Probabilistic Analysis of QoS-Aware Service Composition with Explicit Environment Models

Yu Zhou, Tingting Han, Taolue Chen, Shiqi Zhou

November 6, 2019

Abstract

Service composition is one of the primary ways to provide value-added services on the Internet. Quality-of-Service (QoS) represents a crucial indicator for the underlying composition policy adoption, but it is highly influenced by various environmental factors. Existing composition strategies rarely take the influence of environment into consideration explicitly, which may lead to sub-optimal composition policies in a dynamic environment. In this paper, a model-based service composition approach is proposed. Given the user request, it is possible to first find a set of matching abstract web services (AWSs), and then pull relevant concrete web services (CWSs) based on the AWSs. The set of CWSs can be modelled as a Markov decision process (MDP). In addition, we model the environment as a fully probabilistic system, capturing changes of environment probabilistically. The environment model can be further composed with the MDP from the service models, obtaining a monolithic MDP. The policy of which corresponds the selection of concrete services. We demonstrate how probabilistic verification techniques can be used to find the optimal service selection strategy against their QoS and the environment change. A distinguished feature of our approach is that the QoS of services, as well as the dynamic of environment change, are made parametric, so that the formal analysis is adaptive to the environment which is of paramount importance for autonomous and self-adaptive systems. Examples and experiments confirm the feasibility of our approach.

KEYWORDS

Service Composition, Markov Decision Process, Parametric Model Checking, Quality-of-Service

1 Introduction

Web service has nowadays become one of the most popular forms of software function provision on the Internet. Despite an increasing number of individual services, more complex and systematic user tasks pose a high demand for the composition of underlying atomic services. Web service composition has been an active research topic during the last decade from both the theoretical and practical points of view [1, 2, 3, 4, 5]. During composition, Quality-of-Service (QoS) usually emerges as the primary concern, since it is often the case that multiple services provide similar or the same functionality, but with significantly different QoS measures. Supporting QoS-aware service composition is a challenging task due to the inherent uncertainty from the environment [6, 7, 8]. For instance, it is conceivable that services may exhibit fluctuating performance ensuing the vibrating network bandwidth. A majority of existing approaches, however, merely model the QoS with pre-defined values (which are acquired by, e.g., historical data or experiments), and reduce the composition problem to an optimisation problem, such as [9, 10]. These approaches, while being simple, disregard the uncertainties from the environment which may have a significant impact on QoS in real settings.

In this paper, we are mostly concerned with two forms of uncertainties: (1) Environment uncertainty. It is not difficult to understand that the environment would have an effect on the QoS, and in a dynamic setting, various "modes" in the environment may switch from one to another. Here, the environment could be weather, traffic conditions, tim-

ing, security issues, network performance and any other factors that might affect the QoS. A mode is a typical configuration of an environment. For instance, the mode can be “rainy and peak times” and switch to “cloudy and off-peak times" or the security level is “severe” and switch to “medium", etc. (2) Parameter uncertainty. QoS of individual web services, or the probabilities of the environment modes switch are numeric values. They are acquired via monitoring, test, experiments, etc, and are essentially of statistical nature. It is virtually impossible to obtain a precise value for them. For instance, the time to arrive at an airport at peak hours is one hour on average, but can be in the range of, for instance, 0.8 to 2 hours. To have a more accurate analysis of the time, it is suggested to use a parameter (variable) to represent some probability (e.g., the probability to turn from sunny to cloudy) or measure (e.g., the time it takes to get somewhere), and to explore how the property of interest depends on the concrete parameter values.

To address these issues, in this paper, we propose a *parametric Markov decision process* based approach to support the analysis of QoS-aware service composition. Apart from modelling web services and their composition, we also explicitly model the mode switch of environments using a probabilistic model. The probabilities can usually be extracted from the real-time weather forecast, live traffic data, on-going network performance reports, or other domain-specific historical data that might be related and informative. The parameters may appear as the transition probabilities in the environment model, or as the QoS measures in the service models. Consider the following scenario. A customer wants to buy a product online. The person should make a decision in (1) which shopping websites, e.g., Amazon or Ebay; (2) which shop, e.g., Amazon or third party shops; (3) delivery services at different speeds and rates, e.g., expedite but expensive same-day delivery or free 3-5 days delivery; and (4) payment methods, including credit card fees, possible discounts and cashbacks, etc. In this example, depending on the customer's priorities and needs, different decisions could been made. For instance, if the customer makes this purchase for a birthday party that evening, then only shops that provide same-day delivery would be chosen. If the product is not urgently needed, then the most economical way of delivery would be used.

Our system takes in the user requirements, finds the abstract web services (the type-matching services that are used in the workflow at a high level), searches the eligible and compatible concrete web services (the services that are really used) and builds models for both the web services and the environment. Depending on the user's objectives and constraints, the system recommends the best strategies or decisions in choosing concrete web services. The system can then invoke and compose the chosen web services and deliver the final result to the user.

During the process of composition, we observe that although services are standalone entities, at times, a client's requirements would greatly reduce the number of potentially usable web services. For instance, if a traveller would like to bring a pet along, this basically excludes the option of taking the airplane. Or if a traveller needs to use a lift to get to the platform of an underground train, but the closest station does not provide a lift facility, the underground option could be taken off the table. In our approach, by formalising the user's requirements, we can reduce the state space at an early stage instead of enumerating all the composition possibilities.

Contribution The main contributions of the paper are summarised as follows:

- A modelling language including both the web service model and the environment model; the web service model is extended with variables and pre-/post-conditions to facilitate modelling of composition. The approach, which follows the “separation of concerns" rationale, provides the maximum modularity and flexibility with regards to web services and environments. In other words, we separate the modelling of web services and environment so that either can easily be replaced by other matching services or environment.
- Automated analysis with fully-fledged tool support. Our analysis is parametric, and accounts for Pareto optimality, which is indispensable when factoring in a dynamic environment. More technically, our analysis benefits from advanced, automated probabilistic verification tools, and supports analysis of objectives akin to QoS-aware service composition (based on expected reachability rewards rather than discounted sum which is a least worst surrogate).

Structure of the paper. The rest of the paper is structured as follows. Section 2 briefly reviews related work. Section

3 gives background information and preliminaries. We detail our approach in Section 4, and conclude our paper in Section 5.

2 Related Work

In this section, we briefly overview some most relevant work with focus on QoS-aware service composition.

Canfora et al. [11] proposed a genetic algorithm based approach for QoS service composition. Particularly, genetic algorithms are leveraged to determine a set of concrete services to be bound to abstract services. The approach only supports single objective (fitness) function, not the parametric analysis. Zhou et al. [12] proposed a model based approach to assure the behaviour consistency of service composition during runtime evolution. Wang et al. [13] studied the incomplete information problem of selecting one service among a set of candidates, and employed dynamic pricing strategy to compose web services with the maximum utility and the lowest costs. Ren et al. [14] modelled the service composition problem as a Markov decision process to satisfy the requirements of both functional aspects and non-functional ones, and then a Q-learning based algorithm is applied to solve the model. Rodriguez-Mier et al. [15] proposed a hybrid approach for service composition which could produce a composition strategy that minimizes a single objective QoS function. These pieces of work mainly cast the service composition into a constraint solving problem and leverage optimisation techniques to provide optimal solutions. More remotely, genetic algorithms and swarm optimisation algorithms, have also been used in self-adapted systems [16, 17].

Apart from using different techniques, our work is grounded on a formal framework with rigorous languages for modelling and specification, as well as well-established tool support. One of the advantages of our approaches lies in the theoretical guarantee it provides, owing to the underlying probabilistic verification. Computationally, our approach essentially utilises specific numerical methods for solving Markov decision processes (e.g., value iteration) so could be more efficient than the general optimisation methods, especially when their objective functions are overly complicated.

Wang et al. [9, 10] proposed approaches based on reinforcement learning to enable adaptive and dynamic service composition. During composition, the multi-agent reinforcement learning technique is employed to select the concrete service. Jungmann et al. [18] studied the problem of functional discrepancy during service composition and presented an automated approach for adaptive service composition. Moustafa et al. [19] proposed a reinforcement learning based approach to support QoS-aware service composition with conflicting objectives. Similar to our work, they use Markov decision process to model the service composition. However, they neither explicitly model the environment factors, nor support the parameterisation. On a more technical level, these authors usually take discounted sum of rewards as the optimisation objective in MDPs, which is a tradition in the reinforcement learning literature, but which arguably is not suitable for service composition purposes. Bashari et al. [20] proposed an automated approach to reconfigure the service composition strategy from the changing environment, the reconfiguration policies were derived from software product line techniques. Zhang et al. [8] recognised the necessity of an explicit environment model during service selection and proposed a QoS-aware monitoring approach. However, this approach is restricted to QoS monitoring purposes instead of service composition.

There are several service composition patterns, for instance, atomic pattern, sequential patten, multiplication pattern, parallel pattern and combined pattern, etc [21]. Our work focuses on sequential patterns. Moreover, because of the similarity of web service composition and business process modelling [22], our methods can be applied to the analysis and optimisation of business processes, which would complement the existing approaches [23, 24, 25].

Apart from the previous work on the composition aspect, there is also another line of research on the QoS-aware service prediction, such as [26, 27, 28, 29, 30]. In [26], a temporal QoS-aware web service recommendation framework via non-negative tensor factorisation is proposed by considering third dynamic context information, i.e., the time information. In [27], a probabilistic latent model for QoS prediction has been proposed. The model mainly leverages a user observable variable and a web service observable variable, which constitutes a user-service matrix. [28] extended the traditional collaborative filtering and combined historical QoS value to forecast the

personalised QoS values. [29] proposed a learning based approach for QoS value prediction which integrate multi-dimensional context. Particularly, the approach adopts an unsupervised encoder-decoder framework to generate a hidden feature based on which the similarity between two context entities could be calculated more accurately. In [30], a model was proposed which can embed policies to calculate composite service performance, and based on which, the composite service's performance could be predicted. Different from the work, our approach addresses the QoS awareness in the context of service composition despite the fact that some common composition techniques might have been employed in the previous work mentioned above.

3 Preliminaries

In real-world settings, the web services are hosted in a dynamic and unstable Internet-based environment, which inevitably makes deterministic models inappropriate. Probabilistic model checking has been widely applied in quantitatively analysing systems which exhibit stochastic features. Properties could be specified based on the two probabilistic temporal logics, i.e., Probabilistic Computation Tree Logic (PCTL) [31] and Continuous Stochastic Logic (CSL) [32].

