements; however, the systems that can self-adapt create, test and execute their new requirements at run time. The level of adaptation completely depends on the type of system, i.e. for a mobile interface the adaptation will be shown by screen, for a robot, the adaptation will be executed by the output limbs/channels. The output of these systems varies a lot but generally the output is produced through effectors.

3. Formal Methods

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 [24]. Formal methods provide foundation for describing, analyzing, reasoning, verifying and modeling the complex systems. Formal methods tools provide a comprehensive analysis of the system [25]. Specification of a system is written by using notations which are based on mathematical expressions instead of informal explanations. These notations 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. The syntax and semantic of formal specification languages is a set of precise mathematical expressions based on concrete mathematical principles. The traditional systems development techniques, such as graphical notations and natural languages make the system specification highly ambiguous.

Formal tools facilitate the designers to formally specify the system's requirements and produce its formal model. The tools are also used to check that the model has the desired formally specified requirements. It can be checked that the implementation against the formal model is equivalent and correct with respect to the user requirements. Formal methods provide a methodology that facilitates the development of large scale and complex systems. Petri nets were proposed by Carl Adam Petri [26] in 1962, he proposed these nets based on graph theory and automata to model the dynamic aspects of systems. With the passage of time Petri nets have evolved and multiple variants have been proposed in literature [27]. We have used Petri Nets in this research and have found it to be a very useful formal specification language to analyze, model and verify any system.

4. MAPE-K Feedback Loop

This Loop is also known as MAPE-K Loop [4]. Where all four letters represent four major phases of this loop/cycle. Each phase can be further sub divided in to multiple sub steps where the process for conducting each step varies according to the requirements and system goals. The first major step is Monitor, in this step the system/device takes input from the environment through sensors. This input is then checked with the existing set of requirements, if a requirement exists, no adaptation is performed and the concerned requirement is executed. However, if the inputs do not match with the existing requirements then the loop starts and proceeds to the next phase. In second phase the inputs are analyzed, here multiple approaches can be used for analysis/mapping of data. Once the inputs are analyzed, we proceed with the next phase which is plan. In this phase adaptations are proposed and tested. In this phase validation and verification techniques are applied to check the proposed adaptation. The best adaptation is forwarded to the execute phase and the system acts according to the adaptation. In case the adaption is unsuccessful the loop

executes again [13]. It is named as feedback loop because it takes input from the environment and gives feedback (output) to the same environment and all this is done iteratively [7]. It is to be noted that existing requirements, systems goals, objectives, successful and unsuccessful adaptations are all recorded/present in the knowledge. A simplified version of MAPE-K feedback loop is shown in Fig. 1.

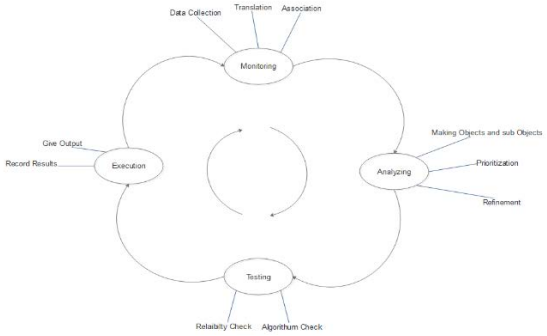

Fig. 1 Phases of MAPE-K loop

5. Modeling and Analysis of MAPE-K using Petri Nets

In this section an abstract Petri net model of MAPE-K loop is described. Fig. 2 gives an initial state of Petri net model for MAPE-K loop. This model is analyzed and verified using CPN tool.

The inputs, conditions and outputs are kept simple for ease of understanding and analysis. Following is a description of each state and transition. The details are elaborated in points to understand the complete process in a proper sequence. We have used colored Petri nets [28] to model this system. The model has been analyzed and verified using CPN tool [29].

