A comprehensive survey on intra and inter organizational agreements

Ikbel Guidara, Tarak Chaari, Kaouthar Fakhfakh, Mohamed Jmaiel

A comprehensive survey on intra and inter organizational agreements

Ikbel Guidara[1-2-3]
[1]CNRS-LAAS
[2]Université de Toulouse; UPS, INSA, INP, ISAE; UT1, UTM, LAAS
7 avenue du colonel Roche, F-31077
Toulouse Codex 4, France
iguidara@laas.fr

Tarak Chaari[3], Kaouthar Fakhfakh[3] and Mohamed Jmaiel[3]
[3]ReDCAD Laboratory, University of Sfax
B.P. 1173, 3038 Sfax, Tunisia
{tarak.chaari, kaouthar.fakhfakh}@redcad.org
mohamed.jmaiel@enis.rnu.tn

***Abstract*—With the continued growth of Service Oriented Architectures (SOA), cross-organizational enterprise collaboration is needed to satisfy complex client needs. To define dependencies between collaborative enterprises and guarantee the required Quality of Service (QoS) at different layers of the SOA, agreements present an effective solution. In fact, using agreements at different layers of SOA gives a clear specification and control of the agreed service levels between different stakeholders involved in the composition. In this context, aggregation functions that allow the management of composite agreements from the atomic ones at the same layer and translation techniques to detect dependencies between agreements at multiple layers are needed. In this paper, we give a survey on intra and inter organizational agreements for enterprise collaborations. We start by presenting the motivation of this paper and give an overview of agreements lifecycle and categories. Then, we give a state of the art of agreements composition approaches. After that, we highlight the techniques used to manage dependencies between agreements composition and we finish this paper by giving a synthetic view of existing works.**

Keywords- Agreements; Service composition; cross-organizational enterprises; multi-layer architecture

I. INTRODUCTION

Nowadays, to satisfy growing client needs, inter-enterprises collaboration is necessary to provide new value added services. In fact, to ensure the required higher QoS levels while offering low cost services, providers usually collaborate together and compose their services to form inter-organizational virtual enterprises. This requires the collaboration between multiple processes, services and partners. Each organization provides services to different customers and requires services from other organizations. Opening organizational boundaries aims to promote building, integrating and maintaining large-scale business applications.

In this context, SOA becomes an architectural style used by multiple enterprises to enhance the management of services across multiple layers: Business layer, Service layer and Infrastructure layer [1]. Business layer is used to define business context and requirements understandable for business people. Service layer allows the specification of services and components needed to realize business layer objectives. Infrastructure layer offers infrastructures which are necessary for the execution of services described in the previous layer. Recently, SOA wasn't applied only to structure service stack within the organization but also between multiple organizations.

Nowadays, enterprises are outsourcing their business applications to enable and automate cross-organization collaboration. To get control over the use and the delivery of services and to supervise relationships between multiple organizations and ensure the performance of the outsourced services, partners often use agreements. In fact, agreements allow the specification of common collaboration parameters between service consumers and producers within the involved parties. They also specify guarantees and obligations of each party. Managing agreements between organizations throughout the different phases of their lifecycle is a very important issue. In particular, managing agreements at multiple architectural layers is actually an active research filed. Two types of techniques are used: top-down approaches [2] that allow deducting lower-layer metrics from higher-layer objectives and bottom-up approaches [3] which are generally used to compute high level indicators from lower level parameters.

Based on the aforementioned motivation, we believe that it is necessary to manage agreements (i) horizontally within each layer (e.g. agreements between partners involved in a service composition at the service layer) and (ii) vertically across multiple layers (e.g. agreements between partners at the business layer and those at service layer) within and across organizations. Obviously, in IT environment where services are outsourced, it is crucial to rely on rigorous agreements management methodologies to get control over services delivery and use. In this paper, we present a state of the art of: (i) standards used to specify agreements, (ii) agreements composition approaches, (iii) techniques used to manage dependencies between agreements composition and finally we give a synthetic view on the existing works.

