SLA Translation in Multi-Layered Service Oriented Architectures: Status and Challenges

Hui Li, Wolfgang Theilmann, Jens Happe

April 17, 2009

Service-Oriented Architecture (SOA) represents an architectural shift for building business applications based on loosely-coupled services. In a multi-layered SOA environment the exact conditions under which services are to be delivered can be formally specified by Service Level Agreements (SLAs). However, typical SLAs are just specified at the top-level and do not allow service providers to manage their IT stack accordingly as they have no insight on how top-level SLAs translate to metrics or parameters at the various layers of the IT stack. This paper addresses the research problems in the area of SLA translation, namely, the correlation and mapping of SLA-related metrics and parameters within and across IT layers. We introduce a conceptual framework for precise definition and classification of SLA translations in SOA. With such a framework, an in-depth review and analysis of the state of the art is carried out by the category, maturity, and applicability of approaches and methodologies. Furthermore, we discuss the fundamental research challenges to be addressed for turning the vision of holistic and transparent SLA translation into reality.

1 Introduction

The paradigm of Service Oriented Architecture (SOA) has changed the way for building IT-based systems [7, 58]. Initially SOA was mainly applied to restructure the IT stack within an organization. More recently it also evolves as a common paradigm for cross-organizational service landscapes where services are considered as tradeable goods. Consequently, services operate under a strong business context where service customers can expect services to be provided under well-defined and dependable conditions and with clearly associated costs.

Service Level Agreements (SLAs) are a common way to formally specify the exact conditions (both functional and non-functional behavior) under which services are or shall be delivered. However, the current SLAs in practice are just specified at the top-level interface between a service provider and a service customer. Top-level SLAs can be used by customers and providers to monitor whether an actual service delivery complies with the agreed SLA terms. In case of SLA violations penalties or compensations can be directly derived.

However, top-level SLAs do not allow service providers to either plan their IT landscapes according to possible, planned or agreed SLAs. Nor they allow to understand why a certain SLA violation might have occurred. The reason for this is that SLA guarantee terms are not explicitly or directly related to actual performance metrics or configuration parameters. This makes it difficult for service providers to derive proper configuration parameters from top-level SLAs and to assess (lower-level) monitoring metrics against top-level SLAs. Overall, the missing relation between top-level SLAs and (lower-level) metrics and parameters is a major hurdle for managing IT stacks in terms of IT planning, prediction or adjustment processes and in accordance with possible, planned or actual SLAs.

Our vision is to use the paradigm of SLAs for managing a complete IT stack in correlation with top-level SLAs which are agreed at business level. This complies very well with the technical trend to apply the paradigm of service-orientation across the complete IT stack (infrastructure/platform/software as a service) but also with the organizational trend in IT companies to organize different departments as service departments (providing infrastructure resources, middleware, applications or composition tools as a service). SLAs will be associated with multiple elements of the stack at multiple layers, e.g. SLAs for elements of the physical infrastructure, virtualized infrastructure, middleware, application level and process-level. Such internal SLAs describe the contract between the lower-level entity and a higher-level entity which consumes the lower one. More precisely, the SLAs specify the required or agreed performance metrics but also the related configuration parameters.

The fundamental challenge for realizing this vision is how to properly correlate the different SLAs in such a scenario so that they form a well synchronized SLA hierarchy. This is basically a translation problem where clear translation rules are needed for translation of terms (metrics and parameters) between the different layers and elements. Obviously, neither a naive top-down nor a bottom-up approach alone can realize this. For example, throughput requirements might be translated in a top-down manner, assuming that sufficient behavior models of the system are available. In contrast to that, response times are more likely to be translated in a bottom-up manner. A comprehensive approach for SLA translation will have to combine those competing approaches in order to allow for the derivation of a comprehensive solution that satisfies all the various SLA terms at all layers.

The contribution of this paper is threefold. Firstly, we introduce a conceptual framework for a more precise definition of the problem of SLA translation. This includes a model for multi-layered SOA and a classification of four basic translation types. Secondly, we present a comprehensive review of the related work by the category of approaches and methodologies. We examine SLA translation with an integrated view from a multi-layered perspective, which we believe is crucial for understanding and classifying SLA translation problems in SOA. Thirdly, based on the analysis of the state of the art, we present the main research challenges ahead in order to achieve the sketched vision.

The rest of the paper is organized as follows: Section 2 sets the context of this paper by introducing a conceptual framework of multi-layered SOA and defining four main types of SLA translations. Section 3 presents a detailed review and analysis of technical solutions for SLA translations, categorized by the main methodologies. These include knowledge and rule based approaches, queuing-analytic model based approaches, and statistical learning based approaches. The optimization methods used for top-down translation are highlighted in these categories. Based on the review and analysis, Section 4 identifies and discusses several main challenges for future research. Concluding remarks are presented in the last section.

2 The Conceptual Framework

In order to enable a proper discussion and comparison in Section 3, we first introduce a conceptual framework comprising a SOA reference model, a definition of the concepts of SLAs and SLA translation as well as a categorization of four fundamental translation types.

There are a few reference models and architectures for SOA proposed by different standard bodies, such as OMG [56], OASIS [57], and SOA Alliance [67]. Similar elements are shared among these proposals, with different perspectives and details. In the context of this paper we use a reference architecture for SOA largely in line with OMG, which is closely related to its Model-Driven Architecture [55]. The target SOA architecture consists of four main layers, namely *Operational Resources*, *Software Components*, *Business Services*, and *Business Processes*. It is also possible that multiple sub-layers exist in one layer. Such a layered architecture [66, 11] has the advantages of clearly separating different levels of abstraction, and makes it easy to associate layers with roles. A layered view serves as an important basis for the discussion of SLA translation in this paper.

Firstly let us define what is a SLA in the context. A Service Level Agreement (SLA) refers to a part of a service contract between customers and service providers where the level of a service is formally defined [34]. In this paper the concept of SLA is broader than the traditional sense of customers and providers. Our SLA definition is in close correspondence with the layered SOA architecture sketched above. Each layer or sub-layer can have SLAs defined, sometimes referred as "sub-SLAs" or "OLAs (Operational Level Agreements)". SLAs can include both functional and non-functional properties (NFPs), where the latter is the main focus in this paper. NFPs can be categorized into *qualitative NFPs* (e.g. operational policies) and *quantitative NFPs* (e.g. response time and availability). Typically certain thresholds or targets are used to constrain a quantitative NFP, which can be referred as a "Service Level Objective (SLO)". Quantitative NFPs in SLOs can sometimes be referred as *metrics* or Key Performance Indicators

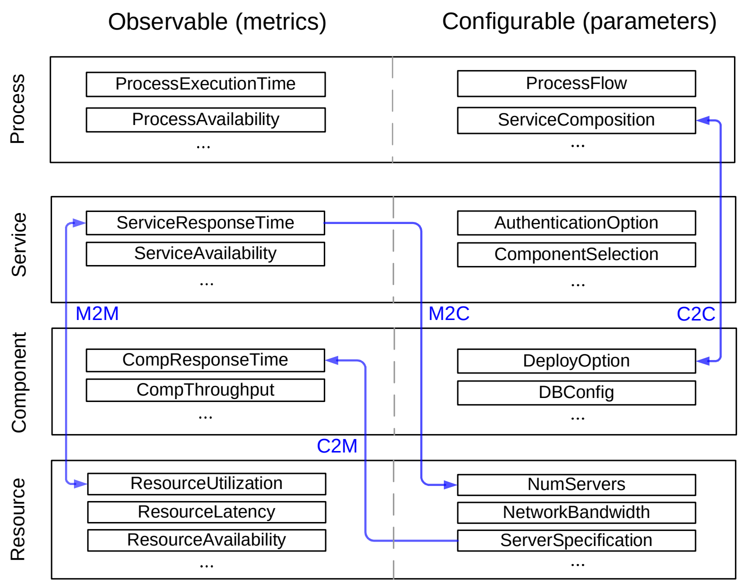


Figure 1: Observables (metrics) and configurables (parameters) in SOA layers, with different types of translations.

(KPIs). Most qualitative NFPs, along with some quantitative ones, are typically configurable and can be set as *parameters*. Different kinds of metrics and parameters exist at different layers in a multi-layered SOA environment. For example, the service layer might have "web service response time" as metric and "authentication method" as a parameter. The resource layer might have "number of cores" as metric and "network latency" as parameter. It is evident that these metrics and parameters at one layer or different layers are somehow correlated, but fully characterizing their relationships can be complex and remains as a challenging task. We treat such problems in a multi-layered SOA environment as "SLA translation".

