QoS WITH RELIABILITY AND SCALABILITY IN ADAPTIVE SERVICE-BASED SYSTEMS

V. RHYMEND UTHARIARAJ[1] P.MERCY FLORENCE[2] A.GEETHA[3]

[1]Prof&Director, Anna University, Chennai
[2]Research Scholar, Anna University, Chennai
[3]Asst.Prof, Madha Engg.College, Chennai

ABSTRACT

Service-based systems that are dynamically composed at runtime to provide complex, adaptive functionality are currently one of the main development paradigms in software engineering. However, the Quality of Service (QoS) delivered by these systems remains an important concern, and needs to be managed in an equally adaptive and predictable way. To address this need, we introduce a novel, tool-supported framework for the development of adaptive service-based systems called QoSMOS (QoS Management and Optimization of Service-based systems). QoSMOS can be used to develop service-based systems that achieve their QoS requirements through dynamically adapting to changes in the system state, environment, and workload. QoSMOS service-based systems translate high-level QoS requirements specified by their administrators into probabilistic temporal logic formulae, which are then formally and automatically analyzed to identify and enforce optimal system configurations. The QoSMOS self-adaptation mechanism can handle reliability and performance-related QoS requirements, and can be integrated into newly developed solutions or legacy systems. The effectiveness and scalability of the approach are validated using simulations and a set of experiments based on an implementation of an adaptive service-based system for remote medical assistance.

KEYWORDS

Service-oriented software engineering, QoS management, QoS optimization, adaptive systems.

1 INTRODUCTION

SERVICE-BASED systems (SBSs) are playing an increasingly important role in application domains ranging from research and health care to defense and aerospace. Built through the dynamic composition of loosely coupled services offered by independent providers, SBSs are operating in environments characterized by continual changes to requirements, state of component services, and system usage profiles. In this context, the ability of SBSs to adjust their behavior in response to such changes through self-adaptation has become a promising research direction [32], [70].

Several approaches to architecting adaptive software systems (i.e., software systems that reconfigure themselves in line with changes in their requirements and/or environment) have

DOI : 10.5121/csit.2011.1305

already appeared in the literature [70], [66]. These approaches involve the use of intelligent control loops that collect information about the current state of the system, make decisions, and then adjust the system as necessary (e.g., [5], [22], [26],. Alternative approaches define self-adaptable architectures that emulate the behavior of biological systems, where the global, complex behavior emerges from the cooperation and interaction among distributed, independent components [41],

Achieving and maintaining well-defined Quality of Service (QoS) properties in a changing environment represents a key challenge for self-adapting architectures. Service-based systems are well positioned to address these challenges, as the exploitation of their different composition patterns (orchestration and choreography) can represent an efficient way to achieve self-adapting architectures [12], [86]. Consider, for example, a highly dynamic system where the set of discoverable services may change over time, either because service providers publish (or withdraw) service descriptions or because the availability of certain services may vary according to the users location or to the network connectivity. In these settings, a more reliable or efficient service might become available, and thus self-adaptation may allow its use to improve the overall QoS. A further example is that, because of the increase of the number of users concurrently accessing the system, the response time experienced by an user could become too high. In this case, the system should adopt appropriate reconfiguration strategies (such as using more computational resources or changing service providers) to tackle the peaks in the workload. Therefore, a significant research effort has been devoted to the definition and analysis of QoS properties in SBS systems (e.g., [43], [47], [79], [93]). As illustrated by the overview of related approaches later in this section, typical QoS properties associated with SBSs include operation cost on one hand and probabilistic quality attributes such as availability, reliability, and reputation [101], [5] on the other hand. Among these QoS properties, the management of probabilistic quality attributes is particularly challenging due to problems arising from the environment variability (e.g., changing service workloads and failure rates). Furthermore, QoS management requires self-adaptive SBSs to take into account aspects such as QoS specification, QoS evaluation, QoS optimization, and QoS-based adaptation. Nevertheless, guaranteeing a given level of QoS in these systems is essential for their success in the envisioned "service market," where service providers will compete by offering services with similar functionality but different quality and cost attributes [12]

To deal with the QoS management of SBSs, we define and realize a generic architecture for adaptive SBSs called QoSMOS (QoS Management and Optimization of Service-based systems). QoSMOS is a tool-supported framework for the QoS management of self-adaptive, service-based systems that combines, in a novel way, existing techniques and tools developed by our research groups:

1. Formal specification of QoS requirements with probabilistic temporal logics and the Propose specification system [49],
2. model-based QoS evaluation with probabilistic verification techniques provided by the PRISM model checker [72],
3. monitoring and Bayesian-based parameter adaptation of the QoS models exploiting KAMI [40], and
4. Planning and execution of system adaptation based on GPAC [20].

The QoSMOS framework supports the practical realization of adaptive SBS architectures by means of two complementary mechanisms. The first mechanism consists of selecting the services that compose a QoSMOS service-based system dynamically. Given a set of functionally equivalent services for each component of an SBS, QoSMOS selects those services whose reliability, performance, and cost guarantee the realization of the QoS requirements for the system. QoSMOS is capable of dynamically adapting its selection of services to runtime changes in both the service characteristics (e.g., reliability or performance) and the system QoS requirements. When service selection cannot achieve the QoS requirements, a warning is issued to alert the SBS administrator

The second adaptation mechanism employed by QoSMOS consists of adjusting the resources (e.g., the CPU) allocated to individual services within a service-based system dynamically. This mechanism is applied to services hosted and administered internally by the organization that implements the SBS. Its key benefit is the ability to adapt the resources allocated to SBS component services to the actual workload of the system, thus ensuring that its QoS requirements are satisfied with minimal cost and environmental impact.

Related approaches. One benefit of SBSs is the ability to build applications through composition of available ser-vices at runtime. This composition involves several activities, including the definition of an integration schema yielding the target application, the selection of concrete services that offer the required functionality, and the fulfillment of QoS constraints. While services are described and listed in public registries, there is still little support for QoS-based service management. To cover this gap, the research area of QoS Management in SBSs has been very active in the last five years.

Domain-sorted summary of recent approaches in the area of QoS-driven service selection, composition, and adaptation is given in Table 1.

Specifically, we summarize the approaches according to:

1. the considered QoS metrics and QoS Specification languages (QoS Requirement Specification),
2. the models/algorithms adopted for the QoS metric evaluation (QoS Evaluation Methods),
3. the type of optimization problem defined and solved and/or the adaptation policies adopted (QoS Optimization or Adaptation Methods), and finally
4. Validation of the proposed approaches (Validation).

Considering these approaches, we identify some common points of weakness that we overcome with our QoSMOS approach. QoS requirement specification. As illustrated in Table 1, a variety of different QoS requirements are considered in the current approaches. However, QoS specifications are often tackled in an abstract way by dealing with simple metrics (e.g., by considering the failure rate as a metric to evaluate reliability). In our view, a detailed and formal specification of QoS requirements is required for a comprehensive management of QoS in service-based systems. A concise and unambiguous specification of the QoS requirements enables, among other benefits, a systematic management of SBSs based on the quantitative analysis of their QoS properties.Current examples of specification languages for QoS aspects in the Web services domain are: Web Service Level Agreement (WSLA) [65], the timed Web Service Constraint Language (timed WSCoL) [9], which is close to a real-time temporal logic, the Web Service Management Language (WSML) [92], and the Web Service Offerings Language (WSOL)

In addition, formal QoS specification can be achieved using formalisms like real-time and probabilistic temporal logics [1], [6], [8], [53], [69], [71], timed Life Sequence Charts [54], probabilistic and timed Message Sequence Charts [90], Performance Trees [97], or Probabilistic/Timed Behavior Trees [36], [37], [50]). To this end, QoSMOS adopts the probabilistic temporal logics PCTL [53] and CSL [8] because these logics are sufficiently expressive to formulate a variety of QoS requirements [49] whose formal verification can then be carried out using existing probabilistic model checkers.

QoS evaluation methods. To be effective, QoS evaluation approaches should rely on models representing the systems in an accurate/realistic way, and whose parameters can be adjusted at runtime according to measured data. Several approaches reported in Table 1 rely on the definition of simple aggregate QoS functions (like sum, product, max, and average) that can be easily defined and managed. However, due to dependencies between different services or between services and resources or the operational profiles, these aggregation functions could lead to quality estimation that represent optimistic (or pessimistic) bounds rather than a realistic estimation.