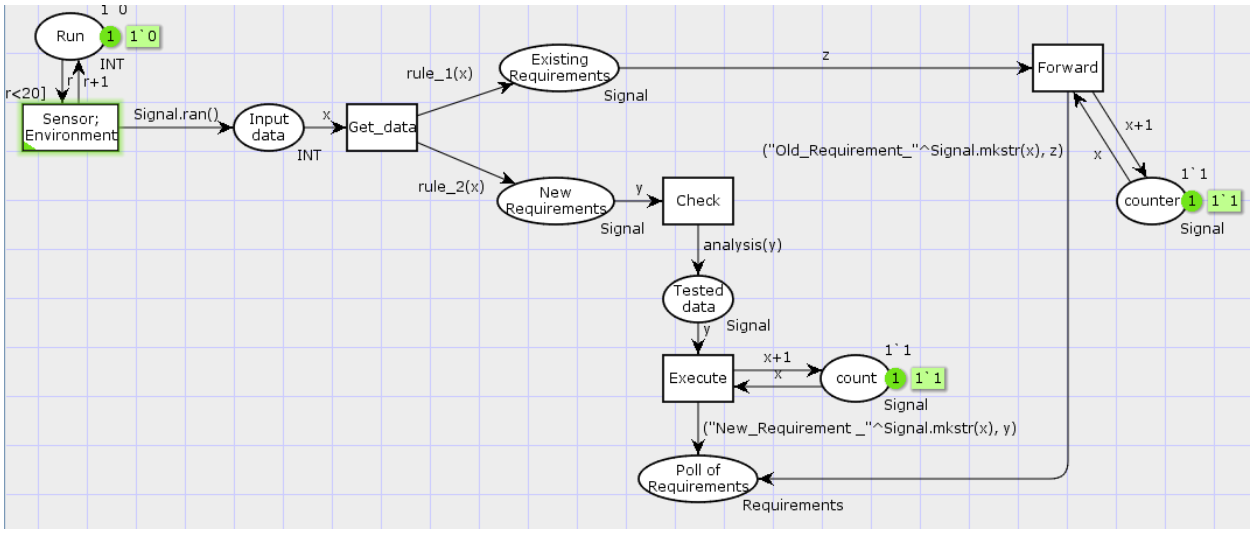

Fig. 2 Initial state of MAPE-K Petri Net

Following is a step wise description of each state and transition:

1. **Run:** This is the initial state of the system, from this state a random integer is generated using the random function; the function is executed by the transition 'sensor environment'. This state keeps the counter of number of transitions fired.
2. **Sensor Environment:** This transition is fired by receiving token from the 'Run' state and generates the random number which is transferred to 'input data' state. Table 1 gives the description of the functions and conditions that were applied during the analysis of the Petri net model for feedback loop.

Table 1: Type and color of inputs of MAPE-K Petri Net

Sr. #	Color Sets	Specification
1	Req	String type
2	Signal	List of integers from 1 to 100
3	Requirements	Product of color set Req and Signal

3. **Input Data:** This is a state which stores/ receives the data received from the transition 'sensor environment'. The data will be received as a result firing of transition. Table 2 gives the

description of the functions and conditions that were applied during the analysis of the Petri net model for feedback loop.

4. **Get Data:** This transition processes the data from the 'Input Data' state and it is checked whether the input is less than 50 or not. It is assumed that 1 to 50 represent the existing requirements.
5. **Existing Requirements:** In case the input is less than 50, the token is moved to this state and no action is further required from this state as the system will perform the action specified in the existing requirements.
6. **New Requirements:** If input is more than 50, we move to New Requirements, in this case the adaptation is required and complete feedback loop has to be executed. From here the analysis phase of MAPE-K will be finalized. Also planning and Execution will be done in next steps.

Table 2: Functions of MAPE-K Petri Net

Sr. #	Function name	Function
1	rule_1	fun rule_1(x:INT)= if x<=50 then 1`x else empty;
2	rule_2	fun rule_2(y:INT)= if y>50 then 1`y else empty;
3	Analysis	fun analysis(x:INT)= if x>59 andalso x<=90 then 1`x else empty;

7. **Check:** In this transition, planning, testing and execution is done and we generate the tested data. Please note that conditions applied here are assumed to keep the model simple and to understand and analyze the MAPE-K loop using Petri nets.
8. **Tested Data:** This state is reached after the data is checked and here we have the requirements which have been tested and are ready to execute.
9. **Execute:** Execute transition executes the action and the new requirement is added in pool of requirements. Additionally number of successful executions are also recorded and stored in 'count' state.
10. **Count:** This keeps the count of the new requirements which have been successfully analyzed, tested and executed.
11. **Pool of Requirements:** This state stores the new requirements which have been generated due to a new set of inputs from the sensors and after performing analysis, planning and testing on the new set of inputs.
12. **Forward:** This transition performs two actions; one is that it appends the existing requirements with the new pool of requirements to form complete set of requirements of the system. Secondly it keeps a counter which is incremented every time after the input data matches with the existing requirements. This helps us to understand the need of adaptation by calculating the ratio of inputs that needed adaptation and the inputs where the adaptation was not required.
13. **Counter:** The counter which is maintained by 'Forward' transition maintains the counter of requirements which did not require any adaptation. Fig. 3 presents the model after fourteen executions and the adaptations were conducted once in accordance with the given inputs and constraints.

