2010 Seventh International Conference on the Quality of Information and Communications Technology

Defining and Observing the Compliance of Service Level Agreements: A Model Driven Approach

Anacleto Correia
CITI / QUASAR / FCT / UNL
and IPS/EST Setúbal
Setúbal, Portugal
accorreia@fct.unl.pt

Fernando Brito e Abreu
CITI / QUASAR / FCT
Universidade Nova de Lisboa
Caparica, Portugal
fba@di.fct.unl.pt

Abstract — **IT Service Management (ITSM) is the set of processes that allow planning, organizing, directing and controlling the provisioning of IT services. Among the concerns of ITSM, namely within the service level management process, are the requirements for services availability, performance, accuracy, capacity and security, which are specified in terms of service-level agreements (SLA). SLA definition and monitoring are open issues within the ITSM domain. This paper overviews an ongoing research initiative concerned with three specific problems in this context: (1) SLAs in the context of ITSM are informally specified in natural language; (2) SLAs specifications are not grounded on models of ITSM processes; (3) SLAs compliance verification in IT services is not performed at the same level of abstraction as service design. To mitigate those problems, we propose a model-based approach to IT services SLA specification and compliance verification. The specification part will be based on a SLA language - a domain specific language (DSL) for defining quality attributes as non functional requirements (NFRs) in the context of ITSM. Its metamodel will be an extension of the metamodel of the adopted process modeling language. As such, it will be possible to ground SLA definition on the corresponding IT service model constructs. SLA monitoring and compliance validation will occur at the same abstraction level as service specification, therefore being understood by all stakeholders.**

Keywords- IT Service Management, ITIL; Service Level Management; Service Level Agreements, Metamodels, MDA; Domain Specific Language, Process Model, BPMN.

I. INTRODUCTION

Most organizations rely on Information Technology (IT) services to support their business processes. IT services are built upon the technical infrastructure (servers and network devices), as well as on systems and application software. The set of processes that allow planning, organizing, directing and controlling the provisioning of IT services is usually called an IT Service Management (ITSM) framework. Several ITSM frameworks have been proposed, such as ITIL [1], ISO 20000 [2] (based on ITIL v.2), MOF [3] (a Microsoft proposal, also influenced by ITIL), ISM [4], or HP ITSM [5]). Specially tailored examples of ITSM frameworks have also been proposed, such as eTOM [6], for the telecom industry. Although these frameworks are mainly abstract and prescriptive (of best practices), some efforts are being made to help practitioners to instantiate them by using process modeling notations such as the Business Process Modeling Notation (BPMN) [7].

In ITIL, the most widely used ITSM framework, some of the processes are: incident management, problem management, change management, service asset and configuration management and service level management [8]. In order to support these processes, IT providers use two kinds of applications: *service management applications*, which allow to track IT services incidents and problems; and *systems management tools*, for monitoring and control networks, systems or applications [9]. Currently, there are many service management ITSM tools [10] , as well as systems management tools [11] available with quite similar functionalities. Service management applications often overlie and rely, to some extent, on information collected by systems management applications.

Among the concerns of ITSM, namely within the service level management process, are the requirements for services availability, performance, accuracy, capacity and security, which are specified in terms of service-level agreements (SLA) [5]. In SLAs, such requirements, built upon quality attributes, are quantitatively bounded (e.g. maximum time to recover or repair, maximum latency or response time, minimal percentage of availability). However, although those requirements are non-functional in nature, and many non-functional requirements techniques have been proposed in the literature [12, 13], to the best of our knowledge, those techniques are not being used in SLA specification.

