Advancements and State of the Art Analysis

Keywords:
State of the Art, service level agreement, negotiation, prediction, business continuity, service management, infrastructure management

Due date of deliverable: 31st July 2011
Actual submission to EC date: 3rd August 2011
Start date of project: 1st June 2008
Duration: 38 months

Lead contractor for this deliverable: UDO
### Document Editors

<table>
<thead>
<tr>
<th>Partner</th>
<th>Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDO</td>
<td>Ramin Yahyapour, Philipp Wieder, Peter Chronz</td>
</tr>
</tbody>
</table>

### Contributors

<table>
<thead>
<tr>
<th>Partner</th>
<th>Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG</td>
<td>Francesco Torelli, Paolo Zampognaro</td>
</tr>
<tr>
<td>SAP</td>
<td>Tariq Ellahi, Hui Li, Ulrich Winkler, Wolfgang Theilmann</td>
</tr>
<tr>
<td>INTEL</td>
<td>John Kennedy, Andy Edmonds</td>
</tr>
<tr>
<td>TID</td>
<td>Juan Lambea, Beatriz Fuentes, Miguel Alvarez Calvo, Oscar L. Dueñas</td>
</tr>
<tr>
<td>UDO</td>
<td>Ramin Yahyapour, Philipp Wieder, Costas Kotsokalis, Peter A. Chronz, Edwin Yaqub, Kuan Lu</td>
</tr>
<tr>
<td>FZI</td>
<td>Jens Happe, Christof Momm, Franz Brosch</td>
</tr>
<tr>
<td>FBK</td>
<td>Annapaola Marconi, Michele Trainotti, Marco Pistore</td>
</tr>
<tr>
<td>PMI</td>
<td>Luciano Baresi, Elisabetta Di Nitto, Liliana Pasquale, Sam Guinea</td>
</tr>
<tr>
<td>CITY</td>
<td>George Spanoudakis, Christos Kloukinas, Marco Comuzzi, Davide Lorenzoli</td>
</tr>
<tr>
<td>XLAB</td>
<td>Gregor Pipan, Gregor Berginc, Miha Vuk</td>
</tr>
<tr>
<td>eTel</td>
<td>Mark Evenson</td>
</tr>
</tbody>
</table>

### Notices

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law. Copyright 2011 by the SLA@SOI consortium.

* Other names and brands may be claimed as the property of others.

This work is licensed under a [Creative Commons Attribution 3.0 License](http://creativecommons.org/licenses/by/3.0/).
Executive Summary

This document is an appendix to Deliverable D.B1a, the "Scientific Evaluation Report", and presents an analysis of SLA@SOI-related state-of-the-art work.

It offers an overview and an analysis of the state of the art with regards to service level management. The contents are aligned with the perspective on service level management taken by SLA@SOI, targeting primarily IT-services throughout various organizational layers as described by D.A1a. The document has been updated throughout the lifetime of the project; new developments have been added as appropriate.

As an introduction an overview on the achieved innovations aligned with the work packages is presented. Furthermore the innovations are presented in more detail with regards to the advancement against the state of the art. The biggest part of the document provides summaries of the related work, which has been analysed and advanced during the course of this project. A list of the related work which has been taken into account is provided at the beginning of Section 3. Finally the work conducted in SLA@SOI is compared to the prominent standards defined by IT Infrastructure Library (ITIL). This comparison discusses the commonalities and differences between SLA@SOI's approach providing an automated software framework and ITIL's recommendations with regards to standardized IT-management processes.

The document is structured as follows. Section 1 provides an overview on the innovations. Section 2 lists the most important related work and the innovations per work package in more detail. Section 3 provides summaries of important related work. Section 4 compares SLA@SOI to ITIL. The appendix finally lists references cited in sections 1 and 2.
1 Introduction ......................................................................................................................... 13
  1.1 Overview ......................................................................................................................... 13

2 Project Advancements and Relevant Related Work .......................................................... 16
  2.1 WP A1 – Architecture and Integration ......................................................................... 16
  2.2 WP A2 – Business Management ................................................................................... 16
  2.2.1 The Information Phase ............................................................................................... 16
  2.2.2 Modelling Business Aspects ..................................................................................... 17
  2.2.3 Business Level Negotiations ..................................................................................... 17
  2.2.4 The Contracting Phase ............................................................................................... 17
  2.2.5 Penalty Management ................................................................................................. 18
  2.3 WP A3 – Service Management ..................................................................................... 19
  2.3.1 Enhanced SOA Modelling ........................................................................................ 19
  2.3.2 Manageability ............................................................................................................ 23
  2.3.3 Service Monitoring ..................................................................................................... 24
  2.4 WP A4 – Infrastructure Management ......................................................................... 24
  2.4.1 Model Standards ........................................................................................................ 25
  2.4.2 Infrastructure Virtualisation ...................................................................................... 25
  2.4.3 Infrastructure Platforms ............................................................................................. 25
  2.4.4 Infrastructure Monitoring .......................................................................................... 26
  2.4.5 Messaging .................................................................................................................. 26
  2.5 WP A5 – SLA Management Foundations .................................................................. 26
  2.5.1 SLA Modeling ............................................................................................................ 26
  2.5.2 SLA Negotiation ........................................................................................................ 27
  2.5.3 Planning and Optimisation ....................................................................................... 28
  2.6 WP A6 – Predictable Systems Engineering ................................................................ 29
  2.6.1 Software Performance and Reliability Prediction .................................................... 29
  2.6.2 Run-time SLA Violation Prediction ......................................................................... 29
  2.6.3 Manageability Modelling and Design ..................................................................... 30
  2.6.4 Resource Usage Prediction ...................................................................................... 30