We leverage the PRISM model checker [33] to construct the probabilistic model. PRISM is a leading open-source model checker and has been successfully applied in many fields, such as communication protocols, distributed algorithms and some other systems of specific subjects like biology. PRISM supports a set of probabilistic models, such as Markov decision processes extended with costs and rewards. Therefore, a wide range of quantitative measures related to model behaviours could be analysed. In addition, PRISM has also integrated the support for a parametric analysis of these models.

Definition 3.1 (FPS) *A* fully probabilistic system $\mathcal{D}$ *is a tuple* $(S, s_0, \mathbf{P})$*, where* S *is a set of states with* $s_0 \in S$ *being the initial state; and* $\mathbf{P} : S \times S \to [0,1]$ *is the transition probability function such that for all states* $s \in S$*,* $\sum_{t \in S} \mathbf{P}(s,t) \in [0,1]$.

Definition 3.2 (MDP) *A* Markov Decision Process $\mathcal{M}$ *is a tuple* $(S, s_0, A, \mathbf{P}, R)$*, where*

- S *is a set of states with* $s_0 \in S$ *being the initial state;*
- A *is a set of actions;*
- $\mathbf{P} : S \times A \times S \to [0,1]$ *is the transition probability function such that for all states* $s \in S$ *and actions* $\alpha \in A$*,* $\sum_{t \in S} \mathbf{P}(s,\alpha,t) \in \{0,1\}$*;*
- $\boldsymbol{\mathcal{R}} : S \to \mathbb{R}_{\geq 0}$.

For each state $s \in S$ *and action* α*, if* $\sum_{t \in S} \mathbf{P}(s,\alpha,t) = 1$*, then we say the action* α *is* enabled *in* s.

Policies play a crucial role in the analysis of MDPs. For our purposes, it suffices to consider *simple* policies, in which for each state s, the policy fixes one of the enabled actions at s and selects the same action every time when the system resides in s. Thus, the choices made by a simple policy are independent of history. Formally, a simple policy is a function $\sigma : S \to Act$ such that $\sigma(s)$ is one of the actions enabled at state s. A *path* in MDP under σ is an infinite sequence of states $\rho = s_0 s_1 \cdots$ such that, for all $i \geq 0$, $\mathbf{P}(s_i, \alpha, s_{i+1}) > 0$ for $\alpha = \sigma(s_i)$. Let $Path_{\mathcal{M},\sigma}$ be the set of paths in $\mathcal{M}$ under σ. Let $Path_{\mathcal{M},\sigma}(s)$ be the subset of paths that start from s, and $Pr_{\mathcal{M},\sigma}$ be the standard *probability distribution* over $Path_{\mathcal{M},\sigma}$ as defined in the literature [34, Ch. 10].

The *expected cumulative reward*, or simply *cumulative reward*, of reaching a set $G \subseteq S$ of *goal* states (called G-states hereafter) in MDP under σ, denoted $\boldsymbol{\mathcal{R}}_{\mathcal{M},\sigma}(G)$, is defined as follows: First, let $\boldsymbol{\mathcal{R}}_{\mathcal{M},\sigma}(s,G)$ be the expected value of random variable $X : Path_{\mathcal{M},\sigma}(s) \to \mathbb{R}_{\geq 0}$ such that (i) if $s \in G$ then $X(\rho) = 0$, (ii) if $\rho[i] \notin G$ for all $i \geq 0$ then $X(\rho) = \infty$, and (iii) otherwise $X(\rho) = \sum_{i=0}^{n-1} R(s_i)$ where $s_n \in G$ and $s_j \notin G$ for all $j < n$. Then, let $\boldsymbol{\mathcal{R}}_{\mathcal{M},\sigma}(G) = \boldsymbol{\mathcal{R}}_{\mathcal{M},\sigma}(s_0, G)$.

4 Our approaches

4.1 Overview

Figure 1 gives an overview of our approach. A user request specifies the specification of what the user needs (e.g., input, output, type of service, etc). When a user sends a request r, the web service configurator matches the functional operations with the ontology of a pool of *abstract web services* (AWSs). The depth-first-search (DFS) algorithm could be applied to the interface of AWSs to match

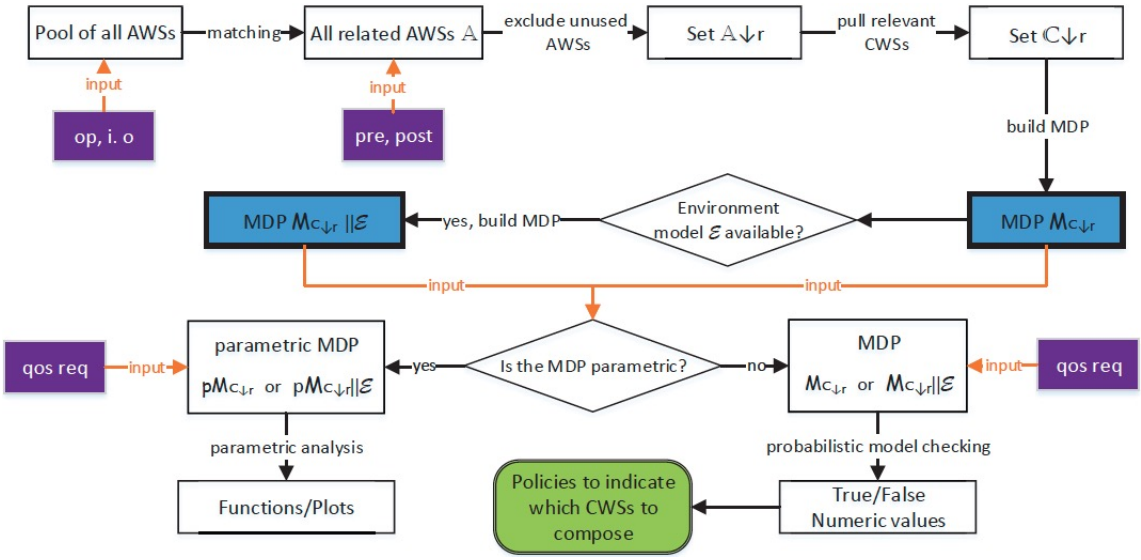

Figure 1: Overview of our approach

the user request. The result is a set $\mathbb{A}$ of AWSs. With additional constraints specified in the user request, AWSs which do not satisfy the user's needs are excluded from $\mathbb{A}$. The result becomes $\mathbb{A}\downarrow r$.

The next step is to pull the *concrete web services* (CWSs) corresponding to $\mathbb{A}\downarrow r$, based on which a Markov Decision Process (MDP) $\mathcal{M}_{\mathbb{A}\downarrow r}$ will be built. In case that the environment model $\mathcal{E}$ is available, it will be integrated resulting the MDP model $\mathcal{M}_{\mathbb{A}\downarrow r}||\mathcal{E}$ which respects the changes in the environment. The formal definitions of AWS, CWS and environment model are given in Section 4.2.

Recall that there might be environment or parameter uncertainty in either the service models or environment models, and the uncertainties are modelled by parameters. In these cases, we will have a parametric MDP (i.e., either $p\mathcal{M}_{\mathbb{A}\downarrow r}$ or $p\mathcal{M}_{\mathbb{A}\downarrow r}||\mathcal{E}$). Parameter analysis can then be applied to the parametric MDPs against user QoS requirements. The results may be closed-form functions or plots, which describe the region of the feasible solutions. One can instantiate the parameters as per any point of the region obtaining a (non-parametric) MDP, whereby obtain an acceptable selection/combination of web services.

In case that the MDP is not parametric, probabilistic model checking can be applied on the user QoS requirement. Either boolean or numeric values may be returned together with a policy that indicates which concrete web services to select and compose.

4.2 Models and specifications of web service selection

4.2.1 Web service models

There are different ways by which individual services can be integrated to build a process, e.g., sequential, parallel, conditional, iterative, etc. In this paper, we focus exclusively on the sequential composition of web services, which is the most fundamental and widely used pattern in service composition. Other ways of composition are convertible to the sequential composition [35] under a configurator, which acts as a policy, resolving all the uncertainties in the composition models. Note that the configurator in this paper is a general manager that does web service selection, composition and adaptation. We refer the readers to [35] for more details.

When a user specifies the requirements, the web service configurator first decides which *abstract web services* (AWS) are needed to accomplish the tasks. AWSs are used in the workflow at a high level. An AWS has input/output

and positive/negative pre-/post-conditions. A similar concept of AWS can be found in e.g., [36].

Definition 4.1 (Abstract web service, AWS) *An* abstract web service *is defined as a tuple* $\mathcal{A} = (OP, I, O, p^+, p^-, e^+, e^-)$*, where each component comprises a set of (atomic) propositions.*

Here the atomic propositions represent the status and/or properties of a state or condition. Intuitively, OP is the semantic description of the service, in the form of a membership statement of a class in an ontology [36]. Input I and output O represent signature information (basic input and output data types) of a service. They ensure syntactically correct solutions and a successful execution. On the other hand, required pre-conditions p^+, prohibited pre-conditions p^-, positive post-conditions e^+, and negative post-conditions e^- represent semantic information which reduces the set of syntactically correct solutions to only those solutions that are really useful. We reserve two propositions *init* and *end* for p^+ and e^-, respectively to indicate the start and end of the sequential composition of the AWSs as shown in Def. 4.4.