QoS delivered by its sub services. Examples can be found in [43], [83], [79], [93], where, starting from the BPEL business processes modeled by UML activity diagrams or by direct acyclic graphs, performance models based on simple queuing networks [83], [79] or reliability models based on Markov models are derived [43], [93]. In line with these approaches, we argue that comprehensive predictive quality evaluation models are needed. Examples of models that can be used for QoS evaluation are: Markov models, state charts like probabilistic UML State Charts [59], [60], queuing networks models [18], [76], stochastic process algebras like PEPA [46]. Toward this end, QoSMOS adopts Markov models as modeling formalisms to determine quantitatively the reliability and performance quality metrics of service-based systems. However, as an enhancement to the existing approaches, we observe composite systems (e.g., usage profiles, branching, and failure probabilities) at runtime and update the quality evaluation models.

To check if a Markov model satisfies its QoS requirements, numerical/symbolic [6], [8], [16], [53] and statistical [100] techniques have been developed, and extensive tool support is available (e.g., PRISM [72]).QoS optimization or adaptation methods. Devising QoS driven adaptation methodologies of SBSs is of utmost importance in the envisaged dynamic environment in which SBS operate. Most of the proposed methodologies for QoS-driven adaptation of SBS address this problem as a service selection problem (e.g., [5], [26], [30], [101]). Other papers have instead considered SBS adaptation through workflow restructuring, exploiting the inherent redundancy of SBS (e.g., [31], [52], [55].) In [28], a unified framework is proposed where service selection is integrated with other kinds of workflow restructuring to achieve greater flexibility in the adaptation.

According to this last approach, we conclude that the service selection and composition problem is really important for SBS QoS-based adaptation, but we also argue that for a comprehensive approach to QoS Management, optimal resource allocation and parameterization of the services is also required.

The QoSMOS framework does not aim to invent new techniques, but includes and integrates optimization techniques and adaptation strategies derived from approaches already present in literature.

Validation. An investigation of the validation strategies shows that several approaches perform experiments based on generated examples or apply a case study-based validation. To validate the QoSMOS approach, we use a similar validation strategy and perform experiments and simulations based on an implementation of a service-based system for remote medical assistance called TeleAssistance [10], [40].Contribution. Based on the review of the related approaches, the main contributions of the QoSMOS framework can be summarized as follows:

- In contrast to the simple and informal metrics that are currently used in the related approaches, QoSMOS uses a precise and formal specification of QoS requirements with probabilistic temporal logics.
- QoSMOS uses a tool-supported model-based quality evaluation methodology for probabilistic QoS attributes(i.e., performance, reliability, and resource usage) of service-based systems that significantly improves current approaches that use simple aggregation functions for QoS prediction because we could model quality dependencies on other services and the operational profile.
- QoSMOS utilizes techniques and tools for monitoring service-based systems and learning the parameters of their model(s) from the observed behavior of the system.
- QoSMOS adds self-adaptation (e.g., self Configuration and self-optimization) capabilities to service based systems through continuous verification of quantitative properties at runtime derived from high-level, user-specified system goals encoded with multi objective utility functions. The self-adaptation capabilities include service selection, runtime reconfiguration, and resource assignment. Consequently, QoSMOS subsumes most of the existing approaches.

2 PRELIMINARIES

2.1 Formal Definition of QoS Requirements

The precise specification of QoS requirements or Service Level Agreements (SLAs) is an important aspect for service composition, service selection, and optimization of service-based systems [42]. In QoSMOS, QoS requirements are specified using real-time temporal logics such as MTL (Metric Temporal Logic) [69] and TCTL (Timed Computational Tree Logic) [1], or probabilistic temporal logics such as PCTL (Probabilistic Computation Tree Logic) [53], PCTL* [6], PTCTL (Probabilistic Timed CTL) [71], and CSL (Continuous Stochastic Logic) [8]. The significant benefits of using logic-based requirement specifications include the ability to define these requirements concisely and unambiguously, and to analyze those using rigorous, mathematically-based tools such as model checkers.

Furthermore, for logic-based specification-formalism, the correct definition of QoS proper-ties is supported with specification patterns [39], [49], [48], [68] and structured English grammars [49], [68].Traditionally, the semantics of the PCTL/CSL is defined with a satisfaction relation j¼ over the states S and possible paths Path^{M}ðsÞ that are possible in a state s 2 S of a discrete/continuous time probabilistic model M. For de-tails about the formal semantics the reader is referred to [8], [33], [53]. Normally, a PCTL/CSL formula is evaluated starting from the initial state of the probabilistic model M. However, for convenience, in tools like PRISM any state and also a set of states can be chosen with a filter. Syntactically, a filter is specified as logical expression inside braces at the end of the PCTL/CSL formula.

2.2 Quality Evaluation Models

Several approaches exist in the literature for the model-based quality analysis and prediction, spanning the use of Petri nets, queuing networks, layered queuing networks, stochastic process algebras, Markov processes, fault trees, statistical models, and simulation models (see [3] for a recent review and classification of models for software quality analysis).In this paper, we focus on Markov models, which are a very general evaluation model that can be used to reason about performance and reliability properties.

Furthermore, Markov models include other modeling approaches as special cases, such as queuing networks, Stochastic Petri Nets [78], and Stochastic Process Algebras [34].Specifically, Markov models are stochastic processes defined as state-transition systems augmented with probabilities. Formally, a stochastic process is a collection of random variables defined on a common sample (probability) space. In Markov models [18], states represent possible configurations of the system being modeled.

Transitions among states occur at discrete or continuous time-steps and the probability of making transitions is given by exponential probability distributions. The Markov property characterizes these models: It means that, given the present state, future states are independent of the past. In other words, the description of the present state fully captures all of the information that could influence the future evolution of the process.

The most used Markov models include Discrete Time Markov Chains (DTMC), which are the simplest Markovian model, where transitions between states happen at discrete time steps.Continuous Time Markov Chains (CTMC), where the value associated with each outgoing transition from a state is intended not as a probability but as a parameter of an exponential probability distribution (transition rate).

Markov Decision Processes (MDP) which are an extension of DTMCs allowing multiple probabilistic behaviors to be specified as output of a state. These behaviors are selected non deterministically. The analytical solution techniques for Markov models differ according to the specific model and to the underlying assumptions (e.g., transient or non transient states, continuous versus discrete time, etc.).

For example, the evaluation of the stationary probability _s of a DTMC model requires the solution of a linear system whose size is given by the cardinality of the state space S. The exact solution of such a system can be obtained only if S is finite or when the matrix of transition probabilities has a specific form. A problem of Markov models, which similar evaluation models also face, is the explosion of the number of states when they are used to model real systems [18]. To tackle this problem, tool support (e.g., PRISM [72]) with efficient symbolic representations and state space reduction techniques [64], [73] like partial-order reduction, bi simulation-based lumping and symmetry reduction is required.

3 QOSMOS ARCHITECTURE

This section introduces the generic QoSMOS architecture of an adaptive service-based system, and describes its realization using existing tools and components. As QoSMOS extends existing service-based systems with the capability to adapt dynamically, we start by presenting the standard architecture of a service-based system.

A typical SBS consists of a composition of web services that are accessed remotely through a software application termed a workflow engine. Several services may provide the same functionality, often with different levels of performance and reliability, and at different costs.Example. We will illustrate the concepts introduced so far by presenting a service-based system for remote medical assistance taken from [10], [40].

This TeleAssistance (TA) system will be used as a running example throughout the rest of the paper, and its associated BPEL workflow is depicted in Fig. 2. The TA system incorporates the following abstract services:Alarm Service, which provides the operation sendAlarm,Medical Analysis Service, which provides the operation analyzeData, andDrug Service, which provides the operations changeDoses and change Drug.

The TA workflow starts executing as soon as a Patient (PA) enables the home device supplied by the TA provider, and this device invokes the start Assistance operation of the workflow. The workflow then enters an infinite loop whose iterations start with a “pick” activity that suspends the execution and waits for one of the following three messages: 1) vitalParamsMsg, 2) pButtonMsg, or 3) stopMsg. The first message contains the patient’s vital parameters, which are forwarded by the BPEL workflow to the Medical Laboratory service (LAB) by invoking the operation analyzeData.