3 General State of the Art Analysis .................................................................................... 32
  3.1 Shared Information and Data Model (SID) ................................................................. 38
  3.2 Web Services Agreement (WS-Agreement) ................................................................. 39
  3.3 Web Services Agreement Negotiation (WS-Agreement Negotiation) ..................... 41
  3.4 NextGRID Service Level Agreement ......................................................................... 42
  3.5 Work by Eid, Alamri and El Saddik on “A reference model for dynamic web service composition systems” ........................................................... 44
  3.6 DIANE - An Integrated Approach to Automated Service Discovery, Matchmaking and Composition ................................................................. 45
  3.7 Governance Interoperability Framework (GIF) .......................................................... 47
  3.8 CentraSite ..................................................................................................................... 48
  3.9 TM Forum SLA Management Handbook ..................................................................... 49
  3.10 TMForum Service Delivery Framework (SDF) Catalyst Project .............................. 51
  3.11 Web Services Agreement Language (WSLA) ............................................................ 52
  3.12 Web-based Enterprise Management (WBEM) Standards ........................................ 53
  3.13 CIM - Common Information Model ........................................................................... 55
  3.14 Web Services Distributed Management (WSDM) Standard .................................... 57
  3.15 Java Management Extensions (JMX) .......................................................................... 59
  3.16 OASIS Data Centre Markup Language (DCML) ...................................................... 60
  3.17 OMG System Modelling Language (SysML) ............................................................ 62
  3.18 GLUE Specification ..................................................................................................... 63
  3.19 Service Modelling Language (SML) .......................................................................... 65
  3.20 Common Model Library (CML) ................................................................................ 67
  3.21 ITIL Configuration Management Databases (CMDB) ............................................... 68
3.22 Making BPEL Flexible - Adapting in the Context of Coordination Constraints Using WS-BPEL
3.23 Towards Aspect Weaving Applications
3.24 A dynamic and reactive approach to the supervision of BPEL processes
3.25 ServiceCom: prototype tool for service composition specification, construction and execution developed by Tilburg University
3.26 OWL-S (Ontology Web Language for Services)
3.27 MoSCoE: (Modeling Web Service Composition and Execution)
3.28 MoDe4SLA
3.29 Service Centric System Engineering European Project – FP6
3.30 SCENE: To offer a language for composition design that extends the standard BPEL language with rules used to guide the execution of binding, re-binding, negotiation, and self-reconfiguration operations
3.31 EVEREST (EVEnt REaSoning Toolkit)
3.32 Dynamo (and related work)
3.33 Work by Ardissono et. al on “Fault Tolerant Web Service Orchestration by Means of Diagnosis”
3.34 Work by Lazovik et.al on “Business process management”
3.35 Colombo (IBM)
3.36 Work by Momm et.al. on “Model-driven Development of Monitored Processes”
3.37 Work by Roth et.al on "Probing and Monitoring of WSBPEL Processes"
3.39 VieDAME (and related work)
3.40 Work by Jurca, R., Binder, W., and Faltings, B. on "Reliable QoS Monitoring Based on Client Feedback"
3.41 Work by W.M.P. Van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and E. Verbeek, on “Conformance checking of Service Behavior”
3.42 Assumption-based monitoring of service compositions
3.43 Monitoring privacy-agreement compliance
3.44 Performance monitoring for utility computing
3.45 Cremona
3.46 Automated SLA monitoring
3.47 Query-based business process monitoring
3.48 Model-driven development for BPM
3.49 Intelligent Business Operation Management (iBOM)
3.50 MAPE-K (Monitor-Analyze-Plan-Execute-Knowledge)
3.51 HAGGLE (An innovative Paradigm for Autonomic Opportunistic Communication)
3.52 CASCADAS (Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services)
3.53 BIONETS (BIOlogically-inspired autonomic NETworks and Services)
3.54 ANA (Autonomic Network Architecture)
3.55 FOCALE: A Novel Autonomic Networking Architecture
3.56 SAHARA - Service Architecture for Heterogeneous Access, Resources, and Applications
3.57 PROSPECT- A Prospect of Multi - Domain Management in the Expected Open Services Marketplace
3.58 Policy-based management on multidomain/multiprovider environments.
3.59 perfSONAR (performance Service Oriented Network monitoring ARchitecture)
3.60 OVF Open Virtualisation Format
3.61 Libvrt
3.62 OpenNebula .......................................................... 147
3.63 Amazon Elastic Compute Cloud (Amazon EC2) .......... 148
3.64 XCalibre Flexiscale ............................................. 150
3.65 ElasticHosts ....................................................... 151
3.66 ServePath GoGrid .............................................. 153
3.67 Zimory ............................................................. 154
3.68 Nirvanix SDN .................................................... 156
3.69 Ganglia ............................................................ 157
3.70 SLA Evaluator (GT4 & .NET versions) ................. 159
3.71 Nagios ............................................................. 161
3.72 Groundwork - Monitor Community Edition .............. 162
3.73 MonALISA ......................................................... 163
3.74 Zabbix ............................................................. 165
3.75 Osmius - The Opensource Monitoring Tool ............ 167
3.77 Java Message Service (JMS) ............................... 170
3.78 AMQP – Advanced Message Queuing Protocol .... 171
3.79 CORTEX (CO-operating Real-time senTient objects: architecture and EXperimental evaluation) .... 172
3.80 Real-time reconfiguration for guaranteeing QoS provisioning levels in Grid environments .................................. 176
3.81 Autonomic policy-based management .................... 179
3.82 SWRL: Semantic Web Rule Language .................... 182
3.83 Rule-based and Ontology-based Policies .................. 184
3.84 Model-based Performance Prediction ..................... 187
3.85 Performance Prediction Techniques for Component-based Systems ................................................. 190
3.86 Measurement-based Approaches to Performance Prediction ......................................................... 195
3.87 Performance Prediction Techniques for Web Services and Service-oriented Architectures .................. 197
3.88 UML Profile for Schedulability, Performance and Time (UML SPT) .... 200
3.89 Core Scenario Model (CSM) .................................. 201
3.90 Kernel Language for Performance and Reliability Analysis (KLAPER) . 202
3.91 Service Component Architecture (SCA) .................... 204
3.92 FTFV (From Failure To Vaccine) ............................ 205
3.93 Self-protective techniques .................................... 207
3.94 Performance Model Driven QoS Guarantees and Optimization in Clouds ............................................. 209
3.95 A Game Theoretic Framework for SLA Negotiation ........ 211
3.96 Turning Software into a Service ............................... 212
3.97 Optimal Web Service Composition Using Hybrid GA-Tabu Search .... 214
3.98 Non-functional Properties In Web Services ................ 216
3.99 Cloud computing state-of-the-art and research challenges .... 216
3.100 SLA-based profit optimization in Automatic Computing Systems ..... 218
3.101 SLA-aware Virtual Resource Management for Cloud Infrastructures . 219
3.102 BREIN: Business objective driven RELiable and Intelligent grids for real busiNess ................................. 220
3.103 SLA-Negotiation in BEinGRID .............................. 222
3.104 SLA-Negotiation in ASSESSGrid ........................... 223
3.105 SLA Monitoring and Evaluation in BEinGRID ......... 224
3.106 IaaS Resource Advanced Reservation .................... 225
3.107 On the Design of Online Scheduling ....................... 226
3.108 Virtualized e-Learning on the IRMOS Real-time Cloud .... 226
3.109 A Negotiation Protocol Description Language for Automated Service Level Agreement Negotiations .. 228

4 ITIL & Service Management Analysis .................................. 230
4.1 Introduction .................................................................. 230
4.2 Information Technology Infrastructure Library (ITIL) .......................................................... 230
4.2.1 ITIL Service Lifecycle ........................................................................................................ 231
4.3 ITIL vs. SLA@SOI: A Comparison ...................................................................................... 235
4.3.1 SLA@SOI vs ITIL: Service Lifecycle .............................................................................. 236
4.3.2 SLA@SOI vs ITIL: Convergence ..................................................................................... 237
4.3.3 SLA@SOI vs ITIL: Divergence ......................................................................................... 238
4.4 Conclusions on ITIL ........................................................................................................... 239
4.5 Self-managing systems ......................................................................................................... 239

5 References .............................................................................................................................. 241

Appendix A: Abbreviations ....................................................................................................... 248
Table of Figures

Figure 1: ServiceCom lifecycle ................................................................. 74
Figure 2: Top level of the service ontology .............................................. 77
Figure 3: MoSCoE framework .................................................................. 80
Figure 4: IBM Architecture ...................................................................... 113
Figure 5: Haggle unlayered architecture .................................................. 116
Figure 6: The ACE Component Model ...................................................... 119
Figure 7: BIONETS networks .................................................................. 125
Figure 8: FOCALE -The Autonomic Computing Element ...................... 130
Figure 9: Real-time reconfiguration framework ....................................... 177
Figure 10: Autonomic policy-based management using web services. ....... 181
Figure 11: ITIL Service Lifecycle ............................................................ 232
Figure 12: SLA@SOI Service Lifecycle................................................... 236
1 Introduction

This section provides an overview of the state-of-the-art that has had the most impact on SLA@SOI in terms of research and development. Furthermore, it concludes the continuous analysis provided by SLA@SOI and summarizes the project’s most influential impacts on the current state-of-the-art. Therefore, this section serves as an entry-point for researchers and practitioners who would like to get an overview on the scientific advancements of SLA@SOI. For those interested in a certain topic, Section 3 contains detailed descriptions of more than a hundred scientific inputs to SLA@SOI. For the convenience of the reader, this section is ordered along SLA@SOI’s work packages.

1.1 Overview

This section provides a short summary of the extension to the state of the art provided by SLA@SOI during its three year runtime. The subsequent section will elaborate on the individual aspects presented here with regards to the publications, which form the basis for our extensions as well as the extensions themselves. These two sections are aligned with the work packages in SLA@SOI and reflect their respective contributions to the state-of-the-art.

The main contributions by work package A1 are a comprehensive integration of the many sub-aspects involved in SLA management. Although various architectures for SLA management are listed in the literature, none of them aims at providing the possibility to combine the various disciplines and concerns involved in a vertical integration of SLA management into an enterprise environment and also supporting the complete service/SLA management lifecycle. A1’s architecture thus brings together all the innovations presented in the following paragraphs and interlinks them in a consistent way, thus achieving a truly integrated SLA management architecture.

Work package A2 provides various enhancements of the state of the art related to business aspects of SLA management. Those are mostly related to the description of SLAs including business-level information as well the management of SLA violations and penalties.

In specific, A2 provides third party management as it has not been found in the literature. Furthermore, a management framework for penalties based on business level objectives is achieved based on specific SLA descriptions. These descriptions are a further advancement of the state of the art since they extend the SLA model with business specific information, referred to as business-terms. Further innovations by A2 include dynamic pricing, providing KPIs based on negotiations and SLAs based post-sales management.

Work package A3 provides innovations in the three main areas enhanced SOA modeling, manageability and service monitoring. With regards to enhanced SOA modeling, various advancements against the state of the art have been achieved and implemented in the SLA@SOI BCM component. As such, the business continuity management (BCM) component is capable to model the behavior of resources i.e. the time-related and causal relationships between resources. Based on this, the management of deadlines has been introduced as novelty to the topic. Furthermore, the BCM takes failover behavior and risk management based on multiple SLAs into account. These features have been combined so that the BCM can analyze combined threat scenarios including various sources of threats. In case of failures, it is possible to model n-out-of-k systems and partially failed
systems, which has not been found in the literature. Finally, the BCM is capable of modeling and analyzing time-based failure propagation to provide recovery and adjustment actions based on its results.

With regards to manageability, A3 makes contributions such as providing an interface, which is capable of receiving high-level runtime information. This makes it possible to allow for high level restructuring of composed services as a means to achieve service adjustment. Furthermore, the manageability component allows taking advantage of task-level parallelization of independent tasks, which has been achieved by adjusting the BPEL syntax based on a dependency analysis. Finally, the literature does not provide any work on instrumentation based service management beyond the collection of XML data. This has been extended by associating this type of information with event correlation technology to achieve a higher level of data correlation.

