Towards Self-adaptation Planning for Complex Service-Based Systems

Azlan Ismail[1] and Valeria Cardellini[2]

[1] Faculty of Computer and Mathematical Sciences
Universiti Teknologi MARA (UiTM), Malaysia
azlanismail@tmsk.uitm.edu.my
[2] Department of Civil Engineering and Computer Science Engineering
University of Roma Tor Vergata, Italy
cardellini@ing.uniroma2.it

Abstract. A complex service-based system (CSBS), which comprises a multi-layer structure possibly spanning multiple organizations, operates in a highly dynamic and heterogeneous environment. At run time the quality of service provided by a CSBS may suddenly change, so that violations of the Service Level Agreements (SLAs) established within and across the boundaries of organizations can occur. Hence, a key management choice is to design the CSBS as a self-adaptive system, so that it can properly plan adaptation decisions to maintain the overall quality defined in the SLAs. However, the challenge in planning the CSBS adaptation is the uncertainty effect of adaptation actions that can variously affect the multiple layers of the CSBS. In a dynamic and constantly evolving environment, there is no guarantee that the adaptation action taken at a given layer can have an overall positive effect. Furthermore, the complexity of the cross-layer interactions makes the decision making process a non-trivial task. In this paper, we address the problem by proposing a multi-layer adaptation planning with local and global adaptation managers. The local manager is associated with a single planning model, while the global manager is associated with a multiple planning model. Both planning models are based on Markov Decision Processes (MDPs) that provide a suitable technique to model decisions under uncertainty. We present an example of scenario to show the practicality of the proposed approach.

Keywords: Self-adaptation, Adaptation planning, Cross-layer services, Markov Decision Process.

1 Introduction

Service-based systems are becoming increasingly complex (also called CSBS) due to their multi-layer structure and the heterogeneous and dynamic execution environment in which they operate [14]. The multi-layer structure of CSBS is referred to the application, platform, and infrastructure layers. The application layer consists of the composite software services to fulfill the business process

A.R. Lomuscio et al. (Eds.): ICSOC 2013 Workshops, LNCS 8377, pp. 432–444, 2014.

activities. The platform layer provides the computing platforms to execute and manage services, while the infrastructure service layer provides the resources (computing, storage, network) to provision software services. These layers are inter-related to fulfill the CSBS's goals. The complexity of CSBS imposes challenges in managing its lifecycle in a multi-cloud environment [2], where multiple Clouds can be used in a concomitant way to offer the service and each of the CSBS layers may be deployed by different Cloud providers.

Self-adaptation in autonomic computing [13] is the prominent paradigm to manage and maintain the quality of service (QoS) of CSBS. The key idea of self-adaptation is to introduce the IBM's MAPE (Monitor, Analyze, Plan, Execute) loop into the system. The self-adaptation goal is to alleviate the software management efforts in managing highly changing and evolving environments. During run-time, the monitoring component observes the CSBS behavior and detects or predicts any problematic situation such as failures and SLA violations [22]. If a problematic condition is detected, the analysis component analyzes the situation to discover more information such as the impact, or the cause of the failure. Then, the planning component decides the appropriate adaptation strategies to be undertaken by the execution component.

The Quality of Service (QoS) of CSBS needs to be maintained during run-time. The QoS characteristics are specified in an agreement known as Service Level Agreement (SLA), which contains the contractual service levels to be met by the services of each layer. The self-adaptation framework can utilize the multiple SLA information [7] to maintain the QoS of CSBS. The monitoring component can use the contractual information to observe, detect, and predict any SLA violation. The analysis component can use the contractual and the observed information to discover new information, such as the impact region [10]. The planning component can use the discovered information to decide about appropriate adaptation strategies, such as which layer to be adapted and what kind of adaptation action to be executed.