SLAs are important during the service design phase [14] and during operation, to guarantee that service delivery matches the agreed-upon levels and that it can be continuously improved [15]. Despite the current availability of applications for supporting service level management, SLA definition and monitoring are still identified as open issues within the ITSM domain [16]. In our research we are particularly concerned with three specific problems:

1. *SLAs in the context of ITSM are informally specified in natural language;*
2. *SLAs definition is not grounded on models of ITSM processes;*
3. *SLA compliance verification is not performed at the same level of abstraction as service design.*

978-0-7695-4241-6/10 $26.00 © 2010 Crown Copyright
DOI 10.1109/QUATIC.2010.32

The conjunction of these problems has hard consequences to ITSM practitioners. The first problem arises from the current practice in SLA specification for IT services, mostly based in templates filled with natural language descriptions [17]. Such descriptions are not amenable to the automation of SLA compliance verification. Since SLA definition is subjective in nature, it renders the second problem: it is not clear to all involved stakeholders which are the activities upon which SLA compliance should be verified, and how their non-compliance will affect the evolution in the process describing the IT service delivery. The third problem concerns a semantic gap which occurs in SLA compliance verification. This gap comes from the fact that compliance checking has been relying mostly in Quality of Service (QoS) information from systems management applications, instead of consolidated data presented at the process model representation of the IT service.

Somehow the first problem is similar to the one object-oriented modeling languages faced previously to the Unified Modeling Language (UML) [18] [19] unification effort. Diagrammatic notations were not able to capture business rules, so the latter were represented in natural language requirements aside from model diagrams. It was not possible to generate code from those requirements and traceability to implementation was hampered. With the advent of UML came the Object Constraint Language (OCL) [20], a constraint language that allows expressing business rules textually, based on the modeling concepts (e.g. classes and associations) represented graphically.

The main expected contribution of this research work is mitigating the above problems by proposing a model-based approach to SLA specification and compliance verification. The SLA specification will be derived by way of domain specific languages (DSL). Through SLA specification we seek to elicit quality attributes, applying techniques like the ones from requirements engineering, and specifically from non functional requirements (NFRs) [21], to the context of ITSM. The metamodel for the SLA specification will be an extension of the metamodel of the adopted process modeling language of IT services (BPMN). As such, it will be possible to ground SLA contracts definition on the corresponding IT service model constructs.

Regarding SLA compliance verification, we will use a SLA evaluator tool that will be able to run statements of the SLA language. This component will process SLA contracts expressed in the SLA language, as well as information of IT service instances execution, delivered by a process animation tool. The current state of the delivered IT services will be depicted, with visual tags, by that process animation tool, after handling events captured by IT service and system management tools. The SLA evaluator will supply information to a SLA visualizer, to display current levels of service being provided, namely if SLA limits have been surpassed. This environment will allow monitoring process execution and SLA compliance at the process modeling abstraction level, which is easier to understand by all stakeholders involved in IT service specification and deployment. We plan to validate this approach in the context of an IT service management project on the domain of financial self-service systems.

This paper is organized as follows: in the following section, we start by surveying the SLA lifecycle; in section III we identify the main challenges of this research work; in section IV we present the conceptual building blocks of our model driven approach for SLA specification and validation; in section V we describe the architecture of the SLAVE environment that is expected to demonstrate the feasibility of our approach; then, in section VI, we describe the validation context of our work; finally, in section VII, we present the main conclusions that could be drawn so far.