With regards to service monitoring, A3 provides a reasoning component to take into account properties in SLAs which have not been pre-configured. The outputs of service monitoring are a dynamically generated configuration for the monitoring infrastructure based on monitoring features provided dynamically by the infrastructure itself. Similarly, monitoring assessment based on SLAs, instead of industrial monitoring standards has not been mentioned in the literature to the best of our knowledge.

Work package A4 provides various achievements in the field of infrastructure management that go beyond the state of the art. With regards to infrastructure-related models, considerable work has been done with regards to OCCI. In this context, the specification of OCCI has been drafted and elaborated during SLA@SOI’s runtime. Related to this, work has also been conducted to combine OCCI with other initiatives such as OVF and CDMI. To promote the usage of OCCI and its applicability, implementations in various stages of maturity have been developed for OpenNebula, Apache Tashi and the infrastructure service manage (ISM). Furthermore, a generic infrastructure monitoring system has been devised which provides a vendor-independent layer to low-level monitoring data. This monitoring layer has been integrated with Ganglia, Nagios, Apache Tashi and OpenStack.

Work package A5 provides innovations in various fields. One of the most prominent outcomes is the SLA model, which has been finalized in Y3. It provides a conceptual model which can be represented through various concrete incarnations such as XML, WS-Agreement, Java and a JSON-like notation. Furthermore, it is possible to extend the model’s language itself by means of external XML-documents. Based on the final SLA model, it is furthermore possible to realize complex querying operations including queries for compositions of QoS-parameters. Based on our state-of-the-art analysis these features are novelties in the field of SLA modeling.

A5 has also developed a negotiation platform which provides an approach to describe negotiation protocols in a simple manner. This includes the possibility to negotiate specific customizations to a negotiation protocol in a pre-negotiation phase at run-time. Finally, the negotiation platform allows profiling of data related to negotiations and SLAs in general. This data can be accessed in subsequent negotiations to adapt the negotiation protocol and the negotiation strategies with regards to the other negotiation party.

Work package A6 provides various novelties regarding runtime predictions of service performance and service reliability. It does this by combining hardware and software considerations in a unique way. These considerations explicitly take input parameter values of service invocations and their propagations throughout the architecture into account. Furthermore, failure-recovery mechanisms are considered by the performance prediction and reliability mechanisms.
The runtime SLA violation mechanism is based on aggregate values such as the MTTR and MTTF of a service. This approach is novel in terms of the algorithmic approach as well as the integration with the EVEREST SLA monitoring environment.

Finally, manageability modeling accepts the description of KPIs and further performance measures as Esper-based specifications, which allows an appropriate handling of stochastic and other fuzzy parameters.

The remainder of this document is structured as follows. Section 2 provides an overview of the main innovations achieved by each work package individually. These descriptions provide a short introduction on the most relevant related work as a basis. Section 3 provides summaries of the relevant publications taken into account. Section 4 provides a comparison to related standards in the ITIL framework. Finally, references cited throughout the document and a list of abbreviations is provided as appendix.
2 Project Advancements and Relevant Related Work

2.1 WP A1 – Architecture and Integration

SLA@SOI developed a consistent SLA-management framework architecture that relates perspectives of relevant stakeholders (software/service/infrastructure provider and customer) and of different resource types (business, software, and infrastructure) in a consistent and transparent way.

The architecture is highly flexible so that it can be used for different scenarios which may require totally different kinds of resource types and related SLAs.

The new architecture can manage arbitrary resource types and SLAs around them in a structured way across layers and stakeholders.

The architecture supports the complete service and SLA management lifecycle, from service/SLA engineering, offering, negotiation, provisioning, monitoring, and adjustment.

It comes with a family of models, founded in the SLA model, which supports all these lifecycle phases across all the layers in a consistently interlinked way. Among these models are prediction models, monitoring models, manageability models, business models, and last but not least the specially developed service construction model. The latter supports the SLA-aware construction of concrete service instances based on a given customer requested service.

To our best knowledge, no such architecture yet exists. Closest work is the management frameworks Governance Interoperability Framework (GIF)[123,124] and CentraSite [125], though both do not support any multi-layered view on SLAs. While these have inspired certain concepts and relations, no specific architecture elements have been directly adopted from other sources.

2.2 WP A2 – Business Management

The management of Service Level Agreements is a complex task that requires to be specialized in the different domains it involves. Since SLA management can eventually be an integral part of contracting environments, several topics have to be tackled in this layer: third parties management, Business Level Objectives, penalties, etc. The scientific work done in the project describes the business terms and conditions that must be taken into account and the architecture of the business layer.

2.2.1 The Information Phase

Service Oriented Architectures (SOA) and the new ways of delivering applications and resources as services have consolidated in a manner to integrate and deliver functionality from vendors and providers. However, one of the main rationales of SOA was to create the roots of business environments in which providers and consumers could trade services. The emergence of Cloud Computing technologies have fostered the need to monetize the services provided, including software (SaaS), platforms (PaaS), infrastructures (IaaS), or any other service.
Typical proposals for interaction and trading of services between providers and consumers are e-Marketplaces [98], e-Contracting environments or services ecosystems [97]. These frameworks usually convey different phases of the trading process.

The work has been addressed in these steps: information, in which the technical and commercial offer is published, shown and rated; negotiation, in which the business and technical aspects are calculated and agreed upon; contracting, in which the agreement is signed and the service provisioned; and runtime, in which the service is delivered, managed, charged, reported, terminated, etc.

The information phase of eContracting covers all the activities related to defining, publishing, browsing, searching and rating the commercial offer of a services e-Marketplace. In the context of SLA management, that means to integrate the business terms and conditions under an SLA model and the management of a products catalogue, reified as a business SLA templates registry.

### 2.2.2 Modelling Business Aspects

There are several initiatives to model SLAs for their automatic management. However, these alternatives mostly focus on the modelling of technical issues related to the services definition and the guarantee terms specifications. For instance, [120] proposes a semantic model to integrate business oriented Service Level Management objectives with technical ones that includes pricing and Management of the Business SLAs for Services payment terms, service installation, revisions and terminations, maintenance, support, problem escalation procedures, etc.

The Universal Service Description Language (USDL) supports the modelling of the business terms in its business perspective [99], supporting availability, payment, pricing, obligations, rights, penalties, binding, security and quality. However, this services specification is decoupled from the SLA specification (WS-Agreement). Usual service registries are based on UDDI (Universal Description Discovery and Integration) and more recently on LDAP (Lightweight Directory Access Protocol) [102]. In the Telco environment, service descriptions are usually registered and stored in service catalogues, which is an essential component in new Operation Support Systems environments [96].

### 2.2.3 Business Level Negotiations

Negotiation is one of the most important and frequently tackled issues in SLA management. SLA negotiation in an eContracting environment requires a broad approach that takes into account business terms for the sake of both providers and customers, providing a more efficient environment for partners management and services trading [106].

From the point of view of eContracting frameworks, several issues can be considered in SLA negotiation: better matching of providers and consumers business goals [106], ranking of services based on prices or quality, sensibility of offers and counter offers on different issues (prices, Key Performance Indicators, etc.), past transactions [98], etc.

### 2.2.4 The Contracting Phase

Another important issue is the building of the offers. Service environments are usually based on the aggregation of services and, therefore, the individual SLAs of the atomic services must be taken into account in the final offer [100]. In
several SLA aggregation patterns which are very useful for the automation of the aggregation process of cross-company hierarchical SLAs are explained. Among other parameters, from the business perspective, it is also required to aggregate terms like price or penalties. Dynamic composition of an offer can include not only the bundling of services, but also the current supply and demand or historical data, as well as the parameters defined above. One added complexity that must be faced when defining service prices is the pricing schema (per transaction, per period) and the relationship between the different QoS levels and the price (absolute value, percentage value).

Electronic contracts are used to specify the terms and conditions under which a service is provided and consumed, and represents the basis for a business based eMarketplace function. Even though the law-conformity of electronic SLAs is still an open issue due to its complexity, and there are still many challenges to solve, it is an important aspect of an eContracting environment. A service contract is a contract associated with a specific service that involves the parties to the agreement, the service, including the description of the interfaces and expected interactions, promises about the service provision and consumption, business issues and legal procedures. One of the most important drawbacks of electronic contracts is information structuring and reuse, and because of that, many e-contract establishing proposals are based on contract templates, empty forms that have to be filled in. SLAs can undoubtedly represent e-Contracts, because they include, in the case of SLA@SOI model, all the information related to the service being provided, including business terms and guarantee terms. As mentioned above, there are a number of languages defined to specify electronic contracts, as Web Service Level Agreement (WSLA) and WS-Agreement, and USDL adds an SLA-decoupled layer to specify business terms.