Translation is a term with different views and interpretations in different research domains. In networking [78] policy translation is defined as "the transformation of a policy from a representation and/or level of abstraction, to another representation or level of abstraction". For instance, a high-level policy on Class of Service (CoS) can be mapped into low level configuration parameters on routers in a straightforward way [76]. In other domains such as computer systems, however, correlating a higher-level objective such as service response time with low-level operational parameters may involve sophisticated queuing-analytic models (e.g. [18]) and/or machine learning techniques (e.g. [20]). In this paper we define SLA translation as *any form of transformation of metrics and parameters, within one layer or from one (sub)layer to another in a multi-layered SOA environment.* Such a definition opens a broader perspective towards SLA translation problems in SOA. It makes it necessary to review the state of art in multiple domains (e.g. networking, computer systems, and software performance engineering), in multiple topic areas (e.g. QoS mapping, SLA decomposition, performance prediction, dependency analysis, fault diagnosis and control), and with different type of methodologies (statistics, queuing theory, optimization, and machine learning).

Based on the concepts of SOA layers and the distinction between metrics and parameters, we distinguish four main translation types. These are shown on an exemplary basis in Figure 1 together with some examples of metrics and parameters in different layers. The four types are:

- *C2C (Configuration to Configuration)*: this type of translation mostly relates to the dependencies within a layer or between layers. Such dependency graphs are useful in configuration management and problem diagnosis.

- *M2C (Metric to Configuration)*: this type of translation translates higher-level objectives to lower-level system parameters. It can also be referred as "top-down" translation or SLA decomposition. It is useful for sizing and capacity planning, mostly at design time.

- *C2M (Configuration to Metric)*: this type of translation predicts higher-level objectives from lower-level system parameters. It can also be referred as "bottom-up" translation or performance prediction. It is useful both what-if analysis at design time and predictive management at run time.

- *M2M (Metric to Metric)*: this types of translation correlates a high-level metric with lower-level metrics. The translation can go both directions, namely decomposition or prediction, depending on the usage scenario. It is useful for forecasting and problem diagnosis at run time.

Noteworthy that the depicted translations are just examples which might be actually composed of several lower level translations (e.g. M2M might consist of a sequence of M2M translations). We will see that most of the work under study can be categorized into one of these types. Therefore, this classification will be heavily used in the discussion of the next Section.

3 Methodologies for SLA Translation

In this section we review and analyze technical solutions by three main categories of methodologies, namely, *knowledge and rule based approaches, queuing-analytic model based approaches*, and *statistical learning based approaches.*

3.1 Knowledge and Rule Based Approaches

In knowledge and rule based approaches, domain-specific expert knowledge and rules are applied to make deductions or choices. The most visible related work to SLA translation are from the network domain. As is well known, the concept of Quality of Service (QoS) and SLA are firstly used in the telecommunication and networking community. The network QoS metrics, mostly quantitative, naturally appear in multiple layers and in different levels of abstraction. The translation problem is addressed either explicitly or implicitly, so it is worthwhile to review the work in the networking domain.

3.1.1 Policy Translation and QoS Mapping

The network traffic has shown strong self-similarity [46] and rich scaling behavior [1]. The realistic packet arrival process is not "smooth" as a Poisson process, instead, the interarrival time distribution can be heavy-tailed (amplitude burstiness) and the arrival rate can exhibit long range dependence (temporal burstiness). The practical implication of such characteristics is that, even when the average arrival rate is low, random bursts of traffic can be expected to saturate network devices. This indicates that packets should not be treated uniformly because different types of traffic have different QoS requirements. DiffServ is such a networking architecture that specifies a mechanism to provide different QoS guarantees on IP networks [12, 8]. It can, for instance, to provide low-latency, guaranteed service to real-time audio and video traffic while serving web traffic or file transfers with best-effort. DiffServ enable this via traffic classification and marking. A 6-bit value is encoded into the Type of Service (TOS) field of the IP header, which can be used to define traffic classes in a flexible way. The commonly-defined types are, for instance, default (best effort), expedited forwarding (low-loss and low-latency), and assured forwarding (different drop rate). DiffServ is simple, easy-to-implement, and scalable. Nevertheless, a main disadvantage is that how routers in different domains deal with the TOS field is somewhat arbitrary, making it difficult to predict end-to-end behavior.

In the DiffServ domain as long as the policy validation step is completed successfully, the translation process is straightforward [76]. Different users (replaced with IP addresses and port numbers) are associated with Class of Services (CoS). The high-level CoS policies are mapped into low-level network parameters such as delays, bandwidth rates, and DiffServ-specific details via *mapping tables*. The common practice is that an domain expert has pre-configured the mapping tables from CoS to different network levels. Such network level parameters can subsequently be used to configure devices.

There are efforts in HPC and Grid community to introduce the concepts of SLAs and service oriented science [31]. On production HPC centers, however, it is largely up to the site administrators to interpret higher level objectives and implement resource management policies based on experience. There are policy engines that provide good management capabilities, such as priority queues, fairshares, resource throttling, and advanced reservation [35]. Nevertheless, the implemented policies and configuration parameters are hand-tuned and sometimes ad-hoc. This results in poor performance time to time, especially given the burst behavior of job arrivals and job computation times [47, 68]. In such cases a historical knowledge base of configurations could be helpful by taking workload patterns into account.

Another translation-related topic is functional QoS mapping. A generic function maps a set of variables to another variable(s). Such a mapping can be derived in different ways, for instance, via a simple pre-determined analytic function [45], via statistic techniques such as regression analysis [63], via operation research methods [48], or via machine learning [32, 81]. Most of the research in networking aims at building pre-determined functions to map low-level network metrics to high-level QoS objectives, which is a simple and effective method given expert knowledge is in place.

In [45] it defines a set of QoS objectives (e.g. availability, latency), network performance metrics (e.g. RTT, $\#PacketLoss$), and associate them with evaluation functions. For example, $latency = f(RTT) = \sum RTT \ / \ \#TestPackets$. In [48] a mapping function is defined as $D = D_N + D_S + D_C$, where D is the response time, D_N, D_S, D_C are network latency, system delay and software delay, respectively. Queuing operational laws are used to correlate network latency with link bandwidth and router throughput. Similar problem is addressed in [73] where an experimentally built graphic function is used to map network latency and DB access time.

3.1.2 Semantic Translation and Rule-Based Engines

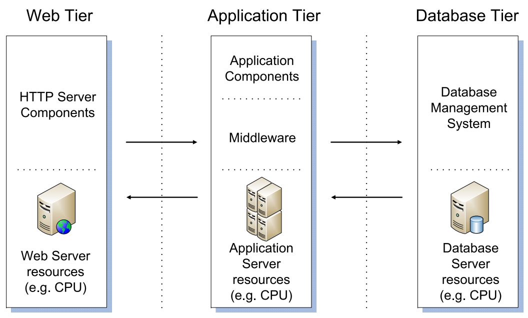
Wichadakul and Nahrstedt [79] presents a translation system for QoS-aware applications such as Video-on-Demand. The translation process can be divided into multiple phases. Firstly, The application components and their dependencies are specified by the developer, where each component is supported by a QoS profile. The QoS profile consists of QoS categories (e.g. performanceVideo) and their quantitative QoS dimensions (e.g. frame rate and resolution). A user-to-application template defines different QoS levels (e.g. high, medium, low) with specific categories and dimensions. Secondly, the application QoS requirements is translated into specific configurations of middleware implementations (e.g. CPU scheduling, bandwidth broker, real-time messaging), enabled by a middleware abstraction layer (MAL). Several methods are developed by the authors to address important issues:

1. A mapping rule engine is pre-defined to map application QoS requirements to genetic middleware services in MAL. For example, one rule is defined as "IF the label is 'real-time end-to-end multimedia transmission' THEN all components need local CPU broker service".

2. The mapping of generic middleware representation to possible configurations of middleware implementations is modeled as a constraint satisfaction problem. For instance, QoS provisions Q_{prov} by a configured middleware implementation satisfies QoS requirements Q_{req}.

3. A semantic-specific translation scheme is defined to map QoS dimensions to specific middleware implementation's semantics and expected parameters.

To sum up, building *rule-based knowledge bases* is the main approach for policy/SLA translation in networking. To facilitate such a process QoS information has to be categorized and differentiated. At the network level standards such as DiffServ exist for classification. At the application level, however, solutions have to be application-specific as described above [79]. Pre-determined analytic functions, if obtainable, can be used as rules for mapping, while sometimes semantic translations are necessary. Although all the information have to be pre-built by domain experts into the knowledge base, this approach remains as a basic yet effective method for enabling translations.



