

Challenges in SLA Translation

Hui Li, SAP Research, December 01, 2009

1 Introduction

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 article 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. Furthermore, we discuss the fundamental research challenges to be addressed for turning the vision of holistic and transparent SLA translation into reality.

2 Conceptual Framework

There are a few reference models and architectures for SOA proposed by different standard bodies, such as OMG, OASIS, and SOA Alliance. 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. 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 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 a SLA in this context is. 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. 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 sublayer 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 (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, 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. 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 and/or machine learning techniques. 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).

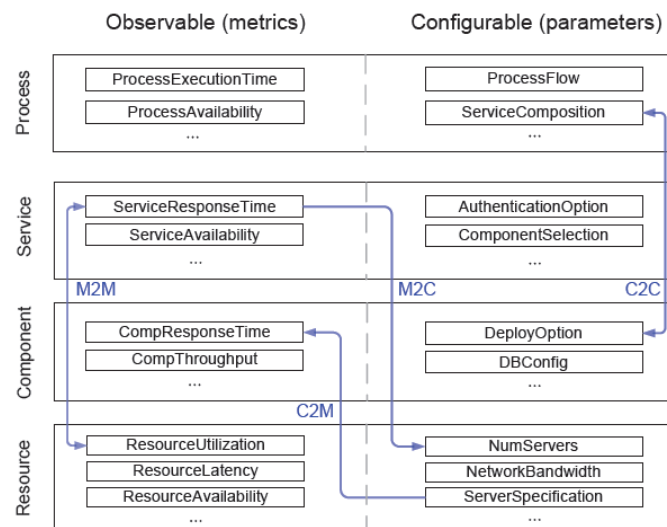


Figure 1: Observables (metrics) and configurables (parameters) in SOA layers, with different types of translations

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 type 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.

3 Research Challenges

As is shown above, problems associated with each translation type are of different kind in nature. In this section, we discuss several main research challenges ahead for making the vision of holistic SLA-driven IT management a reality.

3.1 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 behaviour 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.

3.2 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 too 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 trade-off analysis which contrasts the required accuracy for SLA translation with the needed effort in terms of human, time and compute resources.

3.3 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. The real challenge lies in a performance model which could naturally represent the logical layers in a multi-layered service system. Such a model would be ideal in the context of SLA translation, and more active research is needed 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 works look at their applications in real-world production systems. Kalman filters, which are considered to be the simplest type of DBNs, have been recently applied to track and adapt model parameters.

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 problem. 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 fulfilment 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.

3.4 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 artefacts 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.

3.5 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. 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.

3.6 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.

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. 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. 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. We envision that more upcoming research work on SLA translation adopts standardized benchmarks so their results can be compared and evaluated.