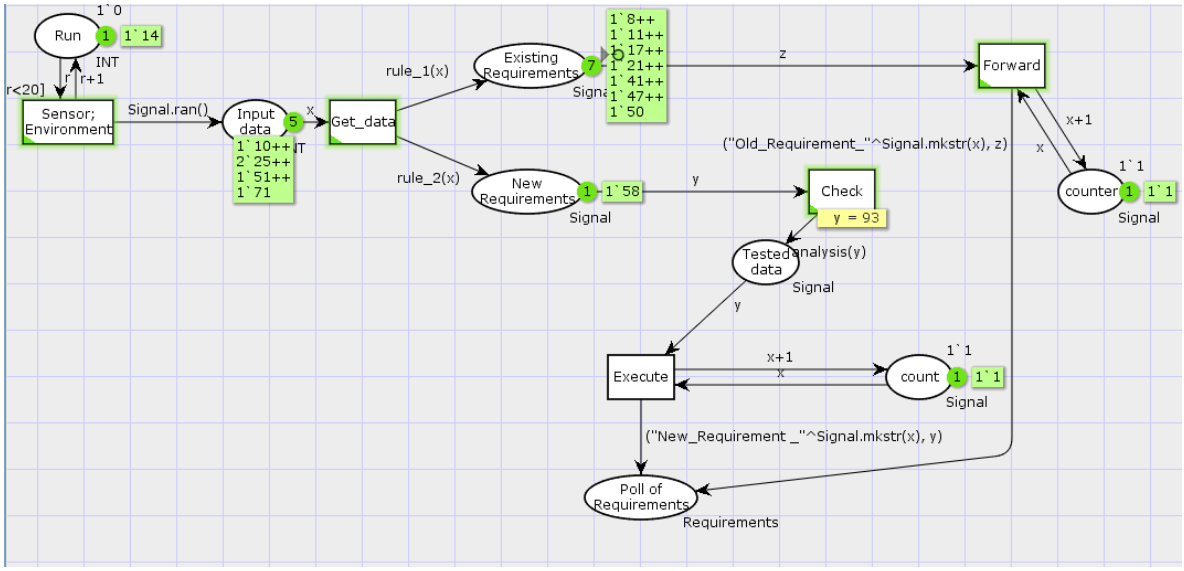

Fig. 3 MAPE-K Petri Net after multiple successful executions

The above model represents the MAPE-K loop and it has been observed that application of Petri nets to model the feedback loop has not only been useful but it has also increased the confidence throughout the development process. This step clearly establishes the need and usefulness of applying Petri nets in analysis, modeling, verifying and simulation of self-adaptive systems. This was an initial attempt and in future more variables, sensors, conditions and scenarios will be added to develop a complete model of a real life adaptive system using Petri nets.

6. Conclusion and Future Work

Efficient and reliable working of feedback loop is the key towards a successful adaptation in self-adaptive system. In this research we have successfully modeled feedback loop using Petri nets. The application of formal methods in modeling the feedback loop using Petri nets has been successful. Due to dynamic properties of Petri nets and their strengths in modeling concurrent systems, this effort will go a long way in development of a complete system using Petri nets. After modeling the MAPE-K loop we can confidently conclude that Petri net will plays key role in analysis, development, verification and modeling of a real life system. It is to be noted that all the steps of adaptation are autonomously performed and we hope that a complete multi-agent formal model will help the efficient and reliable working of self-adaptive systems

In future, this model will be enhanced for multiple sensors inputs, data types and conditions to appreciate the working of the feedback loop for a more complex system. Further, a complete formal model will be developed for a real life system - self driving cars [30]. Finally, we intend to utilize the strength of Petri nets to model, analyze, develop and verify distributed adaptive systems where multiple feedback loops are involved.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] T. Patikirikorala, A. Colman, J. Han, and L. Wang, “A systematic survey on the design of self-adaptive software systems using control engineering approaches,” in 2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012, pp. 33–42.

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

[7] 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.

[8] S. W. Cheng, D. Garlan, and B. Schmerl, “Evaluating the effectiveness of the rainbow self-adaptive system,” in Proceedings of the 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2009, 2009, pp. 132–141.

[9] 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.

[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] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] C. A. Petri, "Kommunikation mit Automaten," Fakultät für Mathematik und Physik, vol. Doktor. p. 128, 1962.

[27] M. Koutny, J. Kleijn, and W. Penczek, "Transactions on petri nets and other models of concurrency XII," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2017, vol. 10470 LNCS, pp. IV–V.

[28] C. Rohr, "The Manual for Colored Petri Nets in Snoopy – QPNC/SPNC/CPNC/GHPNC," Science (80-.)., no. March, 2012.

[29] A. V. Ratzer et al., "CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets," Proc. 24th Int. Conf. Appl. theory Petri nets, vol. 2679, no. Chapter 28, pp. 450–462, 2003.

[30] T. Luettel, M. Himmelsbach, and H.-J. Wuensche, "Autonomous Ground Vehicles ―Concepts and a Path to the Future," Proc. IEEE, vol. 100, no. Special Centennial Issue, pp. 1831–1839, 2012.