II. THE SLA LIFECYCLE

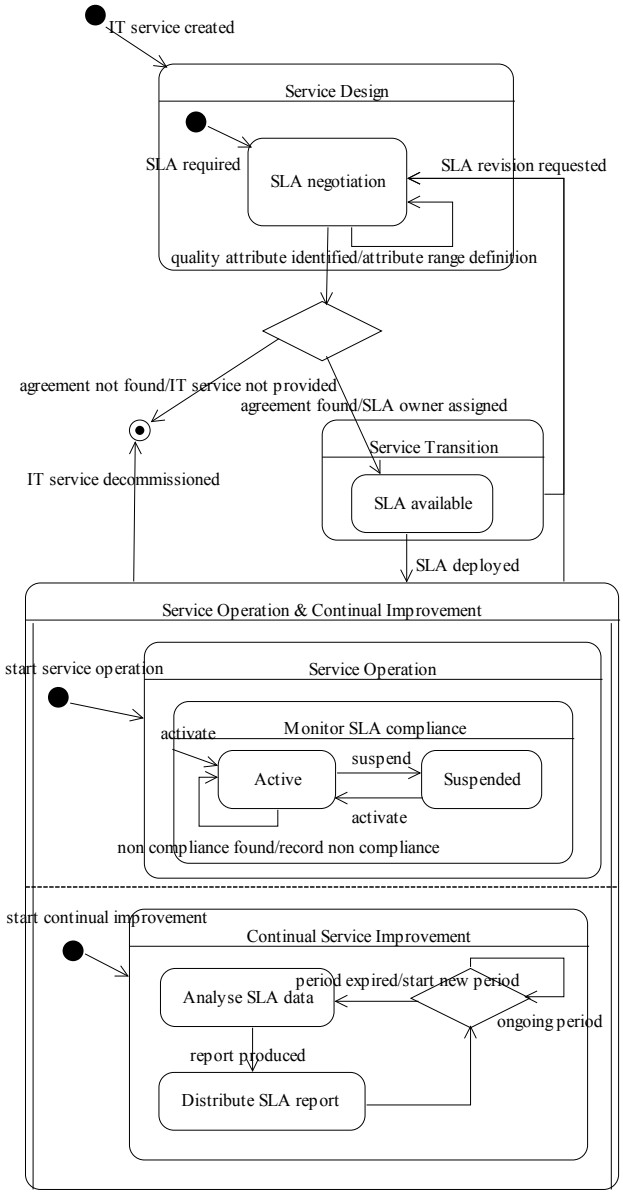

Figure 1. SLA lifecycle, according to ITIL v3

To get a better picture of the context of our proposal, we will now briefly describe the SLA lifecycle. The latter is

associated with ITIL's Service Level Management (SLM) process and is spread out by service phases from *Service Design* [14] to *Continual Service Improvement* [15], as represented in Figure 1.

SLA Specification. When a new IT service is conceived and the need for an SLA is identified by its stakeholders, the SLA negotiation takes place at some point of *Service Design*. For each relevant quality attribute identified, the definition for its acceptable range must be sought. When an agreement is found around the SLA specification, a SLA owner/manager is assigned and the SLA becomes available for *Service Transition* activities, in preparation for deployment.

SLA Validation. When the IT service starts to be provided, the monitoring of SLA compliance can be activated. While active, whenever a non-conformance is detected, it should be recorded. SLA monitoring can be suspended (e.g. when an SLA update is required) and re-activated again. *Continual Service Improvement* occurs in parallel with *Service Operation*. Within it and at regular observation periods, SLA compliance / non-compliance data records should be analyzed, hopefully in a partially automated fashion. The SLA manager should discuss the results of SLA data analysis in a report to distribute to all relevant stakeholders. Based upon consumer satisfaction surveys, the identification of potential improvements, known problems in service support, changing service requirements, or simply the will to improve service levels to get a better market share, the SLA specification may go through a revision.

III. CHALLENGES

As in Software Engineering, where functional and non-functional requirements (NFRs) cannot be seen as completely orthogonal aspects [22], there is also a trade-off among the several quality attributes of the provided IT services, which may vary from service to service. Therefore, the SLA specification should consider that a given level for a quality attribute can never be met in isolation, since thriving to meet the specified level of a quality attribute may have a positive or negative effect on the level of another quality attribute. The requirements engineering techniques for dealing with conflicting non-functional requirements [12] can help us to address quality attributes in the context of ITSM. Some examples of the tension among quality attributes follow:

- security and reliability - a more secure system has less points of failure, whereas a more reliable system should have more redundant elements (points of failure);
- performance and portability - performance is negatively affected by almost every quality attribute, namely portability. A technique for achieve portability is to isolate system dependencies, which introduces overhead into the system's execution, and hurts its performance.

A second challenge concerning our work is related to the fact that some service quality attributes (e.g. scalability, performance) are more amenable to quantification than others (e.g. credibility, auditability). The latter are then less prone to be captured automatically and therefore their compliance verification cannot be deployed in a tool.