Web Tier
Application Tier
Database Tier
HTTP Server Components
Web Server resources (e.g. CPU)
Application Components
Middleware
Application Server resources (e.g. CPU)
Database Management System
Database Server resources (e.g. CPU)

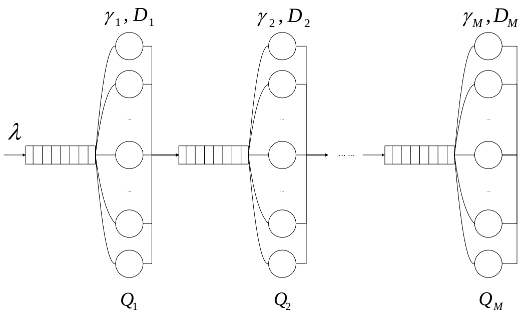


Figure 2: A typical 3-tier architecture for e-commerce applications

Figure 3: A queuing network model (QNM) for multi-tier applications

3.2 Queuing-Analytic Model Based Approaches

In some domains expert knowledge for pre-defined rules and mapping functions are not always manageable, or simply not possible. For example, there is no clearly definable functional relationship between web server response time and CPU load in all situations. Therefore correlating a high-level objective with low-level metrics or parameters is an active research field in computer systems, although different names may be called such as performance modeling, prediction, or autonomic management. In this section we focus on approaches based on analytic modeling, specifically referring to queuing network models. Before introducing the main queuing-analytic methodologies, we briefly discuss some approaches which do not strictly belong to this category but are of particular interest to SLA translation in the distributed systems domain.

Amazon's global web store has a decentralized service oriented infrastructure. In e-commerce sites at such a large scale, a page request typically requires a response constructed by sending requests to a large number of services. These services often have multiple dependencies and the call chain is across multiple layers. To ensure a clear bound on the page delivery at the application level, each and every dependency in the infrastructure needs to deliver its functionality with even tighter bounds. So the Amazon internal SLAs are expressed and measured at the 99.9 percentile of the distribution. For example, The read and write operations have latencies within 200ms for 99.9% of all the requests for a peak load of 500 requests per second. A storage system called Dynamo is designed to deliver such stringent SLAs by leveraging and combining well-established techniques from distributed systems research [25]. This is a typical bottom-up approach: high-level response times have a better chance to be guaranteed by ensuring tight bounds on low-level operations.

3.2.1 Queuing Network Models (QNM)

Performance modeling of web-based e-commerce applications has been an active research field for many years. As is shown in Figure 2, a typical e-commerce application consists of three tiers: a web tier for HTTP web page processing, an application tier for core application/middleware components, and a backend database tier for data storage. Each tier can be scaled to a cluster of servers if necessary. Queuing network models (QNM) are well-established analytic methods to model such systems [44, 52], as is shown in Figure 3. Mean Value Analysis (MVA) and variants as popular analytic solvers have been applied to multi-tier systems with session-based workloads [6, 17, 74]. Most of the MVA-based approaches have strict assumptions on distributions and do not consider burstiness/heavy-tail behavior, which is inherent in the workloads of such systems [51, 80].

3.2.2 Top-Down Translation Solved as a Constraint Satisfaction Problem

In this section we focus on discussing a QNM-based approach by Chen et al. [17, 18] that translates service level objectives (SLOs) into low-level system thresholds. Compared to the common bottom-up approaches for performance modeling and prediction, their solution is of particular interest in that it explicitly addresses the "SLA Decomposition" problem. It presents a typical top-down translation process from the average response time to the system resource thresholds in a multi-tier e-commerce environment. We use an open (request-based) workload to explain the translation process in detail because a close-form solution can be presented.

A QNM-based performance model establishes the relationship between application response time and workload characteristics, application configuration, and resource parameters. Consider an open queuing network model with the following notations:

1. N: no. of transaction types, i: index of transaction type
2. R: no. of resource types, j: index of resource type
3. M: no. of tiers, k: index of tier
4. λ_i: arrival rate of transaction type i
5. γ_k: number of servers at tier k
6. D_{ik}: service demand of transaction type i at tier k
7. U_{jk}: utilization of resource type j at tier k

Assuming even distribution of loads among servers at each tier, one tier with k servers can be modeled as k $M/M/1$ queues. As deduced in [18] the close-form solution for the average response time is

$$T_R = f(\gamma_1, ..., \gamma_M) = \sum_k \frac{\sum_i \frac{\lambda_i}{\gamma_k} * D_{ik} + \sum_j \frac{U_{jk}^2}{1-U_{jk}}}{\sum_i \frac{\lambda_i}{\gamma_k}}, \tag{1}$$

and the utilization of each resource is

$$U = D_0 + \sum_i D_i * \frac{\lambda_i}{\gamma}, \tag{2}$$

where D_0 is the background utilization of the resource. and D_i is the resource demand of transaction type i at the resource. Resource demands D_i are obtained by profiling based on historical data or benchmarking and the problem can be solved using statistical techniques such as linear regression.

The output of the SLA decomposition is a set of system-level parameters such as the number of servers in each tier, and the server specifications such as CPU and memory (contained in the profile).

The process of the SLA decomposition can be formulated as a *constraint satisfaction* problem. Given an open workload with transaction mixes and SLA objectives (e.g. $T_R = f(\gamma_1, ..., \gamma_M) < \theta$, $U_{cpu} < 50\%$), the problem is to find γ_k to satisfy such constraints. Constraint satisfaction problems on finite domains are typically solved using a form of search or an optimization technique [64]. For linear equations there are standard techniques such as linear programming that can be used as solvers. Constraint satisfaction algorithms as generic optimization methods can be combined with other models than queuing networks to address top-down SLA translation problems, in which SLOs can be treated as constraints.

Besides multi-transaction mixes, the technique presented in [18] introduces a multi-class MVA algorithm that can handle multiple customer classes (e.g. Gold, Silver, Bronze). Similar solution is also developed for closed (session-based) workloads. The limitation of the approach, as discussed above, is that only average response time is calculated and it is not able to handle bursty workloads. Nevertheless, the solution by Chen et al. provides a systematic methodology to translate high-level objectives to low-level policies.

3.2.3 Layered Queuing Network (LQN) Models

Layered queuing network models are firstly developed independently in [83] and [62], then a joint effort as a toolset [28]. The name “layers” is coined by Rolia and Sevcik [62], who introduced a heuristic MVA algorithm for hierarchically layered software systems. Woodside et al. [83, 82, 28] introduce the concept of tasks, entries, and phases in a layered approach, and it is able to handle parallelism and multiple scheduling disciplines such as priority queues. The main extension of LQN to ordinary queuing networks is that in LQN a server may become a client to other servers. This makes it possible for LQNs to explicitly model a system as layers of interacting software entities, which generate demands on the physical resources such as CPUs and disks. QNMs can only model software structures implicitly via service demands, as is shown in Section 3.2.1. Therefore LQNs reflect the structure of distributed (software) systems more naturally and they provide models that match the layered view of service-oriented architecture.

An example 3-tier e-commerce server platform modeled using LQNs is shown in Figure 4. We can see that in a 3-tier commerce server platform, “client”, “WebServer”, “AppServer” and “DBServer” can be structured in hierarchical layers. These servers (or tasks) are mapped onto physical resources such as CPUs and disks. It is possible to specify multiple entries and activity graphs in one layer such as “AppServer”, describing in detail its behavior. LQNs can be solved by both analytic solvers [82] and simula-

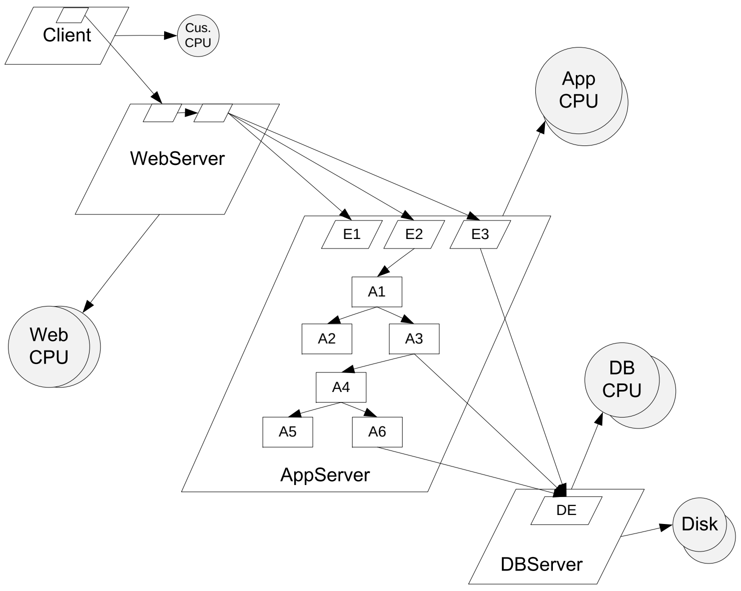


Figure 4: A layered queuing network (LQN) model for a typical 3-tier e-commerce server platform.

tions [28, 40]. Analytic solvers are limited to exponential distributions and mean value analysis (MVA). Recent work have been found on predicting system performance under bursty arrivals. Krishnamurthy et al. [42] proposes a hybrid approach that improve the accuracy of analytic performance models with bursts of user sessions. It exploits a Monte Carlo method that is able to estimate arbitrary distributions for workload parameters such as interarrival time, and combines it with both QNMs and LQNs. It is shown that a LQN-based approach has better accuracy than the classic QNM models.