II. AGREEMENTS LIFECYCLE AND CATEGORIES

A. Agreements lifecycle

An agreement is generally defined as a contract between a consumer and a producer. It specifies guarantees expected by different stakeholders in order to ensure the required

quality of the provision. An agreement can describe functional and non-functional characteristics that are common to providers and consumers. To give a rigorous management of agreements between different stakeholders, multiple steps in agreements lifecycle [4] have to be taken into account. We give a short overview of these steps in the remaining parts of this section.

1) Specification

A model or a template of the agreement has to be specified in order to be used along all the steps of the agreement lifecycle. Generally, this model defines clauses that constitute the agreement. The main components of an agreement are: *parties*, *obligations* and *penalties* [5].

2) Negotiation

This phase allows the negotiation of the agreement in order to establish the appropriate contract between the involved parties. It includes negotiation of cost, functional properties, non-functional properties and guarantees in case of violation. The negotiation phase requires the use of negotiation strategies and common protocols that should be adopted by the involved parties.

3) Establishment

This step consists in establishing and validating the negotiated agreement. In this step, the objectives and penalties are clearly specified and signed by the involved parties.

4) Execution

The execution step consists in monitoring the agreement throughout the service execution in order to detect violations and take the appropriate actions and penalties.

5) Termination

This phase handles the events and the actions to take when the agreement is about to finish after its expiration or violation.

B. Agreements modeling

Multiple agreement modeling languages have been specified in the literature. Web Service Level Agreement language (WSLA) [6], presents three main concepts: parties, services definition and obligations. It is based on *Service Level Objectives (SLOs)* to guarantee the SLA's state for a determined period and on *actions guarantee* to take in case of contract violation. This language defines third parties which can be sub-contracted by signatory parties; it considers only two signatory parties. In SLAng language [7], an SLA captures client and provider *responsibilities* and mutual responsibilities to define obligations of each party. It contains *Service Level Specifications (SLSs)* to present technical QoS metrics. SLAng takes into account multiple layers in technical domain (storage, network, middleware and application). It defines seven types of SLAs (three horizontal SLAs between entities in the same architectural layer and four vertical SLAs between entities in different layers). SLAng provides only a limited set of domain-specific QoS parameters. WS-Agreement [8] allows the specification of SLA templates. It presents three components of an agreement: *agreement name, agreement context* and *agreement terms* (*Service Terms* to define the functional attributes of the agreement and *Guarantee Terms* to define non functional attributes). This language presents the advantage of being easily extended and domain independent specification. WSOL [9] is the most used language for the description of service offerings. It is based on WSDL and it defines multiple levels of services. WSOL enables the specification of QoS properties, metrics and constraints (e.g., non functional constraints, access rights constraints...). It allows the definition of multiple service levels and the specification of third parties and penalties.

All the previous studied languages do not allow the specification of multiple signatory parties in the SLA which is not suitable with the requirement of new enterprises collaboration. In this new emergence, multiple parties can collaborate together and sign the same SLA. For this reason, we focus on the main two works which have extended existing languages to support multiple parties.

WSLA+ [10] extends WSLA language to allow the dynamic collaboration between multiple parties instead of bipartisan collaboration. This extension is based on four main concepts: *multiple parties, multiple services* are offered by parties involved in the collaboration, each party in can play *multiple roles* instead of a single role and *multiple agreements* can be defined between collaborators to achieve the whole contract. GSLA [11] is a role-oriented contract between two or more parties (e.g. provider, subcontractors). Each role characterizes the behavior of each party in the SLA. GXLA is an XML-based language which implements the GSLA information model. It allows the automatic service negotiation and the translation of SLA agreements into SLOs by defining the role of each party and the policies needed to guarantee the contract.

Despite the described extensions above, the major parts of the existing agreement specification languages do not consider the establishment of the agreement at the business layer and they just focused on specifying agreements at the service layer.