IT service outsourcing, nowadays a common practice, poses another challenge for SLA compliance verification. Indeed, many organizations provide IT services that depend on services from other organizations. For example, an online shopping system may interact with (1) banks to authorize credit cards, (2) a warehouse to check availability and reserve the items to be sold, and (3) a delivering company for checking availability of transport. Another example are IT services provided upon technical infrastructures based on cloud computing, where shared resources (e.g. software, hardware infrastructure) are provided on-demand, like a public utility. The unavailability or poor performance of one of the services may compromise the whole IT service.

A final challenging issue is the automation of the SLA life cycle in the context of ITSM, in order to enable the negotiation, provisioning and compliance validation of standard IT services among organizations in real time.

IV. A MODEL DRIVEN APPROACH FOR SLA SPECIFICATION AND VALIDATION

Our proposal is framed by the Model-Driven Engineering (MDE) paradigm. MDE was operationalized by the OMG's initiative known as Model-Driven Architecture (MDA) [23], which refers to the systematic use of models as primary engineering artifacts, to express domain concepts effectively, throughout the engineering lifecycle. MDE assumes that any problem can be represented by a model which captures its essence [24] [25].

In our case, the problem domain is related with the SLA specification, and compliance verification in the context of IT services. In order to address the problem, our intention is to adapt NFR techniques from the requirements engineering domain [26] to the ITSM domain. NFR approaches allow elicitation of non-functional requirements, grounded in knowledge bases and functional requirements documentation [27]. By following the MDE approach, a set of model transformations from the domain specification models (the SLA contracts) to the implementation models (for SLA compliance verification) will be accomplished.

One possible alternative for expressing contracts upon IT service process model would be to extend the general-purpose UML modeling language with a profile that would restrict the scope to a particular domain with stereotypes, tagged values and constraints. This alternative has the advantage of tools availability, although the effort for creating a new profile is usually considerable. However, the main disfavor of this alternative is that the representation richness of UML renders the corresponding modeling tools excessively complex for IT service designers and clients.

To mitigate the previous problem we adopted a Domain-Specific Modeling (DSM) approach [28]. In other words, we are developing a Domain Specific Language (DSL) for expressing SLAs. DSLs allow solutions to be expressed in the idiom and at the level of abstraction of the problem domain. For instance, it will be possible to operate with familiar terms related with service level management (e.g. availability, performance), instead of using QoS terms which are more implementation oriented (e.g. bit rate, bit error, latency).

Consequently, domain experts can more easily understand, modify and validate specifications at the domain level. Summing up, we believe that this choice will reduce the effort to express a contract specification. Moreover, a DSM environment, aka graphical DSL tool, can automate the creation of tools that are costly to build from scratch, such as the editor required for specifying SLA contracts [29].