Example 4.2 *Suppose Adam would like to go from his home in city A to another city B. He could choose to go by train or plane, and whether to take taxi or tube to either the train station or the airport. Suppose there are one train station, two airports that have connection to B. The abstract web services in this case could be TAXI_TS, TAXI_A1, TAXI_A2, TUBE_TS, TUBE_A1, TRAIN (train from A to B) and PLANE_A1, PLANE_A2. More specifically, TAXI_TS and TAXI_A1 mean the taxi service to the train station, and Airport_1, respectively, and PLANE_A1 means the airlines from Airport_1 to city B. Given the input (starting location) and the output (destination), we show some of the AWSs in Fig. 2.*

Recall that the keyworsd init *and* end *mean that the corresponding AWSs are the initial and last service along the composition. In PLANE_A1, pets are not allowed on the planes and are restricted by p^-. In TUBE_A1, the disable access is not available at the local station, which is excluded by p^-.*

To sequentially compose two AWSs $\mathcal{A}_1$ and $\mathcal{A}_2$, one needs to ensure that $\mathcal{A}_1$ and $\mathcal{A}_2$ are *compatible*, which entails the following three conditions: (a) the input of $\mathcal{A}_2$ is a subset of the output of $\mathcal{A}_1$; this is to guarantee that all the inputs of the second AWS would be available as a result of the outputs from the first AWS. (b) the required pre-condition of $\mathcal{A}_2$ should be a subset of the positive post-condition of $\mathcal{A}_1$; (c) the intersection of the prohibited pre-condition of $\mathcal{A}_2$ and the positive post-condition of $\mathcal{A}_1$ should be empty. Formally,

TAXI_TS = {
OP = {taxi_to_train_station},
I = {*home*},
O = {*TrainStation*},
p^+ = {*init*},
p^- = {},
e^+ = {},
e^- = {}
}

TUBE_A1 = {
OP = {tube_to_airport_1},
I = {*home*},
O = {*Airport*1},
p^+ = {*init*},
p^- = {*disable access*},
e^+ = {},
e^- = {}
}

TRAIN = {
OP = {train_to_city_B},
I = {*TrainStation*},
O = {*B*},
p^+ = {},
p^- = {},
e^+ = {*end*},
e^- = {}
}

PLANE_A1 = {
OP = {plane_to_city_B},
I = {*Airport*1},
O = {*B*},
p^+ = {},
p^- = {*pets*},
e^+ = {*end*},
e^- = {}
}

Figure 2: Example AWSs

Definition 4.3 *Two AWSs* $\mathcal{A}_1 = (OP_1, I_1, O_1, p_1^+, p_1^-, e_1^+, e_1^-)$ *and* $\mathcal{A}_2 = (OP_2, I_2, O_2, p_2^+, p_2^-, e_2^+, e_2^-)$ *are* compatible*, written* $\mathcal{A}_1 \bowtie \mathcal{A}_2$*, if*

$$(a)\ I_2 \subseteq O_1; \quad (b)\ p_2^+ \subseteq e_1^+; \quad (c)\ p_2^- \cap e_1^+ = \emptyset.$$

As expected, when two AWSs are compatible, one can define their sequential composition, yielding another AWS.

Definition 4.4 (Service composition) *Given two AWSs* $\mathcal{A}_1 = (OP_1, I_1, O_1, p_1^+, p_1^-, e_1^+, e_1^-)$ *and* $\mathcal{A}_2 = (OP_2, I_2, O_2, p_2^+, p_2^-, e_2^+, e_2^-)$ *such that* $\mathcal{A}_1 \bowtie \mathcal{A}_2$*. The sequential composition* $\mathcal{A}_1 \otimes \mathcal{A}_2$ *of* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *is defined as* $\mathcal{A}_1 \otimes \mathcal{A}_2 = (OP, I, O, p^+, p^-, e^+, e^-)$ *where* $OP = OP_1 \cup OP_2$*,* $I = I_1$*,* $O = O_2$*,* $p^+ = p_1^+ \cup p_2^+$*,* $p^- = p_1^- \cup p_2^-$*,* $e^+ = e_1^+ \cup e_2^+$*,* $e^- = e_1^- \cup e_2^-$*.*

An AWS can be seen as a *service type*. Any *concrete* web service of a particular type would have the same signature (operation, input, output, pre- and post-condition). The difference between the concrete web services of the

same type lies in the quality of services. To this end, a concrete WS is defined as follows.

Definition 4.5 ((Concrete) web service, CWS) *A* (concrete) web service *is defined as a tuple* $C = (ID, AWS, QoS)$*, where ID is the identifier of the web service; AWS is the abstract web service (service type); QoS is the quality of the service represented by an* $(n+1)$*-tuple* $\langle Q_0, Q_1, \cdots, Q_n \rangle$*, where each* $Q_i, i \in \{0, ..., n\}$ *denotes a QoS attribute of the CWS.*

We reserve Q_0 as a primary QoS, representing the availability of the service, usually given as the probability of the service being successfully invoked.

Example 4.6 C_1 = (MiniCab, TAXI_TS, $\langle Q_0, Q_t, Q_s, Q_c \rangle$) *is a taxi service called* MiniCab. $Q_0 = 0.95$ *is the probability that a minicab is available and arrives on time;* $Q_t = 0.9h$ *shows how fast the service is;* $Q_s = 0$ stops *shows that it is a direct journey and no transfer is required;* $Q_c = £50$ *shows how much it charges.*

C_2 = (Uper[1], TAXI_TS, $\langle Q_0 = 0.9, Q_t = 0.9h, Q_s = 0, Q_c = £40 \rangle$) *is another taxi service called Uper, which is cheaper than MiniCab, but with less availability.*

C_3 = (Line1&2, TUBE_A1, $\langle Q_0 = 0.99, Q_t = 1h, Q_s = 1, Q_c = £5 \rangle$) *is a tube service. Note that the probability of a tube running on time (0.99) is much higher than that of a taxi. But this service requires one transfer during the journey.*

C_4 = (Express, TRAIN, $\langle Q_0 = 0.98, Q_t = 2h, Q_s = 0, Q_c = £100 \rangle$) *is an express train and it provides fast and direct connections, but is rather expensive.*

C_5 = (Local, TRAIN, $\langle Q_0 = 0.90, Q_t = 5h, Q_s = 2, Q_c = £50 \rangle$) *is connected by several local trains, which is relatively slow and needs 2 transfers, but is economical.*

After deploying the abstract web service composition at a higher level, the web service configurator then picks a CWS for each AWS so that the concrete chain of services satisfies its own need subject to certain constraints. Inspired by [18] which used transition systems to model the composition, we construct an MDP to model the sequential composition of CWSs. Given a set of candidate CWSs $\mathbb{C}$, we can generate an MDP as follows.

[1] A fictitious company

Definition 4.7 (MDP for composition) *Given a set* $\mathbb{C}$ *of candidate CWSs, the MDP is*

$$\mathcal{M}_{\mathbb{C}} = (S, s_0, A, \mathbf{P}, \boldsymbol{\mathcal{R}}, AP, L),$$

where

- $S \subseteq \mathbb{C} \cup \{s_0, End\}$*, with* s_0 *the initial state and End a special state indicating the end of a process;*
- AP *is the set of atomic propositions;*
- $L : S \to 2^{AP}$ *is a labelling function that assigns subset of* AP *to each state;*
- *The set of actions is* $A = \{\alpha \mid \alpha = C.ID, \forall C \in \mathbb{C}\} \cup \{fin\}$*;*
- *The transition probabilities are defined as*
  - $\mathbf{P}(s_0, \alpha, C_j) = q$ *and* $\mathbf{P}(s_0, \alpha, C_j) = 1 - q$*, if*
    - ⋆ $C_j \in \mathbb{C}$*,*
    - ⋆ $init \in C_j.p^+$ *(C_j is one of the starting web services),*
    - ⋆ $\alpha = C_j.ID$ *(action is the ID of the starting web service), and*
    - ⋆ $q = C_j.Q_0$ *(probability of successfully calling service* C_j*);*
  - $\mathbf{P}(C_i, \alpha, C_j) = q$ *and* $\mathbf{P}(C_i, \alpha, C_j) = 1 - q$*, if*
    - ⋆ $C_i, C_j \in \mathbb{C}$*,* $\alpha = C_j.ID$ *and* $q = C_j.Q_0$*,*
    - ⋆ $C_i \bowtie C_j$ *(C_i and C_j are compatible);*
  - $\mathbf{P}(C_j, fin, End) = 1$*, if* $C_j \in \mathbb{C}$*,* $end \in C_j.e^+$*.*
- $\boldsymbol{\mathcal{R}} : S \times A \times S \to \mathbb{R}^m$ *is the reward function that describes a vector of m (transition) rewards, and for* $C, C' \in \mathbb{C}$ *and* $\alpha \in A$*,*

$$\boldsymbol{\mathcal{R}}_i(C, \alpha, C') = C'.Q_i, \text{ if } C \neq C' \text{ and } \boldsymbol{\mathcal{R}}_i(C, \alpha, C) = 0$$

The state space S consists of the set of CWSs and two extra states - the initial state and the end state. The set of atomic propositions AP is the set of all atomic propositions from the AWSs. The set of actions A is the ID of each CWS in $\mathbb{C}$ with an additional one *fin*.

To lay out the transition probabilities, we distinguish three cases. (1) Starting from the initial state s_0. A transition $init \xrightarrow{\alpha} C_j$ can take place if the target state C_j (which is a CWS) has been labelled with *init* (representing the start of the sequence of web services). The action would be the

ID of the CWS C_j, and the probability of the transition is the initial probability of C_j. (2) For all the intermediate transitions $C_i \xrightarrow{\alpha} C_j$, we require that the source and target states (CWSs) are compatible. The actions are the IDs of the target states and the transition probability is the probability successfully calling the target CWS C_j. This probability is $C_j.Q_0$. (3) For the final transition $C_j \xrightarrow{fin} End$, we introduce another action *fin* and the extra state *End* and the probability is 1.

4.2.2 User requirement

A user requirement defines what a user wants, which guides the configurator to select appropriate abstract or concrete web services. With the user requirement, it is possible to greatly narrow down the range of feasible web services and web service composition, facilitating the selection of web services.

We use the same formalism of (abstract) web services for specifying a user requirement so they can be cast into the same framework in a uniform manner. In general, user requirements consist of two parts, the functional requirement which usually specifies the pre- and post-conditions, but from the user's perspective; and the non-functional requirement which usually focuses on the QoS set by the user. The non-functional requirement is defined as follows.

Definition 4.8 (QoS requirement) *Given a QoS measure Q, a QoS requirement is*
(i) hard*, if it is of the form $Q \sim v$ where $\sim \in \{<, >, \leq, \geq\}$ and v is a real number; or*
(ii) soft*, if it is of the form* $\max Q$ *or* $\min Q$*, which is to maximise or minimise Q.*

Intuitively, the hard requirements are of the boolean type and should be met, and the soft requirements are of the optimisaton type and the target to be achieved.

Definition 4.9 (User requirement) *A user requirement r is defined as a tuple $(op, i, o, pre^+, pre^-, post^+, post^-, req)$, with op denotes the type of operation the user asks, i and o denoting input and output, respectively; pre^+, pre^- and $post^+$, $post^-$ are positive and negative pre- and post-conditions; req is a set of QoS requirements.*

Example 4.10 *A request from a disabled passenger who would like to bring a pet from home to city B would be*

$$r = \big(travel, home, cityB, pre^+ = \{disabled, pets\}, \\ \emptyset, \emptyset, \emptyset, \{\min Q_s, Q_c \leq £110\}\big).$$

Here, $\min Q_s$ *is to minimise the number of transfers and $Q_c \leq £110$ means the total sum of fare should be no greater than £110.*

4.2.3 Web service selection at the abstract level

Given the user requirement, the main task is to select which web services to compose to fulfill the requirement. The selection is carried out at two levels.