Another important aspect of the contracting phase is the contract lifecycle management. Although there are already in the market a big number of commercial contract management tools, this topic presents a number of interesting challenges that improve the efficiency and required time to set up a contract, help to reduce errors and risks or improve revenue forecasts. For instance, presents a flexible framework for the automation of services contracts based on standard SOA middleware and shows a contract management solution for multi-tenant SaaS applications, in which the contractors may customize and configure the contracted services. In, electronic contracting between agents are also tackled, providing interesting Management of the Business SLAs for Services eContracting novelties as violation scale-up to humans, SLAs versioning, SLA hierarchies, contract dependencies and termination, extension and renewal of contracts.

### 2.2.5 Penalty Management

The most important business specific SLA management processes during the runtime phase are those related to calculating the penalties derived from SLA breaches, and, if possible, the self-adjustment processes to use spare resources to avoid those violations and penalties before they happen. In, a method to take into account the economic penalties due to SLA violations is proposed so that it is possible to monitor and allocate resources in the cloud, enforcing the maximization of a single Business Level Objective (BLO): the revenue of the provider. In, it is shown a deep analysis on how the economic penalties due to SLA breaches affect both providers and customers. It recommends giving priorities to different SLOs, in order to minimize the effect of an undesired penalty or contract cancelation. Anyway, the monitoring, SLA
breach detection and eventual penalty must be carried out by a third party, and not by the provider.

Following these arguments, [116] presents a way to define policies that modify the effects of SLA violations and penalties depending on the cause (i.e., if it is not the fault of one of the parties) in order to improve long-term relationships.

E-Contracting is a complex but well known area, since it has been researched and developed since the eCommerce hype. The services science development has renewed the interest on this field, first for web services marketplaces and, currently, with cloud services and applications stores. In this context, SLAs become a critical issue that must be tackled and tightly integrated with the eContracting frameworks. However, eContracting is a large field with many challenging topics. The business management layer of SLA@SOI has covered an SLA aware management of a marketplace from a comprehensive perspective, with innovative contributions in different topics as business terms modeling, post-sales management including penalties and violations or dynamic pricing and negotiation based on KPIs.

On the other hand, the business SLA model presented as an extension of the generic SLA@SOI model offers a first group of terms whose management can be to some extent automated and performed from an eContracting suite. The integration of some of these terms, especially those related to prices and penalties, in some of the SLA processes (negotiation, monitoring, reporting, ...) show the importance of modelling this layer.

## 2.3 WP A3 – Service Management

### 2.3.1 Enhanced SOA Modelling

Failure Modes, Effects, and Critical Analysis (FMECA) [1], developed by a military research project, is a standard to determine the impact on function, mission success, personal safety and performance that may could arise from failures in systems. FMECA uses a systematic approach to analyse systems [1, p. 8f]. First the system under discussion has to be defined. Second, it is decomposed in smaller elements, e.g. components and the dependencies between components are documented. Third, for each system component potential failure modes are identified. Each failure mode is evaluated in terms of the worst potential consequences which may result and a severity classification category is assigned. This enables a priority ranking and a Risk Priority Number is assigned to every failure mode. The goal is to identify failure modes that are hard to detect, very severe and very likely to occur. Although the main objective of FMECA is to eliminate failure modes in early design phases, the standard says about itself: “Probably the greatest criticism of the FMECA has been its limited use in improving designs. The chief causes for this have been untimeliness and the isolated performance of the FMECA without adequate inputs to the design process.” FMECA analyses failures one-by-one and does not consider combinations of them. However, a thorough BCM dependency and risk analysis should consider possible combinations, for example an earthquake may cause a fire and a power breakdown. FMECA also does not evaluate combination of systems, e.g. a database with or without a primary backup server. Furthermore, FMECA provides no means to model failures caused by environmental events, e.g. earthquakes, which are often the primary focus of a BCM analysis. BCM managers are also interested in quantified analysis of timely behaviour of resources. FMECA provides no means to analyse this aspect.
Fault Tree Analysis (FTA) [2] is a technique used in reliability engineering to determine combinations of failures in a system that could lead to undesired event at system level. The modelling process starts with the undesired event (e.g. “valve is closed failed”) and is broken down into a fault tree. Each fault is analysed in more detail (e.g. “valve is closed due to human errors”, “valve is closed due to hardware failure”) and if necessary broken down again, until a reasonable level of understanding has been achieved (e.g. “valve is not opened by operator after last test” or “valve is closed because of malfunctioning switch”). The logical relationship between faults is defined by logical functions, such as AND, OR, XOR or NOT. Probabilities are assigned to basic events and thus the marginal likelihood of an undesired event can be calculated.

Although a fault tree analysis is able to provide a better estimation of the probability of adverse events to occur, such an analysis is not able to model the dynamic behaviour of systems since Boolean functions are not able to capture the order in which events occur, nor is it possible to model time constraints, such as deadlines. This limits the application of FTA in ICT BCM to very simple analyses.

The SLA@SOI BCM component is capable to model the behaviour of resources, the timely and causal relationship between resources and introduces deadlines (Maximum Tolerable Outage Time) to business process activities.

Tropos Goal-Risk Framework: In his doctoral thesis [3], Asnar developed the Tropos Goal-Risk (GR) Framework, for requirement analysis and risk assessment for critical socio-technical systems, such as Air Traffic Management. This framework elicits interesting features. First, it provides the means to model combinations of failures similar to FTA. Second, it provides semantics to model other aspects, such as time dependencies, treatments and assets, which are useful for Business Continuity Management. In fact, in [4] Asnar and Giorgini demonstrated that the GR framework could be used to analyse and compare the effectiveness and cost-efficiency of different treatment strategies. However, the analysis does not provide any means to determine the business impact nor does it provide the means to determine BCM metrics, such as RTO. Furthermore, the treatment analysis is not sufficient to model complex examples. The example provided in [3, Fig 2, p5] models a database failure with an outage time of exactly three hours. Business continuity management normally assumes a distribution of outage time, mostly estimated by a three-point-estimation (minimal outage time, most-likely outage time and worst-case outage time) to fit a beta-distribution. Treatments in GR can be modelled by reducing likelihood and/or reducing the time period of disruption.

In the example, provided in [3], a treatment is modelled as well. The treatment (“backup-server”) reduces the likelihood of the database to fail by 90%, but does not provide any means to model the behaviour of a switch from primary to backup database server, nor does it provide means to model why a switch fails. Also, steps of treatments are not detailed and hence are not well suited to model and analyse recovery plans. Furthermore, the algorithm proposed in [3] only selects a combination of possible treatment; the algorithm does not propose treatment levels, e.g. proposes that a backup-server should reduce the risk by 99% to guarantee certain level of business continuity.

The SLA@SOI BCM component is capable to model failover behaviour and to propose a set of SLAs or multiplicity of resources to achieve an acceptable risk level.

ROPE: The Risk Aware Process Evaluation (ROPE) [4] aims to combine business process modelling with risk and business continuity management. ROPE diagrams consist of three layers: the business process layer, the Condition, Action,
Resource and Environment (CARE) layer and the Threat Impact Processes (TIP) layer. The business process layer models business process activities. A CARE layer element described in a simplified fault-tree-like structure resources (e.g. "personal computer") and an environment (e.g. "office") needed by a business activity. The top-element of this fault-tree structure is the business activity. At the time of writing, the ROPE framework only considers resource availability and resource utilisation as modelled aspects. If the fault-tree evaluates to false, the associated business activity is on hold and therefore the business process does not proceed.

The TIP layer models three kinds of processes: threat processes, detection sub-processes and countermeasure processes. A threat process step is linked against a set of resources or an environment of a CARE element and makes those resources or environment unavailable. If a threat activity (e.g. "virus") makes the associated resource set unavailable (e.g. "personal computer") and, if the fault-tree of the CARE element evaluates to false, the associated business activity is paused. A threat process does not proceed by itself and is paused at each single threat activity. The detection sub-process and the countermeasure sub-process processes are linked against single threat activities. A countermeasure activity releases threat-activities in such a way that the threat-process is able to proceed. If the threat-process reaches its final state, a threat has been removed.

The ROPE framework has some novel features. First, it uses workflows for all four activities: to model business processes, to model threats, to describe detection behaviour and to model countermeasure activities, which are in its essence business continuity plans. This allows business process manager to model all activities using the same concept. Second, it assigns resources and business activities and enables resource dependency modelling and resource multiplicity modelling. Finally, business processes, threats, and countermeasure processes are independent activities but influence each other. This simplifies the modelling phase since every process can be modelled independent of each other.