Although conceptually LQNs are proven techniques to model layered system architecture and analyze its quantitative behavior, there are practical considerations that limit its usage. On one hand, software developers usually build system models that deal with functional aspects only. On the other hand, adhoc performance models are developed by performance experts, which are difficult to build and validate for complex software systems. A new paradigm that tries to bridge this gap is to annotate software models with performance data (e.g. UML [61], system scenarios [60]), and transform them into performance models such as LQN [5, 40]. Tools and platforms have been developed for facilitating this process [4, 5]. In [22] a framework is proposed to integrate a software model and a platform model for performance analysis via simulations. There are tradeoffs with different approaches. Analytical models are fast and inexpensive but have strict assumptions that limit its applicability. Simulations is general and more applicable, however, it is relatively slow and involves time-consuming sensitivity analysis.

3.3 Statistical Learning Based Approach

Analytic performance models are powerful mathematical tools but their practical usage may be limited by the simplified assumptions. Knowledge bases of "if-then" rules are themselves hard to build, maintain, and adapt to system changes. An alternative approach is to apply statistics and machine learning techniques [32, 81]. These techniques work on trace data to induce models and rules, assuming little or no domain knowledge. However, significant research efforts are needed to identify the most applicable statistical learning techniques and apply them in a way that can be deployed efficiently and effectively in practice. In this section we review several statistical learning based approaches that are applied successfully to problems related to SLA translation.

3.3.1 Probabilistic Models for Inference

Cohen et al. [20] applies a class of probabilistic models to induce the relationship between system-level metrics and service level objectives (SLOs) on a typical 3-tier e-commerce sever platform. SLOs are defined on high-level metrics such as response time or throughput of transactions. For instance, an SLO can be expressed as "response time should be less than 5 seconds for 90% of the requests, measured at a 10-minute interval". An SLO state S indicates whether an SLO is in compliance or violation, denoted as $S = \{s^+, s^-\}$. A set of n system-level metrics, denoted as $M = \{m_0, ...m_n\}$, are measured at application server (AS) or DB server, such as mean/variance of AS CPU time and mean DB swap space. The monitoring data are collected for $< S, M >$, which are used as the input training data set. The problem is to learn a classifier that can predict the SLO state S given a set of system metrics as input values.

The proposed probabilistic model is an extended naive Bayesian network called *tree-augmented naive Bayesian network* (TAN) [29], which is illustrated in Figure 5. A classic naive Bayesian classifier learns from the training data set the conditional probability of attributes L_i given the class label H $(Pr(L_i|H))$. Bayes rule is applied to compute the probability of H given L_i $(Pr(H|L_i))$ and classification is done by predicting the class with the highest posterior probability. In naive Bayesian classifiers all L_i are conditionally independent given class H. In TAN, as an addition, an edge from L_i to L_j is introduced besides H to show that the influence of attribute L_i on H also depends on L_j. As is shown in figure 5, each attribute L_i has one and only one augmenting edge from another attribute pointing to it. TAN imposes a directed acyclic graph among L_i so the relationships among metrics are captured. This is particularly useful in diagnosis as the TAN model becomes interpretable and expert knowledge can be incorporated.

Based on the technique presented above, Cohen et al. [21] shows how multiple machine learning techniques can be combined systematically in solving performance diagnosis problems. A first and important step is to manipulate the set of input attributes. One method is feature selection, namely, selecting a subset of the most relevant attributes for use in TAN. A directed search strategy such as greedy search can be used for feature selection. A more sophisticated technique is to transform the raw attributes/metrics into another representation, the so-called "*signatures*", which are more suitable for learning

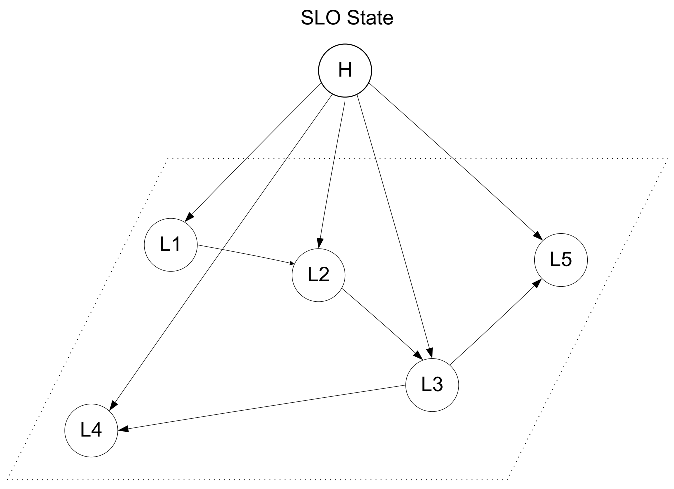


Figure 5: A tree-augmented naive Bayesian network (TAN). H is the high-level SLO state to be predicted/classified, $\{L_1, L_2, ..., L_5\}$ is a set of low-level attributes/system-level metrics.

and clustering. The signature construction process is as follows:

1. A given data trace is divided into epochs. For each epoch an ensemble of TAN models are learned and the ones with higher accuracy for estimating SLO violations are selected (one ensemble method called "bagging" [32]).

2. Corresponding to the raw metrics $M = \{m_i\}$, a vector of signatures $Sig = \{sig_i \subset (1, -1, 0)\}$ can be constructed: $sig_i = 1$ (*attributed*) if metric m_i is selected by the models and its value is "abnormal". $sig_i = -1$ (*not attributed*) if metric m_i is selected by the models and its value is "not abnormal". $sig_i = 0$ (*irrelevant*) if metric m_i is not selected.

It is proven that it is more effective to cluster signatures Sig than raw metrics M, using clustering techniques such as K-means or K-medians [81]. The center of the cluster can be considered as syndrome for the problem. Signatures can have annotations (e.g. root cause, solutions), which can be searched and retrieved for fast problem solving. In this approach we can see that the TAN model is combined with other machine learning techniques such as feature selection, ensemble methods, and information retrieval for a practical solution.

3.3.2 Constraint Satisfaction via Exhaustive Search

In [43] Kumar et al. proposes a TAN-based approach from deriving low-level metrics from high-level objectives. It introduces the system state represented by a set of variables V, which includes both a subset V_ϕ for high-level objectives (e.g. responseTime, accuracy), and a subset of low-level parameters V_τ such as cacheRefreshTime and allocatedServers

$(V_\phi \subset V, V_\tau \subset V)$. The SLA compliance or violation is defined using the high-level objectives:

$$\Gamma(V_\phi) = \prod_i \gamma(o_i, v^\phi_{\lambda(i)}), \tag{3}$$

where o_i represents the objectives such as operational range, $\lambda(i)$ is the mapping between o_i and variables in V_ϕ, $v^\phi_{\lambda(i)}$ is the objective-defined variable in V_ϕ, and γ is a boolean function which returns true if $v^\phi_{\lambda(i)}$ values conforms o_i.

The approach by Kumar et al. firstly partitions the data from system states V into a small number of sub-spaces that have a reduced set of controllable variables V_τ, by defining a specific distance function between variables in V. Then in each sub-space a TAN model is applied to determine the probability for SLA compliance/violation:

$$p = Pr(\Gamma(V_\phi)|V). \tag{4}$$

The suitable ranges of variables in V_τ are calculated using exhaustive search: an enumeration of possible values for V_i^τ are substituted into V and p is calculated when $\Gamma(V_\phi)$ is set to SLA compliance. If p is larger than a pre-defined threshold, the value is then recorded in a SLA-compliance set for low-level metrics.

To sum up, statistical learning techniques such as Bayesian networks has the advantages over queuing-analytic models in that they assume little domain knowledge and such black-box approaches are generic enough to apply to translation problems between arbitrary layers. However, it is assumed that certain amount of training data has to be available for learning and validation. It also takes substantial design and experimental efforts to apply such techniques successfully in systems.

3.3.3 Adaptability and Online Optimization

Most of the work we covered so far concern "static" or "offline" translation, namely, constructing or inducing a model that captures the relationship between high- and low-level metrics/parameters. We also briefly review some approaches that focus on more "dynamic" or "online" perspective of SLA-related management. Techniques such as control theory [27, 59] and reinforcement learning [69] are applied to implement online controllers that measure and keep high-level objectives in a (sub)optimal state by adaptively adjusting certain low-level system parameters according to the changing environment. Statistical signal processing techniques are frequently used as well. In [39] Koliver et al. introduce fuzzy-logic to represent QoS information and use filtering algorithms to adapt QoS parameters dynamically. In [87] Kalman filters are used to track and adapt performance model parameters in LQNs, making the model continuously updated over time. These methods are not covered in detail in the context of this paper.