The LAB is in charge of analyzing the data, and replies by sending a result value stored in a variable analysis Result. A field of the variable contains a value that can be change Drug, change Doses, or send Alarm. A send Alarm value triggers the intervention of a First-Aid Squad (FAS) comprised of doctors, nurses, and paramedics whose task is to visit the patient at home in case of emergency. To alert the squad, the TA workflow invokes the operation alarm of the FAS. The message pButtonMsg caused by pressing a panic button also generates an alarm sent to the FAS.

Finally, the message stopMsg indicates that the patient decided to cancel the TA service, deleting each pending invocation to the FAS service. Different providers could be involved in providing concrete implementations for the abstract services in the TA service-based system. For example, we will consider that the Alarm Service and the Medical Analysis Service are implemented by n_1 ¼ 3 and n_2 ¼ 5 telecommunication operators, respectively—each such concrete service being provided with different cost, performance and reliability characteristics. Finally, we will consider that a single, in-house implementation of the Drug Service is available (i.e., n_3 ¼ 1).

3.1 Generic Architecture of QoSMOS

As illustrated in Fig. 3, QoSMOS augments the standard SBS architecture with a component termed an autonomic manager. This component employs the autonomic computing monitor analyze-plan-execute (MAPE) loop [66], [56] to ensure that the SBS adapts continually in order to achieve a set of high-level, multi objective QoS requirements specified by its administrator. The four stages of the QoSMOS MAPE loop are described below.

3.1.1 Monitoring Stage

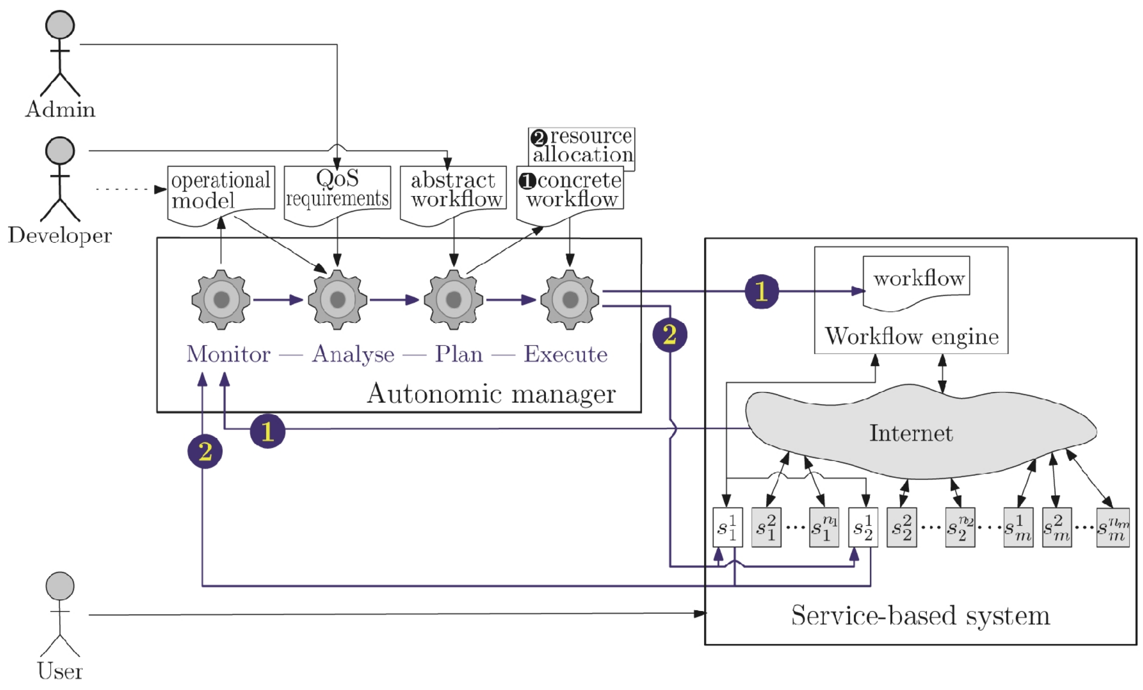
The first stage of the MAPE loop involves monitoring either or both of:

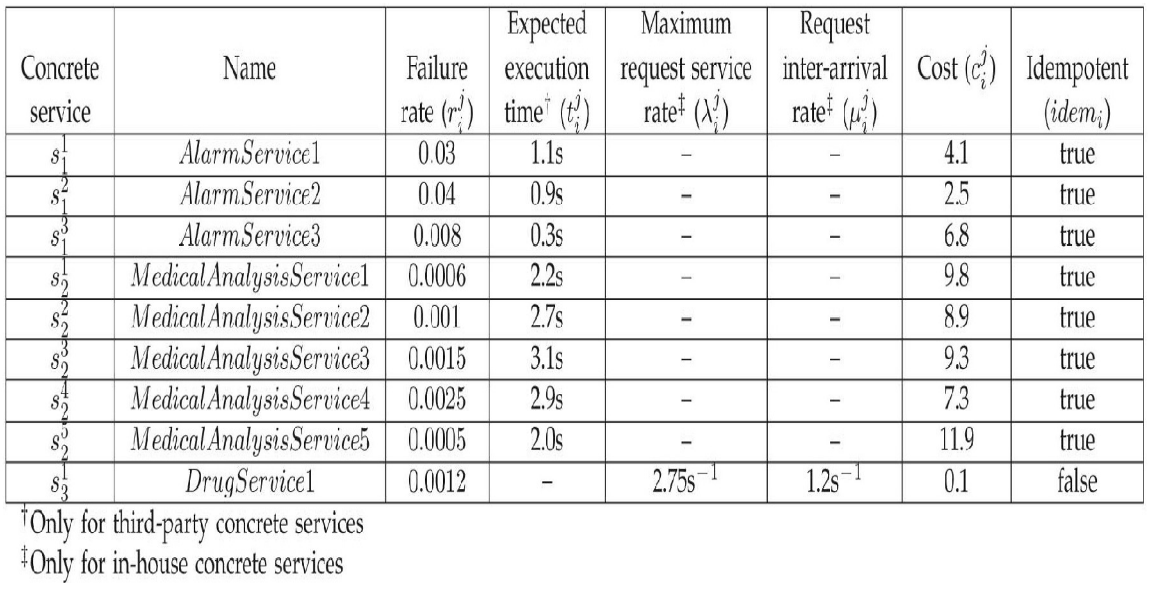
1. The performance (e.g., response time) and reliability (e.g., failure rate) of the SBS services. These parameters can be monitored for both in-house and third-party services.
2. The workload of individual concrete services (e.g., their request inter arrival rates) and the resources allocated to these services (e.g., CPU, memory, and bandwidth). Note that this is possible only for in-house services; these characteristics cannot be monitored for third-party services.

This information is used to build and/or to update an operational model of the SBS, an initial version of which can be provided by the developer of the service-based system. The model updates can happen periodically or when the monitor identifies significant changes in the parameters of the system. The types of operational models supported by the QoSMOS approach are those described earlier in Section 2.2, i.e., Markovian models. The maximum request service rate for this concrete, in-house service represents the request service rate when the service is allocated the maximum amount of CPU resources on the server(s) on which it is running.

3.1.2 Analysis Stage

The operational model from the monitoring stage is then employed to analyze the QoS requirements specified by the SBS administrator. The model is parameterized by the configurable parameters of the SBS, and this analysis step is intended to identify SBS configurations that satisfy the QoS requirements for the system. The analysis step includes a preprocessing step in which the QoS requirements specified by the SBS administrator in a high-level language are converted automatically into formally defined QoS requirements of the form presented in Section 2.1. Example. The high-level requirements for the TA service-based system from our running example are comprised of reliability and performancerelated requirements. Note that the reliability-related requirements take into account the fact that the average number of alarms associated with a particular patient throughout his or her utilization of the TA service-based system (i.e., the lifetime of the system) is 10





Concrete service	Name	Failure rate (r_i^j)	Expected execution time† (t_i^j)	Maximum request service rate‡ (λ_i^j)	Request inter-arrival rate‡ (μ_i^j)	Cost (c_i^j)	Idempotent ($idem_i$)
s_1^1	*AlarmService1*	0.03	1.1s	-	-	4.1	true
s_1^2	*AlarmService2*	0.04	0.9s	-	-	2.5	true
s_1^3	*AlarmService3*	0.008	0.3s	-	-	6.8	true
s_2^1	*MedicalAnalysisService1*	0.0006	2.2s	-	-	9.8	true
s_2^2	*MedicalAnalysisService2*	0.001	2.7s	-	-	8.9	true
s_2^3	*MedicalAnalysisService3*	0.0015	3.1s	-	-	9.3	true
s_2^4	*MedicalAnalysisService4*	0.0025	2.9s	-	-	7.3	true
s_2^5	*MedicalAnalysisService5*	0.0005	2.0s	-	-	11.9	true
s_3^1	*DrugService1*	0.0012	-	$2.75s^{-1}$	$1.2s^{-1}$	0.1	false

†Only for third-party concrete services
‡Only for in-house concrete services

3.1.3 Planning Stage

The planning stage of the QoSMOS MAPE loop uses the results of the analysis stage to build a plan for adapting the configuration of the SBS. The two types of adaptation made possible by the QoSMOS approach and implemented in the execution step of its MAPE loop are described below.