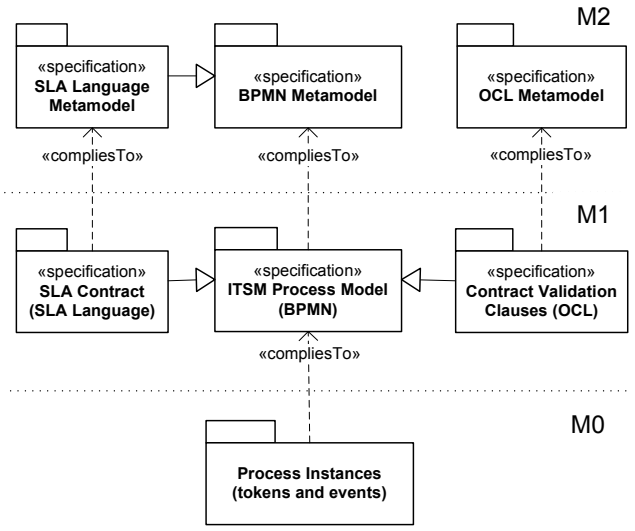

Figure 2. MDD approach to SLA specification and validation

In the following description of our approach we will use OMG's MDA abstraction levels M2 (metamodel level), M1 (model level) and M0 (model instances / objects level) [23] as represented in Figure 2. A SLA contract is a set of contractual clauses. To grant the required hooks for the SLA contracts, which stipulate levels for quality attributes upon specific activities in the process model, our SLA language metamodel will be an extension of the process modeling language to be used (BPMN). A given contract, expressed with our SLA language, will therefore be conformant to the SLA language metamodel and will extend the corresponding process model. While editing a contract, we must therefore provide SLA language metamodel checking ability (M1 to M2 compliance), while allowing to hook contractual clauses on abstractions of the process model under consideration (M1 to M1 relationships). The metamodel of the SLA language is being built upon ECore[1], a meta-metamodel language very close of the Essential MOF (EMOF) [18]. Notice that for the same process we can have more than one contract (e.g. different scenarios for different clients). The IT service process model under consideration (e.g. configuration management, incident management or problem management) should also be conformant with the BPMN metamodel (M1 to M2 compliance). Since this compliance is supposedly granted by existing process modeling tools, it will not be a concern here.

Using a M1 to M1 model transformation, we will generate the contract validation clauses (an OCL specification) out of the SLA contract. These clauses extend the ITSM process model, should be conformant to the OCL metamodel and will be used to validate the ITSM process model loaded with process instances (events and tokens) originating from run-time capture or simulation. By executing the OCL clauses upon a process model we can detect SLA compliance or violation.

Model transformation is a critical component of MDA. The Query/Views/Transformations (QVT) is the current standard compatible with the MDA recommendation suite (UML, MOF, and OCL). Transformations take model specifications as input (the SLA contracts expressed in the SLA language, in our case) and perform actions upon them, resulting in a different model specification (the contract validation clauses, in our case). Several model transformation languages are presently available (e.g. GReAT [30], UMLX [31], VIATRA [32], MOLA [33], ATL [34], Kermeta, Stratego/XT, MT [35] and Tefkat), with different levels of compliance to the QVT standard. In the present research work the transformations are exogenous, i.e. their source and target metamodels are different.

V. Environment Architecture

To illustrate the feasibility of our approach for SLA specification and compliance validation upon ITSM services modeled in BPMN, we are building the *SLA specification and Validation Environment* (SLAVE). Its architecture addresses two main stages in the SLA life cycle: the SLA specification stage, when a *negotiation phase* takes place among the stakeholders; and the *validation stage*, which comprises the SLA life cycle phases of monitoring, reporting, evaluating and improving [14]. Each of those stages is implemented by a different subsystem, schematically depicted by the component diagrams in Figure 3 and Figure 4. Grayed components are produced by a third-party.

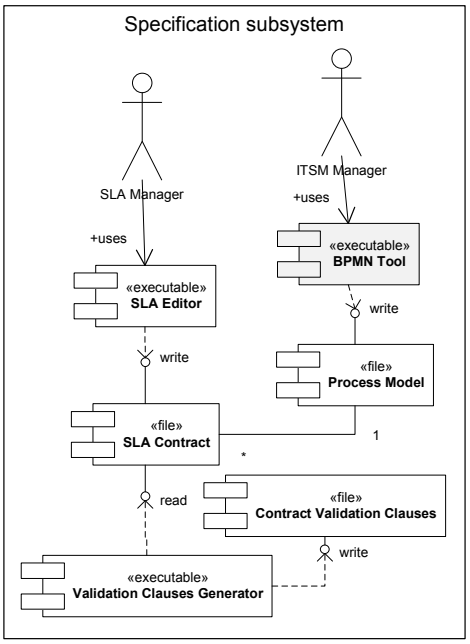

Figure 3. SLAVE (Specification subsystem)

[1] - The ECore has been defined in the Eclipse Modeling Framework (http://www.eclipse.org/modeling/emf/)

Our approach considers that ITSM processes were previously modeled with BPMN, by the *ITSM Manager* role, with the support of a BPMN modeling tool, as represented in Figure 3. The resulting *Process Model* can be persisted in a standard format (e.g. XMI).

The *negotiation phase* aims to define the conditions of the SLA contract, agreed between the IT service provider and the customer, and expressed with the SLA language, which will be inspired in techniques of NFRs elicitation [36]. An *SLA Editor* supports this activity. This tool is used by the SLA Manager role and serves to specify the SLA terms. This editor will be built using a DSL tool.