In some cases a hybrid approach can achieve good results by getting the best of both worlds, such as the approach proposed in [36]. Comparing to the online algorithms for resource provisioning, it optimizes the configuration policies for different types of workloads offline using a discrete gradient-based search algorithm. Such (*workload, config*) pairs are then passed through a decision-tree learner to generate the rule sets, which in turn are used in run time dynamically according to a given workload at a given time. Utility functions are necessary components to be carefully defined in these optimization problems [15, 85]. And there are approaches proposed to deal with multiple conflicting objectives [75, 77].

3.3.4 Dependency and Critical Path Analysis

Statistical learning techniques are able to map a set of input variables to another set of variable(s) via probabilistic modeling and learning from data, and the system under study is considered as a black box. This can be considered as an advantage in one perspective, but may be a disadvantage in another perspective. Sometimes we may want to know the system structure in one more level of detail: the underlying components and their dependencies. In a layered view it also applies to the relationships between layers. As far as translation is concerned, it becomes clear once the dependencies among components are identified and the delays of components are calculated. To manage the complexity the component itself can be a black box. We refer such techniques following a "*gray box*" approach.

Firstly we look at the methodologies for dependency analysis in distributed systems. In [2] Aguilera et al. proposes an approach to infer *causal path* patterns from traces in distributed systems of black-box nodes. Data are obtained passively by tracing the messages between nodes, without knowledge on message semantics or nodes themselves. Two offline algorithms are developed to extract path information from traces. The first algorithm, called the "nesting" algorithm, is a heuristic method that combines all per-edge traces into a single global trace and individual traces are examined to determine how calls are nested. The output of the nesting algorithm is a call graph that shows the hierarchical structure of a particular causal path. The average latency of the nodes and the edges can also be calculated. Such information is useful in determining sub-SLAs for nodes in the system. The second algorithm makes use of a well-known signal processing technique called "convolution". In contrast to the nesting algorithm, the convolution algorithm separates the whole trace into per-edge traces, which in turn can be treated as time signals. The convolution method can then be used to find cross-correlations between signals. The result of convolution is a directed graph where nodes can appear multiple times. Comparing the two algorithms the intuitive path structure inferred by the nesting algorithm shows more potential to be used in SLA translation, although the accuracy of the nesting algorithm suffers from the increased parallelism in the trace. Other techniques for dependency analysis include static dependence model construction [37], which uses information from system-wide configuration repositories such as Windows registry and Linux RPM. Different than most passive techniques, Brown et al. [9] propose an active approach combined with statistical methods to characterize dynamic dependencies between system components. However, active "perturbation" is invasive and it impacts the performance, availability, and behavior of the production system, which is normally not acceptable in production environment.

Secondly we focus on the critical path of a dependency graph. In a distributed application there may be many parallel executions but only some of the components are responsible to the overall response time. And the longest execution path along these components is called a “critical path”. A further step after dependency analysis is to identify a critical path that determines the overall response time. In [84] Yang and Miller constructs a program activity graph (PAG, acyclic) using the data collected during the execution of a program. The critical path (i.e. the longest path) can be calculated by easily modifying most available shortest path algorithms, for instance, Chandy and Misra’s algorithm [14]. Statistical extension is also proposed for comparing execution lengths which are in the form of statistical distributions [86].

Last but not least, successful dependency analysis requires efficient, accurate, end-to-end measurement and monitoring tools. Chen et al. [16] proposes an approach for end-to-end trace collection of client requests through system components by tagging calls with request-IDs on J2EE servers. Cherkasova et al. [19] measure end-to-end Internet service performance by server side monitoring of HTTP packet traces. Web page time is broken down to detailed response times of network and processing, and statistical filtering mechanism is applied to reconstruct page accesses. Hellerstein et al. [33] externalize the definition of transactions and use it to construct response time measurements from event streams. It is not a black-box approach as the components have to be instrumented to generate events with specific structures. The advantage of such an approach is that transactions can be decomposed into multiple layers of sub-transactions which relate to different components such as authentication, HTTP and DNS. A layered monitoring framework is necessary for enabling dependency analysis and SLA translation across layers.

3.4 Classification and Discussion

After detailed review and analysis of state of the art, we summarize our findings in this section. Serving as a quick reference, Table 1 includes summaries of the main literature under study. We present the discussions by the SLA translation types which are defined in Section 2.

• We find that C2C (Configuration to Configuration) type of translation is investigated heavily in DiffServ and policy conversion in the networking domain. This is reasonable since the separation of classes of services (CoS) naturally involves specifications of low-level device (e.g. router) configurations. In computer systems C2C translation largely relates to dependency analysis and critical paths. By combining both static system information and dynamic tracing data, the dependency structures of components can be discovered and characterized to a useful level. These are important information in correlating distributed components within or across layers. Techniques that are used for C2C translation include semantic-specific translation, rule-based systems, and statistical analysis.

• C2M (Configuration to Metric) type of translation constitutes a large portion of research as performance predictions are popular topics across domains, including computer system management and software performance. At design time analytic models such as queuing networks and layered queuing networks are commonly used for characterizing

Reference	Topic	Approach	Category	Type	Source	Target
[45]	Mapping QoS metrics to network metrics	Simple functional mapping by defining evaluation functions	Knowledge and rules	M2M	Packet loss, delay, utilization	Availability, latency, MTTF
[73, 48]	Mapping application SLA to network parameters	Experimental method to build graphic functions [73]; queuing-analytic and operational laws [48]	Queuing-analytic; statistical	M2C	Application response time	network latency
[8, 12]	Diffserv - Differentiated services in the Internet	Classification of services based on QoS requirements	Knowledge and rules	M2C	Latency, loss	CoS coding
[76]	Policy validation and translation	Translate classes of services (CoS) into device configurations	Knowledge and rules	C2C	CoS coding	router parameters
[39]	QoS mapping and adaptation	Fuzzy-logic controller and filter algorithms, rule based inference engine	Rule based; statistical	C2M	Packet loss	Frame rate
[79]	Application specific middleware configuration	QoS profiling, semantic translation, constraint satisfaction	Knowledge and rules	M2C	Frame rate, size	middleware and resource parameters
[35, 31]	Resource management on HPC clusters	scheduling algorithms including priority queues, fairshare, throttling, etc.	Knowledge and rules	M2C	Application response time, throughput	Resource parameters
[25]	E-commerce platform management	Distributed systems techniques on data partitioning, replication, and failure management	Analytical algorithms	M2M	DB access time, availability	Web response time
[6, 74]	Performance modeling of multi-tier systems	Queuing network models and mean value analysis (MVA)	Queuing analytic	C2M	Resource parameters	Response time
[42]	Performance modeling of multi-tier systems with burst sessions	Combine queuing network models with Monte Carlo simulation that deals with heavy-tail distributions	Queuing analytic	C2M	Resource parameters	Response time
[17, 18]	SLA decomposition in multi-tier systems	Queuing network models and a regression-based profiling technique	Queuing analytic	M2C	Response time	Resource parameters
[62, 83, 28]	Performance modeling and prediction of software	Layered queuing network (LQN) models and mean value analysis (MVA)	Queuing analytic	C2M	Resource parameters	Response time, throughput
[61, 60, 40]	Performance prediction of software systems	Annotate software models (e.g. UML, system scenarios) and transform them into performance models (e.g. LQN)	Queuing analytic	C2M	Resource parameters	Response time, throughput
[22, 5]	Performance prediction of software systems	Integrate software and resource models for performance prediction via simulation	Queuing analytic, simulation	C2M	Resource parameters	Response time
[43]	Derivation of component level objectives from SLAs	State space partitioning and Bayesian networks	Statistical	M2C	Response time	Resource parameters
[59, 27, 69, 36]	Autonomic SLA management in multi-tier systems	Control theory, reinforcement learning, signal processing, rule-based systems	Statistical	C2M	Resource parameters	Response time
[20, 21, 38]	Performance diagnosis and fault management	Multivariate regression analysis, metric transformation, Bayesian networks	Statistical	M2M	Utilization, DB access time, etc	Response time
[2, 86, 9]	Dependencies of system components and critical path analysis	Heuristic correlation algorithms, critical path profiling, active perturbation and statistical modeling	Statistical	C2C	System configuration	Critical paths and dependency graphs
[16, 19, 33]	Measurement and monitoring techniques	instrumentation, tagging, trace analysis	Statistical	-	-	Response time, throughput
[15, 75, 77]	Resource allocation in hosting centers, service composition	Utility and economic approach, multiobjective optimization	Statistical	C2M	Resource parameters	Response time, profit, energy

Table 1: Summary of the main references, including topics, approaches, method categories, translation types, source and target metrics/parameters.

relationships between metrics and parameters. Given the specified low-level resource parameters, the high-level metrics can be predicted based on performance models. At run time statistical methods such as time series and control theory proves to be successful in tracking target metrics and adjusting parameters accordingly. Other machine learning techniques such as reinforcement learning can also be applied for online optimization.