Planning and deciding the appropriate adaptation actions are challenging research problems. There are two core factors, namely, the complexity of CSBS and the uncertainty effect, which need to be taken into consideration at the same time. In general, there are several sources of uncertainty, such as those discussed in the context of service delivery [20] and self-adaptive software systems [8]. Herein, the uncertainty effect refers to the CSBS state as a result of executing the adaptation actions. Meanwhile, the complexity refers to the multi-layer interactions among services and the CSBS dynamism. These factors demand a robust adaptation planning and failing to consider these factors may cause a failure in maintaining the overall QoS of CSBS.

This paper contributes to twofolds. First, a conceptual framework of multi-layer self-adaptive service-based system. The uniqueness of the framework is the decentralized adaptation managers to support the multi-layer planning. Second, a planning model based on Markov Decision Processes (MDPs) to appropriately select the adaptation action for the respective service layer. This model can

complement the existing decision making proposals in selecting the adaptation strategy.

The remainder of this paper is organized as follows. We present a decentralized self-adaptation architecture for multi-layer services in Section 2. Then, we elaborate the model and the method for solving the adaptation planning problem in Section 3. We discuss a motivating example to illustrate the practicality of the approach in Section 4. In Section 5, we analyze the related work. Finally, we summarize and highlight future work in Section 6.

2 Self-adaptation Framework for CSBS

The key idea to enable self-adaptation in service-based systems is to adopt the IBM's MAPE reference framework [13]. It consists of four components (Monitor, Analyze, Plan, and Execute) that interact in a feedback control loop. The Monitor is responsible to observe the system behavior and to detect or predict any problematic situation, e.g., a SLA violation. The Analyze component is responsible to gain more information about the identified problem and to decide whether it occurs to trigger an adaptation. The Plan component is responsible to produce a policy or a plan to support the adaptation. The generated plan can contain which service layer(s) to be adapted and what adaptation action(s) to be executed. Finally, the Execute component is responsible to execute the planned adaptation actions.

A single MAPE has been argued as to be insufficient to adapt a CSBS due to its multi-layer architecture. The main challenge of a multi-layer architecture is the complexity of the system in dealing with changes and evolution. The complexity is attributed to the vertical and horizontal dependencies among the services in the CSBS. Thus, in this paper, we propose a self-adaptation framework for CSBS with multiple MAPE loops as illustrated in Figure 1.

The framework consists of the adaptation managers and the CSBS. The adaptation manager is classified into two types of managers, namely the *global adaptation manager* (GAM) and the *local adaptation managers* (LAMs), on which we focus below. The proposed framework follows the hierarchical control pattern as discussed in [21]. This pattern is suitable to manage the complexity of self-adaptation by providing a hierarchy of MAPE loops. The higher-level MAPE loop concerns with the global adaptation, while the lower-level MAPE loops concern with the local adaptation. In the case of demanding the local adaptation only, this pattern can potentially reduce the adaptation time. Its drawback is the possibility of not being able to achieve a global adaptation due to conflict of interests from the lower level. Furthermore, the global MAPE loop might be dealing with a considerable workload of adaptation requests triggered by the lower MAPE loops. We will investigate in future work the evaluation of the effectiveness and efficiency of the hierarchical control pattern.

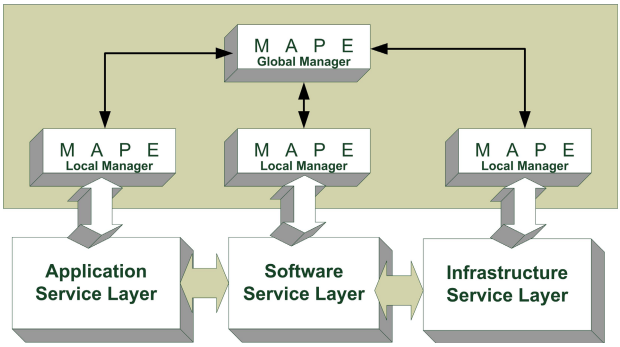

Fig. 1. Self-adaptation framework for CSBS

2.1 Adaptation Managers

The framework contains two types of adaptation managers, GAM and LAMs, which operate at different levels of abstraction and may operate at different time scales.