The output of the SLA Editor will be an *SLA Contract* complying with the SLA metamodel, with hooks on the corresponding BPMN process model. Those hooks specify where in the process a specific contractual clause (which will refer to given SLA quality attributes) applies (e.g. diagram elements such as activities, gateways or data objects). The *Validation Clauses Generator* is a component that performs a transformation between SLA language and OCL and it will also be built with a DSL tool. This generator takes the *SLA Contract* (expressed in the SLA language) as input and generates the *Contract Validation Clauses* (expressed in OCL) as output.

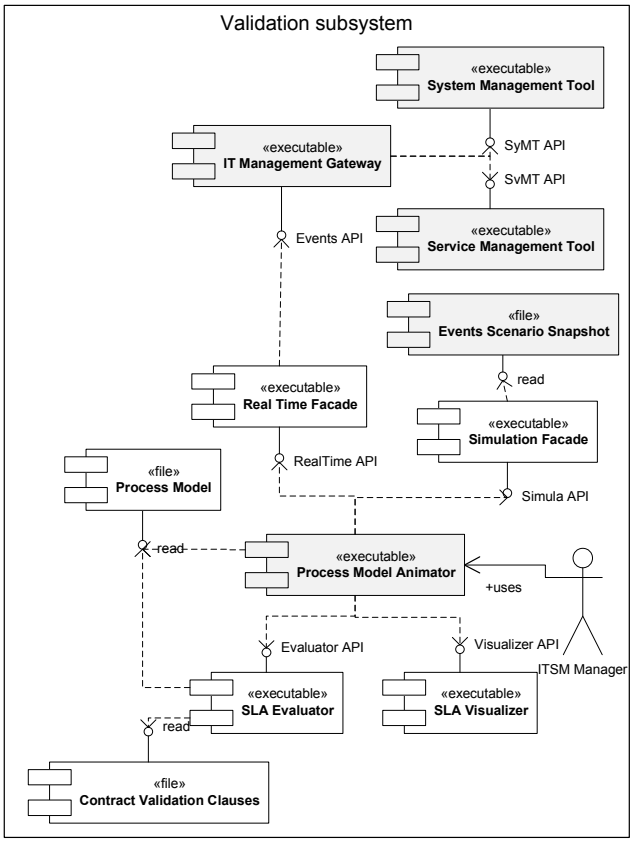

Figure 4. SLAVE (Validation subsystem)

In the validation stage (Figure 4), the *ITSM Manager* role interacts with the *Process Model Animator*. The latter allows observing the dynamics of the execution of the ITSM process on a BPMN diagram. To do so it takes the *Process Model* as input and processes its events received from one of two façades [37]. The *Real-Time Façade* is a front-end to an existing *IT Management Gateway* that performs a centralized processing of events coming either from *System Management Tools* (e.g. network management, software distribution) or *Service Management Tools* (concerning the execution of ITIL processes such as incident management or problem management). The *Simulation Façade* reads previously recorded *Event Scenario Snapshots*. The latter can be artificially generated (e.g. taking as basis the process model) or created by logging *IT Management Gateway* sessions and storing events in a standard format such as CIMOM [38], WBEM [39] or a CMDB [40].

The *Process Model Animator* will have two plugins: the *SLA Evaluator* and the *SLA Visualizer*. The former is basically an OCL interpreter that is able to check the *Contract Validation Clauses* against the instantiated *Process Model*, thereby being able to deliver information regarding SLA conformance or non-conformance. The SLA Evaluator is the cornerstone of the SLAVE environment since it joins the specification and validation stages.

Finally, the *SLA Visualizer* will extend the *Process Model Animator* with graphical views (e. g. dashboard or balanced scorecard) upon the running or simulated IT service processes. For instance, they will be filtered by SLA fulfillment or non-fulfillment in a specified timeslot.

VI. VALIDATION CONTEXT

We have established a joint research project between our research center and an industrial partner specialized in producing financial software, whose main operational processes will provide an adequate background for this work. In the realm of this joint project we will participate in the conception of a new events management system for financial self-service terminals networks (ATMs, kiosks and POS). Events management is concerned with faults or malfunctions related to the use or operation of the terminals, either by final users or by support team members (supervisors, maintenance and replenishment agents and helpdesk). Events are caused by total or partial unavailability of terminals (due to hardware or software problems), as well as other causes such as security breaks, communications failure or inadequate user operation.