At the abstract level, the pre- and post-conditions in the request are crucial to reduce the state space of the MDP for composition. For instance, the positive pre-condition of disabled would rule out the option of taking a tube from a station that does not have disable access; meanwhile, taking a pet would exclude travelling by plane. The above two restrictions leave the option of taking a taxi connected by a train.

At the concrete level, the composition may only choose the available *concrete* web services from the remaining AWSs and build an MDP accordingly. The QoS requirement can restrict certain policies and CWSs in the MDP, as such policies may induce a model that dissatisfies the QoS requirement.

In the following, we will show how to exclude some AWS from the given set of AWSs and a user request at the abstract level.

Definition 4.11 (AWS exclusion) *Given a set of AWSs $\mathbb{A}$, the corresponding CWSs $\mathbb{C}$ and a user request $r = (op, i, o, pre^+, pre^-, post^+, post^-, req)$, we define the set of AWSs exluded by r as $\mathbb{A}\backslash r = \{\mathbb{A} \setminus \mathcal{A}\}$, for each $\mathcal{A} = (OP, I, O, p^+, p^-, e^+, e^-) \in \mathbb{A}$ if one of the following is true*

$$\begin{array}{lll} (1)\ pre^+ \cap p^- \neq \emptyset, & & (2)\ post^+ \cap e^- \neq \emptyset, \\ (3)\ pre^- \cap p^+ \neq \emptyset, & or & (4)\ post^- \cap e^+ \neq \emptyset. \end{array}$$

$\mathbb{C}\backslash r$ is the set of concrete web services that belong to $\mathbb{A}\backslash r$.

Example 4.12 (Continued from Example 4.10) *The set of abstract web services $\mathbb{A}$ =*

$\{TAXI\_TS, TAXI\_A1, TAXI\_A2, TRAIN, TUBE\_TS, TUBE\_A1, PLANE\_A1, PLANE\_A2\}$ *is reduced to be* $\mathbb{A}\downarrow r = \{TRAIN, TAXI\_TS\}$ *by the user request* r. *The corresponding set of concrete web services is* $\mathbb{C}\downarrow r = \{Express, Local, MiniCab, Uper\}$.

Given a set of AWSs $\mathbb{A}$, the corresponding CWSs $\mathbb{C}$ and a user requirement r, the resulting MDP is $\mathcal{M}_{\mathbb{C}\downarrow r}$. Intuitively, the MDP produces all the functionally feasible solutions that conform to the user requirement. Each path that starts from s_0 and ends at *End* represents a way to compose web services to achieve the functional requirement of the user.

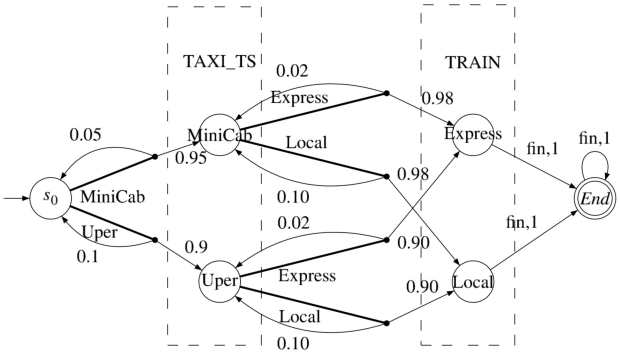

Figure 3: Example of MDP from composing CWSs

Example 4.13 (Continued from Example 4.12) *The MDP for composing set of CWSs* $\mathbb{C}\downarrow r$ *is shown as in Fig. 3, where the probabilities were taken from all the availability probabilities* Q_0*s in Example 4.6.*

The exclusion at the AWS level could have led to a (significant) state space reduction. The MDP in Fig. 3 has 6 states and this is after the exclusion due to positive and negative pre- and post-conditions. The MDP before the exclusion has 18 states, assuming there are 2 CWSs per AWS. In PRISM, due to some auxiliary variables that are used to synchronise, there would be more states than 6 or 18.

Table 1 summarises the number of states and transitions before and after the exclusion due to pre- and post-conditions for the travel example. Note that #A is the number of AWSs and #C is the number of CWSs per AWS. To test the magnitude of state space reduction, we scaled the MDP model by enlarging the number of AWSs and CWSs. From the table, the number of states/transitions after the exclusion using pre-/post-conditions is approximately reduced by 80% of that before the exclusion.

4.2.4 Web service selection at the concrete level

At the concrete level, the goal is to decide which CWS to select from each type of AWS to satisfy the QoS requirements. In the MDP, it is to find a policy under which the rewards along the set of paths that go from s_0 to the *End* state fulfiling the QoS requirements.

Remark that there are two types of QoS requirements, the hard ones (e.g., $Q_c \leq$ £110, $Q_t \leq 5h$) and the soft ones (e.g., min Q_s minimise the number of transfers). As a result, we may have eight different patterns of the QoS requirements in the first column of Table 2. We write 1H1S a short form of "one hard and one soft requirement" and mHmS is short for "multiple hard and multiple soft requirements", etc.

Six out of the eight QoS patterns can be calculated in PRISM. The only limitation is that there cannot be multiple software requirement *and* one or more hard requirements. As a very useful by-product, PRISM provides a policy on how the result is or could be obtained. In other words, the resulting policy points out which concrete web services are being selected.

Example 4.14 • [1H] *R{"Qc"}<=110 [F End] is false. The formula asks whether under ALL policies, one can reach city B with no more than £110. The result is false, and a counterexample policy is given - taking MiniCab (£50) + Express train (£100) will add up to £150, which exceeds £110.*

- [1S] *R{"Qs"}min=? [F End] is 1. The minimum number of transfers is achieved when taking Uper followed by the Express train. The result 1 comes from changing means of transportations.*
- [mH] *Multi(R{"Qc"}<=200[C], R{"Qs"}<=1[C]) asks whether there exists a policy under which both boolean requirements are true. An evidence policy is provided - taking MiniCab then Express trains.*
- [mS] *Multi(R{"Qc"}min=?[C], R{"Qs"}min=?[C]) is a pareto curve. The result is a region with two extreme*

#A	#C	before		after		#A	#C	before		after	
		#states	#trans	#states	#trans			#states	#trans	#states	#trans
8	2	62	91	14	19	11	2	142	191	30	39
	4	202	301	42	61		4	842	1181	170	237
	6	422	631	86	127		6	2582	3691	518	739
	8	722	1081	146	217		8	5842	8441	1170	1689
	10	1102	1651	222	331		10	11102	16151	2222	3231
	20	4202	6301	842	1261		20	8.4×10^4	1.2×10^5	1.7×10^4	2.5×10^4
	30	9302	13951	1862	2791		30	2.8×10^5	4.1×10^5	5.6×10^4	8.3×10^4

#A	#C	before		after		#A	#C	before		after	
		#states	#trans	#states	#trans			#states	#trans	#states	#trans
14	2	302	391	62	79	17	2	622	791	126	159
	4	3402	4701	682	941		4	13642	18781	2730	3757
	6	15542	22051	3110	4411		6	93302	132211	18662	26443
	8	4.7×10^4	6.7×10^4	9.4×10^3	1.3×10^4		8	3.7×10^5	5.4×10^5	7.5×10^4	1.1×10^5
	10	1.1×10^5	1.6×10^5	2.2×10^4	3.2×10^4		10	1.1×10^6	1.6×10^6	2.2×10^5	3.2×10^5
	20	1.7×10^6	2.5×10^6	3.4×10^5	5.0×10^5		20	3.4×10^7	5.0×10^7	6.7×10^6	1.0×10^7
	30	8.4×10^6	1.2×10^7	1.7×10^6	2.5×10^6		30	2.5×10^8	3.7×10^8	5.0×10^7	7.5×10^7

Table 1: Comparison of #states and #transitions before/after AWS exclusion

Pattern	Example	Result	Policy
1H	R{"Qc"}<=110[F End]	false	MiniCab→Express
1S	R{"Qs"}min=?[F End]	1	Uper→Express
mH	Multi(R{"Qc"}<=200[C], R{"Qs"}<=1[C])	true	MiniCab→Express
mS	Multi(R{"Qc"}min=?[C], R{"Qs"}min=?[C])	[(94, 3), (140, 1)]	Uper→Local MiniCab→Express
1H1S	Multi(R{"Qs"}min=?[C], R{"Qc"}<=95[C])	3	Uper→Local
mH1S	Multi(R{"Qs"}min=?[C], R{"Qc"}<=110[C], R{"Qt"}<= [C])	2.33	MiniCab→Express
1HmS	Multi(R{"Qs"}min=?[C], R{"Qc"}min=?[C], R{"Qt"}<=5[C])	Error	–
mHmS	Multi(R{"Qs"}min=?[C], R{"Qc"}min=?[C], R{"Qt"}<=5[C], R{"Qc"}<=130[C])	Error	–

Table 2: The QoS pattern and the PRISM properties

points, one with minimum cost and the other with minimum number of transfers. For each extreme point, a policy is given.

- [1H1S,mH1S] *Multi(R{"Qs"}min=?[C], R{"Qc"}<=95[C]) and Multi(R{"Qs"}min=?[C], R{"Qc"} <=110[C], R{"Qt"}<=5[C]) calculate the minimum expected cumulative value of reward structure Qs, given one or more constraints on other expected cumulative values. The policy returns a feasible solution on how to achieve this.*

4.3 Adaptive Web Service Composition

As stated previously, QoS depends heavily on the external environmental factors [7, 37, 38], and thus a web service would not always withhold the same QoS performance in different environment. For instance, the availability of taxi services would drastically decrease if it is at rush hours or the weather is bad. At some extreme weathers, flights might be cancelled (so the availability drops to very low $Q_0 = 0.1$). Meanwhile, the tube services are more robust to weather influences.

To match the different mode of the environment, each concrete web service will have to be equipped with different sets of QoS measures.

Example 4.15 (Continuing Example 4.6) *The set of QoS of the Uper service C_2 was $QoS_1 = \langle Q_0 = 0.9, Q_t = 0.9h, Q_s = 0, Q_c = £40\rangle$ and let us assume it is the QoS when the taxi service is running normally at off-peak*

time. In the case of peak time, the QoS might be $QoS_2 = \langle Q_0 = 0.7, Q_t = 1.7h, Q_s = 0, Q_c = £51\rangle$ *and in the case of night time when there is little traffic, the QoS might be* $QoS_3 = \langle Q_0 = 0.95, Q_t = 0.68h, Q_s = 0, Q_c = £26\rangle$.