1. Adaptation through changing the workflow implemented by the service-based system. This type of adaptation is possible for all service-based systems considered by the QoSMOS framework, including those that employ third-party services. It requires that the SBS developer provide a workflow that is defined in terms of the abstract services needed to implement the intended SBS functionality, i.e., an abstract workflow.

   It is worth emphasizing that developing an abstract workflow is identical to developing a concrete workflow, minus the step in which the addresses of the concrete services to use are decided. This last step is carried out at runtime, when the analysis results are used to map the abstract services within this original workflow to concrete services—a process that takes place during the planning stage.

   Note that it is possible to restrict this adaptation to a subset of the workflow services by associating a single concrete service with each abstract service that does not belong to this subset. We actually envisage this as a common use case, and we will illustrate it by means of a number of experiments in Section 4.3. This use case is supported without having to specify in the abstract workflow which services should be considered for runtime adaptation and which services should always be implemented using the same concrete service.

2. Adaptation through modifying the resources allocated to individual services. When internally administered services are used to implement the SBS, it may be possible to adapt the resources allocated to these services in line with the variation in their workloads and in the QoS requirements for the system Potential applications of this type of adaptation include: achieving performance- related QoS requirements with minimal cost and environ-mental impact and achieving dependence QoS requirements by running services across a variable number of servers for redundancy purposes.

The mapping of abstract to concrete services within the QoSMOS architecture can be performed using one of the mapping patterns described below:

1. In a single mapping (SGL), a concrete service with suitable performance, reliability, and cost characteristics is used for the abstract service.

2. In sequential one-to-many mapping (SEQ), an abstract service is mapped to a sequence of concrete services. When the workflow is executed, these services are used one at a time, starting with the first service in the sequence and carrying on through the sequence until either a non erroneous response is obtained or all services in the sequence fail to respond success-fully. This concretization of an abstract service is useful for improving the reliability-related QoS of an SBS, but can elongate its response time. Note that the sequence of concrete services for an SEQ mapping pattern may include several instances of the same concrete service, or even a single concrete service to be invoked repeatedly for redundancy purposes.

3. Finally, in parallel one-to-many mapping (PAR), an abstract service is mapped to a set of concrete services, all of which are called during the execution of the workflow. This ensures that an increase in the reliability-related QoS metrics is obtained without impacting the SBS response time, but potentially at a higher cost the service response time varies linearly with the value assigned to this parameter, which can therefore be used to adapt the service behavior to its request arrival rate and to the system requirement

3.2.2 PRISM

PRISM [72], [74] is an open-source probabilistic model checker developed originally at the University of Birming-ham and currently supported and extended at the University of Oxford. The tool supports the analysis of a growing number of model types, including discrete and continuous-time Markov chains (DTMCs and CTMCs), Markov decision processes (MDPs), and extensions of these models with costs and rewards.The models to be analyzed are specified in the PRISM modeling language, which is based on the Reactive Modules formalism of Alur and Henzinger [2]. The proper-ties to be established are specified using PCTL (Probabilistic Computation Tree Logic) [53] for DTMCs and MDPs, and CSL (Continuous Stochastic Logic) [8] for CTMCs.

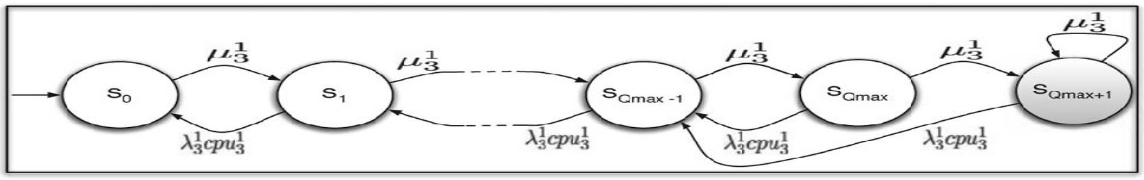
The tool works by first building a symbolic, MTBDD (multiterminal binary decision diagram) representation of the reachable state space of the analyzed model [72]. It then performs the analysis by induction over syntax, being capable of handling both bound properties—i.e., deciding whether a probability is above or below a specified threshold—and quantitative properties—i.e., calculating the actual probability of an event or the expectation for cost/ reward formulas. Particularly important for its integration in the QoSMOS architecture, PRISM supports the concept of experiments, which allows the automated analysis of several versions of a parameterized model. We will use this capability within the QoSMOS MAPE loop to automatically carry out the analysis of a range of possible configurations for a service-based system.

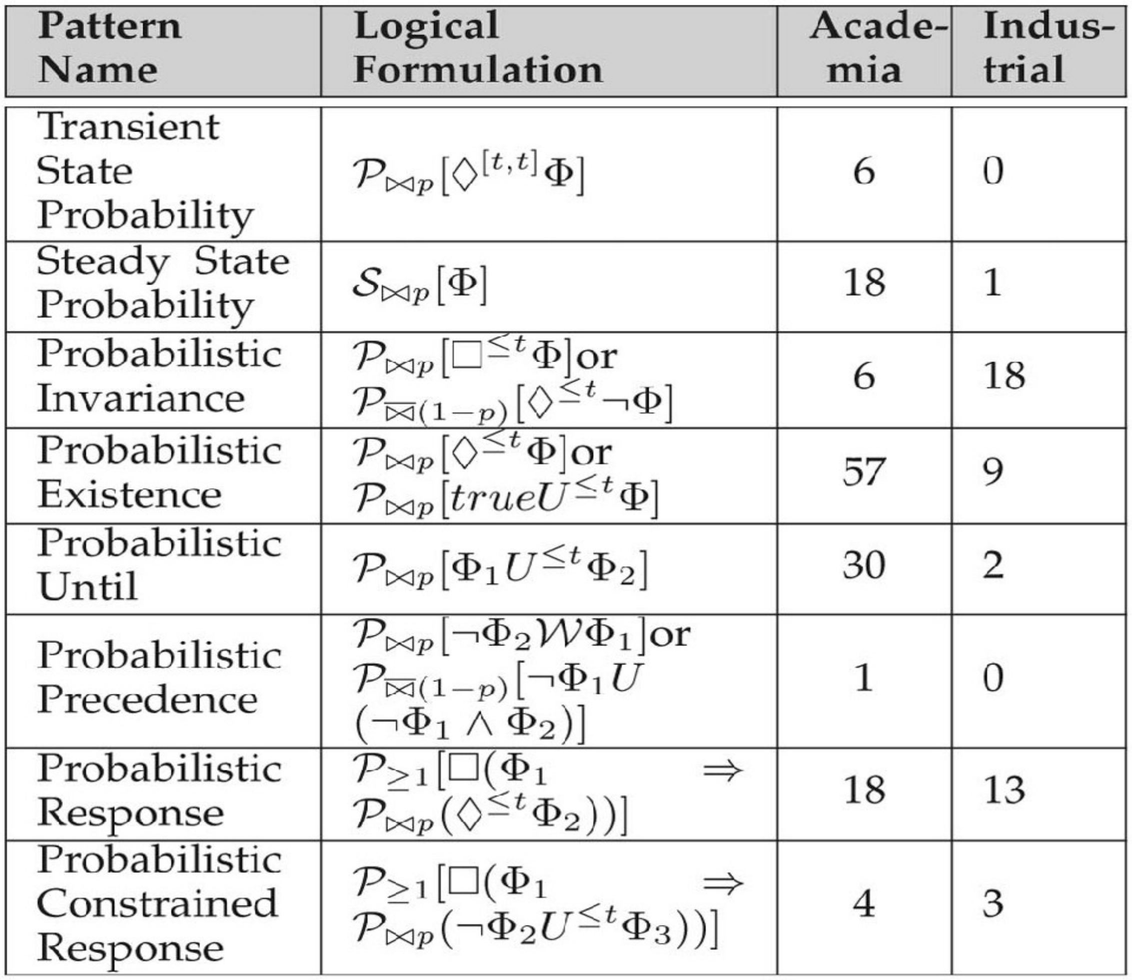
The model checking algorithms employed by PRISM involve a combination of graph-theoretical algorithms and numerical computation. The first type of algorithms operates on the underlying graph structure of the analyzed Markov model, e.g., to determine the reachable states within a model. Numerical computation (typically using iterative methods) is required for the solution of linear equation systems and the calculation of the transient probabilities of Markov chains.