C. Agreements categories

Different types of agreements have been specified in the literature. According to IBM [12], there are three types of agreements used between business and IT layers which are:

- *BLA (Business Level Agreement)*: agreement which states business objectives between two business partners participating in the business process (e.g. business value creation and revenue generation)
- *BSLA (Business Service Level Agreement)*: agreement which states operational objectives used to operate the business (e.g. business hours spent on a given task)
- *SLA (Service Level Agreement)*: agreement which states IT operations objectives to measure the performance of technical applications (e.g. service response time)

ITIL[1] has proposed another classification of agreements:

[1] http://www.itil-officialsite.com/

- *SLA (Service Level Agreement)*: agreement between a service consumer and a service producer to manage consumer expectations (e.g. response time)
- *OLA (Operational Level Agreement)*: agreement between teams within the same organization to manage operational objectives related to the service delivery (e.g. availability)
- *UC (Underpinning Contract)*: agreement between service providers and external suppliers or third parties which support providers to deliver their services and goods.

In this section, we gave an overview of the different steps of agreements lifecycle, then, we presented the most used agreement modeling languages and categories. In the next section, we discuss agreement composition approaches at both single and multiple layers.

III. AGREEMENT COMPOSITION APPROACHES

Several works in the literature have focused on the management of agreements signed between different partners in order to offer a composite service. These approaches can be classified into two categories: single-layer approaches which have treated the management of agreements composition at a single layer and multi-layer approaches which have studied the composition of agreements in multiple layers (e.g. business layer, service layer and infrastructure layer).

A. Single-layer composition approaches

COSMA [13] is a framework for dynamically managing SLAs in a service composition. Two types of SLAs are defined in this approach: composite SLA (CSLA) between the consumer and the provider of the composite service and atomic SLA (ASLA) between the composite service provider and each provider of an atomic service participating in the service composition. Although this framework allows the management of atomic and composite SLAs, collaboration between different partners is not taken into account and a simple orchestration between the composite service provider and each atomic service provider is adopted. Defining SLAs at a single level of the composition is very restrictive especially in actual enterprises collaborations when each partner can be a consumer and a producer at the same time. In [14], authors proposed a hierarchical SLA aggregation approach taking into account multi-hierarchical levels. Each partner can sign SLA as a producer or/and a consumer. In this work, multiple levels are considered to give partners the opportunity to outsource services from other partners. However, this approach does not take into account collaboration relationships between providers at the same level. Winkler et al [15], proposed an approach to automatically manage and validate composite SLAs in services composition. They define a dependency model to handle dependencies between SLAs. This model is then used to validate and evaluate SLAs during the negotiation. Despite that this work takes into account dependencies between services in the composition, it does not give methodologies to generate SLAs composition. Blake et al, [16] have also highlighted the importance of SLAs composition associated with web service workflow compositions. They defined three principles which are *compliance*, *sustainability* and *resiliency*. Their approach is limited to predefined aspects and measures. In [17], authors defined an approach to convert agreement of a composite service into individual agreements for elementary services by exploring dependencies between them. Two types of dependencies are defined: explicit dependencies between services in the composition and implicit dependencies which are related to infrastructural services.

All the previous approaches define SLAs at a single-layer and do not support multi-layer SLAs management. With these approaches, service providers con not identify metrics and parameters at the lowest layer that can have impacts on metrics and parameters at the highest layers and vice versa.

B. Multi-layer composition approaches

As mentioned in the previous section, specifying SLAs only at the top layer, do not allow service providers to control and manage their IT services since they cannot deduce IT service metrics from business parameters. In this section, we give a review of multi-layer agreements composition approaches.

In the context of the European Research Project SLA@SOI, Comuzzi et al [18] proposed multi-layer SLA management approach. The implemented SLA hierarchy captures dependencies between three layers: business layer, software layer and infrastructure layer. This allows providers to manage their internal services in different layers. In [19], authors defined a framework for multi-layer SLA management. They presented a reference architecture to manage multi-layer SLAs throughout the overall agreement lifecycle. These two works are interested in the translation of metrics from business layer to the lower layers of the SOA stack. They especially focus on managing dependencies between departments within the same organization and do not consider composite SLAs between multiple organizations. In [20], authors presented a generic framework to manage multi-layer SLA across four layers: business, application, middleware and infrastructure. They defined an SLA management architecture to manage SLAs throughout all phases of SLA lifecycle. In this work, authors gave an overview of SLA terms for each layer as well as a set of translation techniques which can be adopted to translate these terms from one layer to another.