The LAM is concerned with a single, specific layer of the service-based system. It consists of all the MAPE components:(1) **M**onitor, which is responsible to determine the abnormality within the layer; (2) **A**nalysis, which is responsible to determine whether an adaptation is required and what needs to be adapted; (3) **P**lanning, which is responsible to plan the appropriate adaptation action for the layer; (4) **E**xecutor, which is responsible to execute the adaptation plan.

Meanwhile, the GAM is concerned with the overall service-based system. It consists of all the MAPE components: (1) **M**onitor, which is responsible to monitor the abnormality notification triggered by the local monitors; (2) **A**nalysis, which is in charge to determine the joint effect of the cross-layer adaptation; (3) **P**lanning, which is in charge to plan the appropriate adaptation strategy for the entire system; and finally,(4) the **E**xecutor, which is in charge to properly instruct the local executors to perform the adaptation.

2.2 Adaptation Interaction Process

The interaction among GAM, LAMs, and CSBS can be presented in terms of a UML sequence diagram as depicted in Figure 2. The adaptation process is perceived as a continuous activity of monitoring and adapting the CSBS.

Each LAM monitors the CSBS behavior at runtime, while the GAM monitors each LAM. If an abnormal condition is detected at a specific layer, the respective LAM Analyzer is executed and the respective LAM notifies the GAM. The LAM proceeds with the local planning and then waits to synchronize with the GAM's decision.

From the GAM perspective, it performs a global analysis upon receiving the notification. Then, the GAM analyzes the abnormal conditions to understand the

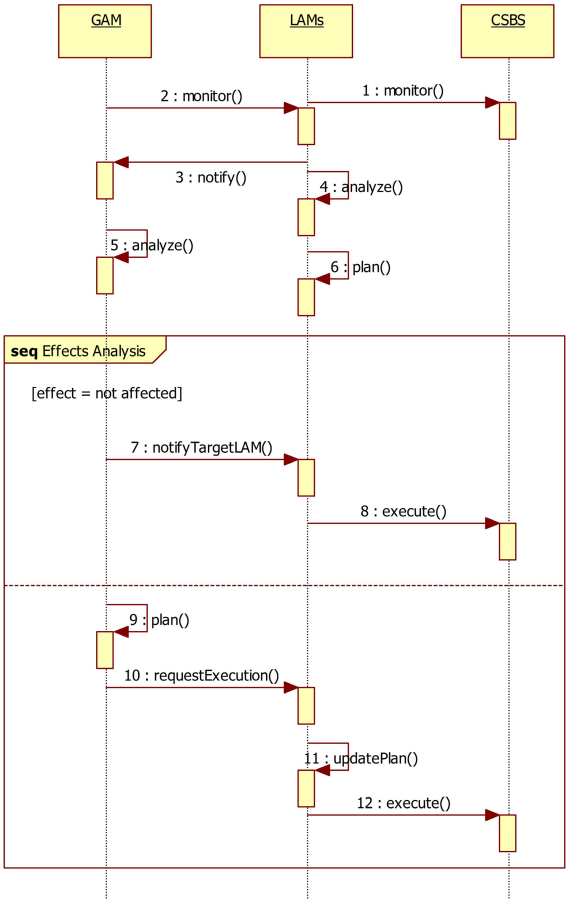

Fig. 2. Interaction model of the adaptation process

effect on the cross-layer system. The outcomes of the analysis may fall into one of these categories: (1) *Affected*, which is identified when more than one CSBS layer is affected; (2) *Not Affected*, which is identified when the other layers are not affected by the abnormal condition notified by the LAM.

Based on this outcome, the GAM performs either of the following: (1) let the LAM operate locally; (2) handle the situation. The first case is triggered if the GAM's analysis results in *Not Affected*. In this case, the GAM skips the planning activity and notifies the targeted LAM to perform the local adaptation. Upon receiving this notification, the respective LAM executes the established plan. For instance, if the LAM refers to the application layer, then a replanning process will be executed by invoking an existing planner, i.e., the MOSES planner [6,5].