The probabilistic model checker PRISM has been used in a large number of case studies that spawn application domains ranging from communication protocols and security systems to biological systems and dynamic power management. Many of these case studies are presented in detail on the PRISM website (www.prismmodelchecker. org). An extensive, independent performance analysis of a broad selection of probabilistic model checkers [61] ranked PRISM as the best tool for the quantitative analysis of large models such as the ones encountered in the adaptive service-based systems targeted by our QoSMOS work.First, the DTMC model depicted in Fig. 6 is used for the analysis required to achieve the reliability QoS requirements R_0 to R_3. This model follows the structure of the BPEL workflow, and assigns probabilities to branches and failure probabilities to service invocations (failures are represented by states highlighted in gray). Our approach relies on initial estimates for transition probabilities that come from domain experts and from monitoring previous versions of the system. Transition probabilities corresponding to service failure rates are unspecified in the DTMC model and represented by the unknown parameters a, b, and c because they depend on the mapping patterns and concrete services selected by the QoSMOS MAPE loop.

3.2.3 ProProST

To ease the formalization of QoS properties as required by the QoSMOS architecture, the idea of specification patterns [39], [68] has recently been investigated for probabilistic logics [49]. The outcome of an investigation of 152 proper-ties from academia and 48 properties from CTMC model for the in-house concrete service s^1_3.





Pattern Name	Logical Formulation	Acade-mia	Indus-trial
Transient State Probability	$\mathcal{P}_{\bowtie p}[\Diamond^{[t,t]}\Phi]$	6	0
Steady State Probability	$\mathcal{S}_{\bowtie p}[\Phi]$	18	1
Probabilistic Invariance	$\mathcal{P}_{\bowtie p}[\Box^{\leq t}\Phi]$or $\mathcal{P}_{\overline{\bowtie}(1-p)}[\Diamond^{\leq t}\neg\Phi]$	6	18
Probabilistic Existence	$\mathcal{P}_{\bowtie p}[\Diamond^{\leq t}\Phi]$or $\mathcal{P}_{\bowtie p}[trueU^{\leq t}\Phi]$	57	9
Probabilistic Until	$\mathcal{P}_{\bowtie p}[\Phi_1 U^{\leq t}\Phi_2]$	30	2
Probabilistic Precedence	$\mathcal{P}_{\bowtie p}[\neg\Phi_2 \mathcal{W}\Phi_1]$or $\mathcal{P}_{\overline{\bowtie}(1-p)}[\neg\Phi_1 U (\neg\Phi_1 \wedge \Phi_2)]$	1	0
Probabilistic Response	$\mathcal{P}_{\geq 1}[\Box(\Phi_1 \Rightarrow \mathcal{P}_{\bowtie p}(\Diamond^{\leq t}\Phi_2))]$	18	13
Probabilistic Constrained Response	$\mathcal{P}_{\geq 1}[\Box(\Phi_1 \Rightarrow \mathcal{P}_{\bowtie p}(\neg\Phi_2 U^{\leq t}\Phi_3))]$	4	3

3.3 QoSMOS Scalability

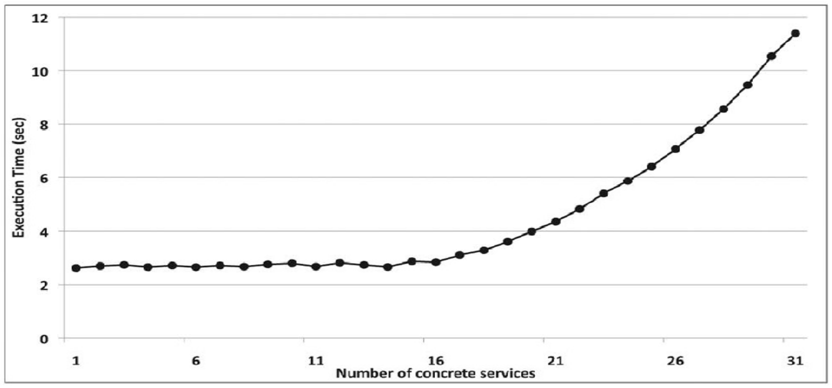
The main overhead of using the QoSMOS approach to add adaptiveness to a service-based system corresponds to the execution of the PRISM experiments in the analysis stage of the QoSMOS MAPE loop. All other operations performed by the QoSMOS autonomic manager including the monitoring of the system state and workload, updating the QoSMOS operational model, parsing the results of the PRISM experiments, and using these results to plan and enforce a new system configuration—take a negligible fraction of the overall MAPE loop processing time. For the QoSMOS-enabled TA system in our case study, each full PRISM evaluation of the PCTL and CSL properties associated with the QoS requirements R0 to R5 took between 2-3 milliseconds on a 2.4 GHz Intel Core 2 Duo server with4 GB of DDR3 RAM at 1067 MHz.

Given the number of possible configurations examined and the time spent in the communication steps between the QoSMOS components, the end-to-end execution of the MAPE loop and the adaptation of the SBS configuration to a new system state and workload can be completed in between 2.7-3.4 seconds. Note that this time represents the time required to react to changes in the system objectives, state and/or workload; it does not represent system downtime. Furthermore, this overhead does not need to be accommodated by a production server running one of the SBS components such as the BPEL workflow engine or one of the in-house concrete services.

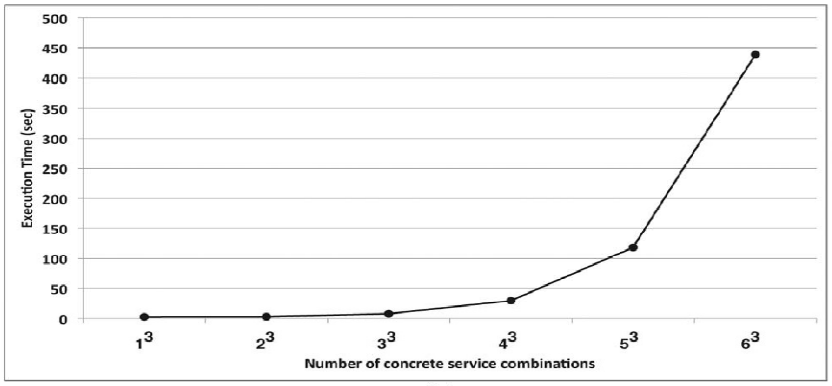
Instead, the GPAC autonomic manager employed by QoSMOS is itself a service-based system and can therefore be executed on a separate, management server. In this way, retrofitting adaptiveness to an existing SBS system can be done without modifying the original system or adding overheads to the physical servers that are used to execute its components. As these encouraging results were obtained for a service based system comprised of only three abstract services and nine associated concrete services, we carried out experiments to assess the scalability of the QoSMOS approach for service based systems comprising larger numbers of services. We first considered scenarios involving the original three service abstract TeleAssistance workflow and larger sets of concrete services.

The increases in the MAPE loop execution time for two such scenarios are presented shows the MAPE loop execution time required for gradually increasing sizes of the set of concrete services implementing the AlarmService (i.e., CS1). The size of the other concrete service sets (i.e., sets CS2 and CS3 implementing the Medical Analysis Service and the Drug Service, respectively) were maintained at the values from Table 2. As expected, the MAPE loop execution time grows exponentially due to the background quantitative model checking from the analysis stage. However, the execution time does not exceed five seconds for CS1 sizes of up to 22 concrete services, which is well over the number of concrete alarm services that can be expected for our case study.