In [21], Juan et al proposed a multi-tier industrial architecture approach that allows converging business process management to SOAs management. Three types of agreements are defined in this work: BLA to control the business quality required by the customer; SLA to define services and measurement parameters; and ALA to measure the performance of the application layer. Based on indicators at each level, the authors proposed to govern and monitor the performance of the entire business process horizontally and the performance of each task of the business process across the different layers vertically. This approach is based on reports provided by each level indicator to locate and correct violations. In [22], Bratanis et al extended SLAs to define BLAs to guarantee the required business level. They

proposed a monitoring approach of behavioral and qualitative aspects of business services during the execution of web services in order to evaluate their implementations and satisfy the quality level required at the business layer. Nevertheless, these works are mainly focused on the monitoring phase and don't cater for the specification and the translation of agreements across different layers.

Nowadays, agreements are considered also as an important mean to align business processes to IT services and improving assessment of services composition management. In [23], authors defined SLA as a mean that allows the formalization and the evaluation of the relationship between business processes and IT services and the measurement of the alignment level between them. Other works [24] have not used agreements when mapping business processes with IT layers. This limits the benefits of using agreements which are very useful to make decisions, to specify the behavior at the business layer and IT service layer and to evaluate the performance of the alignment.

IV. AGREEMENT DEPENDENCIES MANAGEMENT TECHNIQUES

In this section, we give an overview of different approaches used to manage dependencies between SLAs involved in a service composition according to two main categories: approaches that focus on SLA parameters aggregation functions to detect dependencies between SLAs in a single layer and those that have defined translation methods to map agreements across multiple layers.

A. Single-layer aggregation techniques

In the framework adopted by [13], a generic SLA document is specified. This document contains contractual information about atomic and composite SLAs and composite-specific dependencies and relationships between them. To manage parameters of involved SLAs, the authors defined aggregation formulas based on composition patterns (e.g. sequence, parallel or loop). For instance, response time value in the composite service is the sum of response time values of atomic services if there are sequentially executed. An extension of the WS-Agreement language has been defined in [14] in order to take into account different aggregation patterns. To aggregate multiple metrics into one global parameter, authors defined different aggregation functions such as sumtype, maxtype and mintype. In addition, they used logical functions (e.g. and, or, xor) to form rule-based aggregation expressions to manage SLOs and guarantee terms. In these two works, complex SLA composition patterns cannot be applied (e.g. a parameter value may depend on multiple values of other parameters in other SLAs). In [15], Winkler et al proposed several functions to specify dependencies between SLAs in service compositions. Four types of dependencies are used which are Price, Location, QoS and Time dependencies. Multiple formulas and operators are specified to manage these dependencies (e.g. time dependencies are managed using time operators such us finish-to-start, start-to-start and before). To compute composite SLA measures, Blake et al [16] used functions to compose parameters of all agreements which have dependencies with the composite SLA. However, only a limited and predefined set of measures is taken into account. In [17], Constantinos et al defined two graphs to manage dependencies between metrics: services dependency graph and properties dependency graph. In fact, properties of a dependent service may be related to different properties of its antecedent services in the composition. For these reasons, Constantinos et al proposed to use functions and rules in order to explore and map each property to its dependent properties. Nevertheless, they do not explain how they define these functions and how to compute the composite properties according to other atomic parameters.

B. Multi-layer translation techniques

To translate SLA parameters across layers, Comuzzi et al [18] use Palladio Component Model (PCM) [25] to predict service terms and simulate non-functional behavior across layers. However, in this work authors do not give any details on how the PCM is used to predict terms from business layer to the lower layers. In [20], authors highlighted the most important SLA terms translation techniques across the service hierarchy. For instance, with regard to the workload term, authors focus on users' profiles to translate the users' classes (e.g. single user, department) to expected transaction rates. These rates, which are specified in the application layer, are then translated into a middleware load characterization using an application-based benchmark called SAPS measure [26]. Finally, this measure is also used to translate middleware workloads into infrastructure specifications. Multiple other SLA terms translation methods have been proposed such us the queuing model [2], statistical methods [27] and simple aggregation functions which can be used to aggregate energy or cost terms. In this work, two types of translation are specified: top-down translation to derive lower-level SLA terms from higher-level ones (e.g. workload) and bottom-up translation to predict higher-level SLA terms from the lower-level ones (e.g. response time).