The second case is triggered when the GAM's analysis results in *Affected*. The GAM performs a global planning which determines the adaptation actions

for the multiple layers. Then, the GAM's executor instructs all the LAMs to execute the plan. Herein, the GAM decision will supersede the decision taken by the respective LAM. Hence, the LAM will update the existing plan with a new plan and execute the latter.

We provide the design for the planning component in the following section, while the remaining framework components are beyond the scope of this paper and are left to future work.

3 Adaptation Planning

In this section we present the adaptation planning component for the CSBS, first introducing the selected methodology and then analyzing how we determine the single and multiple planning strategies.

3.1 Overview of the Methodology

The technique we used to model the planning is based on decision-theoretic planning and specifically on Markov Decision Processes (MDPs), that provide a suitable framework to model the decision making process under uncertainty and to take forward-looking decisions [3,17].

The basic MDP model is also known as *centralized MDP* and is suitable to model a single planning problem. MDP has been applied in various application domains, including multi-robot coordination and sensor network management. The common MDP model consists of states, actions, transition probabilities, and rewards. In addition, the model can be associated with finite and infinite horizon. The optimal solution can be obtained by using stochastic dynamic programming algorithms such as value iteration and policy iteration.

The centralized MDP is also determined as fully observable, which means the agent has a full knowledge about the underlying state environment. However, in certain situations an agent may only have a partial information about the underlying state environment. Hence, a generalization of MDP was proposed, also known as Partially Observable Markov Decision Process (POMDP) [11]. In POMDP, the agent needs to establish its belief to the state environment in order to determine the appropriate policy.

Several variants and generalizations of MDP and POMDP models have been proposed in literature, especially related to the multi-agent decision processes. Cooperative multi-agent systems are often modeled by Multi-agent MDP, Multi-agent POMDP, decentralized MDP, or decentralized POMDP [18]. There are some common elements to model the multi-agent systems [12], which include the set of agents, the set of global states, the set of joint actions, the set of joint observation, the joint transition function, the global reward, and the set of belief states.

In this study, we identify two types of planning approaches (i.e., single and multiple) to support the adaptation planning process. The single planning approach concerns only a single layer of the CSBS. Hence, we model this type of planning based on the centralized MDP. On the other hand, the multiple planning approach deals with multiple layers of the CSBS and therefore we model this type of planning based on the *multi-agent MDP* (MMDP) [3], which is a generalization of the centralized MDP.

3.2 Single Planning

We first present the problem formulation of the single planning using a single MDP and explain how to achieve the optimal policy. The single planning will be executed by a LAM. The single planning problem is modeled as a single MDP with a tuple (S, A, P, R, H), where:

- S refers to the set of violation states associated to the services in a specific layer. There are three possible stati for the state: *normal*, *expected violation*, and *violated*.
- A refers to the set of possible adaptation actions to be executed associated to a specific violation state. The action to be taken will change a state to the next state. Herein, we consider a significative subset of adaptation actions that can be executed at a specific service layer. For instance, the actions that can be taken at the infrastructure layer include adding a new virtual machine instance or migrating a virtual machine instance from one physical machine to another machine. The possible actions at the platform layer may include updating or redeploying a Web server. Meanwhile, the actions to be considered at the application layer include replanting the workflow or rebinding to different component services (i.e., through service selection) that provide the same functionality but with different non-functional parameters (e.g., response time, cost, availability, reputation).
- P is a transition function $P : S \times A \rightarrow \Delta(S)$. $P(s'|s, a)$ denotes the transition probability (uncertainty effect) of taking action a in state s which results in a transition to state s'. For instance, a state s, e.g., *violated*, may change to the next state s', e.g., *normal*, if action a is taken with 0.9 probability.
- R is the reward function $R : S \times A \rightarrow \Re$. $R(s, a)$ denotes the reward obtained when action a is taken from a state s which a state transition to s' occurs. The reward can be viewed as a utility value of a specific layer in fulfilling the layer objective. For instance, the objective of the application layer can be to minimize the response time, and thus the utility value represents the response time of taking an adaptation action. The objective of the infrastructure layer can regard the minimization of the energy consumption and thus the utility value represents the energy consumed in taking an adaptation action. For simplicity, we assume the reward value takes either 1, 0, or -1. The reward 1 is assigned when state s' holds the *normal* status. The 0 reward is assigned for the *expected violation* status, while reward -1 is assigned for the *violated* status.