The project's broad scope will extend the current version of a solution, which manages hundreds of ATMs in Portugal, to enable its simultaneous use, spanning several countries, with tens of thousands of terminals. An innovative approach to the delivery and processing of messages will take place in this project. The goal is to implement a model where the dispatching of events will result from the conjunction of rules, namely settled in service level agreements contracts. This will differ from solutions deployed in the market today, that are based in the paradigm of division of the total of terminals in groups, by one or two criteria, allocating each group to one professional helpdesk.

VII. Conclusions and Future Work

The main expected contribution of this research work is a model-based approach to IT services SLA specification and compliance verification. The specification part will encompass the proposal of a domain-specific language for expressing SLAs in the context of ITSM – the *SLA language*, for short. Its metamodel will be an extension of the metamodel of the adopted process modeling language, which is BPMN. As such, it will be possible to ground contractual clauses on the corresponding IT service model constructs.

Regarding the SLA compliance verification, a process model animation tool will be used, fed by the events that will be captured by IT service and system management tools. The process model animator will be integrated with the SLA evaluator to allow visual information related to current levels of service being provided, namely when SLA limits are exceeded. Therefore, process execution and SLA compliance validation will be made at the level of abstraction that is understood by all the stakeholders involved in IT service specification and deployment.

Acknowledgment

The work presented herein was partly supported by the *VALSE project* of the CITI research center within the Department of Informatics at FCT/UNL in Portugal.

References

[1] ITIL3Sm, OGC-Office of Government Commerce: *Summary, ITIL Version 3*, ITSMF- IT Service Management Forum, 2007.

[2] ISO20000-1, *Information technology -- Service management -- Part 1: Specification;* ISO20000-2, *Information technology -- Service management -- Part 2: Code of practice*, International Organization for Standardization, 2005.

[3] MS, M.C.-. *Microsoft Operations Framework (MOF) 4.0*, Microsoft Corporation, 2008.

[4] Lindquist, D., Madduri, H., Paul, C.J., and Rajaraman, B.: 'IBM Service Management architecture', *IBM Systems Journal*, 2007, 46(3), pp. 423-440.

[5] Sallé, M.: '*IT Service Management and IT Governance: Review, Comparative Analysis and their Impact on Utility Computing'*, Hewlett-Packard Research Labs, 2004.

[6] TMF, TeleManagement Forum: *Enhanced Telecom Operations Map® (eTOM) - The Business Process Framework - Version 7.1*, 2007.

[7] BPMN, OMG: '*Business Process Model and Notation (BPMN) '*, dtc/2009-08-14, 2009.

[8] Bon, J.v., Pieper, M., and Veen, A.v.d. (eds.): *Foundations of IT Service Management: Based on ITIL, ITIL Version 2*, Van Haren Publishing, 2005.

[9] Mayerl, C., Vogel, T., and Abeck, S.: 'SOA-based Integration of Service Management Applications', *Proc. of the IEEE International Conference on Web Services*, 2005.

[10] *http://www.pinkelephant.com/pinkverify/pinkverifytoolsetv3.htm*, accessed on 30-06-2010.

[11] Gläßer, L.: '*Development and Operation of Application Solutions 2005/2006'*, Siemens Business Services, 14.09.2005, 2005.

[12] Mylopoulos, J., Yu, E., Nixon, B.A., and Chung, L.: *Non-Functional Requirements in Software Engineering*, Springer, 1st edn, 0792386663, 1999.

[13] Kassab, M.: *Non-Functional Requirements: Modeling and Assessment*, VDM Verlag, ISBN 3639206177, 2009.

[14] ITIL3D, OGC-Office of Government Commerce: *Service Design, ITIL Version 3*, ITSMF- IT Service Management Forum, 2007.