When the sizes of all concrete service sets were increased at the same time, the execution overheads were observed for the QoSMOS MAPE loop. These results suggest that the QoSMOS approach can be used for systems of similar size to the TA SBS with sets of up to four concrete services for each abstract service (MAPE loop execution time under 30 seconds) or even up to five concrete services of similar size to the TA SBS with sets of up to four concrete services for each abstract service (MAPE loop execution time under 30 seconds) or even up to five concrete services for each abstract service (MAPE loop execution time under 2 minutes). One way to accommodate larger sets of concrete services is to preselect and use within the QoSMOS service based system subsets of three to five concrete services that are most likely to be useful based on criteria such as cost or provider trustworthiness.



(a)



(b)

QoSMoS scalability with the number of concrete

This preselection can be done periodically, either by the SBS administrator or by another instance of the QoSMOS MAPE loop. One last set of experiments that we present in this section involves examining the scalability of the QoSMOS framework for larger service-based systems. To perform these experiments, we increased the size of the abstract TA workflow by considering that the medical analysis part of the workflow requires the sequential execution of several services, each of which performs one part of the analysis. In order to choose a realistic range of workflow sizes, we first carried out a study of the SBS development platform Taverna [57] Taverna is widely used in the development of scientific workflows in application domains, including bioinformatics, chemoinformatics, astronomy, and social sciences.

Our study focused on the Taverna workflows with the tag “bioinformatics” and with a download count of 50 or more from the Taverna workflow repository my Experiment [85]. We selected this particular set of workflows because it represents the most used set of workflows from an application domain in which the Taverna platform is used regularly. Out of the 28 workflows in this set, 13 are comprised of five services or less, seven are comprised of between six and eight services, five are comprised of 10 services, two have 11 services, and one consists of 32 services.Wetherefore focused our experiments on extensions of theTAabstract workflow of similar size to these Taverna workflows. Loop for TA workflow variants comprised of up to 13 additional abstract medical services (i.e., up to 16 abstract services in total).

In all experiments, we considered that the sets of concrete services for all but the first three abstract services contained a single concrete service, i.e., adaptation was applied only to the original abstract services. The experiments were run for three adaptation scenarios, namely, when sets of two, three, and four concrete services, respectively, were available for each abstract service for which QoSMOS adaptation was employed. The results indicate that QoSMOS-based adaptation can be applied to workflows comprised of up to 16 abstract services, with overheads of under four seconds in the first scenario, under 20 seconds in the second scenario, and up to two minutes in the last scenario.

When more than one concrete service is available for every abstract service within the QoSMOS based workflow (Fig. 18), the workflow sizes for which the QoSMOS MAPE loop completes within 140 seconds are: eight—when that these experiments cover over 71 percent of the Taverna workflows from the study described above (i.e., 20 out of 28 workflows), which we consider a good result for the QoSMOS prototype realization and this scenario in which the adaptation is applied to every single component of the service-based system.

Furthermore, remember that the SBS objectives in our case study consist of no less than six QoS requirements, each of which brings an almost equal contribution to the execution times obtained for the experiments in this section. Adaptation in service-based systems with less complex SLAs can be achieved with significantly lower overheads. There are several options that we are investigating in our effort to increase the scalability of QoSMOS.

4.CONCLUSIONS AND FUTURE WORK

In this paper, we have presented QoSMOS, a tool-supported framework for QoS management of self-adaptive service based systems. QoSMOS defines and implements an autonomic architecture that combines formal specification of QoS requirements, model-based QoS evaluation, monitoring and parameter adaptation of the QoS models, and planning and execution of system adaptation.

The proposed framework has been built through the integration of extended versions of existing tools and components developed by the authors. Essential strengths of QoSMOS are the use of a precise and formal specification of QoS requirements with probabilistic temporal logics and the definition of a model-based quality evaluation methodology for probabilistic QoS attributes taking into account quality dependencies on other services and on the operational profile.

The monitoring phase of QoSMOS and the consequent possible online update of the quality models allow discovering requirements violations and triggering adaptation strategies for the SBS. The possible strategies are based on techniques for service selection, runtime reconfiguration, and resource assignment to in-house managed services. Furthermore, the quality models in QoSMOS represent the overall system architecture, so it is possible to detect requirement violations generated by different causes and not only related to unexpected behaviors associated with single services of theSBS (e.g., unexpected variations in the usage profile).

The validation of the proposed framework has been performed through the application of QoSMOS capabilities to a common case study of a service-based system for remote medical assistance. The results obtained with a high number of numerical experiments and simulations proved the effectiveness of our solution. On the other hand, we have to also acknowledge some limitations that should be considered when selecting the QoSMOS framework. One limitation of the QoSMOS framework is that, due to the statistical methods behind the monitoring and QoS analysis, it is hard to deal with models that contain extreme probabilities.

As an example, with a Bayesian filter it would require an unfeasibly large number of observations to change the value of a transition probability. Additionally, we acknowledge that the quality evaluation with our more realistic modelbased QoS models and probabilistic verification can take longer than the quality evaluation with simple aggregation functions. Consequently, there is a trade-off between the improved accuracy of our QoS evaluation compared to the existing approaches and the time needed to obtain these results.

For most practical service-based systems where QoSMOS was applied the time efficiency was not a problem. However, when dealing with a workflow with several thousand services and multiple parameters, a very long time could be necessary to get a result of the quality evaluation. Furthermore, our approach currently only applies to probabilistically quantifiable and externally observable QoS properties, such as reliability, availability, and performance. Due to the underlying techniques for the adaptation and planning procedures, an application to qualitative nonquantifiable QoS properties is currently not possible.

Besides working on the above-mentioned limitations, our future work will consist of refining the QoSMOS approach by investigating its range of applicability. We plan to enrich the ongoing implementation by: enlarging the set of supported models (e.g., Markov Decision Processes, etc.),integrating black-box monitoring techniques [51], and defining a language aimed at managing multimodel consistency. Additionally, it would be interesting to explore and extend other QoS specification formalisms (such as, for example, ALBERT [10] or probabilistic and timed MSCs [58], [90]) and map them into the ProProST pattern system and the provided structured English grammar. ACKNOWLEDGMENTS This work was partly supported by the FP7 European projects CONNECT (FP7 231167), Q-ImPrESS (FP7 215013), and IDEAS-ERC SMScom (227977), and by the UK Engineering and Physical Sciences Research Council Grants EP/F001096/1 and EP/H042644/1.

REFERENCES:

[1] R. Alur, C. Courcoubetis, and D. Dill, "Model-Checking in Dense Real-Time," Information and Computation, vol. 104, no. 1, pp. 2- 34,1993.
[2] R. Alur and T.A. Henzinger, "Reactive Modules," Formal Methods in System Design, pp. 207-218, IEEE CS Press, 1999.
[3] D. Ardagna, C. Ghezzi, and R. Mirandola, "Rethinking the Use of Models in Software Architecture," Proc. Fourth Int'l Conf. Quality of Software-Architectures, pp. 1-27, 2008.
[4] D. Ardagna and B. Pernici, "Global and Local QoS Constraints Guarantee in Web Service Selection," Proc. IEEE Int'l Conf. Web Services, pp. 805-806, 2005.
[5] D. Ardagna and B. Pernici, "Adaptive Service Composition in Flexible Processes," IEEE Trans. Software Eng., vol. 33, no. 6, pp. 369-384, June 2007.
[6] A. Aziz, V. Singhal, and F. Balarin, "It Usually Works: The Temporal Logic of Stochastic Systems," Proc. Seventh Int'l Conf. Computer Aided Verification, P. Wolper, ed., pp. 155-165, 1995.
[7] E. Badidi, L. Esmahi, and M.A. Serhani, "A Queuing Model for Service Selection of Multi-Classes QoS-Aware Web Services," Proc. IEEE Int'l Conf. Web Services, pp. 204-213, 2005.
[8] C. Baier, J.-P. Katoen, and H. Hermanns, "Approximate Symbolic Model Checking of Continuous-Time Markov Chains," Proc. 10th Int'l Conf. Concurrency Theory, J.C.M. Baeten and S. Mauw, eds., pp. 146-161, 1999.