– H is the finite horizon during which the policy can be computed. This period is essential to ensure the proposed policy can maintain the quality assurance of CSBS. For instance, the adaptation cycle can take up to a maximum of t time to maintain the overall execution time.

Based on these elements, the optimal policy for the adaptation planning can be formulated. The optimal policy refers to the best adaptation action to be executed for evolving the state of a service to the next state. The policy is based on Bellman equation, given as follows:

$$\pi^*(s) = \arg max_{a \in A}\{R(s,a) + \sum_{s'} P(s'|s,a)V^*(s')\} \quad (1)$$

In Eq. 1, $\pi^*(s)$ is the optimal policy (the best action to be taken) for state s. The best action is obtained based on the maximum reward of the possible action rewards $R(s,a)$, the transition probability $P(s'|s,a)$ and the value function of the next state from the previous adaptation cycle $V^*(s')$.

In contract to reward, the V value represents the long term objective to be achieved by the specific layer through a set of adaptation cycle. Technically, the V value can be formulated as follows:

$$V^k(s) = \{R(s,a) + \sum_{s'} P(s'|s,a)V^{k-1}(s')\} \quad (2)$$

In Eq. 2, k refers to one of the steps in the finite horizon H. The optimal solution can be achieved by using the standard algorithms, namely, the value iteration in a finite horizon. In the algorithm, the initial V value of all states can be set 0. Then, for each k, the V value will be computed iteratively until the value converges, namely, $V^k(s) - V^{k-1}(s') < \epsilon$. After that, the best V value is used to select the best action at step k.

3.3 Multiple Planning

The multiple planning is implemented whenever two or more layers are analyzed as affected. This planning is essential to avoid conflicting adaptation objectives at the different layers. For instance, the local planning at the infrastructure layer aims to minimize the energy consumption by migrating some virtual machines. This decision may affect the application layer which aims to minimize the response time by replanning the workflow. Thus, a joint decision is needed and can be achieved through multiple planning.

We model the multiple planning problem as multi-agent MDP [3] with a tuple (I, S, A, P, R, H) where I is a set of agents, S is a set of global states, A is a set of joint actions, P is the global transition function, R is the global reward function and H is the horizon.

We map the elements in multi-agent MDP to the adaptation planning problem as follows:

– I is a set of layers, namely the local adaptation managers (LAMs).

- S is a set of global states of the CSBS. The global states can be factored into local states observed by LAMs such as, $S = S_i \times S_j$ where $(i, j) \in I$. This means, each LAM will have a full observation on the states within a specific service layer.
- A is a set of joint adaptation actions, $A = A_i \times A_j$ where $(i, j) \in I$. Each A_i represents the possible local adaptation actions available to a specific service layer.
- P is a global transition function $P : S \times A \to \Delta(S)$. $P(s'|s, a)$ denotes the global transition probability of taking joint adaptation action a in global state s which results in a transition to the global state s'. The global transition probability can be decomposed into a set of independent local transition probabilities, given as follows:

$$P(s'|s, a) = P(s'_i|s_i, a_i) \times P(s'_j|s_j, a_j) \tag{3}$$

- R is the global reward function $R : S \times A \to \Re$. $R(s, a)$ denotes the global reward obtained for taking the joint adaptation action a in state s and transitioning to state s'. The global reward is obtained by the following:

$$R(s, a) = \sum_{i \in I} R_i(s_i, a_i) \tag{4}$$

- H is the finite horizon during which the policy can be computed.