• M2C (Metric to Configuration) translation is the opposite direction compared to C2M. Similar models as in C2M can be used (e.g. the queuing network model), which characterize the relationship between metrics and parameters. The problem then becomes how to derive parameters given pre-defined thresholds on metrics. Such problem can be formulated as constraint satisfaction problem and techniques such as heuristic search algorithms or linear programming are commonly-used solvers. This type of translation, or so-called SLA decomposition, becomes increasingly important for SLA-aware management during design and planning phase.

• M2M (Metric to Metric) translation relies on analytic or statistical models that correlate high-level and low-level metrics. Bayesian networks is such a class of probabilistic models that are successfully applied to M2M problems. Trained on historical trace data, relationship between a high-level metric and several low-level metrics can be built using Bayesian classifiers. On one hand, the sub-SLAs associated with low-level metrics can be used to "predict" high-level SLA violation. This is called a forecasting/classification process. On the other hand, high-level SLAs can be broken down to define several sub-SLAs by leveraging constraint satisfaction algorithms. This is called a decomposition process. Metric to metric translation shows its importance for SLA-driven operation management at the run time phase.

We believe that these four main types of translations are crucial topics to be addressed for managing SLAs in multi-layered SOA.

4 Research Challenges

As is shown above, problems associated with each translation type are of different kind in nature. We have reviewed a number of solutions and approaches which have been developed successfully for each type of SLA translations. In this section, we discuss several main research challenges ahead for making the vision of holistic SLA-driven IT management a reality.

1. Realistic workloads and usage patterns

Most of the queuing-network based performance models under review rely on unrealistic Markovian workload assumptions to be analytically tractable. However, real user behavior and real world workloads exhibit heavy-tailed distributions and high-level burstiness in many situations. And they are unknown or hard to anticipate at design time. Such statistical properties, in turn, may have great impacts on the system performance and resource consumption. Therefore workloads and usage patterns need to be taken into account in defining SLAs. For example, guaranteeing average or percentile of response times may have dramatically different implications on the service provider side under bursty or none-bursty usage patterns. The challenges associated with workloads come from two aspects: firstly performance models need to be improved, so that burstiness and high variability can be considered during the capacity planning phase. Secondly, usage profiles and workloads have to be parameterizable for facilitating the SLA specification and design time optimization.

2. Tradeoff-analysis for scalable approaches

Translation lies in the core of SLA management in that it correlates metrics and parameters within and across layers. As such it has to possibly consider a huge number of properties. This includes identifying relevant properties, identifying relevant correlations between properties, and deriving property values in order to answer certain SLA management questions. Consequently, SLA translation faces two scalability issues. Firstly, the issue of compute scalability, i.e. the scalability of actual algorithms (e.g. analytical or statistical ones) in terms of required compute resources, respectively in terms of time needed to generate a response. Though this issue is well known in the performance prediction community many scientific results simply do not scale with industrial problem-sizes. Secondly, there is the related issue of modeling scalability which is about the required (and typically manual) effort to discover and model relevant properties as well as their correlations. Many of the scientific results assume from scratch engineering of systems which obviously does not work for legacy applications but even for new applications the associated effort is often far to cost intensive and therefore not used in industrial practice.

Both these issues are well known in research around SLA translation or performance engineering and there is most likely no silver bullet to solve them. However, the challenge we see in order to make the best out of existing techniques is a thorough tradeoff analysis which contrasts the required accuracy for SLA translation with the needed effort in terms of human, time and compute resources.

3. Innovation and integration of methodologies

Innovating methodologies is crucial for more efficient and effective problem solving in SLA translations. We have seen how SLA decomposition is solved as a constraint satisfaction problem and optimization techniques such as linear programming and exhaustive search are applied as solvers. There is much room for improvement by applying more advanced optimization methods. For instance, in addition to deliver the SLA guarantees the service provider mostly cares about reducing operational costs on its side. This is essentially a problem with multiple objectives (e.g. performance and cost), which are conflicting with each other. In such cases multi-objective optimization (MOO) methods and utility theory prove to be more applicable than single-objective based methods. The search strategies for design time exploration can also be improved, for example, from exhaustive search to stochastic and heuristic search.

The performance models for mapping parameters to metrics needed to be improved as well. We have seen recent research on the accuracy gain of layered queuing models over classic queuing networks [42]. The real challenge lies in a performance model which could naturally represent the logical layers in a multi-layered service systems. Such a model would be ideal in the context of SLA translation, and more active research are need towards this direction.

So far we have talked about the translation problem from a static view, namely, at design time. There are also many challenges at run time. We have seen that the application of Bayesian networks as probabilistic models to correlate metrics at different layers. Since the system and the workload evolve along time, it would be desirable that the model be adaptive as well. There are advances in theory of dynamic models, for example dynamic Bayesian networks (DBNs), but few work looks at their applications in real-world production systems. Kalman filters, which is considered as the most simple type of DBNs, have been recently applied to track and adapt model parameters in LQNs [87].

On one hand, integration is to combine the best of the breed methodologies from the performance modeling and optimization to address a particular SLA translation problems. On the other hand, integration comes from the need of problem solving across multiple domains. The landscape of today's service provider is inherently integrated. It consists of all kinds of elements, namely networks, servers, storage, and software stacks. Therefore the fulfillment of any higher-level objective requires proper enforcements not on a single resource, but on multiple resources at low levels. For example, in order to guarantee certain bounds on the response times for ERP-type of requests there are potentially many configuration steps. It involves the ERP software, the application and database servers, the network configuration, and more. In the research community it is common-practice, or even necessary to focus on a specific domain (e.g. server) by making assumptions on other domains. However, great amount of work are needed to deliver practical solutions in real service-oriented environments, for which integrated approaches are necessary and have to be well developed.

4. Model integration and transformation

After discussing the problem of methodology integration, we address the issues with model integration. The full SOA stack typically includes independently developed models and various model artifacts provided by different parties for different purposes. For example, there are infrastructure-level models such as CIM, software component models such as SCA or UML component diagram, and business process models such as BPEL. Potential usage scenarios include architecture design, deployment description, testing, and operation. Information available in these models could be highly important for solving SLA translation related problems, for example, the resource specification in CIM and the component information by SCA. To obtain such information efficiently from many different models, it is necessary to extract and represent the information in a harmonized and integrated way. Therefore a systematic approach to model integration, transformation, and mapping play an important role for supporting SLA translations. A harmonized representation of available model information can also generate new insights for a broader range of translation problems. For instance, sub-SLAs for software components can be defined, monitored, and enforced only if component-level model information are exposed and made available for translation algorithms.

5. The definition of layers and layer interfaces

In certain SOA application scenarios, it is not possible to build a complete holistic view of the system, or to clearly identify all the layers as defined in Section 2. Therefore the definition and separation of layers should be flexible and reflect the context and customized view of a specific environment. It is a challenging task to find a balance between the abstraction of layers and the exposure of layer internal information for enabling SLA translation, especially when the layers are across administrative boundaries. For instance, a Software-as-a-Service (SaaS) provider makes use of the infrastructure provided by a cloud provider. The correlation between high-level metrics such as response time and some low-level operational metrics in the infrastructure becomes not possible because low-level monitoring information may be only available to the cloud provider. In this case the interfaces between layers must be well-defined on what information should be exchanged between two parties. This is also in close relationship on the concrete guarantee terms to be specified in the SLAs.

6. Business values and reference benchmarks

Estimating the business values of IT services probably represents a even higher level of translation problems beyond the technical scope of this paper. Nevertheless, it is necessary to point out its importance as the economic structure of service systems has increased greatly in terms of complexity. Quantitative assessment of business values and impact provides valuable guidelines for IT service deployment and change management. For some recent work towards this direction we refer to [13, 26].

Benchmarks are needed in order to compare and evaluate different approaches. For E-Commerce and dynamic web sites widely-used benchmarks include TPC-W and RUBiS [3]. In the scope of this paper we are primarily interested in application server and web services benchmarks. Many industry vendors provide their own ones (e.g. SAP, Oracle), and TPC also provides the TPC-App benchmark [72]. TPC-App models processes and services that a retail distributor support ordering and retrieving product information, which represents a typical business-to-business (B2B) transactional scenario. Some early work has used this benchmark in comparing middleware platforms such as J2EE and .NET [30]. We envision that more upcoming research work on SLA translation adopt standardized benchmarks so their results can be compared and evaluated.

5 Summary

In this paper we reviewed and analyzed the state of the art of SLA translation in SOA. We adopt a multi-layered perspective and distinguish four main types of translation problems, which prove to be successful in classifying and evaluating a broad range of approaches across multiple domains. Our research is driven by the importance of layers and a holistic view in service-oriented systems, and the growing needs for translating metrics and parameters across multiple layers.