[15] ITIL3C, OGC-Office of Government Commerce: *Continual Service Improvement, ITIL Version 3*, ITSMF- IT Service Management Forum, 2007.

[16] Coyle, D.M., and Brittain, K.: '*Magic Quadrant for IT Service Desk'*, Gartner, 2009.

[17] Wedemeyer, M., and Menken, I.: 'The ITIL V3 Service Management Awareness Pocket Guide': (The Art of Service, 2007).

[18] MOF: '*Meta Object Facility (MOF) Core Specification - Version 2.0'*, No. formal/06-01-01, OMG-Object Management Group, January, 2006.

[19] UMLs, OMG-Object Management Group: *OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2,* 2007.

[20] OCL, OMG- Object Management Group: *OCL - Object Constraint Language Version 2.0,* 2006.

[21] Röttger, S., and Zschaler, S.: 'Tool support for refinement of non-functional specifications', *Software and Systems Modeling Journal (SoSyM) 6(2),* 2006, pp. 185–204.

[22] Bass, L., Clements, P., and Kazman, R.: *Software Architecture in Practice*, Addison Wesley, 2nd edn, 0-321-15495-9, 2003.

[23] MDA, OMG: '*Model Driven Architecture (MDA)'*, No. ormsc/2001-07-01, July 9, 2001, 2001.

[24] Frankel, D.S.: *Model Driven Architecture Applying MDA to Enterprise Computing*, Wiley Publishing, Inc, 0-471-31920-1, 2003.

[25] Mellor, S.J., Scott, K., Uhl, A., and Weise, D.: *MDA Distilled: Principles of Model-Driven Architecture*, Addison Wesley, 0-201-78891-8, 2004.

[26] Chung, L., Nixon, B.A., Yu, E., and Mylopoulos, J.: *Non-Functional Requirements in Software Engineering*, Boston, Kluwer, 2000.

[27] Cysneiros, L.M., Leite, J.C.S.d.P., and Neto, J.d.M.S.: 'A Framework for Integrating Non-Functional Requirements into Conceptual Models ', *Requirements Engineering*, 2001, Springer-Verlag London Limited(6), pp. 97-115.

[28] Langlois, B., Jitia, C.-E., and Jouenne, E.: 'DSL Classification', *Proc. of OOPSLA'07 – Domain-Specific Modeling Workshop*, October 21-22, 2007, 2007.

[29] Kalnina, E., and Kalnins, A.: 'DSL tool development with transformations and static mappings', *Proc. of MODELS 2008 Workshops - Doctoral Symposium*, Toulouse, 29.9.2008, 2008.

[30] Balasubramanian, D., Narayanan, A., vanBuskirk, C., and Karsai, G.: 'The Graph Rewriting and Transformation Language: GReAT', *Proc. of the Third International Workshop on Graph Based Tools (GraBaTs 2006), Electronic Communications of the EASST, Volume 1*, 2006.

[31] Willink, E.D.: '*UMLX : A graphical transformation language for MDA'*, , Proc. 18th Annual ACMSIGPLAN Conf. on OOPLA, Anaheim,CA(2003), pp. 13–24, 2003.

[32] OptXware: '*The Viatra-I Model Transformation Framework Pattern Language Specification'*, Technical report August 2006.

[33] *http://mola.mii.lu.lv/, accessed on 30/06/2010*

[34] *http://wiki.eclipse.org/index.php/ATL*, accessed on 30-06-2010.

[35] Tratt, L.: '*The MT model transformation language'*, Technical report TR-05-02, Department of Computer Science, King's College London, 2005.

[36] Matoussi, A., and Laleau, R.: '*A Survey of Non-Functional Requirements in Software Development Process'*, Technical report TR-LACL-2008-7, Departement d'Informatique, Université d Paris 12, 2008.

[37] Gamma, E., Helm, R., Johnson, R., and Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[38] *http://www.dmtf.org/standards/cim/*, accessed on 30-06-2010.

[39] *http://www.dmtf.org/standards/wbem/*, accessed on 30-06-2010.

[40] *http://www.dmtf.org/standards/published_documents/DSP0252_1.0.0c.pdf, accessed on* 30-06-2010.