Based on the given model, the global optimal policy, which consists of a set of joint policies, can be obtained on the basis of Eq. 1. In the latter, the reward function refers to Eq. 4 and the transition function refers to Eq. 3.

4 Example and Discussion

In this section, we provide an example of scenario to show the practicality of the proposed approach. The scenario refers to the citizen service center scenario adopted from [1].

The public service center can be abstractly presented as a multi-layer service as shown in Fig 3. At the highest level there is the application service layer, which refers to the application of citizen service center. The middle layer is the software service layer which consists of composite health and mobility services. This composite service comprises of the booking service, healthcare service, and mobility service. The lowest layer is the infrastructure service later which refers to the actual service providers for each service in the software service layer.

Each of these layers is associated to the LAMs. For instance, the application layer is managed by a specific LAM. Meanwhile, all LAMs are controlled by the GAM.

To show the adaptation needs, let us assume that the infrastructure service layer is detected as having a problematic behavior by the LAM's monitor. The problem can be due to many reasons, such as one of the call center providers has

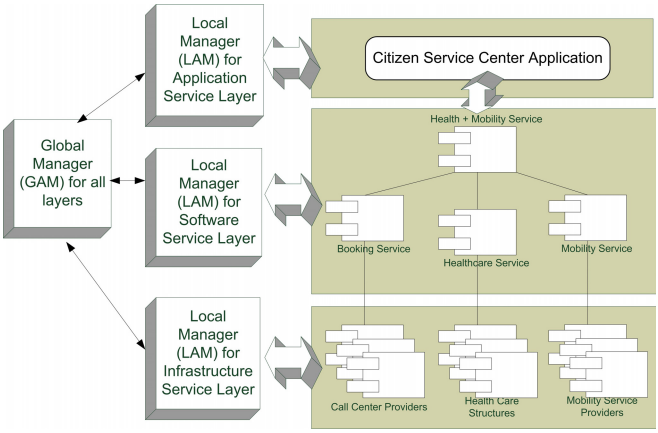

Fig. 3. Citizen service center considered as use case scenario

been detected as unavailable. Due to this problem, the LAM's analysis of the infrastructure layer will notify the GAM's monitor for a global analysis (refer to Fig. 2).

The GAM's analysis will assess the impact level of the problematic situation. Conceptually, there are two possible outcomes; affected or not affected. If the outcome is not affected, this means the other layers are not affected by the condition, that is in our example the booking service can behave as a normal service. In this case, the GAM will notify the LAM of the infrastructure layer to handle locally the adaptation problem.

If the analysis outcome is affected, it means the other layers may violate their QoS constraints due to the problematic condition, in this case the software service layer. Thus, the GAM performs a further analysis to identify the impact region (a set of affected services) which results in the booking service. Based on such outcome, the GAM performs a global planning to determine the appropriate adaptation action for each service in each layer, namely, the booking service and the respective call center provider.

In the context of local planning, the mapping between the scenario we consider in this example and the MDP can be viewed as follows:

- States: the state of the call center provider which is violated;
- Actions: renegotiating the SLA or replacing the call center provider;
- Probability: the effect value of taking the possible actions from the current state to the next state;
- Reward: the value assigned to the next state;
- Horizon: the number of iterations to address the problematic situation;
- Value: a value to quantify the goodness of the action.

By solving the MDP, a set of values will be obtained in relation to the possible actions. Thus, the best action that maximizes the objective (as specified in Eq. 1) will be selected and executed by the Executor component.

5 Related Work

Adaptivity is one of the most challenging research problems for service-based systems and the existing approaches can be mainly divided into three areas [4]: dynamic context-aware adaptation, user-centric adaptation, and multi-layer adaptation. In this paper, we address the multi-layer adaptation challenges.