However, ROPE has several drawbacks. First, the simplified fault-tree dependency model does not allow the analysis of temporal failure propagation. Although a threat process could be used to overcome this limitation the restriction that a threat process cannot proceed on its own, does not permit this modelling approach. The SLA@SOI BCM component is capable of modelling timely failure propagation. Second, ROPE only delays business process activities; there is no means to model situations where certain business process activities have to be repeated due to a threat. This is needed in order to analyse the backlog caused by a threat, e.g. if a database failed and data has to be re-entered manually. Third, ROPE does not provide any means to model effects. A threat always makes a set of resources completely unavailable. ROPE does not consider other threat characteristics, e.g. a threat might gradually decrease a resource's performance. An effect-model should also be used to model the effects of n-out-of-k system on dependent resources. For example, if two out of five air-condition units fail, then the sever room will overheat within two days. If all air-condition units fail the sever room will overheat within five minutes. The SLA@SOI component is capable to model n-out-of-k systems and partial failed systems. Fourth, threat processes can be combined in ROPE, but do not interact with each other. This makes it hard to create advanced risk scenarios where, for example, threat processes trigger or enable each other. The SLA@SOI BCM model is enabled to analyse combined threat scenarios. Lastly, countermeasure processes models do not have resources assigned. Therefore it is not possible to model scenarios in which a threat may affect the countermeasure process itself. These scenarios are needed to detect dead-locks or live-locks; for example a login-server might be unavailable but the administrator depends on the availability of this particular login-sever to login to the system to resolve the problem. That is one problem
that has been encountered by one of SAP's customer and has been rated as an important feature. The SLA@SOI model is capable to model and analyse such problems.
2.3.2 Manageability

There have been three main contributions with respect to the state of the art: the definition of a unified management interface for services, the definition of an algorithm for BPEL process restructuring, and the implementation of an instrumentation-based infrastructure for managing software services.

The management interface allows us to manage a service's lifecycle, to install and configure sensors for capturing runtime information about the service's behaviour, and to request corrective measures through effectors that are available for the service. The main novelties with respect to the state of the art [5, 6] are that we provide a richer and more detailed information model for configuring sensors that can also take into account execution context properties. The information model also explicitly distinguishes between low-level information and high-level information which can be correlated such as general-purpose KPIs (e.g., reliability, average response time, etc.) and domain-specific aggregations. This is not the case for [7] which only concentrate on low-level runtime information. The unified interface also explicitly acknowledges the need to be able to issue higher-level commands to the resource than simply asking the resource to set some property. In our manageability approach, we consider dynamic binding and process restructuring as high-level adjustment measures. This goes well beyond the reactive capabilities of previous work [8].

Our restructuring approach relies on program analysis and parallelization introduced for compilers [9]. We model business-process encoded in BPEL as a graph, where nodes are BPEL activities and edges are dependencies among the activities. Our dependency analysis considers not only data-, control-flow dependencies but also partner protocols which are extracted from the original BPEL file. Based on the dependency analysis, we define a partial ordering over graph nodes. This partial ordering serves later to eliminate redundant edges and to find independent activities that can be run in parallel. The graph dependency is also used in [10], [11], [12] correspondingly for minimization of dependencies, optimization of monitor workload and maximization of process throughput, while the process structure remains sequential and no additional parallelism is added. Innovation of our research lies in introduction of BPEL syntax modification based on dependency analysis in such a way that the process could be executed correctly yet with maximum parallelization of independent activities.

The instrumentation-based infrastructure for managing services consists of a massive evolution of previous work achieved in [7], in which the instrumentation was only used to collect simple XML data types from a running BPEL process. The instrumentation now works in tandem with event correlation technology (Esper) to achieve higher-level data collection. We have also applied our instrumentation techniques to BPEL processes to enable instance- and class-based runtime process restructuring. A similar approach is presented in [13], yet the authors do not explicitly support class-based process evolution. Instead, they concentrate entirely on instance-based modification. Their prototype implementation is entirely implemented with .NET technology, meaning it could not be easily used or extended in the context of SLA@SOI. By adopting WF Sequential Workflows, the authors were able to directly exploit Microsoft’s WorkflowChanges API. This API allows the authors to suspend, modify, and resume any running instance.
2.3.3 Service Monitoring

Our contributions in this area are highly related to automatic monitoring from software requirements such as reported in [14], where the authors describe flexibility in generating monitoring configurations based upon some constraints specified. In this work, the user expresses his/her requirements and assumptions for monitoring in FLEA (Formal Language for Expressing Assumptions). This language provides a set of tailored constructs, which may be composed, for the convenient expression of a wide range of monitoring concerns including sequences, combinations and time sensitive events. We base the constraints on those already specified in the SLA, and automatically derive suitable monitoring configurations based upon some monitorability calculation. Several publications have focused on reasoning about monitorability based upon calculations of the total cost of monitoring the SLA between providers. For example, in [15, 16], the authors describe an approach to determine the monitorability of systems of SLAs (a system of SLAs is closely related to a set of agreement terms in the SLA notation used in our work). Analysis of SLAs in their work assumes a set of pre-configured properties (e.g. availability, satisfaction etc) and does not dynamically seek reasoning components as our work provides. Hence our work is different as it aims to report on those terms that are supported and unsupported, based upon constraints specified in the SLA and also service consumer monitor selection preferences. The TrustCOM project [17] has also produced a reference implementation for SLA establishment and monitoring. This implementation, however, does not involve the dynamic setup of monitoring infrastructures or reporting. The SLA Monitoring and Evaluation architecture presented within the BeinGrid project [18] has several similarities with the approach presented in this paper, such as the need to separate SLA from service management. Their focus of work, however, is on statically binding services and monitors, whilst ours is on dynamically allocating monitors to SLA parts, based upon matching the exact terms that need to be monitored and the monitoring capabilities available in different services.

Secondly, our contributions in this period have focused on service monitorability assessment and reporting. Our infrastructure-design for service monitoring and initiation shares many related aspects with broader interacting communications equipment. For example, in [19, 20] the authors describe monitorability assessment for communications infrastructure based upon a specification for monitoring (synonymous for an SLA) and how the capabilities of the monitoring infrastructure can be improved for effective reporting. Our work aligns with this but proposes how this is achieved for service level agreements rather than industrial communication standards. Bridging business and technical monitorability reporting requirements has been discussed in the NextGrid project [21], where the authors propose a human-centric architecture for SLA composition and checking. In particular, they stipulate the importance of business-level objectives such as "utilize my resources a hundred percent" or "get the maximum productivity, while spending as little money as possible" alongside the SLA quality of service terms. Where these works have not provided designs and implementation for these preferences, our design and tools allow engineers to support service consumer preferences for service monitoring through optimisation of monitor configurations.

2.4 WP A4 – Infrastructure Management

The state of the art in infrastructure models, virtualization, platforms, monitoring and messaging has been examined over the course of SLA@SOI.
2.4.1 Model Standards

Regarding model standards, Common Information Model (CIM) [21], Open Virtual Machine Format (OVF) [22], and GLUE v. 2.0 [23] were all examined. Regarding implementations of infrastructure models, Amazon EC2, Sun Cloud API, Flexiscale, ElasticHosts, GoGrid, Enomalism, OpenNebula, Slicehost, Eucalyptus, Globus Nimbus, AppNexus, and Apache Tashi were all analysed. It was observed that no current or draft model existed that met all the needs of SLA@SOI. OVF was considered to be the most relevant model, and indeed SLA@SOI has the potential to propose extensions to OVF to enable support for non-functional parameters that are at the core of SLA-enabled infrastructures. From a design and implementation point of view, it was foreseen that SLA@SOI should be engineered to accommodate requests for infrastructure in arbitrary formats. The SLA@SOI architecture was designed to support the recommendations of this analysis. Whilst potential contributions to OVF from SLA@SOI were considered, our engagement with OCCI was prioritised in order to make it as useful a standard as possible. It was designed in such a way as to support OVF descriptors and a paper was published explaining how the various standards could be used together to build a standards based cloud [24].

2.4.2 Infrastructure Virtualisation

Regarding infrastructure virtualisation, Libvirt [25] is an open-source API that provides a generic way to interact with different types of open source virtualization technologies (including Xen [26] and KVM [27]). OpenNebula [28] is an open-source distributed VM manager that enables the dynamic placement of VMs on a pool of physical resources, and key concepts such as federation and scheduling are dealt with comprehensively. Apache Tashi [29] is an internet-scale provisioning system designed to provide safe access to ‘big-data’. NASA and Rackspace have combined forces to produce a family of open-source cloud computing enablers called OpenStack [30]. Now receiving significant attention in industry, it is a rapidly maturing project. The SLA@SOI architecture was thus designed to support arbitrary virtualisation technologies, with the OCCI interface specifically conceived to provide a generic interface to heterogeneous virtualisation providers. An OCCI interface has been successfully developed and open-sourced by SLA@SOI into both Apache Tashi and OpenNebula. Although timing did not facilitate complete integration with OpenStack, integration with the Infrastructure Monitoring System was successfully demonstrated.