[9] L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini, “A Timed Extension of WSCoL,” Proc. IEEE Int’l Conf. Web Services, pp. 663-670, 2007.

[10] L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini, “Validation of Web Service Compositions,” IET Software, vol. 1, no. 6, pp. 219-232, Dec. 2007.

[11] L. Baresi and S. Guinea, “Towards Dynamic Monitoring of WS-BPEL Processes,” Proc. Third Int’l Conf. Service Oriented Computing, 2005.

[12] L. Baresi, E.D. Nitto, and C. Ghezzi, “Toward Open-World Software: Issue and Challenges,” Computer, vol. 39, no. 10, pp. 36-43, Oct. 2006.

[13] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, “Heuristics for QoS-Aware Web Service Composition,” Proc. IEEE Int’l Conf. Web Services, pp. 72-82, 2006.

[14] J.O. Berger, Statistical Decision Theory and Bayesian Analysis, second ed. Springer, 1985.

[15] C. Bettini, D. Maggiorini, and D. Riboni, “Distributed Context Monitoring for the Adaptation of Continuous Services,” World Wide Web, vol. 10, no. 4, pp. 503-528, 2007.

[16] A. Bianco and L. de Alfaro, “Model Checking of Probabilistic and Nondeterministic Systems,” Proc. 15th Conf. Foundations of Software Technology and Theoretical Computer Science, P.S.Thiagarajan, ed., pp. 499-513, 1995.

[17] D. Bianculli and C. Ghezzi, “Monitoring Conversational Web Services,” Proc. Second Int’l Workshop Service Oriented Software Eng., pp. 15-21, 2007.

[18] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi, Queuing Network and Markov Chains. John Wiley, 1998.

[19] B. Boonea, S. Van Hoeckea, G. Van Seghbroecka, N. Jonckheereb, V. Jonckersb, F.D. Turcka, C. Develdera, and B. Dhoedta, “SALSA: QoS-Aware Load Balancing for Autonomous Service Brokering,” J. Systems and Software, vol. available online, p. in print, 2010.

[20] R. Calinescu, “General-Purpose Autonomic Computing,” Autonomi Computing and Networking, M.K. Denko, L.T. Yang, and Y. Zhang, eds., pp. 3-30, Springer, 2009.

[21] R. Calinescu, “Reconfigurable Service-Oriented Architecture for Autonomic Computing,” Int’l J. Advances in Intelligent Systems, vol. 2, no. 1, pp. 38-57, June 2009

[22] R. Calinescu and M. Kwiatkowska, “CADS*: Computer-Aided Development of Self-* Systems,” Fundamental Approaches to Software Eng., M. Chechik and M. Wirsing, eds., pp. 421-424, Springer, Mar. 2009.

[23] R. Calinescu and M.Z. Kwiatkowska, “Using Quantitative Analysis to Implement Autonomic IT Systems,” Proc. 31st Int’l Conf. Software Eng., pp. 100-110, 2009.

[24] G. Canfora, M. Di Penta, R. Esposito, and M.L. Villani, “An Approach for QoS-Aware Service Composition Based on Genetic Algorithms,” Proc. Conf. Genetic and Evolutionary Computation,pp. 1069-1075, 2005.

[25] G. Canfora, M.D. Penta, R. Esposito, and M.L. Villani, “QoSAware Replanning of Composite Web Services,” Proc. IEEE Int’l Conf. Web Services, pp. 121-129, 2005.

[26] G. Canfora, M.D. Penta, R. Esposito, and M.L. Villani, “A Framework for QoS-Aware Binding and Re-Binding of Composite Web Services,” J. Systems and Software, vol. 81, no. 10, pp. 1754- 1769, 2008.

[27] L. Cao, J. Cao, and M. Li, “Genetic Algorithm Utilized in Cost- Reduction Driven Web Service Selection,” Proc. Int’l Conf. Computational Intelligence and Security, Y. Hao, J. Liu, Y. Wang, Y.M. Cheung, H. Yin, L. Jiao, J. Ma, and Y.-C. Jiao, eds., pp. 679- 686, 2005.

[28] V. Cardellini, E. Casalicchio, V. Grassi, F. Lo Presti, and R. Mirandola, “QoS-Driven Runtime Adaptation of Service Oriented Architectures,” Proc. European Software Eng. Conf. and ACM SIGSOFT Symp. Foundations of Software Eng., pp. 131- 140, 2009.

[29] V. Cardellini, E. Casalicchio, V. Grassi, and R. Mirandola, “A Framework for Optimal Service Selection in Broker-Based Architectures with Multiple QoS Classes,” Proc. Services Computing Workshops, pp. 105-112, 2006.

[30] V. Cardellini, E. Casalicchio, V. Grassi, and F.L. Presti, “Scalable Service Selection for Web Service Composition Supporting Differentiated QoS Classes,” Technical Report RR-07.59, Dip. Di Informatica, Sistemi e Produzione, Univ. di Roma Tor Vergata, 2007.

[31] G. Chafle, P. Doshi, J. Harney, S. Mittal, and B. Srivastava, “Improved Adaptation of Web Service Compositions Using Value of Changed Information,” Proc. IEEE Conf. Web Services, pp. 784- 791, 2007.

[32] B.H.C. Cheng, H. Giese, P. Inverardi, J. Magee, and R. de Lemos, “08031—Software Engineering for Self-Adaptive Systems: A Research Road Map,” Software Eng. for Self-Adaptive Systems, Springer-Verlag, 2008.

[33] F. Ciesinski and M. Gro¨ ßer, “On Probabilistic Computation Tree Logic,” Validation of Stochastic Systems—A Guide to Current Research, C. Baier, B.R. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, eds., pp. 147-188, Springer, 2004.

[34] A. Clark, S. Gilmore, J. Hillston, and M. Tribastone, “Stochastic Process Algebras,” Seventh Int’l School Formal Methods, pp. 132- 179, Springer, 2007.

[35] M. Colombo, E. Di Nitto, M. Mauri, “Scene: A Service Composition Execution Environment Supporting Dynamic Changes Disciplined through Rules,” Lecture Notes in Computer Science, vol. 4294, pp. 191-202, Springer, 2006.

[36] R. Colvin, L. Grunske, and K. Winter, “Probabilistic Timed Behavior Trees,” Proc. Int’l Conf. Integrated Formal Methods, J. Davies and J. Gibbons, eds., pp. 156-175, 2007.

[37] R. Colvin, L. Grunske, and K. Winter, “Timed Behavior Trees for Failure Mode and Effects Analysis of Time-Critical Systems,” J. Systems and Software, vol. 81, no. 12, pp. 2163-2182, 2008.

[38] A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef, “Web Services on Demand: WSLA-Driven Automated Management,” IBM Systems J., vol. 43, no. 1, pp. 136-158, 2004.

[39] M.B. Dwyer, G.S. Avrunin, and J.C. Corbett, “Property Specification Patterns for Finite-State Verification,” Proc. 21st Int’l Conf. Software Eng., pp. 411-420, 1999.

[40] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli, “Model Evolution by Runtime Parameter Adaptation,” Proc. 31st Int’l Conf. Software Eng., pp. 111-121, 2009.

[41] R. Frei, G.D.M. Serugendo, and J. Barata, “Designing Self- Organization for Evolvable Assembly Systems,” Proc. Second IEEE Int’l Conf. Self-Adaptive and Self-Organizing Systems, pp. 97-106, 2008.

[42] S. Frolund and J. Koistinen, “Quality-of-Service Specification in Distributed Object Systems,” Distributed Systems Eng. J., vol. 5, no. 4, pp. 179-202, Dec. 1998.

[43] S. Gallotti, C. Ghezzi, R. Mirandola, and G. Tamburrelli, “Quality Prediction of Service Compositions through Probabilistic Model Checking,” Proc. Fourth Int’l Conf. Quality of Software-Architectures, S. Becker, F. Plasil, and R. Reussner, eds., pp. 119-134, 2008.

[44] R. Garcia, J. Jarvi, A. Lumsdaine, J. Siek, and J. Willcock, “A Comparative Study of Language Support for Generic Programming,” ACM SIGPLAN Notices, vol. 38, no. 11, pp. 115-134, Nov. 2003.