Modelling the environment change is challenging. In this section, we will abstract the environmental change as a fully probabilistic system (FPS) and describe the interaction and influence between the environment and the web services as a composition between the FPS and the MDP.

4.3.1 Environment model

As discussed before, we abstract the environment's possible behaviours as a fully probabilistic model (as defined in Section 3), which is to capture the environment uncertainty.

Definition 4.16 (Environment) *An* environment *is* $\mathcal{E} = (S^{\mathcal{E}}, s_0^{\mathcal{E}}, \mathbf{P}^{\mathcal{E}})$ *where*

- $S^{\mathcal{E}}$ *is a finite set of environment states (or modes), with* $s_0^{\mathcal{E}} \in S^{\mathcal{E}}$ *the initial state;*
- $\mathbf{P}^{\mathcal{E}} : S \times S \to [0, 1]$ *is the transition probability function of the system.*

Example 4.17 *We model the environment for the travel example as in Fig. 4. The probabilities are chosen such that the long-run distribution is approximately* $(\frac{5}{24}, \frac{11}{24}, \frac{8}{24})$ *for* peak, normal *and* night *mode, respectively. The long-run distribution is based on the fact that peak hours are 6am-9am, 5pm-7pm (5 hours), normal hours are 9am-5pm and 7pm-10pm (11 hours), and quiet night time is 10pm-6am (8 hours). Note that there is no transition from peak hours to quiet time directly. The traffic will first reduce to normal and then to the night time.*

4.3.2 Web services in environment

Given a versatile environment, the web services adapt themselves to fit in. The composition of the underlying models is of two-folds. At any state $(s^{\mathcal{M}}, s^{\mathcal{E}})$ it is possible to have an environmental change or carry out a web service. Therefore, we stratify the model states into $(s^{\mathcal{M}}, \underline{s^{\mathcal{E}}})$ and $(\underline{s^{M}}, s^{\mathcal{E}})$, where only environmental changes can take place in the former states and only web service execution

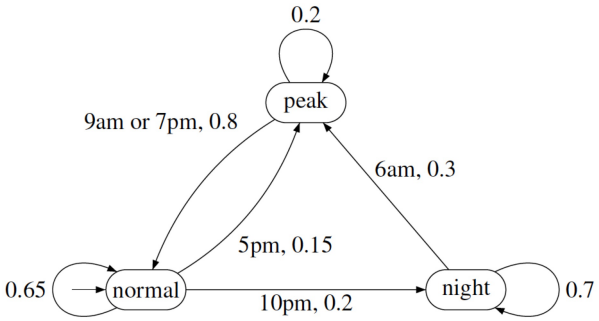

Figure 4: Environment model

could take place in the later states. The resulting model is an MDP and is defined as follows:

Definition 4.18 *Given a CWS MDP model* $\mathcal{M} = (S^{\mathcal{M}}, s_0^{\mathcal{M}}, A^{\mathcal{M}}, \mathbf{P}^{\mathcal{M}}, \mathcal{R}^{\mathcal{M}}, AP^{\mathcal{M}}, L^{\mathcal{M}})$ *and the environment model* $\mathcal{E} = (S^{\mathcal{E}}, s_0^{\mathcal{E}}, \mathbf{P}^{\mathcal{E}})$*, the composed model is an MDP* $\mathcal{M}||\mathcal{E} = (S, s_0, A, \mathbf{P}, \mathcal{R}, AP, L)$*, where*

- $S = \{(\underline{s}, t), (s, \underline{t}) \mid s \in S^{\mathcal{M}}, t \in S^{\mathcal{E}}\}$*, with* $s_0 = (s_0^{\mathcal{M}}, \underline{s_0^{\mathcal{E}}})$ *as the initial state;*
- $A = A^{\mathcal{M}} \cup \{env\}$*, where* env *is a new action;*
- $\mathbf{P}\big((s, \underline{t}), env, (\underline{s}, t')\big) = \mathbf{P}^{\mathcal{E}}(t, t')$ *and* $\mathbf{P}\big((\underline{s}, t), \alpha, (s', \underline{t})\big) = \mathbf{P}^{\mathcal{M}}(s, \alpha, s')$*;*
- $\mathcal{R}\big((s, \underline{t}), env, (\underline{s}, t')\big) = 0$ *and* $\mathcal{R}\big((\underline{s}, t), \alpha, (s', \underline{t})\big) = \mathcal{R}^{\mathcal{M}}(s, \alpha, s')$*;*
- $AP = AP^{\mathcal{M}}$*, and* $L\big((\underline{s}, t)\big) = L^{\mathcal{M}}(s)$*.*

Example 4.19 *Given the CWS MDP in Fig. 3 and the environment model in Fig. 4, the resulting MDP is shown in Fig. 5. Note that due to the space limit, we only show the part of the MDP (the first two steps from the CWS MDP).*

The model starts from the initial state $(s_0, \underline{normal})$ *where a probability distribution (labelled with action env) takes place, which is from the state* normal *in the environment model. In the next level, from state* $(\underline{s_0}, peak)$*, two actions are taken from the CWS MDP, but the probabilities are from the peak time availability* Q_0 *in Table 3.*

From state $(MiniCab, \underline{normal})$*, it has a similar distribution as the one mentioned above; and from state* $(Uper, \underline{night})$*, it takes the probabilities from state night in the environment model in Fig. 4. The dashed lines are other probability transitions of which we omit the details.*

	normal				peak				night			
	Q_0	$Q_t(h)$	Q_s	$Q_c(£)$	Q_0	Q_t	Q_s	Q_c	Q_0	Q_t	Q_s	Q_c
MiniCab	0.95	0.9	0	50	0.6	1.7	0	57	0.98	0.68	0	49
Uper	0.9	0.9	0	40	0.7	1.7	0	51	0.95	0.68	0	26
Express	0.98	2	0	100	0.98	2	0	115	0.98	2	0	100
Local	0.9	5	2	50	0.85	5.5	2	60	0.91	5	2	50

Table 3: The QoS in different environment

The reward structure will depend on the environment model and will only have positive rewards on transitions that start from CWS-active states, i.e., $(\underline{s}, t)$*. In this example, the travel time is* $\boldsymbol{\mathcal{R}}_{Q_t}\big((\underline{s_0}, night), Uper, (Uper, \underline{night})\big) = 0.68h$*, the number of transfers is* $\boldsymbol{\mathcal{R}}_{Q_s}\big((\underline{s_0}, night), Uper, (Uper, \underline{night})\big) = 0$*, and the cost is* $\boldsymbol{\mathcal{R}}_{Q_c}\big((\underline{s_0}, night), Uper, (Uper, \underline{night})\big) = £26$*.*

4.3.3 Analysis

We now show the results of the running example against some typical properties listed in Table 4. We will explain what it means to the model and, in particular, how they can be used for the web service composition. The policies in the last column specify which CWSs to select and in which order.

Boolean results The boolean results tell whether a given property is true or false.

- The first property in Table 4 - P>=0.98[F End] - is a *reachability probability* property. For all policies the probability of reaching B is no less than 0.98. An example policy is Uper *to* Local.
- The next three are *pareto* properties. When the journey is finished, either it is achieved by the express train or the local train, but not both (hence Multi(P>=0.98[F Express], P>=0.95[F local]) is false). A counterexample policy for the false property is given.
- The next two are related to rewards. The first one is a *reachability reward* property and the second one is a multi-objective referring to the expected total cumulative values.

Numerical results

- The first class of properties we compute is the *(unbounded) reachability probability*. As it is an MDP, where uncertainty arises, it is to compute the maximum or minimum reachability probability. In Table 4, the minimum probability of reaching city B by an express train is 0. The maximum probability of arriving at B is 1.0, as the probability of taking a loop (whose probability is less than 1) infinitely many times is 0. See the $*$ rows in Table 4.
- The second class of properties we compute is the *reachability rewards*. It is to compute the maximum or minimum reward when reaching a certain goal. In Table 4, we listed three properties (rows marked with +) to calculate the minimum time, cost, number of stops when reaching City B (It takes 2 hours, £99.6 and 1 stop at the minimum. Note that this is the weighted mean average between different modes of the environment.
- The third class of properties is *single objective single constraint*. It has one soft and one hard QoS requirement. To study these properties, one way is to first fix a pair of QoS measures, say Q_t (time) and Q_c (cost), and then change the bounds of the restriction (Q_t <= $2, 2.3, 3$, etc) and see how the value of the other property changes (minimum Q_c is £104, £99.3, £87.8, respectively). As the travel time increases, the budget reduces. See & marked in Table 4. The other direction also holds - when budget increases (£100, £110, £120), the travel time decreases (6.42, 5.75, 5.09 hours), see rows marked with $-$.
  The other way is to fix the restrictive properties (say the budget is no more than £200, marked with #) and see what the other metrics are. Compared with the case when budget is £100 (marked with §), the number of stop of the high budget case is lower, travelling time is

