Design and implementation of autonomic simulator

Design and Implementation of Autonomic Simulator

Zulfiqar Ali
School of Computing
Ulster University
United Kingdom
e-mail: z.ali@ulster.ac.uk

Botond Virginas
Research and Innovation, BT
Ipswich, United Kingdom
e-mail: botond.virginas@bt.com

Bryan Scotney
School of Computing
Ulster University
United Kingdom
e-mail: bw.scotney@ulster.ac.uk

Darryl Charles
School of Computing, Eng & Intel. Sys
Ulster University
Londonderry, United Kingdom
e-mail: dk.charles@ulster.ac.uk

Anousheh Ramezani
School of Computing
Ulster University
United Kingdom.
e-mail: ramezani-a@ulster.ac.uk

***Abstract*—Autonomic systems have broad scope in the telecommunication industry, where the prime objective is to provide quality services to customers whilst obeying certain financial constraints. Therefore, an autonomic system is designed and developed in this study to determine a trade-off between cost and quality. The developed simulator is capable of responding to unforeseen changes appearing over time as well as in business policy without human intervention. It learns from data using a machine learning algorithm and performs classification to assign the instances of data to corresponding groups. Various experiments are carried out to observe the performance and behaviour of the simulator.**

Keywords-self-adaptive system; cost and quality; decision tree; business policy; stability.

I. INTRODUCTION

Autonomic computing is an approach that equips a system with an intelligent capability to modify its behaviour during execution corresponding to the unpredicted dynamic changeable environmental conditions or internal state modification without human intervention [1, 2]. In addition to robustness against changes, self-adaptive systems (SAS) are also rapidly responsive in learning and converge to the steady state [3, 4]. The objective of autonomic computing is to give assistance to customers and construct complex platform to reduce unforeseen issues such as errors and cost [5]. Systems with autonomic computing provide a means to modify the level of responsiveness and automation to handle the complex platform in business organizations, including telecommunications industries. An autonomic system can be developed for providing a trade-off between cost and quality by proposing a framework using suitable self-adaptive approaches.

As investigated in [6], different methods can be implemented for building SAS. The methods may be classified in different categories: agent-based, model-based [7], service-oriented, nature-inspired, architecture-based [8], reflection-based, control theory, programming paradigms, learning, formal modelling & verification, and requirements-oriented. A recent study [9] also presented a comparison of various approaches for developing SASs. Rainbow framework is an architecture-based approach in which adaptation is accomplished through the implementation of an adaptation strategy chosen at runtime [10]. Model-Driven approach is a framework that provides model-based support for developing pervasive systems in order to add a resource or eliminate resources from a system [11, 12]. Meta-Self is a service-oriented framework that provides controllable and dependable engineering requirements for supporting the development of SASs [13]. Graph-based runtime adaptation framework (GRAF) is a runtime adaptation framework which exploits TGraphs and accompanying technologies, as the enabling technology, for manipulating and modelling runtime models [14]. The dynamic and unpredictable circumstances of many systems challenge the stability and Quality of Service. Self-adaptivity brings into concerns the requirement for a shift to a wider concept of stability that tackles runtime changes of systems and their environment, as well as the runtime uncertainty. Moreover, a stable system can reduce maintenance costs. From the standpoint of being economic, stability is desirable to safeguard customers' satisfaction and reputation of service provider. Lyapunov-based approaches are applied to guarantee stability [15].

978-1-7281-4577-8/19/$31.00 ©2019 IEEE

In this study, a self-adaptive system (autonomic simulator) using machine learning is designed and implemented. The autonomic process in the simulator evolves iteratively over time in discrete stages. The time between successive stages, which might be, for example one week or a month, depends on the nature of the problem. At each stage, the outcomes of process instances up to that point are analysed to achieve high-level organisational objectives in terms of costs and user experience (quality). The autonomic process can be started through an in-practice algorithm, which divides data into two groups, or by assigning data directly to any of the groups. A machine learning approach is applied to learn from data for generation of models. There are various machine learning algorithms, including Support Vector Machine, Naive Bayes, Decision Tree, Random Forest and Neural Networks [16, 17, 18, 19]. In this study, Classification and Regression Tree (CART) is used to generate models for classification of instances [20]. One of the positive aspects of CART is that it does not require any assumptions of linearity in the data, unlike some other approaches such as regression models [21]. The process will continue until trade-off is reached between cost and quality (an external decision criteria) or enough elements of each category are available to generate more models. Various experiments are performed to analyse the proposed simulator.