2.4.3 Infrastructure Platforms

In terms of infrastructure platforms, a comprehensive review of APIs was undertaken and documented in Y1. Amazon EC2, Sun Cloud API, Flexiscale, ElasticHosts, GoGrid, Enomalys – Enomalism, OpenNebula, Slicehost, Globus Numbus, Eucalyptus AppNexus, F5.com, Apache Tashi and CohesiveFt were all reviewed. This helped identify key generic requirements of APIs that SLA@SOI needed to comprehend in its efforts to develop an open standard for heterogeneous virtualisation platforms. This analysis was contributed to OCCI, and heavily influenced our contributions to the development of this standard.

2.4.4 Infrastructure Monitoring

Regarding monitoring, EVEREST, Ganglia, Nagios, Groundwork, MonALISA and Zabbix were all examined. Although Nagios, Groundwork, MonALISA and Zabbix all have interesting aspects, from a low-level infrastructure point of view the most...
relevant framework was determined to be Ganglia [31]. This was adopted as the default lowest level monitoring framework on which our monitoring system has been built, and extended with additional sensors to capture data needed for SLA monitoring. However, the project did implement a generic interface into its instrumentation layer, and the SLA@SOI-developed infrastructure monitoring system has also been demonstrated to work on top of Nagios [32].

2.4.5 Messaging

To deliver a truly scalable SLA-aware infrastructure layer, it was clear that a traditional RPC-style interface would soon run into issues and so the potential of various messaging protocols was examined as part of the state-of-the-art review. XMPP, the Extensible Messaging and Presence Protocol [33], JMS, the Java Message Service [34], and AMQP, the Advanced Message Queuing Protocol [35], were all examined. It was determined that XMPP offered the most appropriate messaging solution for our initial implementations. However, with the maturing of AMQP it was observed that the messaging layer in SLA@SOI should be implemented in such a fashion that the messaging protocol can be easily replaced. The current SLA@SOI infrastructure implementation has adopted this layered approach.

2.5 WP A5 – SLA Management Foundations

2.5.1 SLA Modeling

Modeling SLAs is a central theme throughout the project’s conceptual work, implementation as well as the adoption guidelines. The project’s use cases pose a broad range of requirements to modeling SLAs. As such we have developed our own SLA model, since the state of the art did not provide us with solutions which could be used ad hoc in a satisfactory way in SLA@SOI.

The most prominent standard for representing and negotiating SLAs is WS-Agreement [126], which build on established standards from the WS-* family. WS-Agreement is a recommendation by the OGF and incorporates the web service description language (WSDL) and the web service resource framework (WSRF) at its core. SLAs in WS-Agreement are represented as XML-documents and validated using a provided XML schema document. WS-Agreement offers extension and validation mechanisms by means of XML-Schema extensions, which in turn requires extensions in the form of XML documents. We have advanced the state of the art with respect to WS-Agreement by providing an SLA model, which does not depend on the representation in one specific language such as XML. Instead SLA@SOI’s SLA model, which we refer to as SLA*, offers a formal description and various concrete incarnations (BNF, XML, Java, JSON-like). Furthermore the negotiation interface and negotiation protocol are detached from the SLA representation. Another core feature of SLA* is simple extensibility. In the final model it is possible to extend the SLA vocabulary by providing a description of the desired extension in form of a XML-document at runtime. Structurally SLA* furthermore differs from WS-Agreement by the possibility to implement advanced querying mechanisms. These querying mechanisms works also for extensions of the SLA model in the case that these are not yet known at compile-time. Finally SLA* also drops the dependency on WSRF and relies mostly on custom code in practice.
Despite those deviations from WS-Agreement’s structure we offer a compatibility layer, which can render and parse representations of SLA* as WS-Agreement documents.

And older proposed model for representing SLAs is called Web Service Level Agreement (WSLA) [127] and has been developed by IBM. This work provides a structure to represent SLAs, which has inspired the conceptual design of SLA*. WSLA however does not provide advanced mechanisms such as extensibility, support for querying mechanisms and negotiations. Furthermore WSLA has been discontinued and no current implementation of it is provided anymore.

Another notable SLA model is SLAng [128], which provides a certain level of language independence since it makes use of OMG’s MetaObject Facility (MOF). However SLAng is targeted at electronic services and thus provides only a limited set of domain-specific quality of service constraints.

CC-Pi [129] is an even more generic solution to the SLA modeling problem than SLAng, however it is also tightly coupled with the negotiation process itself. This leads to the lack of concepts such as agreement parties or service interfaces. As a result, it is not suitable for the requirements posed by SLA@SOI.

Conclusively SLA* offers novelties by providing an abstract syntax, which makes it language independent. At the same time it is decoupled from the notions of a service and from particular modes of expression. Furthermore SLA* can be extended to satisfy a wide variety of domains without sacrificing formality or semantics.

2.5.2 SLA Negotiation

Several projects and solutions were considered in order to understand and further the state of the art on SLA Negotiations. As a result, it was realized that the use cases, which SLA@SOI focuses on specifically, target bilateral negotiations which may potentially be part of hierarchies. It was understood that the Y2 approach of keeping protocol-governed interaction behavior decoupled with the negotiation strategy and communication mechanism is the most flexible way to provide automated negotiations for a host of possible scenarios including those of SLA@SOI’s use cases. We learned from several design aspects of the SecSe project [37] which also caters for bilateral negotiations but couples rules related to negotiation strategies with the ones defining interaction behavior - this surfaced as an engineering difference of the SLA@SOI approach of writing protocols and SecSe. Shortcomings from previous projects like Kasbah [38], ASAPM [39] and CAAT [40] were also studied. Kasbah for instance does not have a well defined counter-offer support, ASAPM and CAAT as well as BREin couple low level communication mechanisms (i.e. synchronous and asynchronous modes of messaging) with protocol description – a requirement that SLA@SOI defines as strictly separable in order to not limit adoption for scenarios that favour e.g. web service-based communication over the FIPA CNP styled asynchronous communication. Approaches from earlier projects like OPELIX[41], Inspire [42], Aspire [43] and e-Agora [44] were also considered. We reached an understanding that there cannot be a single protocol that would amicably serve the needs for all possible scenarios. (cf. SLA@SOI Book - chapter 15).

We found strength in SLA@SOI’s design principles of developing as generic an approach as possible to encode protocols that concretely define only an agent interaction-behavior, while loosely coupling the negotiation intelligence in domain specific components – the POCs - which are consulted during negotiation time by the Protocol Engine. Similar lessons have been drawn by several other publications that strive for a generic protocol description and execution (cf. Section 1.109). Thirdly, we also abstract over the underlying communication
mode that could be either synchronous (as was required by all SLA@SOI use cases) or asynchronous. For the latter or any other communication mode per se, merely a separate implementation of the Syntax Converter could be used which provides for message invocation and delivery in addition to SLA model-based transformations among various serializable forms – needed to marshal parameters.

Existing SLA standards were also considered. Although a one-for-all global standard does not exist, the closest work is that delivered by the OGF in the form of WSAG [45] and WSAGN [46]. These have emerged as de-facto standards to represent an SLA (agreement) template and negotiate using given protocol. Since SLA@SOI has developed its own model, the challenge lies in 1) bridging the two models and 2) allowing a compatibility of SLA@SOI based projects to be able to interoperate with WSAG/WSAGN based projects. With regards to 1, reasonable effort was invested to be able to transform essential elements and terms of WSAG-based templates to those based on SLA@SOI. For 2, a feasibility study was conducted with more interest towards WSAGN which supports multi-round negotiation capability that is of basic importance in SLA negotiations. It is worth noting here that the SLA@SOI negotiation platform caters for several protocol level parameters as part of protocol description which are not available in WSAG/WSAGN. These enrich negotiations by allowing a pre-negotiation mechanism that fixes protocol parameters among negotiating parties in a mutually consensual manner. WSAGN on the other hand advocates a host of signaling scenarios which allow parties to communicate with each other using a certain mode of communication. Among these, the client-server asymmetric signaling scenario in principle can easily be made interoperable with the SLA@SOI negotiation platform since most required engineering artefacts are available. The ongoing work on WSAGN also posed a concern for interoperability efforts, nevertheless our timely assessment and resultant influence on the SLA@SOI negotiation platform benefitted us in terms of situating our work and furthering the state of art.