We find that semantic-specific translation and rule-based systems are common practices for policy configuration and translation in the networking domain, given expert information in place. This applies to qualitative NFP translation (mostly configuration) in computer systems and software engineering as well. For quantitative translations queuing-analytic models are popular and well-established methodologies. On one hand, such models can be directly applied for performance prediction problems (namely configuration to metric translation). On the other hand, similar models can be combined with constraint satisfaction and optimization algorithms for decomposing high-level objective into low-level parameters (namely metric to configuration translation). Statistics and machine learning techniques can also be applied in various situations. It can be combined with analytic models to form a comprehensive approach. It is particularly useful in situations when only a black-box approach is feasible. One class of probabilistic models called Bayesian networks prove to be effective to capture relationships between a high-level metric with a set of low-level metrics with or without the system knowledge. Such metric to metric translations are powerful tools for problem diagnosis of SLA violations.

Though many sophisticated approaches for partial SLA translations exist, comprehensive and general purpose solutions are still to be developed. On the way forward we see major challenges in terms of integrated modelling of relevant perspectives (workloads, layers, business value) and innovative and integrated algorithms (multi-objective, layered approaches, multi-domain). We envision that more active research and development will be carried out for SLA translation topics in SOA-enabled IT systems, and we believe that such efforts will contribute to drive SOA to be a success.

Acknowledgments

This research is funded in part via project SLA@SOI in the EU 7th framework programme, grant agreement no. FP7-216556.

References

[1] P. Abry, R. Baraniuk, P. Flandrin, R. Riedi, and D. Veitch, "The multiscale nature of network traffic: discovery, analysis, and modelling," *IEEE Signal Processing magazine*, vol. 19, no. 3, pp. 28–46, May 2002.

[2] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen, "Performance debugging for distributed systems of black boxes," in *proc. of SOSP*. ACM, 2003, pp. 74–89.

[3] C. Amza, A. Ch, A. L. Cox, S. Elnikety, R. Gil, K. Rajamani, and W. Zwaenepoel, "Specification and implementation of dynamic web site benchmarks," in *proc. of 5th IEEE Workshop on Workload Characterization*, 2002, pp. 3–13.

[4] S. Balsamo, A. D. Marco, P. Inverardi, and M. Simeoni, "Model-based performance prediction in software development: A survey," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 295–310, 2004.

[5] S. Becker, H. Koziolek, and R. Reussner, "The palladio component model for model-driven performance prediction," *J. of Systems and Software, to appear*, 2008.

[6] M. Bennani and D. Menasce, "Resource allocation for autonomic data centers using analytic performance models," in *proc. of Intl. Conf. on Autonomic Computing (ICAC).* IEEE, 2005, pp. 229–240.

[7] M. Bichler and K.-J. Lin, "Service-oriented computing," *IEEE Computer*, vol. 39, no. 3, pp. 99–101, 2006.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Service," RFC 2475 (Informational), 1998.

[9] A. Brown, G. Kar, and A. Keller, "An active approach to characterizing dynamic dependencies for problem determination in a distributed environment," in *proc. of 7th IFIP/IEEE Intl. Sym. on Integrated Network Management*, 2001, pp. 377–390.

[10] M. J. Buco, N. R. Chang, L. Z. Luan, C. Ward, L. J. Wolf, and P. S. Yu, "Utility computing SLA management based upon business objectives," *IBM System J.*, vol. 43, no. 1, pp. 159–178, 2004.

[11] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*, 1st ed. Wiley, 1996.

[12] B. E. Carpenter and K. Nichols, "Differentiated services in the internet," *Proc. of the IEEE*, vol. 90, no. 9, pp. 1479–1494, 2002.

[13] N. S. Caswell, C. Nikolaou, J. Sairamesh, M. Bitsaki, G. D. Koutras, and G. Iacovidis, "Estimating value in service systems: A case study of a repair service system," *IBM System J.*, vol. 47, no. 1, pp. 87–100, 2008.

[14] K. M. Chandy and J. Misra, "Distributed computation on graphs: Shortest path algorithms," *Commun. of ACM*, vol. 25, no. 11, pp. 833–837, 1982.

[15] J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centres," in *proc. of SOSP.* ACM, 2001, pp. 103–116.

[16] M. Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. A. Brewer, "Pinpoint: Problem determination in large, dynamic internet services," in *proc. of DSN.* IEEE, 2002, pp. 595–604.

[17] Y. Chen, S. Iyer, X. Liu, D. Milojicic, and A. Sahai, "Translating service level objectives to lower level policies for multi-tier services," *Cluster Computing*, vol. 11, pp. 299–311, 2008.

[18] Y. Chen, A. Sahai, S. Iyer, and D. Milojicic, "Systematically translating service level objectives into design and operational policies for multi-tier applications," HP Labs Palo Alto, Tech. Rep. HPL-2008-16, 2008.

[19] L. Cherkasova, Y. Fu, W. Tang, and A. Vahdat, "Measuring and characterizing end-to-end internet service performance," *ACM Trans. Internet Techn.*, vol. 3, no. 4, pp. 347–391, 2003.

[20] I. Cohen, J. S. Chase, M. Goldszmidt, T. Kelly, and J. Symons, "Correlating instrumentation data to system states: A building block for automated diagnosis and control," in *proc. of OSDI.* USENIX, 2004, pp. 231–244.

[21] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox, "Capturing, indexing, clustering, and retrieving system history," in *proc. of SOSP.* ACM, 2005, pp. 105–118.

[22] V. Cortellessa, P. Pierini, and D. Rossi, "Integrating software models and platform models for performance analysis," *IEEE Trans. Software Eng.*, vol. 33, no. 6, pp. 385–401, 2007.

[23] A. D'Ambrogio, "A Model-driven WSDL Extension for Describing the QoS of Web Services," in *Proceedings of the IEEE International Conference on Web Services (ICWS'06), Chicago, USA, September*, 2006, pp. 18–22.

[24] A. D'Ambrogio and P. Bocciarelli, "A Model-driven Approach to Describe and Predict the Performance of Composite Services," in *WOSP '07: Proceedings of the 6th International Workshop on Software and Performance.* New York, NY, USA: ACM, 2007, pp. 78–89.

[25] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," in *proc. of SOSP.* ACM, 2007, pp. 205–220.

[26] Y. Diao and K. Bhattacharya, "Estimating Business Value of IT Services through Process Complexity Analysis," in *proc. of IEEE/IFIP NOMS*, 2008, pp. 208–215.

[27] Y. Diao, J. L. Hellerstein, and S. S. Parekh, "Using fuzzy control to maximize profits in service level management," *IBM Systems Journal*, vol. 41, no. 3, pp. 403–420, 2002.

[28] G. Franks, A. Hubbard, S. Majumdar, J. E. Neilson, D. C. Petriu, J. A. Rolia, and C. M. Woodside, "A toolset for performance engineering and software design of client-server systems," *Perform. Eval.*, vol. 24, no. 1-2, pp. 117–136, 1995.

[29] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[30] D. F. García, J. García, M. García, I. Peteira, R. García, and P. Valledor, "Benchmarking of Web Services Plattforms - An Evaluation with the TPC-APP Benchmark," in *proc. of WEBIST*, 2006, pp. 75–80.

[31] P. Hasselmeyer, B. Koller, L. Schubert, and P. Wieder, "Towards SLA-Supported Resource Management," in *proc. of Intl. Conf. on High Performance Computing and Communications (HPCC).* Springer, 2006, pp. 743–752.

[32] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, August 2001.

[33] J. L. Hellerstein, M. M. Maccabee, W. N. M. III, and J. Turek, "ETE: A Customizable Approach to Measuring End-to-End Response Times and Their Components in Distributed Systems," in *proc. of ICDCS.* IEEE, 1999, pp. 152–162.

[34] ITIL version 3, "Service operation," 2008, online available: http://www.itilversion3.com/, accessed Feb 2009.

[35] D. B. Jackson, Q. Snell, and M. J. Clement, "Core algorithms of the maui scheduler," in *proc. of Intl. Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP).* Springer, 2001, pp. 87–102.

[36] G. Jung, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and C. Pu, "Generating adaptation policies for multi-tier applications in consolidated server environments," in *proc. of Intl. Conf. on Autonomic Computing (ICAC).* IEEE, 2008, pp. 23–32.

[37] G. Kar, A. Keller, and S. Calo, "Managing application services over service provider networks: Architecture and dependency analysis," in *proc. of the 7th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2000, pp. 61–75.

[38] T. Kelly, "Detecting performance anomalies in global applications," in *proc. of 2nd Conf. on Real Large Distributed Systems (WORLDS).* USENIX, 2005, pp. 42–47.

[39] C. Koliver, K. Nahrstedt, J.-M. Farines, J. da Silva Fraga, and S. A. Sandri, "Specification, Mapping and Control for QoS Adaptation," *Real-Time Systems*, vol. 23, no. 1-2, pp. 143–174, 2002.

[40] H. Koziolek, "Parameter dependencies for reusable performance specifications of software components," Ph.D. dissertation, Universitat Oldenburg, 2008.