The rest of the paper is organised as follows: Section II describes the main functionality of the proposed simulator. All components of the simulator, including statistics of the dataset and objective function for cost and quality are discussed in this section. Section III illustrates the learning of the autonomic simulator and provides experimental results. Finally, Section IV draws some conclusions and suggests future directions.

II. Proposed Autonomic Simulator

All major components of the proposed autonomic simulator to explain its functionality are illustrated in this section. In addition, the use case of the developed simulator with statistics of the dataset is also explained.

A. Customer Services Dataset

All experiments are performed using the historical data of BT customer services. Customers make calls to customer services to place new orders or to register complaints, and they should be dealt with a priority for better customer experience when they reach at a certain duration of calls. If customers are unable to receive attention when needed, then resolution of customers' issues will be delayed, and this phenomenon can be referred to as a failure. The failure cases in the dataset are simulated using the static algorithm with a threshold of 200 minutes. Any customer with the duration of more than 200 minutes is considered as a failure case. Due to confidentiality, the static algorithm cannot be disclosed.

The dataset contains 834,235 customers' journeys. In the dataset, 33,904 customers have a failure (abbreviated as FAIL), and need priority/escalation in their matters through human involvement. The remaining customers (800, 331), abbreviated as NOR (normal), are handled in due time and don't need any kind of special attention. The number of calls made by FAIL and NOR customers are 812,360 and 2,424,113, respectively. So, the total number of calls in the dataset is more than 3 million (3,236, 473), where each call is an instance and represented by a row in the dataset. Each call contains various attributes such as unique identifier for a customer, age of broadband, television, and landline connections.

To improve the customer experience, ideally, each customer should be dealt with a priority, but it is not cost-effective nor essential essential because most of the time customers' issues are resolved within a given timeframe. From the statistics of the dataset, it can be observed that the average number of calls by FAIL customers is approximately 24, whereas, for NOR customers the average is 3, which indicates that the matter is resolved without causing any trouble to the customers. So, the proposed simulator will determine whether a customer should be escalated or not?

Although the simulator must be reliable in decision making, it is vital that it should achieve a trade-off between cost and quality. In other words, by giving priority to every customer, the user experience would improve significantly, but result in increase of cost. Similarly, if many customers who need escalation are ignored, this would result in a bad reputation for an organization, which is also not acceptable. Therefore, the simulator should be optimal in the context of cost and quality. The functionality of the simulator is described in the following subsections.

B. Different Starting Conditions of the Simulator

The dataset is divided into discrete stages, say periods, with the assumption that each period represents data over a month. Every period contains 30,000 instances approximately, and the autonomic process takes a period each time for processing.

The simulator can be initiated using various starting conditions. One of the ways to generate starting conditions is direct allocation of all instances in any of the groups or their portioning into different groups. The other way to create the conditions is the use of the static algorithm. The following three staring conditions are created by adjusting a variable *baseScore* of the static algorithm. The values of the variable are determined through experiments.

For the first starting condition, all instances are assigned to a group (say Group1), where customers do not need escalation. For this starting condition, the autonomic process escalates customers' issues by shifting the instances from Group1 to Group2 (where all customers need escalation). This is referred to as the forward autonomic process. For the second starting condition, all instances can be allocated to Group2. Moreover, for the third starting condition, the instances can be partitioned into both groups (Group1 and Group2).

In this study, we are considering the simulator with the first starting condition (forward autonomic process).

C. Modes of Simulator

There are two important modes of the simulator: offline and online modes. In the offline mode, the simulator learns from data through a machine learning algorithm and

generates models. In this study, we have implemented CART, which generates models in the form of decision trees. Later, these decision trees are used for the classification of instances in online mode. The simulator shifts instances into Group2 if they are expected to have a failure (in our current case, which is the forward autonomic process).