2.5.3 Planning and Optimisation

Cloud computing effectively implements the vision of utility computing by employing a pay-as-you-go cost model and allowing on-demand (re-)leasing of IT resources. Small or medium-sized IaaS-providers find it challenging to satisfy all requests immediately due to their limited resource capacity. In that situation, both providers and customers may benefit greatly from advanced reservation of virtual resources [48] [49]. In our work, we assume SLA-based resource requests and introduce an advanced reservation methodology during SLA negotiation by using computational geometry. Thereby, we are able to verify record and manage the infrastructure resources efficiently. Based on that model, service providers can easily verify the available capacity for satisfying the customer's Quality-of-Service requirements. Furthermore, we introduce flexible alternative counter-offers, when the service provider lacks resources. Therefore, our mechanism increases the utilization of the resources and attempts to satisfy as many customers as possible.

By using computational geometry [50] as a means, we can represent the reserved grid resources into a computational geometry context. In there, some strategies can be deployed for gaining better resource utilization and guaranteeing QoS terms. In Grids, it is used by [51] to represent the advance reservation into a 2-D diagram. Therefore starting time and end-time represent x-axis and y-axis. However, due to the differences between cloud and grid computing with regard to application and virtualization [52], resources could be partitioned in a virtual way rather than be occupied completely by a job.
Customers set their VM configuration and QoS requests through SLAs. Therefore, there could be several VMs that run simultaneously in a server, which will introduce a more complicated fragmentation situation. Thus, we can take one step further to represent resources in 3 dimensions and resource availability and fragmentation of the virtual resources dynamically. Last but not least, service providers can also provide alternative solution during SLA negotiation with customer by verifying resource availability in each physical server and overall data center.

2.6 WP A6 – Predictable Systems Engineering

2.6.1 Software Performance and Reliability Prediction

For the software performance and reliability prediction, the most fundamental related work was the Palladio Component Model (PCM) approach [51]. The PCM meta-model served as a basis for the QoS meta-model used in SLA@SOI to specify service-based architectures. The existing performance simulation was used for service performance prediction in SLA@SOI. Extensions realized in SLA@SOI include reliability modelling and prediction, as well as an automated prediction service as part of the SLA management framework. Architecture-based software reliability prediction as realized in SLA@SOI is highly innovative compared to related approaches [52] as it (i) integrates the consideration of software and hardware failure potential, (ii) explicitly considers input parameter values of service invocations and their propagation throughout the architecture and (iii) includes mechanisms for failure recovery into the consideration.

2.6.2 Run-time SLA Violation Prediction

Prediction techniques have been developed to generate forecasts for different properties and aspects of software systems (e.g., [64], [69], [70]). These techniques may be classified with respect to different criteria, including the property of the software system that a technique aims to predict and the basic algorithmic approach deployed by it (see for a recent survey [64]).

With respect to the prediction property, there have been techniques focusing on prediction of software systems failures [65][59][62], software aging [57], trends in different system parameters such as server workloads, CPU loads and network throughput [61][54], or violations of special types of software system properties such as security [63].

With respect to the algorithmic approach, we can distinguish between techniques using: time series analysis [57][54]; Markov models (e.g. [65][68]); regression models (e.g., weighted regression [55], linear regression [53], auto-regression [56], trend slope estimation [66]); other statistical models (e.g. seasonal Kendall test [58], various mean time prediction techniques [64]); belief based reasoning [63], system model based predictions (e.g. FSA based prediction [62]), machine learning [60] or hybrid techniques (e.g., [59] which uses Markov models and clustering). Approaches based on time series have been used for predicting system parameters, e.g., CPU, memory, disk, or network throughput (see [57][54] for example). Approaches based on regression analysis are used to find out dependencies between system parameters [56][53], predict threshold violations for web server throughput, and estimating software aging [57].
The approach developed for predicting runtime SLA violations focuses on aggregate mean values of properties of software services (specified as SLA guarantee terms), as for example the mean-time-to-repair (MTTR) and mean-time-to-failure (MTTF) of a service, is – to the best of our knowledge – novel both in terms of its algorithmic basis and its focus on prediction of threshold constraints for these availability properties. Furthermore, it is fully integrated with the generic SLA monitoring environment of EVEREST thus providing integrated support for SLA monitoring and prediction.

2.6.3 Manageability Modelling and Design

Management has been an important topic for standardization by part of different consortiums. Oasis has proposed the Web Services Distributed Management (WSDM) [71], a protocol for the interoperability of management information and features. It focuses on two main aspects: how to use web service technology as the foundation of a resource management framework, and how these notions can be adapted to Web services themselves. An alternative standard, called WS-Management [72] has been proposed by the Distributed Management Task Force (DMTF). Its goal is similar to WSDM’s, as it provides support for generic resource management using Web service standards. It also proposes a special binding for managing resources that are defined using their Common Information Model (CIM).

Different MDE methodologies for managing extra-functional properties of service compositions have been proposed: Chowdhary et al. [73] address the specification of business-level performance indicators and their direct transformation to platform specific models, Debusmann et al. [74] define SLA parameters together with the specific indicators needed within the SLA itself, and Chan et al. [75] address the automatic generation of component-based instrumentations for monitoring specified QoS concerns. The work that most resembles our own is that of Christof et al. [76]. The main difference with our work lies in the different level of abstraction proposed. Our meta-models provide an extensible set of high-level off-the-shelf KPIs. The definition of the KPIs themselves is an issue for platform-specific implementations. In their work, the definition of the KPI is something that needs to be modelled; to this end they propose lower-level processing operations (e.g., addition, subtraction, etc.) for defining the calculation that needs to be performed. We believe that this level of detail in the modelling makes matters unnecessarily complex, especially with KPIs that are stochastic or more fuzzy in nature. To cope with similar issues, we allow for domain-specific correlation through appropriate Esper-based specifications.

2.6.4 Resource Usage Prediction

Initial work in the prediction of resource usage has addressed how to calculate predicted values of a metric or a set of metrics using various algorithms including Multiple Predictor Integration, Fuzzy Logic and Clustering, and Periodic Pattern Prediction (see Deliverable D.A6a, Y1). More recent effort has been focused on addressing the performance and scalability challenge: identifying approaches to predicting a set or subset of metrics that are related to a relatively large number of live resources (e.g. physical servers used for VM provisioning), as in the case of cloud-scale applications. This work has been grounded in requirements for scalable architectures as defined by Anderson [77], and follows principles in autonomic computing documented by Sterritt et al [78].

Some research projects have studied how the autonomic agent approach can be used to provide an elastic SOA backend infrastructure that can adjust to different levels of demand in an ad-hoc manner [79][80]. There are many systems in the
literature that implement scalable de-coupled and message-based infrastructures, including Amazon’s Dynamo [81]. Dynamo has been implemented following the “Eventually Consistent” philosophy [82] to overcome the “CAP” theorem limitation [83]. Hadoop [84] and its sibling projects and reference implementations of standard machine learning algorithms that can run on such infrastructures [85] have also proved to be very relevant.
3 General State of the Art Analysis

The following table summarises the contributions to SLA@SOI’s State-of-the-Art analysis. With the title, for each item a classification, keywords and the relevance to SLA@SOI are listed.

Regarding the classification the following classes are distinguished:
- Standard
- Model
- Literature
- Language
- Algorithm
- Software
- Protocol

The “Standard” classification is used in addition to the other ones to indicate that an item has been standardised by a standardisation body.

The relevance column indicates the relevance of an item for SLA@SOI. We distinguish the following states:
- Y: Is or will be adopted, used, integrated, … in SLA@SOI.
- (Y): Is or will be adopted, used, integrated … partially in SLA@SOI.
- N: Definitely not relevant to SLA@SOI.
- observe: A clear recommendation to further investigate the entity is given by the authors of this contribution.
- tbd: The entity might be relevant to SLA@SOI, but the current situation does not allow to make a definite decision.
- ?: No information about the relevance is provided.