The complexity and the uncertainty of a complex service-based system demand for a robust adaptation planning in order to decide the appropriate adaptation actions during the runtime operations. The approach by Pernici and Siadat [15] proposed a fuzzy-based solution to select the adaptation actions based on QoS satisfaction. It differs from our work since the uncertainty is associated to the service behavior rather than to the adaptation action effect. The works in [9,19,24] proposed and utilized Cross-layer Adaptation Manager (CLAM) to handle the complexity of service-based systems in the adaptation process. CLAM covers three aspects: first, the integrated platform to plug in existing adaptation and analysis tools; second, the cross-layer model of the service-based system as core input; third, the rule-based analysis to construct alternative cross-layer adaptation strategies. In contrast to our work, we address the uncertainty planning challenge in deciding the adaptation strategies/actions. This is essential since the actual effect of executing any adaptation action is unknown and cannot be guaranteed. In [16] Popescu et al. proposed a methodology for the dynamic and flexible adaptation of multi-layer applications that uses adaptation templates and taxonomies of adaptation mismatches. However, their approach focus on functional properties and cannot handle multiple mismatches that occur at the same time. Zeginis et al. proposed in [23] a framework that can deal with both reactive and proactive cross-layer adaptation of service-based systems. While they focus on a cross-layer monitoring mechanism, we investigate the adaptation phase.

The Model-based Self-Adaptation of SOA Systems (MOSES) framework [5] aims to dynamically adapt service-based systems according to non-functional QoS properties by acting at the SaaS layer. Our work aims to complement such framework with a planning mechanism that takes the complexity (i.e., the multi-layer model) and the uncertainty effect of adaptation actions into account.

6 Conclusions and Future Work

In this paper, we have proposed a conceptual architecture and interaction process for self-adapting a complex service-based system. The architecture consists of two adaptation managers to cater the global and local perspectives of the adaptation. Furthermore, we have proposed a planning model based on MDP techniques by considering the uncertainty and complexity factors of adapting a complex service-based system. Our study shows the viability of MDP techniques for realizing a multi-layer self-adaptation planning component.

In the future, we plan to analyze the performance of the proposed approach as well as its computational complexity with respect to the system scale. We also

plan to explore reinforcement learning for realizing a decentralized multi-layer adaptation planning where full knowledge of system dynamics is not required.

Acknowledgement. This work is supported by the Fundamental Research Grant Scheme (600-RMI/FRGS 5/3 (164/2013)) funded by the Ministry of Higher Education Malaysia (MOHE) and Universiti Teknologi MARA (UiTM), Malaysia.

V. Cardellini also acknowledges the support of the European ICT COST Action IC1304 Autonomous Control for a Reliable Internet of Services (ACROSS).

References

1. Armellin, G., Chiasera, A., Frankova, G., Pasquale, L., Torelli, F., Zacco, G.: The eGovernment use case scenario service level agreements for Cloud computing. In: Service Level Agreements for Cloud Computing, pp. 343–357. Springer (2011)
2. Baryannis, G., Garefalakis, P., Kritikos, K., Magoutis, K., Papaioannou, A., Plexousakis, D., Zeginis, C.: Lifecycle management of service-based applications on multi-clouds: a research roadmap. In: Proc. of 2013 Int'l Workshop on Multi-Cloud Applications and Federated Clouds, MultiCloud 2013, pp. 13–20. ACM (2013)
3. Boutilier, C., Dean, T.L., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research 11(1), 1–94 (1999)
4. Bucchiarone, A., Kazhamiakin, R., Marconi, A., Pistore, M.: Adaptivity in dynamic service-based systems. In: Proc. of 1st Workshop on European Software Services and Systems Research - Results and Challenges, pp. 36–37 (2012)
5. Cardellini, V., Casalicchio, E., Grassi, V., Iannucci, S., Lo Presti, F., Mirandola, F.: MOSES: A framework for QoS driven runtime adaptation of service-oriented systems. IEEE Transactions on Software Engineering 38(5), 1138–1159 (2012)
6. Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F.: Adaptive management of composite services under percentile-based service level agreements. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 381–395. Springer, Heidelberg (2010)
7. Comuzzi, M., Kotsokalis, C., Rathfelder, C., Theilmann, W., Winkler, U., Zacco, G.: A framework for multi-level SLA management. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 187–196. Springer, Heidelberg (2010)
8. Esfahani, N., Malek, S.: Uncertainty in self-adaptive software systems. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) Self-Adaptive Systems. LNCS, vol. 7475, pp. 214–238. Springer, Heidelberg (2013)
9. Guinea, S., Kecskemeti, G., Marconi, A., Wetzstein, B.: Multi-layered monitoring and adaptation. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 359–373. Springer, Heidelberg (2011)
10. Ismail, A., Yan, J., Shen, J.: Incremental service level agreements violation handling with time impact analysis. Journal of Systems and Software 86(6), 1530–1544 (2013)
11. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1-2), 99–134 (1998)