[41] H. Koziolek, S. Becker, and J. Happe, "Predicting the Performance of Component-based Software Architectures with different Usage Profiles," in *Proceedings of the 3rd International Conference on the Quality of Software Architectures (QoSA2007)*, ser. LNCS, vol. 4880. Boston, USA: Springer, Juli 2007.

[42] D. Krishnamurthy, J. Rolia, and M. Xu, “Wam –the weighted average method for predicting the performance of systems with bursts of customer sessions,” HP Labs, Tech. Rep. HPL-2008-66, 2008.

[43] V. Kumar, K. Schwann, S. Iyer, Y. Chen, and A. Sahai, “A State Space Approach to SLA based Management,” in *proc. of IEEE/IFIP NOMS*, 2008.

[44] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queuing Network Models.* Prentice-Hall, Inc., 1984.

[45] H. Lee, M. Kim, J. Hong, and G. Lee, “Mapping between QoS Parameters and Network Performance Metrics for SLA monitoring,” in *proc. of Asia-Pacific NOMS*, 2002.

[46] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, “On the self-similar nature of ethernet traffic (extended version),” *IEEE/ACM Trans. on Networking*, vol. 2, no. 1, pp. 1–15, 1994.

[47] H. Li, “Workload characterization, modeling, and prediction in grid computing,” Ph.D. dissertation, Universiteit Leiden, 2007.

[48] B. H. Liu, P. Ray, and S. Jha, “Mapping Distributed Application SLA to Network QoS parameters,” in *proc. of 10th IEEE Intl. Conf. on Telecommunications*, 2003.

[49] M. Marzolla and R. Mirandola, “Performance Prediction of Web Service Workflows,” in *Proceedings of the 3rd International Conference on Quality of Software Architectures*, ser. Lecture Notes in Computer Science, vol. 4880, 2007, pp. 127–144.

[50] D. Menascé, H. Ruan, and H. Gomaa, “QoS management in service-oriented architectures,” *Performance Evaluation*, vol. 64, no. 7-8, pp. 646–663, 2007.

[51] D. Menasce, B. Abrahao, D. Barbara, V. Almeida, and F. Ribeiro, “Fractal characterization of web workloads,” in *In proc. of the 11th Intl. World Wide Web Conf. (WWW2002*, 2002, pp. 7–11.

[52] D. A. Menasce and V. A. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods.* Prentice-Hall, PTR, 2001.

[53] D. A. Menascé, H. Ruan, and H. Gomaa, “A framework for QoS-aware software components,” *SIGSOFT Softw. Eng. Notes*, vol. 29, no. 1, pp. 186–196, 2004.

[54] J. E. Neilson, C. M. Woodside, D. C. Petriu, and S. Majumdar, “Software bootlenecking in client-server systems and rendezvous networks,” *IEEE Trans. Software Eng.*, vol. 21, no. 9, pp. 776–782, 1995.

[55] Object Management Group (OMG), “Technical guide to model driven architecture (mda) v1.0.1,” 2003, online available: http://www.omg.org/mda/, accessed Oct 2008.

[56] ——, “The omg and service oriented architecture,” 2008, online available: www.omg.org/attachments/pdf/OMG-and-the-SOA.pdf, accessed Oct 2008.

[57] Official OASIS Standard, “Oasis reference model for service oriented architecture 1.0,” 2006, online available: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf, accessed Oct 2008.

[58] M. P. Papazoglou and W.-J. van den Heuvel, “Service oriented architectures: approaches, technologies and research issues,” *VLDB J.*, vol. 16, no. 3, pp. 389–415, 2007.

[59] S. S. Parekh, N. Gandhi, J. L. Hellerstein, D. M. Tilbury, T. S. Jayram, and J. P. Bigus, “Using control theory to achieve service level objectives in performance management,” *Real-Time Systems*, vol. 23, no. 1-2, pp. 127–141, 2002.

[60] D. B. Petriu and C. M. Woodside, “Software performance models from system scenarios,” *Perform. Eval.*, vol. 61, no. 1, pp. 65–89, 2005.

[61] D. C. Petriu and X. Wang, “From uml descriptions of high-level software architectures to lqn performance models,” in *proc. of Intl. Workshop on Applications of Graph Transformations with Industrial Relevance (AGTIVE)*, 1999, pp. 47–62.

[62] J. A. Rolia and K. C. Sevcik, “The method of layers,” *IEEE Trans. Software Eng.*, vol. 21, no. 8, pp. 689–700, 1995.

[63] S. M. Ross, *Introduction to Probability Models*, 8th ed. Academic Press, 2003.

[64] F. Rossi, P. van Beek, and T. Walsh, Eds., *Handbook of Constraint Programming.* Elsevier, 2006.

[65] A. Sahai, A. Durante, and V. Machiraju, “Towards automated sla management for web services,” HP Labs Palo Alto, Tech. Rep. HPL-2001-310, 2002.

[66] M. Shaw, “Some patterns for software architectures,” in *Pattern Languages of Program Design.* Addison-Wesley, 1996, vol. 2, pp. 255–269.

[67] SOA Blueprint, “Soa practitioners guide part 2: Soa reference architecture,” 2006, online available: http://soablueprint.com/yahoo_site_admin/assets/docs/SOAPGPart2.290211443.pdf, accessed Aug 2008.

[68] M. S. Squillante, D. D. Yao, and L. Zhang, “The impact of job arrival patterns on parallel scheduling,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 4, pp. 52–59, Dec. 1999.

[69] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, “On the use of hybrid reinforcement learning for autonomic resource allocation,” *Cluster Computing*, vol. 10, no. 3, pp. 287–299, 2007.

[70] The Zachman Institute for Framework Advancement, “The zachman framework for enterprise architecture,” 2008, online available: http://www.zifa.com/framework.pdf, accessed Oct 2008.

[71] W. Theilmann, R. Yahyapour, and J. Butler, “Multi-level SLA Management for Service-Oriented Infrastructures,” in *proc. of ServiceWave08.* Springer, 2008.

[72] Transaction Processing Performance Council (TPC), “Tpc benchmark app specification v1.3,” 2008, online available: http://www.tpc.org/tpc_app/spec/TPC-App_V1.3.pdf, accessed Aug 2008.

[73] E. Tse-Au and P. Morreale, “End-to-end QoS measurement: analytic methodology of application response time vs. tunable latency in IP networks,” in *proc. of IEEE/IFIP NOMS*, 2000, pp. 129–142.

[74] B. Urgaonkar, G. Pacifici, P. J. Shenoy, M. Spreitzer, and A. N. Tantawi, “Analytic modeling of multitier internet applications,” *ACM Trans. on the Web*, vol. 1, no. 1, 2007.

[75] A. v.d. Kuijl, M. Emmerich, and H. Li, “A new multi-objective optimization scheme for grid resource allocation,” in *proc. of Middleware conf., Workshop on Middleware for Grid Computing.* ACM, 2008.

[76] D. C. Verma, *Policy-Based Networking: Architecture and Algorithms.* New Riders Publishing, 2000.

[77] H. Wada, P. Champrasert, J. Suzuki, and K. Oba, “Multiobjective Optimization of SLA-Aware Service Composition,” in *proc. of IEEE Congress on Services*, 2008, pp. 368–375.

[78] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser, “Terminology for Policy-Based Management,” RFC 3198 (Informational), 2001.

[79] D. Wichadakul and K. Nahrstedt, “A Translation System for Enabling Flexible and Efficient Deployment of QoS-Aware Applications in Ubiquitous Environments,” in *Component Deployment*, ser. LNCS 2370. Springer, 2002, pp. 210–221.

[80] A. Williams, M. Arlitt, C. Williamson, and K. Barker, “Web workload characterization: Ten years later,” in *Web Content Delivery.* Springer, 2005, vol. 2, pp. 3–21.

[81] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.

[82] C. M. Woodside, “Tutorial introduction to layered modeling of software performance,” *Online available: http://www.sce.carleton.ca/rads/lqns/lqn-documentation/tutorialg.pdf*, Aug 2008.

[83] C. M. Woodside, J. E. Neilson, D. C. Petriu, and S. Majumdar, "The stochastic rendezvous network model for performance of synchronous client-server-like distributed software," *IEEE Trans. Computers*, vol. 44, no. 1, pp. 20–34, 1995.

[84] C.-Q. Yang and B. P. Miller, "Critical path analysis for the execution of parallel and distributed programs," in *proc. of ICDCS*. IEEE, 1988, pp. 366–373.

[85] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 311–327, 2004.

[86] R. Zhang, "Identifying performance-critical components in transaction-oriented distributed systems," in *proc. of Intl. Conf. on Autonomic Computing (ICAC)*. IEEE, 2007, p. 13.

[87] T. Zheng, C. M. Woodside, and M. Litoiu, "Performance model estimation and tracking using optimal filters," *IEEE Trans. Software Eng.*, vol. 34, no. 3, pp. 391–406, 2008.