D. Recursive Autonomic Process

When the autonomic process starts, first it generates a decision rule if the criterion of acceptance is satisfied. The process may take several periods to find a rule. Once a rule is found, then it is applied to the subsequent period to shift the instances that are expected to have a failure from Group1 to Group2. After shifting, the remaining instances in Group1 will be used to determine another rule. If the second rule is found, then both rules will be applied to the next period one by one to shift instances, and the remaining instances will be used to determine a new rule. The process will continue until there are not enough instances remaining to determine a new rule or a trade-off between cost and quality is achieved.

The decision criteria for acceptance of a rule depends on the performance metrics, which is defined in the following section.

E. Performance Metrics for Acceptance Criteria of Rule

The accuracy (ACC) is computed for the acceptance criteria to make sure that the accepted rules are reliable and accurate in shifting of instances from one group to another. The accuracy is calculated using the relation given in Eq. (1).

- $$ACC = \frac{TP + TN}{TP + FN + TN + FP} \quad (1)$$

ACC is used to evaluate all rules that are generated using a cross validation approach.

F. K-Fold Cross Validation

K-fold cross validation is a reliable approach to capture intra-variation of data. This approach divides the data into disjoint subsets. Each time one of the subsets is used to evaluate a generated rule, the remaining *k*-1 subsets are used to train the machine learning algorithm. The number of instances for the folds is a user input, and the autonomic simulator automatically determines the value of *k* (number of folds). ACC is computed for all generated decision trees. The rule which satisfies the selection criteria is accepted for shifting of instances.

G. Selection Criteria to Accept Rule

An appropriate selection criterion to accept a rule for shifting of instances is crucial. If a rule is not accepted carefully, it will shift the wrong instances to the other group. Resultantly (in the case of the forward process), it will not only increase the cost by allocating human effort to escalate customers' issues when it is not necessary, but also compromises the quality as failure instances will stay in the same group when they should have been escalated. The following criterion is implemented to accept a rule.

- $$\max(ACC) \geq threshold \quad (2)$$

According to this criterion, a rule will be accepted if the maximum value of ACC is greater or equal to a *threshold*. In this study, initially, the threshold is 70%. Some more experiments can be performed in the future to analyse the impact of the threshold on the behaviour of the proposed simulator.

H. Trade-off between Cost and Quality

One of the terminating conditions for the recursive process is a trade-off between cost and quality. The objective function, given by Eq. 3, is optimised to determine the trade-off.

In the objective function, $TotalCost_i$ represents the sum of costs associated with escalation of matters (*EscCost*) and failure of instances (*FailureCost*) for any i^{th} period. For the forward recursive process, *SI* stands for the number of shifted instances to Group2 and *RFI* means remaining failure instances in Group1 after shifting. Moreover, *SI* is the sum of truly and erroneously shifted instances, and *P* (positive class for the forward process) denotes the total failure instances in a period. In addition, *x* and *y* are the costs for each escalation and failure instance, respectively. In this study, we assumed that the cost of one failure instance is double than an escalation instance, i.e., y= 2x.

$$\begin{aligned} &\text{minimise}(TotalCost_i) \\ &\text{where} \\ &TotalCost = EscCost + FailureCost \\ &= (SI \times x) + (RFI \times y) \\ &= [(TP + FP) \times x] + [(P - TP) \times y] \\ &\text{and} \\ &Quality = \frac{F_0 - F}{F_0} \times 100 \\ &\text{with } F = \frac{P - TP}{TotalInst} \times 100 \end{aligned} \quad (3)$$

Furthermore, *Quality* represents the reduction in the failure rate (*F*) for the i^{th} period with respect to the initial failure rate (F_0), where the failure rate is a percentage of the failure instances in a period and *TotalInst* denotes the total number of instances in a period. The performance and behaviour of the simulator are analysed in the following section.

III. Experimental Results and Behaviour of Simulator

Several experiments are performed to observe the performance of the simulator.