[45] C. Ghezzi and G. Tamburrelli, “Predicting Performance Properties for Open Systems with KAMI,” Proc. Fifth Int’l Conf. Quality of Software Architectures, pp. 70-85, 2009.

[46] S. Gilmore and J. Hillston, “The PEPA Workbench: A Tool to Support a Process Algebra-Based Approach to Performance Modelling,” Proc. Seventh Int’l Conf. Computer Performance Evaluation, Modeling Techniques and Tools, G. Haring and G. Kotsis, eds., pp. 353-368, 1994.

[47] V. Grassi, “Architecture-Based Reliability Prediction for Service- Oriented Computing,” Proc. Workshop Architecting Dependable Systems, pp. 279-299, 2004.

[48] V. Gruhn and R. Laue, “Patterns for Timed Property Specifications,” Electronic Notes in Theoretical Computer Science, vol. 153, no. 2, pp. 117-133, 2006.

[49] L. Grunske, “Specification Patterns for Probabilistic Quality Properties,” Proc. 30th Int’l Conf. Software Eng., Robby, ed., pp. 31-40, 2008.

[50] L. Grunske, K. Winter, and R. Colvin, “Timed Behavior Trees and Their Application to Verifying Real-Time Systems,” Proc. Australian Software Eng. Conf., pp. 211-222, 2007

[51] L. Grunske and P. Zhang, “Monitoring Probabilistic Properties,” Proc. Seventh Joint Meeting of the European Software Eng. Conf. and the ACM SIGSOFT Int’l Symp. Foundations of Software Eng., H. VanVliet and V. Issarny, eds., pp. 183-192, 2009.

[52] H. Guo, J. Huai, H. Li, T. Deng, Y. Li, and Z. Du, “ANGEL: Optimal Configuration for High Available Service Composition,” IEEE Int’l Conf. Web Services, pp. 280-287, 2007.

[53] H. Hansson and B. Jonsson, “A Logic for Reasoning about Time and Reliability,” Formal Aspects of Computing, vol. 6, no. 5, pp. 512- 535, 1994.

[54] D. Harel and R. Marelly, “Playing with Time: On the Specification and Execution of Time-Enriched LSCs,” Proc. 10th Int’l Workshop Modeling, Analysis, and Simulation of Computer and Telecomm. Systems, pp. 193-202, 2002.

[55] J. Harney and P. Doshi, “Speeding Up Adaptation of Web Service Compositions Using Expiration Times,” Proc. Int’l Conf. World Wide Web, pp. 1023-1032, 2007.

[56] M.C. Huebscher and J.A. McCann, “A Survey of Autonomic Computing—Degrees, Models, and Applications,” ACM Computing Surveys, vol. 40, no. 3, pp. 1-28, 2008.

[57] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn, “Taverna: A Tool for Building and Running Workflows of Services,” Nucleic Acids Research, vol. 34, no. Web Server issue, pp. 729-732, July 2006.

[58] ITU-TS, “ITU-TS Recommendation Z.120: Message Sequence Chart 1999 (MSC99),”technical report, Int’l Telecomm. Union (ITU-TS), 1999.

[59] D.N. Jansen, H. Hermanns, and J.-P. Katoen, “A Probabilistic Extension of UML Statecharts,” Proc. Seventh Int’l Symp. Formal Techniques in Real-Time and Fault-Tolerant Systems, W. Damm and E.-R. Olderog, eds., pp. 355-374, 2002.

[60] D.N. Jansen, H. Hermanns, and J.-P. Katoen, “A QoS-Oriented Extension of UML Statecharts,” Proc. Sixth Int’l Conf.—The Unified Modeling Language. Model Languages and Applications,P. Stevens, J. Whittle, and G. Booch, eds., pp. 76-91, 2003.

[61] D.N. Jansen, J.-P. Katoen, M. Oldenkamp, M. Stoelinga, and I.Zapreev, “How Fast and Fat Is Your Probabilistic Model Checker? An Experimental Comparison,” Hardware and Software: Verification and Testing, K. Yorav, ed., pp. 69-85, Springer, 2008.

[62] M.B. Juric, B. Mathew, and P. Sarang, Business Process Execution Language for Web Services. Packt Publishing, 2004.

[63] D. Karastoyanova and F. Leymann, “BPEL‘n’Aspects: Adapting Service Orchestration Logic,” Proc. IEEE Int’l Conf. Web Services, pp. 222-229, 2009.

[64] J.-P. Katoen, T. Kemna, I.S. Zapreev, and D.N. Jansen, “Bisimulation Minimisation Mostly Speeds Up Probabilistic Model Checking,” Proc. Int’l Conf. Tools and Algorithms for the Construction and Analysis of Systems, O. Grumberg and M. Huth, eds., pp. 87-101, 2007.

[65] A. Keller and H. Ludwig, “The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services,” J. Network and Systems Management, vol. 11, no. 1, pp. 57-81, 2003.

[66] J.O. Kephart and D.M. Chess, “The Vision of Autonomic Computing,” Computer, vol. 36, no. 1, pp. 41-50, Jan. 2003.

[67] J.M. Ko, C.O. Kim, and I.-H. Kwon, “Quality-of-Service OrientedWeb Service Composition Algorithm and Planning Architecture,”J. Systems and Software, vol. 81, no. 11, pp. 2079-2090, 2008.

[68] S. Konrad and B.H.C. Cheng, “Real-Time Specification Patterns,”Proc. 27th Int’l Conf. Software Eng., G.-C. Roman, W.G. Griswold, and B. Nuseibeh, eds., pp. 372-381, 2005.

[69] R. Koymans, “Specifying Real-Time Properties with Metric Temporal Logic,” Real-Time Systems, vol. 2, no. 4, pp. 255-299, 1990.

[70] J. Kramer and J. Magee, “Self-Managed Systems: An Architectural Challenge,” Proc. Future of Software Eng., pp. 259-268, 2007.

[71] M.Z. Kwiatkowska, G. Norman, D. Parker, and J. Sproston, “Performance Analysis of Probabilistic Timed Automata Using Digital Clocks,” Formal Methods in System Design, vol. 29, no. 1, pp. 33-78, 2006.

[72] M.Z. Kwiatkowska, G. Norman, and D. Parker, “Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach,” Int’l J. Software Tools for Technology Transfer, vol. 6, no. 2, pp. 128-142, Aug. 2004.

[73] M.Z. Kwiatkowska, G. Norman, and D. Parker, “Symmetry Reduction for Probabilistic Model Checking,” Proc. 18th Int’l Conf. Computer Aided Verification, T. Ball and R.B. Jones, eds., pp. 234- 248, 2006.

[74] M.Z. Kwiatkowska, G. Norman, J. Sproston, and F. Wang, "Symbolic Model Checking for Probabilistic Timed Automata," Information and Computation, vol. 205, no. 7, pp. 1027-1077, 2007.
[75] D.D. Lamanna, J. Skene, and W. Emmerich, "Slang: A Language for Defining Service Level Agreements," Proc. Ninth IEEE Int'l Workshop Future Trends of Distributed Computing Systems, pp. 100- 107, 2003.
[76] E.D. Lazowska, J. Zahorjan, G.S. Graham, and K.C. Sevcik, Quantitative System Performance: Computer System Analysis Using Queueig Network Models. Prentice-Hall, 1984.
[77] Y. Ma and C. Zhang, "Quick Convergence of Genetic Algorithm for QoS-Driven Web Service Selection," Computer Networks, vol. 52, no. 5, pp. 1093-1104, 2008.
[78] M.A. Marsan, "Stochastic Petri Nets: An Elementary Introduction," Advances in Petri Nets, pp. 1-29, Springer, June 1989.
[79] M. Marzolla and R. Mirandola, "Performance Prediction of Web Service Workflows," Proc. Int'l Conf. Quality of Software Architectures, pp. 127-144, 2007.
[80] D.A. Menasce´, E. Casalicchio, and V. Dubey, "A Heuristic Approach to Optimal Service Selection in Service Oriented Architectures," Proc. Seventh Int'l Workshop Software and Performance, A. Avritzer, E.J. Weyuker, and C.M. Woodside, eds., pp. 13-24, 2008.
[81] D.A. Menasce´ and V. Dubey, "Utility-Based QoS Brokering in Service Oriented Architectures," Proc. Int'l Conf. Web Services, pp. 422-430, 2007.
[82] D.A. Menasce´, J.M. Ewin