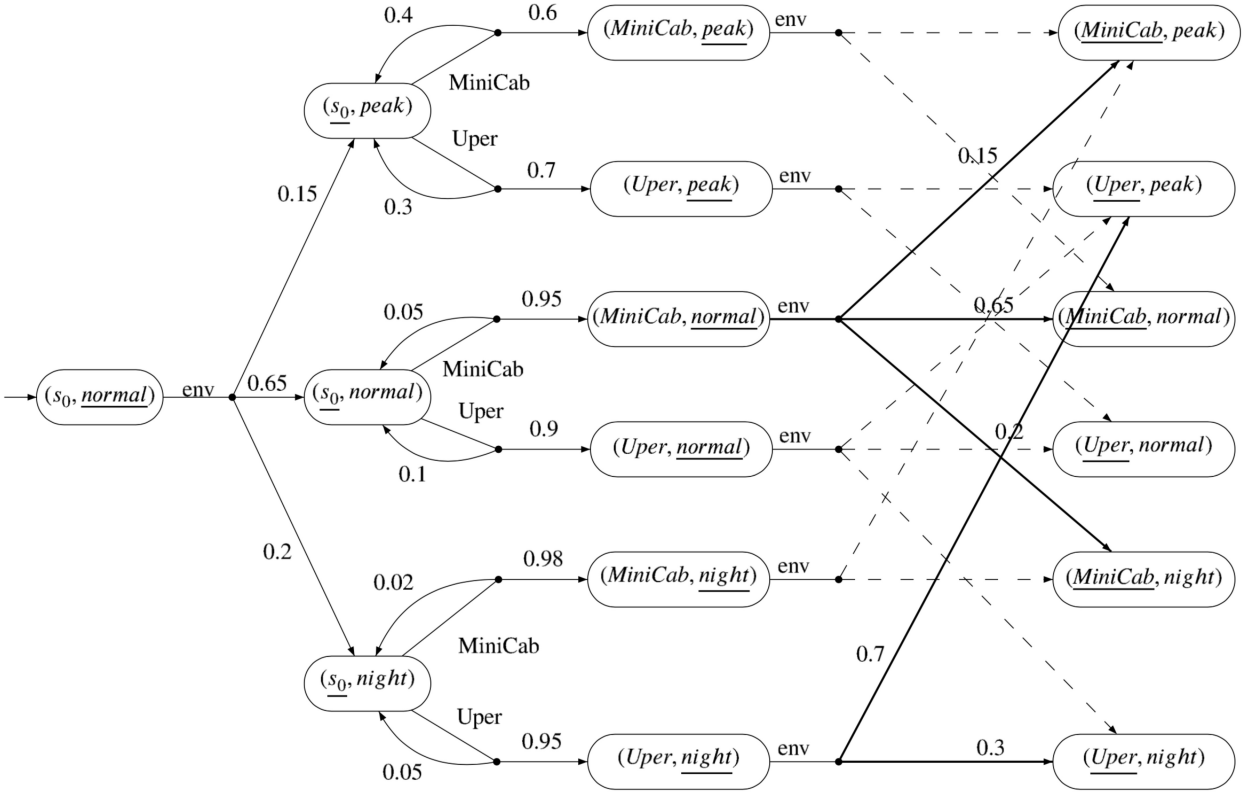

Figure 5: Concrete web service model and environment composition

shorter and the cost stays the same.

- The fourth class is a generalisation of the third class - *single objective multiple constraints*. It has one soft and multiple hard QoS requirement. In Table 4, it shows in ✠ that when the travelling time is no more than 3.5 hours and as the budget increases (£149, £150, £155), the minimum number of transfers reduces (1.09, 1.05, 1.00). This is expected as a higher budget means the affordability to take an express train. Similarly, the ✠-marked rows show the case when a minimum travel time is calculated for a restricted budget and time.
- The last class is *multiple objective properties*. The results contain the extreme points in the region. For instance, Multi(R{"Qc"}min=?[C], R{"Qs"}min=?[C]) has four extreme points. All the points included within the four points form a region where the two factors (cost and number of stops) are competing with each other. Often one is not able to obtain a policy that can minimise both objectives and has to compromise.

In summary, the QoS requirements from all aforementioned classes can be checked. Either none of the existing CWSs would satisfy the requirement and, if this is the case, a counterexample is given, or a policy on which CWSs to select and how to compose them is returned.

4.4 Web service composition via parametric analysis

In this section, we show how to perform web service composition by an approach based on parametric probabilistic verification. The overall strategy is to reduce the problem of finding an optimal combination of concrete web services to the policy synthesis of MDPs. The main challenges are the environment uncertainty and complicated

		Property	Result	Policy
Non−parametric	Boolean	P>=0.98[F End]	True	U→L
		Multi(P>=0.98[F Express], P>=0.95[F Local])	False	M→E
		Multi(P>=0.98[F End], P>=0.95[F Local])	True	M→L
		Multi(P>=0.98[F Express], P>=0.95[F End])	True	M→E
		R{"Qc"}<=110 [F End]	False	M→E
		Multi(R{"Qc"}<=200[C], R{"Qs"}<=1[C])	True	M→E
	Numerical	∗ Pmin=?[F Express]	0.0	M→E
		∗ Pmax=?[F End]	1.0	M→E
		+ R{"Qt"}min=?[F End]	3.28	U→E
		+ R{"Qc"}min=?[F End]	99.6	M→L
		+ R{"Qs"}min=?[F End]	1	M→E
		− Multi(R{"Qt"}min=?[C], R{"Qc"}<=100[C])	6.42	U→L
		− Multi(R{"Qt"}min=?[C], R{"Qc"}<=110[C])	5.75	U→L
		− Multi(R{"Qt"}min=?[C], R{"Qc"}<=120[C])	5.09	U→L
		& Multi(R{"Qc"}min=?[C], R{"Qt"}<=4[C])	137	U→E
		& Multi(R{"Qc"}min=?[C], R{"Qt"}<=5[C])	121	U→E
		& Multi(R{"Qc"}min=?[C], R{"Qt"}<=6[C])	106	U→L
		# Multi(R{"Qs"}min=?[C], R{"Qc"}<=200[C])	1.00	U→E
		# Multi(R{"Qc"}min=?[C], R{"Qc"}<=200[C])	99.60	U→L
		# Multi(R{"Qt"}min=?[C], R{"Qc"}<=200[C])	3.28	U→E
		§ Multi(R{"Qs"}min=?[C], R{"Qc"}<=100[C])	3.00	M→L
		§ Multi(R{"Qc"}min=?[C], R{"Qc"}<=100[C])	99.60	U→L
		§ Multi(R{"Qt"}min=?[C], R{"Qc"}<=100 C])	6.42	U→L
		✓ Multi(R{"Qs"}min=?[C], R{"Qc"}<=99[C], R{"Qt"}<=3.5[C])	NaN	–
		✓ Multi(R{"Qs"}min=?[C], R{"Qc"}<=130[C], R{"Qt"}<=3.5[C])	NaN	–
		✓ Multi(R{"Qs"}min=?[C], R{"Qc"}<=155[C], R{"Qt"}<=3.5[C])	1.00	M→E
		✠ Multi(R{"Qt"}min=?[C], R{"Qc"}<=130[C], R{"Qt"}<=6.5[C])	4.45	U→E
		✠ Multi(R{"Qt"}min=?[C], R{"Qc"}<=150[C], R{"Qt"}<=6.5[C])	3.36	U→E
		✠ Multi(R{"Qt"}min=?[C], R{"Qc"}<=170[C], R{"Qt"}<=6.5[C])	3.28	U→E
		% Multi(R{"Qc"}min=?[C], R{"Qs"}min=?[C])	[151,1]	M→E
		[137,1.5], U→E; [121,2.1], U→E	[100,3]	U→L
		% Multi(R{"Qc"}min=?[C], R{"Qt"}min=?[C])	[151,3.28]	M→E
		[156,3.28], U→E; [137,4], U→E; [121,5], U→E	[100,6.44]	U→L
		% Multi(R{"Qs"}min=?[C], R{"Qt"}min=?[C])	[1,3.27]	M→E

Table 4: Non-parametric Analysis Results for Travelling Example with Environment M – MiniCab, U – Uper, L – Local, E – Express

QoS requirement, which will be addressed by a parametric multi-objective analysis with the aid of the state-of-the-art probabilistic model checker PRISM.

4.4.1 Parametric environment and QoS models

As mentioned in Section 1, there are generally two occurrences of parameters in our models, i.e., in the environment and in the QoS of web services. Fig. 4 gives an example of the first case. The probability to go from *normal* to *night* (*peak*) could be p_1 (p_2) instead of 0.2 (0.15), naturally, the probability to go from *normal* to itself is $1 - p_1 - p_2$. This probability parameters will remain to be a parameter for probability in the resulting MDP.

In the second case, parameters represent some QoS measures of the transitions in the CWS models, They can either be the availability of a CWS or other QoS measures usually with given upper- and lower-bounds. After the

composition, those QoS parameters continue to be QoS parameters in the resulting parametric MDPs. The following example introduces three parameters, two of which are the probabilities in the environmental model (p_1, p_2) and the last one is a QoS measure q_t.

Example 4.20 *In this case study, we will have three parameters:* $p_1 \in [0.1, 0.3]$ *(the probability from normal to night),* $p_2 \in [0.1, 0.4]$ *(the probability from normal to peak) and* $q_t \in [0.6, 2]$ *(the average time to drive from home to train station). Note that* q_t *is introduced as there is a high variation in the arriving time and the Uper/MiniCab fare closed related to* q_t*. Here we assume the taxi fare* Q_c *is a linear function in* q_t*. Depending on how jammed the traffic is, one would choose MiniCab or Uper to minimise the cost. The parametric QoS measures are set in Table 5. The other measures remain the same as in Table 3.*

	MiniCab		Uper	
	$Q_t(h)$	$Q_c(£)$	$Q_t(h)$	$Q_c(£)$
normal	$0.5q_t + 0.3$	$34q_t + 10$	$0.5q_t + 0.3$	$3q_t + 4$
peak	$0.5q_t + 0.5$	$24q_t + 22$	$0.5q_t + 0.5$	$35q_t + 9$
night	$0.4q_t + 0.2$	$35q_t + 7$	$0.4q_t + 0.2$	$20q_t + 2$

Table 5: The parametric QoS in three environment modes

Probabilistic verification tools support automated solving parametric MDPs against various kinds of properties specified before. (In probabilistic verification terms, they are unbounded until and reachability rewards properties.)

Depending on the property under consideration, the result is then given as either a rational function over the parameters or as a mapping from regions of these parameters to rational functions or truth values [39, 40]. To be more specific, given a property, we usually have three types of results which can be exploited for web service selection: it is either a boolean result, a numerical result of probabilities/rewards, or, in the presence of a parametric model, an explicit form of a rational function that can be optimised or plotted.

4.4.2 Parametric analysis

In the parametric analysis, it is possible to have one or more parameters the ranges of which are given. For a reachability probability or a reachability reward property, the closed-form expressions for the objective could be calculated. For instance, we have described three parameters $p_1 \in [0.1, 0.3], p_2 \in [0.1, 0.4], q_t \in [0.6, 2]$ in Section 4.4. We are to show cases with 1, 2 and 3 parameters. The results are summarised in Table 6.