The forward process is started by assigning all instances into Group1. As no rule exists, there is no shifting of instances. Using the first period, the first rule is generated by satisfying the selection criteria. The rule is accepted when ACC is more than 70%. A summary of the forward process during execution is provided in Fig. 1 and learning of the

autonomic process by generating new rules as time evolves is shown in Fig. 2.

```
Forward Process
Initial Failure Rate = 27.7494
1 Rule(s) found: OPT1 --> OPT2
Period: 2, 1479 Orders are Shifted, LRSO = 1479: OPT1 --> OPT2, Failure Rate = 23.7334%, Reduction: 15%
Period: 3, 2582 Orders are Shifted, LRSO = 1089: OPT1 --> OPT2, Failure Rate = 21.0393%, Reduction: 26%
Period: 4, 3292 Orders are Shifted, LRSO = 839: OPT1 --> OPT2, Failure Rate = 18.3579%, Reduction: 35%
Period: 5, 4126 Orders are Shifted, LRSO = 892: OPT1 --> OPT2, Failure Rate = 15.576%, Reduction: 44%

Period: 21, 8200 Orders are Shifted, LRSO = 23: OPT1 --> OPT2, Failure Rate = 6.4935%, Reduction: 78%
Period: 22, 8250 Orders are Shifted, LRSO = 37: OPT1 --> OPT2, Failure Rate = 5.8737%, Reduction: 79%
Period: 23, 8558 Orders are Shifted, LRSO = 18: OPT1 --> OPT2, Failure Rate = 6.098%, Reduction: 79%
Period: 24, 8244 Orders are Shifted, LRSO = 93: OPT1 --> OPT2, Failure Rate = 5.8708%, Reduction: 79%
Period: 25, 8033 Orders are Shifted, LRSO = 29: OPT1 --> OPT2, Failure Rate = 5.9956%, Reduction: 78%
```

Figure 1. Summary of the forward process during execution.

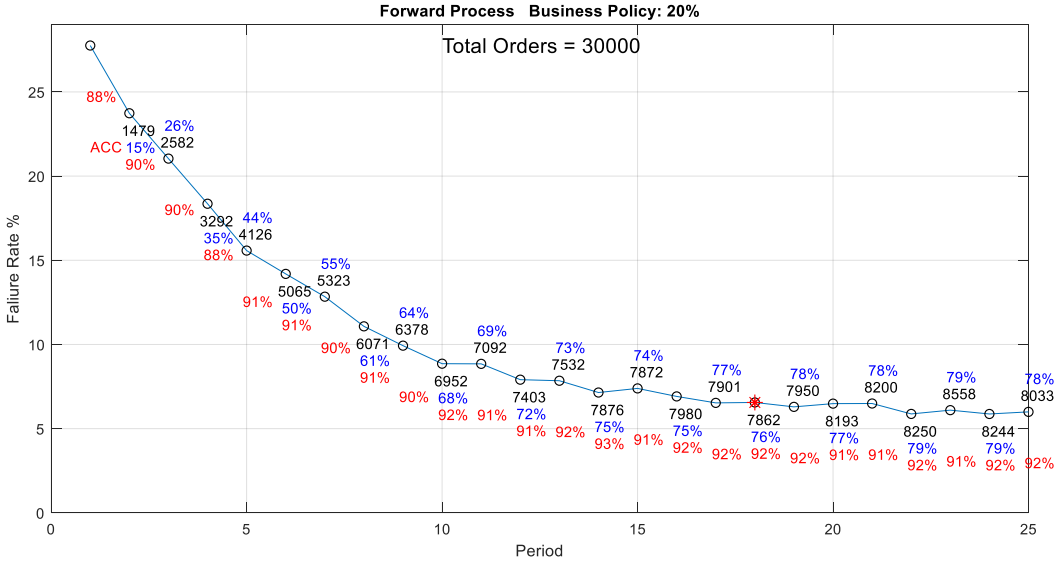

Figure 2. Learning of autonomic process (offline mode) and optimal point for cost and quality.