12. Kaufman, M., Roberts, S.: Coordination vs. information in multi-agent decision processes. In: Proc. of 5th Workshop on Multi-agent Sequential Decision Making in Uncertain Domains, MSDM 2010 (2010)
13. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Computer 36(1), 41–50 (2003)
14. Marconi, A., Bucchiarone, A., Bratanis, K., Brogi, A., Camara, J., Dranidis, D., Giese, H., Kazhamiakink, R., de Lemos, R., Marquezan, C.C., Metzger, A.: Research challenges on multi-layer and mixed-initiative monitoring and adaptation for service-based systems. In: 2012 Workshop on European Software Services and Systems Research - Results and Challenges (S-Cube), pp. 40–46. IEEE Computer Society (2012)
15. Pernici, B., Siadat, S.H.: A fuzzy service adaptation based on QoS satisfaction. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 48–61. Springer, Heidelberg (2011)
16. Popescu, R., A., Staikopoulos, P.L., Brogi, A., Clarke, S.: Taxonomy-driven adaptation of multi-layer applications using templates. In: Proc. of 4th IEEE Int'l Conf. on Self-Adaptive and Self-Organizing Systems, SASO 2010, pp. 213–222 (2010)
17. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic, Dynamic Programming. Wiley, New York (1994)
18. Pynadath, D.V., Tambe, M.: The communicative multiagent team decision problem: Analyzing teamwork theories and models. Journal of Artificial Intelligence Research 16, 389–423 (2002)
19. Siadat, S.H., Zengin, A., Marconi, A., Pernici, B.: A fuzzy approach for ranking adaptation strategies in CLAM. In: Proc. of 5th IEEE Int'l Conf. on Service-Oriented Computing and Applications, SOCA 2012 (2012)
20. Varshney, L.R., Oppenheim, D.V.: Coordinating global service delivery in the presence of uncertainty. In: Proc. of 12th Int'l Research Symposium on Service Excellence in Management (2011)
21. Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., Wuttke, J., Andersson, J., Giese, H., Göschka, K.M.: On patterns for decentralized control in self-adaptive systems. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) Self-Adaptive Systems. LNCS, vol. 7475, pp. 76–107. Springer, Heidelberg (2013)
22. Zeginis, C., Kritikos, K., Garefalakis, P., Konsolaki, K., Magoutis, K., Plexousakis, D.: Towards cross-layer monitoring of multi-cloud service-based applications. In: Lau, K.-K., Lamersdorf, W., Pimentel, E. (eds.) ESOCC 2013. LNCS, vol. 8135, pp. 188–195. Springer, Heidelberg (2013)
23. Zeginis, C., Konsolaki, K., Kritikos, K., Plexousakis, D.: Towards proactive cross-layer service adaptation. In: Wang, X.S., Cruz, I., Delis, A., Huang, G. (eds.) WISE 2012. LNCS, vol. 7651, pp. 704–711. Springer, Heidelberg (2012)
24. Zengin, A., Kazhamiakin, R., Pistore, M.: CLAM: cross-layer management of adaptation decisions for service-based applications. In: Proc. of 2011 IEEE Int'l Conf. on Web Services, ICWS 2011, pp. 698–699 (2011)