		Property	Result
Parametric	1 parameter	R{"Qc"}min=?[F End] $(p_2 = 0.4, q_t = 1.2)$	$h^{c1}(p_1)$
		R{"Qc"}min=?[F End] $(p_1 = 0.2, q_t = 1.2)$	$h^{c1}(p_2)$
		R{"Qc"}min=?[F End] $(p_1 = 0.2, p_2 = 0.4)$	$h^{c1}(q_t)$
		R{"Qt"}min=?[F End] $(p_2 = 0.4, q_t = 1.2)$	$h^{t1}(p_1)$
		R{"Qt"}min=?[F End] $(q_1 = 0.2, q_t = 1.2)$	$h^{t1}(p_2)$
		R{"Qt"}min=?[F End] $(p_1 = 0.2, p_2 = 0.4)$	$h^{t1}(p_1)$
		R{"Qs"}min=?[F End] $(p_2 = 0.4, q_t = 1.2)$	1
		R{"Qs"}min=?[F End] $(q_1 = 0.2, q_t = 1.2)$	1
		R{"Qs"}min=?[F End] $(p_1 = 0.2, p_2 = 0.4)$	1
	2 paras	R{"Qc"}min=?[F End] $(q_t = 1.2)$	$f^{c2}(p_1, p_2)$
		R{"Qt"}min=?[F End] $(q_t = 1.2)$	$f^{t2}(p_1, p_2)$
		R{"Qs"}min=?[F End] $(q_t = 1.2)$	1
	3 paras	R{"Qc"}min=?[F End]	$g^{c3}(p_1, p_2, q_t)$
		R{"Qt"}min=?[F End]	-
		R{"Qs"}min=?[F End]	1

Table 6: Parametric Analysis Results for Travelling Example with Environment

One parameter If we instantiate the two parameters $p_2 = 0.4, q_t = 1.2$, that would leave only one parameter p_1. The function for the minimum cost is captured by $h^{c1}(p_1)$, where c means it is to calculate the cost and 1 means there is one parameter. The function is of the following form for $p_1 \in [0.1, 0.3]$ is shown in (1). The functions $h^{c1}(p_2)$ and $h^{c1}(q_t)$ are in the similar fashion with p_2 and q_t as the parameter, respectively.

$$h^{c1}(p_1) = (145397874025p_1^2 + 117281115430p_1 + 2189373409209) / (244893675p_1^2 + 4572125610p_1 + 21340212843) \quad (1)$$

$$h^{c1}(p_2) = (36960320250p_2^3 - 168561353275p_2^2 + 776963615860p_2 + 1932464456940) / (136890075p_2^2 + 3382066380p_2 + 20889704748) \quad (2)$$

$$h^{c1}(q_t) = (31377704725q_t + 54790318799)/927684738 \quad (3)$$

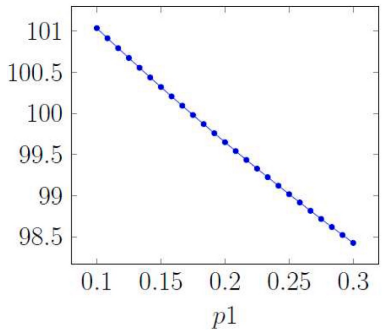

Figure 6: $h^{c1}(p_1)$

When the functions are plotted, we can see how the three parameters are related to the minimum expected cost of reaching City B, respectively.

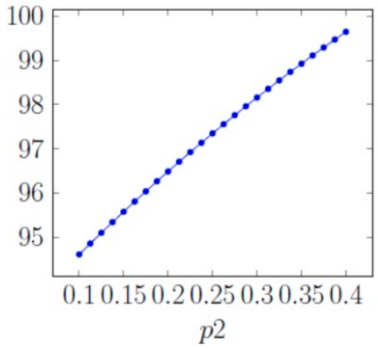

Figure 7: $h^{c1}(p_2)$

It shows from Table 3 that it is cheaper to travel in the night than at normal time. As a result, when p_1 increases, the price drops, and that explains the trend in Fig. 6. On the contrary, it costs more to travel at the peak time. Thus when more weights are towards the peak time, i.e., p_2 increases, the cost would rise, see Fig. 7. As the MiniCab or Uper fare is positively linearly related to q_t, it means that when q_t increases the taxi cost also increases, see Fig. 8. With the closed-form expression, it is possible to evaluate the value (in this case the minimum cost) given the value to the parameter. For instance, $h^{c1}(p_1) = £101.04$ when instantiating $p_1 = 0.2$ in Eq. (1).

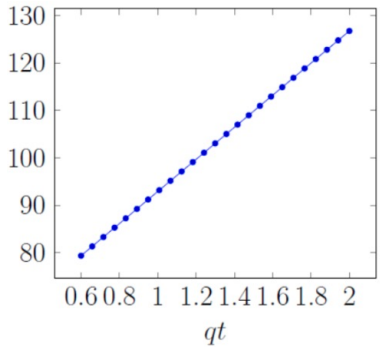

Figure 8: $h^{c1}(q_t)$

Two parameters If the model has two parameters, the resulting closed-form expression could be quite involved. Let's still take the minimum cost Q_c as the objective and let $q_t = 1.2$.

The function $f^{c2}(p_1, p_2)$ calculates the minimum cost when parameters $p_1 \in [0.1, 0.3]$ and $p_2 \in [0.1, 0.4]$.

$$
\begin{aligned}
& f^{c2}(p_1, p_2) \\
=&(66121292250p_1^2p_2 + 98870908500p_1p_2^2 + 36960320250p_2^3 \\
&+118949357125p_1^2 - 246838623450p_1p_2 - 188335534975p_2^2 \\
&+200197219450p_1 + 823686488860p_2 + 1887667038765) \\
&/(244893675p_1^2 + 366188550p_1p_2 + 136890075p_2^2 \\
&+4425650190p_1 + 3308828670p_2 + 19994778963)
\end{aligned}
$$

Given this closed-form expression, we can evaluate the function $f^{c2}(0.1, 0.1) = 95.75$, which means that when the probability to go from normal to night and from normal to peak are both 0.1, then the minimum expected cumulative value of total cost (Q_c) is £95.75. The plot can be found in Fig. 9.

Likewise we can have a similar analysis on how two parameters vary to affect the measure Q_t (time to reach B). The different combinations of parameters can be found in Fig. 10 and 11.

Three or more parameters. When there are more than two parameters, it is not easy to visualise the results in one plot. Instead, we present the results in a series of plots, each with several surfaces. When there are more parameters, the computation time increases and the expression

could be (sometimes very) complicated.

For minimum Q_c, where $p_1 \in [0.1, 0.3]$, $p_2 \in [0.1, 0.4]$ and $q_t \in [0.6, 2.0]$:

$$
\begin{aligned}
& g^{c3}(p_1, p_2, q_t) \\
=& (423635301250p_1^2 q_t - 191557642500p_1 p_2 q_t - 351326263750p_2^2 q_t \\
& +83056825250p_1^2 - 177233878500p_1 p_2 - 418554481500p_1 q_t \\
& -168447287750p_2^2 + 1509743875500q_t * p_2 - 57386583300p_1 \\
& +575587328100p_2 + 3353377815450q_t + 482741425890) \\
& /(627074175p_1^2 + 936527550p_1 p_2 + 349672575p_2^2 \\
& +17599103940p_1 + 13147688820p_2 + 107824464333)
\end{aligned}
$$

For instance, we could have three figures (Fig. 12, 13 and 14), where three surfaces are shown.

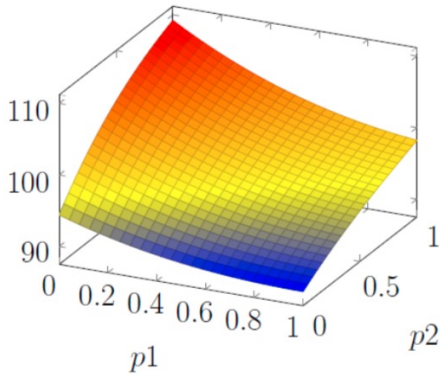

Figure 9: $h^{c2}(p_1, p_2)$ at $q_t = 1.2$

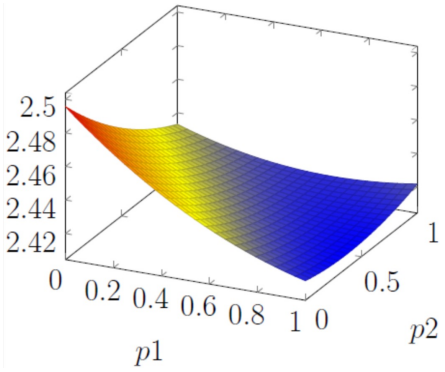

Figure 10: $h^{t2}(p_1, p_2)$ at $q_t = 1.2$

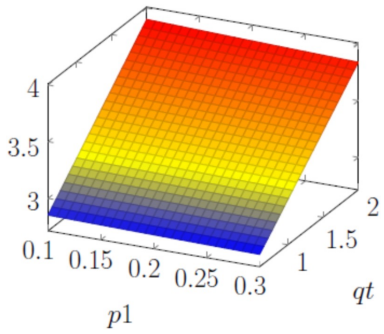

Figure 11: $h^{t2}(p_1, q_t)$ at $p_2 = 0.4$

5 Conclusion

In this paper, we present a model-based approach for QoS-aware service composition based on MDPs. We show how to obtain an MDP model based on the models of abstract and concrete services, and the environment model as a fully probabilistic system. Armed with probabilistic verification, we are able to find the optimal service selection strategy against their QoS and the environment changes. The analysis can be made parametric with parametrised models for QoS of services and dynamics of environment changes, which makes them amicable to autonomous and self-adaptive systems. Our approach is based on a formal framework on a rigourous basis which provides a holistic, independent support for QoS-aware service composition, including modelling, specification and automated synthesis of composition strategies and parameters. This model-based approach offers great flexibility, owing to the expressiveness of the MDP models we have adopted, and allows to take the environment into account explicitly. One of the strengths of our approach is to provide, not only qualitative, but also quantitative analysis results, which is indispensable especially for the parametric analysis. While bearing the modelling capability, as the experiments show, our framework is computationally feasible, and may demonstrate superiority in performance because of the optimised methods and tools to solve MDPs.

Further work includes a learning approach to find out the parameters online so we can have a more comprehensive framework in support of environment-sensitive service composition.

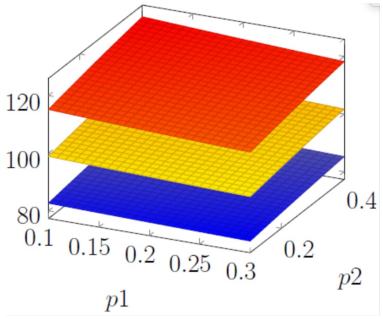

Figure 12: $f^{c2}(p_1, p_2)$ at $q_t = 0.8, 1.3, 1.8$

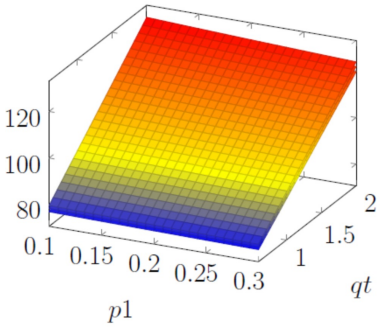

Figure 13: $f^{c2}(p_1, q_t)$ at $p_2 = 0.15, 0.25, 0.35$

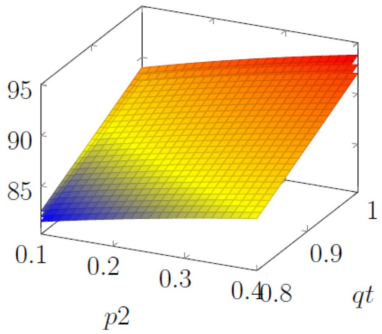

Figure 14: $f^{c2}(p_2, q_t)$ at $p_1 = 0.12, 0.2.0.28$

Acknowledgements The work was partially supported by the National Key R&D Program of China (No. 2018YFB1003902), the National Natural Science Foundation of China (No. 61972197), the Fundamental Research Funds for the Central Universities (No. NS2019055), Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Qing Lan Project. T. Chen is partially supported by Birkbeck BEI School Project (ARTEFACT), ARC Discovery Project (DP160101652, DP180100691), NSFC grant (No. 61872340), and Guangdong Science and Technology Department grant (No. 2018B010107004). T. Han is supported by EPSRC grant (EP/P015387/1).