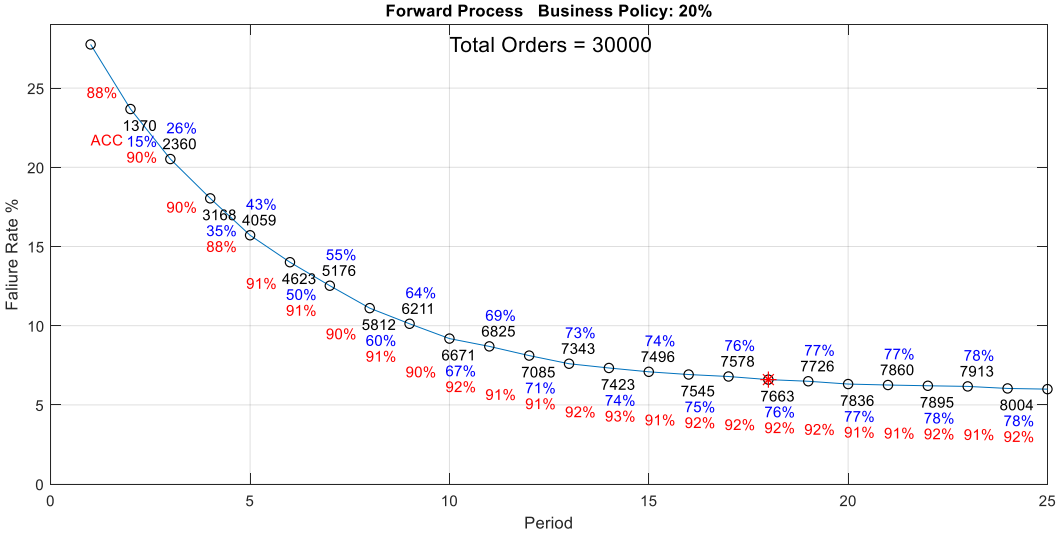

Figure 3. All generated rules during learning applied on a randomly selected period (online mode).

Fig. 1 shows that the first rule is found in the first period. This rule is applied on the second period, and it shifted 1479 instances to Group2 for escalation of the customers' issues. LRSO represents the number of instances shifted by the most recent rule. So far, we have only one rule; therefore, the first and the most recent rule are same. The failure rate informs that 23% failure instances still existed in period 2; initially it was 27%. Moreover, reduction describes that the quality

(reduction in failure instances) is improved by 15%. After shifting of 1479 instances, the forward autonomic process updated Group1. Then, the remaining instances in Group1 determine the second rule. Now, both rules are applied to the instances in period 3, and they shifted 2582 instances. The number of instances shifted by the last rule (second rule) is 1089. The failure rate reduced to 21%, and the quality is improved by 26% using the two rules.

Fig. 2 shows the trend of failure rate as time progressed. After 25 periods, 24 rules are generated, which reduced the failure rate from 27% to 6% and the number of failure instances reduced by 78%. In addition, Fig. 2 shows that when the first rule was accepted, ACC was 88%, and for the second rule was 90%. Similarly, the ACC for all rules is shown. The percentage in blue colour describes the quality achieved for each period. A red asterisk on period 18 shows the optimal point for cost and quality. The total cost is minimum at this point, which means that 7862 instances can be afforded for escalation, with the quality of 76%. This learning procedure represents the offline mode of the forward process.

Now all learned rules are applied to a randomly selected period, which represents the online mode of the autonomic process. After applying all 24 generated rules on the selected period, the performance of the simulator is shown in Fig. 3. The optimal point for this period is also achieved at the 18th period, with 7663 shifted instances and quality of 76%. The location of the optimal point may vary as it is not necessary that all periods contain the same number of failure instances. In Fig. 2 and 3, the business policy is adjusted to 20% which means that only 20% of the instances suggested by a rule will be shifted to Group2.

IV. Conclusion

A simulator to achieve high level organisational objectives in the context of cost and user experience is designed and implemented in this study. The adaptiveness in the proposed simulator is introduced by applying a machine learning approach that iteratively analyses the simulator at each stage. Iterative learning makes the simulator responsive to the unforeseen changes. In the future, the responsiveness of the simulator as well as its behaviour when business policy changes during execution will be observed. Moreover, the other cases when all instances are allocated to Group2 and/or they are partitioned to both groups will be considered. This will help to determine whether the simulator will converge to the same point when starting conditions are different. Moreover, the criteria for the stability of the simulator will be found and implemented.