In [28] multiple SLA translation techniques in multi-layered SOAs have been discussed. Authors proposed four types of translations: configuration to configuration (C2C), metric to configuration (M2C), configuration to metric (C2M) and metric to metric (M2M) which can be used within one layer or between multiple layers. Based on these translation types, authors discussed multiple SLA translation methodologies classified in three main categories. *Knowledge and rule based approaches* which are based on policy translation techniques, QoS mapping and semantic rule-based translation techniques [29]. These approaches are mainly based on rules to make deductions and choices. *Queuing-analytic model based approaches* which are generally top-down approaches that translate SLOs into low-level parameters [2]. In this work, the decomposition of SLAs is formulated as a constraint satisfaction problem. *Statistical learning based approaches* which are generally based on statistics and machine learning techniques to predict SLOs from system metrics values [27].

All the previous approaches treated either the top-down translation or the bottom-up prediction techniques and not both of them.

In [30], Branimir et al proposed an SLA-aware service compositions management based on Key Performance Indicators (KPIs) which specify the performance of Business Processes. Authors defined techniques to aggregate SLO values in order to estimate QoS properties (e.g. benchmarking [31] and capacity planning [32] techniques). Nevertheless, they don't provide top-down techniques to map KPIs to SLOs which guarantee the values of service level parameters.

V. DISCUSSIONS

In this section, we give a summary on the existing approaches that we have mentioned in the previous sections. Table 1 shows a classification of these approaches according to four main criteria:

- Layers: specify if the management of the agreements has been taken at single or multiple layers.
- Dependencies techniques: specify the technique used in the composition or the translation of the agreements.
- Agreement lifecycle: specify which agreement lifecycle steps are taken into account.
- Specification language: specify which language is used to specify agreements.

TABLE I. SUMMURY OF THE EXISTING APPROACHES

	Layers	**Dependencies techniques**	**Agreement lifecycle**	**Specification language**
[13]	Single layer	Aggregation formulas	All the lifecycle	Extends WS-Agreement (Monitoring Negotiation)
[14]	Single layer	Aggregation and logical functions	Specification and validation	Extends WS-Agreement (Typea)
[15]	Single layer	Time, location operators Aggregation formulas	Validation and monitoring	-
[16]	Single layer	Composition functions	Specification and negotiation	Extends WSDL (SLA annotations)
[17]	Single layer	Dependency graph	Specification and negotiation	Extends WS-Agreement (counter-offers)
[18]	Multiple layers	Palladio Component model (PCM)	All the lifecycle	Extends WS-Agreement (counter-offers)
[19]	Multiple layers	-	All the lifecycle	Extends WS-Agreement (counter-offers)
[20]	Multiple layers	Queuing model Statistical methods SAPS measure	All the lifecycle	Extends WS-Agreement (counter-offers)
[22]	Multiple layers	-	Monitoring	-
[30]	Multiple layers	Benchmarking and capacity planning techniques	Specification and monitoring	-

On the first criterion, we can conclude that several existing works [13, 14, 15, 16, 17] lack an efficient agreement management approach across multiple layers. These approaches generally adopt simple composition and aggregation functions to manage dependencies horizontally at the same layer (second criterion). This presents limits and seems insufficient in SOA environments in which interactions between multiple parties at different layers can be required. Other works like [18] and [20] used techniques such as PCM and SAPS measure to translate metrics and parameters of agreements from higher layers to lower ones (top-down approach). In [30] authors used benchmarking and capacity planning techniques to estimate QoS parameters from SLO values (bottom-up approach). These approaches generally handle a single direction transformation and don't provide a generic framework that takes into account both top-down and bottom-up approaches. In addition, they don't manage agreements horizontally at the same layer and they are restricted to manage agreements between multiple layers. This is very restrictive since agreements involved in the composition at the same layer may influence other agreements at other layers. Moreover, there is a considerable heterogeneity between terms and indicators used at multiple layers (business, service and application). In fact, experts at each layer may not understand parameters of other layers. According to the previous study, the existing approaches use syntactic techniques and mathematical functions to manage and translate agreements across multiple layers. This cannot effectively resolve the problem of heterogeneity due to the lack of semantics. On the third criterion in table 1, we notice that the majority of existing works don't address all the phases of agreements lifecycle and they limited their researches to one or two steps [14, 15, 16, 17, 22, 30]. On the last criterion, the most used language to model agreements is WS-agreement. This is justified by the fact that this language is very easy to extend. However, we note that a model that specify agreements at the business layer is not yet clearly defined and only the specification of agreements at service layer is taken into account.