References

[1] Sheng, Q.Z., Qiao, X., Vasilakos, A.V., Szabo, C., Bourne, S., Xu, X.: 'Web services composition: A decades overview', *Information Sciences*, 2014, **280**, pp. 218 – 238

[2] Chen, W., Paik, I.: 'Toward better quality of service composition based on a global social service network', *IEEE Transactions on Parallel and Distributed Systems*, 2015, **26**, (5), pp. 1466–1476

[3] Lemos, A.L., Daniel, F., Benatallah, B.: 'Web service composition: A survey of techniques and tools', *ACM Comput Surv*, 2016, **48**, (3), pp. 33:1–33:41

[4] Zhou, Y., Chen, T.: 'Software Adaptation in an Open Environment: A Software Architecture Perspective'. (CRC Press, Taylor & Francis Group, 2017)

[5] Hayyolalam, V., Kazem, A.A.P.: 'A systematic literature review on qos-aware service composition and selection in cloud environment', *Journal of Network and Computer Applications*, 2018, **110**, pp. 52 – 74

[6] Mabrouk, N.B., Beauche, S., Kuznetsova, E., Georgantas, N., Issarny, V. 'Qos-aware service composition in dynamic service oriented environments'. In: ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing. (Springer, 2009. pp. 123–142

[7] Silic, M., Delac, G., Srbljic, S.: 'Prediction of atomic web services reliability for qos-aware recommendation', *IEEE Transactions on Services Computing*, 2015, **8**, (3), pp. 425–438

[8] Zhang, P., Zhuang, Y., Leung, H., Song, W., Zhou, Y. 'A novel qos monitoring approach sensitive to environmental factors'. In: Web Services (ICWS), 2015 IEEE International Conference on. (IEEE, 2015. pp. 145–152

[9] Wang, H., Wu, Q., Chen, X., Yu, Q., Zheng, Z., Bouguettaya, A. 'Adaptive and dynamic service composition via multi-agent reinforcement learning'. In: Web Services (ICWS), 2014 IEEE International Conference on. (IEEE, 2014. pp. 447–454

[10] Wang, H., Ma, P., Yu, Q., Yang, D., Li, J., Fei, H.: 'Combining quantitative constraints with qualitative preferences for effective non-functional properties-aware service composition', *Journal of Parallel and Distributed Computing*, 2017, **100**, pp. 71–84

[11] Canfora, G., Di.Penta, M., Esposito, R., Villani, M.L. 'An approach for qos-aware service composition based on genetic algorithms'. In: Proceedings of the 7th annual conference on Genetic and evolutionary computation. (ACM, 2005. pp. 1069–1075

[12] Zhou, Y., Ge, J., Zhang, P., Wu, W.: 'Model based verification of dynamically evolvable service oriented systems', *Science China Information Sciences*, 2016, **59**, (3), pp. 32101

[13] Wang, P., Liu, T., Zhan, Y., Du, X. 'A bayesian nash equilibrium of qos-aware web service composition'. In: Web Services (ICWS), 2017 IEEE International Conference on. (IEEE, 2017. pp. 676–683

[14] Ren, L., Wang, W., Xu, H.: 'A reinforcement learning method for constraint-satisfied services composition', *IEEE Transactions on Services Computing*, 2017,

[15] Rodriguez.Mier, P., Mucientes, M., Lama, M.: 'Hybrid optimization algorithm for large-scale qos-aware service composition', *IEEE transactions on services computing*, 2017, **10**, (4), pp. 547–559

[16] Ali, N., Solís, C. 'Self-adaptation to mobile resources in service oriented architecture'. In: 2015 IEEE International Conference on Mobile Services, MS 2015, New York City, NY, USA, June 27 - July 2, 2015. (, 2015. pp. 407–414. Available from: https://doi.org/10.1109/MobServ.2015.62

[17] Yu, Y., Ma, H., Zhang, M. 'An adaptive genetic programming approach to qos-aware web services composition'. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013. (IEEE, 2013. pp. 1740–1747. Available from: https://doi.org/10.1109/CEC.2013.6557771

[18] Jungmann, A., Mohr, F.: 'An approach towards adaptive service composition in markets of composed services', *Journal of Internet Services and Applications*, 2015, **6**, (1), pp. 5

[19] Moustafa, A., Zhang, M. 'Multi-objective service composition using reinforcement learning'. In: International Conference on Service-Oriented Computing. (Springer, 2013. pp. 298–312

[20] Bashari, M., Bagheri, E., Du, W.: 'Self-adaptation of service compositions through product line reconfiguration', *Journal of Systems and Software*, 2018, **144**, pp. 84–105

[21] Tilsner, M., Fiech, A., Zhan, G., Specht, T. 'Patterns for service composition'. In: Fourth International C* Conference on Computer Science & Software Engineering, C3S2E 2011, Montreal, Quebec, Canada, May 16-18, 2011, Proceedings. (, 2011. pp. 133–137. Available from: https://doi.org/10.1145/1992896.1992913

[22] Aguilar.Saven, R.S.: 'Business process modelling: Review and framework', *International Journal of Production Economics*, 2004, **90**, pp. 129–149

[23] Wong, P.Y.H., Gibbonsa, J.: 'Formalisations and applications of bpmn', *Science of Computer Programming*, 2011, **76**, pp. 633–650

[24] Corradini, F., Fornari, F., Polini, A., Re, B., Tiezzi, F., Vandin, A. 'Bprove: tool support for business process verification'. In: Rosu, G., Penta, M.D., Nguyen, T.N., editors. Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03,

2017. (IEEE Computer Society, 2017. pp. 937–942. Available from: https://doi.org/10.1109/ASE.2017.8115708

[25] Kheldoun, A., Barkaoui, K., Ioualalen, M.: 'Formal verification of complex business processes based on high-level petri nets', *Inf Sci*, 2017, **385**, pp. 39–54. Available from: https://doi.org/10.1016/j.ins.2016.12.044

[26] Zhang, W., Sun, H., Liu, X., Guo, X. 'Temporal qos-aware web service recommendation via non-negative tensor factorization'. In: Proceedings of the 23rd international conference on World wide web. (ACM, 2014. pp. 585–596

[27] Madi, B.M.A., Sheng, Q.Z., Yao, L., Qin, Y., Wang, X. 'Plmwsp: Probabilistic latent model for web service qos prediction'. In: IEEE International Conference on Web Services, ICWS 2016, San Francisco, CA, USA, June 27 - July 2, 2016. (, 2016. pp. 623–630

[28] Wu, C., Qiu, W., Wang, X., Zheng, Z., Yang, X. 'Time-aware and sparsity-tolerant qos prediction based on collaborative filtering'. In: IEEE International Conference on Web Services, ICWS 2016, San Francisco, CA, USA, June 27 - July 2, 2016. (, 2016. pp. 637–640

[29] Xiong, W., Wu, Z., Li, B., Gu, Q. 'A learning approach to qos prediction via multi-dimensional context'. In: 2017 IEEE International Conference on Web Services, ICWS 2017, Honolulu, HI, USA, June 25-30, 2017. (, 2017. pp. 164–171

[30] Trang, M.X., Murakami, Y., Ishida, T.: 'Policy-aware service composition: Predicting parallel execution performance of composite services', *IEEE Transactions on Services Computing*, 2018, **11**, (4), pp. 602–615

[31] Hansson, H., Jonsson, B.: 'A logic for reasoning about time and reliability', *Formal Aspects of Computing*, 1994, **6**, (5), pp. 512–535

[32] Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J. 'Model checking continuous-time markov chains by transient analysis'. In: Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings. (, 2000. pp. 358–372. Available from: https://doi.org/10.1007/10722167_28

[33] Kwiatkowska, M., Norman, G., Parker, D.: 'Prism 4.0: verification of probabilistic real-time systems', *Lecture Notes in Computer Science*, 2011, **6806**, pp. 585–591

[34] Baier, C., Katoen, J.: 'Principles of model checking'. (MIT Press, 2008)

[35] Yu, T., Lin, K. 'A broker-based framework for qos-aware web service composition'. In: 2005 IEEE International Conference on e-Technology, e-Commerce, and e-Services (EEE 2005), 29 March - 1 April 2005, Hong Kong, China. (, 2005. pp. 22–29

[36] Wu, Z., Gomadam, K., Ranabahu, A., Sheth, A.P., Miller, J.A. 'Automatic composition of semantic web services using process mediation'. In: ICEIS 2007 - Proceedings of the Ninth International Conference on Enterprise Information Systems, Volume SAIC, Funchal, Madeira, Portugal, June 12-16, 2007. (, 2007. pp. 453–462

[37] Zhu, J., He, P., Xie, Q., Zheng, Z., Lyu, M.R. 'Carp: Context-aware reliability prediction of black-box web services'. In: 2017 IEEE International Conference on Web Services (ICWS). (, 2017. pp. 17–24

[38] Wang, H., Wang, L., Yu, Q., Zheng, Z., Yang, Z.: 'A proactive approach based on online reliability prediction for adaptation of service-oriented systems', *Journal of Parallel and Distributed Computing*, 2018, **114**, pp. 70 – 84

[39] Hahn, E.M., Hermanns, H., Zhang, L.: 'Probabilistic reachability for parametric markov models', *International Journal on Software Tools for Technology Transfer (STTT)*, 2011, **13**, (1), pp. 3–19

[40] Hahn, E.M., Han, T., Zhang, L. 'Synthesis for PCTL in parametric markov decision processes'. In: Proc. 3rd NASA Formal Methods Symposium (NFM'11), volume 6617 of LNCS. Springer. (, 2011.