Acknowledgment

This work was supported by BTIIC (BT Ireland Innovation Centre), funded by BT and Invest Northern Ireland.

References

[1] H. Tabassum and S. Sagar, “International Journal of Advanced Research in Approaches and Concepts of Self Optimization in Autonomic Computing Systems,” vol. 6, no. 2, pp. 31–34, 2016.

[2] 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 Syst. Appl., vol. 40, no. 18, pp. 7267–7279, 2013.

[3] M. Stefanovic, R. Wang, and M. G. Safonov, “Stability and Convergence in Adaptive Systems,” pp. 1923–1928, 2004.

[4] A. Paz and H. Arboleda, “A Model to Guide Dynamic Adaptation Planning in Self-Adaptive Systems,” Electron. Notes Theor. Comput. Sci., vol. 321, pp. 67–88, 2016.

[5] M. C. Huebscher and J. A. Mccann, “A survey of Autonomic Computing,” ACM Comput. Surv., vol. 40, no. 3, pp. 1–28, 2008.

[6] 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.

[7] R. France and B. Rumpe, “Model-driven development of complex software: A research roadmap,” FoSE 2007 Futur. Softw. Eng., pp. 37–54, 2007.

[8] J. Cámara, A. Lopes, D. Garlan, and B. Schmerl, “Adaptation impact and environment models for architecture-based self-adaptive systems,” Sci. Comput. Program., vol. 127, pp. 50–75, 2016.

[9] C. Krupitzer, M. Pfannemüller, V. Voss, and C. Becker, “Comparison of Approaches for developing Self-adaptive Systems,” 2018.

[10] A.-C. Huang, B. Schmerl, P. Steenkiste, D. Garlan, and S.-W. Cheng, “Rainbow: architecture-based self-adaptation with reusable infrastructure,” Computer (Long. Beach. Calif)., vol. 37, no. 10, pp. 46–54, 2004.

[11] V. P. Carlos Cetina, Pau Giner, Joan Fons, “A Model-Driven Approach for Developing Self-Adaptive Pervasive Systems,” Models@RT, 2008.

[12] B. Magableh, “A Framework for Evaluating Model-Driven Self-adaptive Software Systems,” pp. 1–11, 2019.

[13] G. Di Marzo Serugendo, J. Fitzgerald, and A. Romanovsky, “MetaSelf,” no. May, p. 457, 2010.

[14] 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.

[15] R. J. Lian, “Design of an enhanced adaptive self-organizing fuzzy sliding-mode controller for robotic systems,” Expert Syst. Appl., vol. 39, no. 1, pp. 1545–1554, 2012.

[16] N. Couellan, S. Jan, T. Jorquera, and J. P. Georgé, “Self-adaptive Support Vector Machine: A multi-agent optimization perspective,” Expert Syst. Appl., vol. 42, no. 9, pp. 4284–4298, 2015.

[17] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, “Machine learning: A review of classification and combining techniques,” Artif. Intell. Rev., vol. 26, no. 3, pp. 159–190, 2006.

[18] F. Lolli, E. Balugani, R. Gamberini, and B. Rimini, “Quality cost-based allocation of training hours using learning-forgetting curves,” Comput. Ind. Eng., vol. 131, pp. 552–564, May 2019.

[19] M. Praveena and V. Jaiganesh, “A Literature Review on Supervised Machine Learning Algorithms and Boosting Process,” Int. J. Comput. Appl., vol. 169, no. 8, pp. 32–35, 2017.

[20] C. Yu, "Adaptive Japanese Teaching Optimization Based on Classification and Regression Tree," in 2017 International Conference on Robots & Intelligent System (ICRIS), 2017, pp. 15-18.

[21] J. Chen, Y. Lin, and Y. Leu, "Predictive model based on decision tree combined multiple regressions," in 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2017, pp. 1855-1858.