VI. CONCLUSION

In a service composition, agreements between involved parties have multiple dependencies between each other across multiple layers. In fact, the separation between multiple layers offers a coherent agreement modeling, reduces the complexity of the management of the agreements involved in the composition and facilitates the adaptation of services in case of agreements violations. Managing agreements at different layers requires techniques to generate and evaluate dependencies between them. Once the composition is generated, methods that allow the assessment, the monitoring and the global coherence of these agreements are very important. In fact, in many cases, if a violation of an agreement at the lower level has not been addressed appropriately and on time, this may affect other agreements that depend on it at the same layer or at other layers.

According to the survey we made on the existing works on the composition of agreements between multi-organizational collaborations, we conclude that their

management across multiple layers is not sufficiently explored. In fact, the major part of the translation techniques between these layers is based on mathematical and syntactical aggregation. In addition, generally existing works treated either the top-down translation or the bottom-up prediction techniques and not both of them.

In our ongoing work, we plan to implement a generic framework that allows the management of agreements throughout all the steps of their lifecycle. Our aim is to assist business decision makers to automatically generate and manage a composition of agreements to fulfill end-to-end functional and non functional requirements across multiple layers. To build this framework, it is crucial to define an expressive and rigorous agreements specification models to clearly describe functional and non-functional properties. After that, we need to define a complete chain that allows building complex agreements, selecting the most suitable ones, negotiating the incompliant requirements, and finally choosing the most suitable composite agreements and evaluate this composition. We intend to take into account both top-down techniques to specify and generate agreements at different layers and bottom-up techniques to detect violations of agreements at the lowest layer and their impact on other agreements at the same layer or at the highest layers.

REFERENCES

[1] T. Mitra, Business-driven development, IBM, Software Group, http://www.ibm.com/developerworks/webservices/library/ws-bdd/ (2007)

[2] 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).

[3] Y. Diao and K. Bhattacharya. "Estimating Business Value of IT Services through Process Complexity Analysis". In proc. of IEEE/IFIP NOMS, pp. 208-215, (2008).

[4] W. Sun, Y. Xu and F. Liu, "The Role of XML in Service Level Agreements Management. Network Management Research". Center, Beijing Jiaotong University, IEEE, (2005).

[5] K. Fakhfakh, T. Chaari, S. Tazi, K. Drira, M. Jmaiel, "Service level agreement modeling and monitoring using ontologies", Journées Francophones sur les Ontologies (JFO), France, pp.106-111, (2008).

[6] E. Keller and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services". Journal of Network and Systems Management, 11(1), (2003).

[7] D. D. Lamanna, J. Skene and W. Emmerich: "SLAng: A Language for Defining Service Level Agreements". FTDCS, (2003)

[8] A. Pichot, O. Wäldrich, W. Ziegler and P. Wieder, "Towards Dynamic Service Level Agreement Negotiation: An Approach Based on WS-Agreement". Web Information Systems and Technologies 4th International Conference, WEBIST (2008).

[9] V. Tosic, B, Pagurek, K. Patel, B. Esfandiari, and W. Ma, "Management applications of the web service offerings language" . Inf. Syst. 30, 7, pp. 564-586, (2005).

[10] S. Nepal, J. Zic and S. Chen, "WSLA+: Web Service Level Agreement Language for Collaborations". IEEE SCC (2), pp. 485-488, (2008).

[11] B. Tebbani and I. Aib, "GXLA a Language for the Specification of Service Level Agreements". Autonomic Networking, pp. 201-214, (2006)

[12] C. Newton and I. Salvage, "Beyond IT Service Management -bringing business and IT together". IBM Global Technology Services, (2007).

[13] A. Ludwig and B. Franczyk, "Managing Dynamics of Composite Service Level Agreements with COSMA". The 5th International Conference on Fuzzy Systems and Knowledge Discovery, (2008)

[14] I.U. Haq, A.A. Huqqani and E. Schikuta, "Hierarchical aggregation of Service Level Agreements". Data Knowl. Eng. 70 (5), pp. 435-447, (2011).

[15] M. Winkler, T. Springer and A. Schill. "Automating Composite SLA Management Tasks by Exploiting Service Dependency Information". The 8th IEEE European Conference on Web Service ECOWS, (2010).

[16] M.B. Blake and D.J. Cunnings. "Workflow composition of service level agreements". In: International Conference on Services Computing, SCC, pp.138-145, (2007).

[17] Constantinos Kotsokalis and Ulrich Winkler "Translation of Service Level Agreements: A Generic Problem Definition". ICSOC/ServiceWave Workshops, pp. 248-257, (2009).

[18] M. Comuzzi, C. Kotsokalis, C. Rathfelder, W. Theilmann, U. Winkler and G. Zacco, "A Framework for Multi-level SLA Management". ICSOC/ServiceWave Workshops. 187-196, (2009).

[19] W. Theilmann, J. Happe, C. Kotsokalis, A. Edmonds, K. Kearney and J. Lambea, "A Reference Architecture for Multi-Level SLA Management". Journal of Internet Engineering, Vol. 4, No. 1, (2010).

[20] W. Theilmann, U. Winkler, J. Happe, I. M. de Abril, "Managing On-Demand Business Applications with Hierarchical Service Level Agreements". FIS 2010, pp. 97-106, (2010).

[21] J. Li, F. Biennier and C. Ghedira, "An Agile Governance Method for Multi-tier Industrial Architecture", APMS 2011 (IFIP WG5.7), Stavanger –Norvège (2011)

[22] K. Bratanis, D. Dranidis, A. J. H. Simons, "Towards Run-Time Monitoring of Web Services Conformance to Business-Level Agreements". TAIC PART 2010, pp. 203-206, (2010).

[23] E. Silva and Y. Chaix, "Business and IT Governance Alignment Simulation Essay on a Business Process and IT Service Model". In Proceedings of the 41st Annual Hawaii International Conference on System Sciences, HICSS, (2008).

[24] M. Valiente, E. Garcia-Barriocanal and M. Sicilia, "Applying an ontology approach to IT service management for business-IT integration". Knowledge-Based Systems, pp. 76-87, (2012).

[25] S. Becker, H. Koziolek and R. Reussner, "The Palladio component model for modeldriven performance prediction". Journal of Systems and Software 82(1), pp. 3–22, (2009).

[26] A. Janssen and U. Marquard, "Sizing SAP Systems". SAP Press, ISBN 978-1-59229-156-4, http://www.sap-press.com/product.cfm?account=&product=H2904

[27] 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, pp. 231-244, (2004).

[28] H. Li, W. Theilmann, J. Happe, "Sla translation in multi-layered service oriented architectures: Status and challenges". Technical Report 2009-8, Universitat Karlsruhe (TH), (2009).

[29] 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, pp. 210-221, (2002).

[30] B. Wetzstein, D. Karastoyanova and F. Leymann, "Towards Management of SLA-Aware Business Processes Based on Key Performance Indicators", 9th Workshop on Business Process Modeling, Development, and Support (BPMDS'08), France, (2008).

[31] M. Marzolla and R. Mirandola, "Performance Prediction of Web Service Workflows", Proc. QoSA'07, Software Architectures, Components and Applications Third International Conference on Quality of Software Architectures, QoSA 2007, USA, (2007).

[32] D. A. Menascé, V. A. F. Almeida, "Capacity planning for web services: metrics, models, and methods". Prentice Hall (2002).