<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
<th>Classification</th>
<th>Keywords</th>
<th>Relevant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shared Information and Data Model (SID)</td>
<td>Standard, Model</td>
<td>Information Model</td>
<td>Y</td>
</tr>
<tr>
<td>3</td>
<td>WS-Agreement Negotiation</td>
<td>Standard, Protocol</td>
<td>SLA Negotiation</td>
<td>Y</td>
</tr>
<tr>
<td>4</td>
<td>NextGRID SLA Model</td>
<td>Model</td>
<td>SLA Model</td>
<td>observe</td>
</tr>
<tr>
<td>5</td>
<td>A reference model for dynamic web service composition systems</td>
<td>Literature</td>
<td>Service Management, Dynamic Service Composition</td>
<td>observe</td>
</tr>
<tr>
<td>6</td>
<td>DIANE - An Integrated Approach to Automated Service Discovery, Matchmaking and Composition</td>
<td>Language, Algorithm</td>
<td>Service Management, Dynamic Service Composition, (Semantic) Matchmaking</td>
<td>observe</td>
</tr>
<tr>
<td>7</td>
<td>Governance Interoperability Framework (GIF)</td>
<td>Framework</td>
<td>Information System, Service Oriented Architecture</td>
<td>observe</td>
</tr>
<tr>
<td>#</td>
<td>Title</td>
<td>Classification</td>
<td>Keywords</td>
<td>Relevant?</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>----------------</td>
<td>--------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>8</td>
<td>CentraSite</td>
<td>Software, commercial</td>
<td>SOA registry, SOA repository</td>
<td>observe</td>
</tr>
<tr>
<td>9</td>
<td>TeleManagement Forum’s SLA Management Handbook</td>
<td>Literature, Model</td>
<td>SLA MAnagement</td>
<td>Y</td>
</tr>
<tr>
<td>11</td>
<td>Web Services Agreement Language (WSLA)</td>
<td>Model</td>
<td>SLA Model</td>
<td>Y</td>
</tr>
<tr>
<td>12</td>
<td>Web-based Enterprise Management (WBEM) Standards</td>
<td>Model, Software, Standard</td>
<td>WS Management, Common Information Model</td>
<td>(Y)</td>
</tr>
<tr>
<td>13</td>
<td>Common Information Model (CIM)</td>
<td>Model, Standards</td>
<td>Information Model</td>
<td>(Y)</td>
</tr>
<tr>
<td>14</td>
<td>Web Services Distributed Management (WSDM)</td>
<td>Standard, Software</td>
<td>Web Services management</td>
<td>observe</td>
</tr>
<tr>
<td>15</td>
<td>Java Management Extensions (JMX)</td>
<td>Software Development</td>
<td>Resource Management, Java</td>
<td>observe</td>
</tr>
<tr>
<td>16</td>
<td>Data Centre Markup Language (DCML)</td>
<td>Model, Standard</td>
<td>Data Format, Data Model</td>
<td>(Y)</td>
</tr>
<tr>
<td>17</td>
<td>System Modelling Language (SysML)</td>
<td>Language, Standard</td>
<td>Modelling, System Engineering Applications</td>
<td>(Y)</td>
</tr>
<tr>
<td>18</td>
<td>GLUE</td>
<td>Model, Standard</td>
<td>Information Model</td>
<td>N</td>
</tr>
<tr>
<td>19</td>
<td>Service Modelling Language (SML)</td>
<td>Model, Standard</td>
<td>System Modelling, Service Modelling</td>
<td>(Y)</td>
</tr>
<tr>
<td>20</td>
<td>Common Model Library (CML)</td>
<td>Model Library</td>
<td>System Modelling, Service Modelling, SML-based</td>
<td>observe</td>
</tr>
<tr>
<td>21</td>
<td>Configuration Management Databases (CMDB)</td>
<td>Standard</td>
<td>Configuration management, ITIL</td>
<td>observe</td>
</tr>
<tr>
<td>22</td>
<td>Making BPEL Flexible - Adapting in the Context of Coordination Constraints Using WS-BPEL</td>
<td>Literature</td>
<td>Buiness Processing, Constraints, BPEL</td>
<td>N</td>
</tr>
<tr>
<td>23</td>
<td>Towards Aspect Weaving Applications</td>
<td>Concept, Software, Literature</td>
<td>BPEL, AOP</td>
<td>Concept – Y, Software - N</td>
</tr>
<tr>
<td>24</td>
<td>A dynamic and reactive approach to the supervision of BPEL processes</td>
<td>Architecture, Software, Literature</td>
<td>BPEL, ActiveBPEL, Process Monitoring, WSoC, WSRel</td>
<td>Y</td>
</tr>
<tr>
<td>25</td>
<td>ServiceCom: prototype tool for service composition specification, construction and execution</td>
<td>Research, Software</td>
<td>Service Life Cycle, Service Composition</td>
<td>observe</td>
</tr>
<tr>
<td>26</td>
<td>Ontology Web Language for Services (OWL-S)</td>
<td>Language, Standard</td>
<td>Semantic Service Description</td>
<td>Y</td>
</tr>
<tr>
<td>27</td>
<td>Modeling Web Service Composition and Execution (MoSCoE)</td>
<td>Literature</td>
<td>Service Composition</td>
<td>N</td>
</tr>
<tr>
<td>28</td>
<td>MoDe4SLA</td>
<td>Literature</td>
<td>SLA Violation, Monitoring, Petri Nets</td>
<td>observe</td>
</tr>
<tr>
<td>#</td>
<td>Title</td>
<td>Classification</td>
<td>Keywords</td>
<td>Relevant?</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>29</td>
<td>Service Centric System Engineering (SeCSE)</td>
<td>Project</td>
<td>Service Discovery, Service Composition</td>
<td>(Y)</td>
</tr>
<tr>
<td>30</td>
<td>SCENE: To offer a language for composition design that extends the standard BPEL language with rules used to guide the execution of binding, re-binding, negotiation, and self-reconfiguration operations</td>
<td>Literature</td>
<td>BPEL, Dynamic Business Processes</td>
<td>observe</td>
</tr>
<tr>
<td>31</td>
<td>EVEnt REaSoning Toolkit (EVEREST)</td>
<td>Software</td>
<td>Event-based Monitoring</td>
<td>Y (Software used in SL@SOI)</td>
</tr>
<tr>
<td>32</td>
<td>Dynamo</td>
<td>Software</td>
<td>Process Monitoring, BPEL</td>
<td>Y</td>
</tr>
<tr>
<td>33</td>
<td>Fault Tolerant Web Service Orchestration by Means of Diagnosis</td>
<td>Concept, Literature</td>
<td>Failure Monitoring, Failure Diagnosis</td>
<td>observe</td>
</tr>
<tr>
<td>34</td>
<td>Business process management</td>
<td>Architecture, Literature</td>
<td>Business Process Management</td>
<td>observe</td>
</tr>
<tr>
<td>35</td>
<td>Colombo</td>
<td>Product, Commercial</td>
<td>SOA Development, SOA Deployment, SOA Execution Platform</td>
<td>(Y) - conceptually</td>
</tr>
<tr>
<td>36</td>
<td>Model-driven Development of Monitored Processes</td>
<td>Model, Literature</td>
<td>Automatic BP Development, Monitoring, BPEL, BPMN</td>
<td>Y</td>
</tr>
<tr>
<td>37</td>
<td>Probing and Monitoring of WSBPEL Processes</td>
<td>Model, Literature</td>
<td>Business Processes, Event Tracing, BPEL</td>
<td>observe</td>
</tr>
<tr>
<td>38</td>
<td>An Agent-based Architecture for Analyzing Business Processes of Real-Time Enterprises</td>
<td>Architecture, Literature</td>
<td>BP Analysis, Agents</td>
<td>observe</td>
</tr>
<tr>
<td>39</td>
<td>VieDAME</td>
<td>Framework, Literature</td>
<td>Business Processes, Design, BPEL</td>
<td>(Y)</td>
</tr>
<tr>
<td>40</td>
<td>Reliable QoS Monitoring Based on Client Feedback</td>
<td>Concept, Literature</td>
<td>System Monitoring, QoS Properties</td>
<td>observe (after Y1)</td>
</tr>
<tr>
<td>41</td>
<td>Conformance checking of Service Behavior</td>
<td>Model, Literature</td>
<td>Service Monitoring, Status Check, Petri Nets</td>
<td>Y</td>
</tr>
<tr>
<td>42</td>
<td>Assumption-based monitoring of service compositions</td>
<td>Architecture, Software, Research</td>
<td>Service Composition, BPEL, Monitoring</td>
<td>Y</td>
</tr>
<tr>
<td>43</td>
<td>Monitoring privacy-agreement compliance</td>
<td>Concept, Literature</td>
<td>Right Management, Monitoring</td>
<td>observe</td>
</tr>
<tr>
<td>44</td>
<td>Performance monitoring for utility computing</td>
<td>Concept, Literature, Software</td>
<td>SLA Monitoring</td>
<td>N</td>
</tr>
<tr>
<td>45</td>
<td>Cremona</td>
<td>Architecture, Literature</td>
<td>SLA Lifecycle, SLA Management, WS-Agreement</td>
<td>Y</td>
</tr>
<tr>
<td>46</td>
<td>Automated SLA monitoring</td>
<td>Concept, Literature</td>
<td>SLA Monitoring</td>
<td>N</td>
</tr>
<tr>
<td>47</td>
<td>Query-based business process monitoring</td>
<td>Framework, Literature, (Prototype) Software</td>
<td>BP Monitoring, BPEL</td>
<td>Y</td>
</tr>
<tr>
<td>48</td>
<td>Model-driven development for</td>
<td>Model, Model-Driven</td>
<td>tdb</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Title</td>
<td>Classification</td>
<td>Keywords</td>
<td>Relevant?</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>--------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>49</td>
<td>Intelligent Business Operation Management (IBOM)</td>
<td>Concept, Platform, Literature</td>
<td>Business Metrics, Monitoring, KPI, Prediction</td>
<td>observe</td>
</tr>
<tr>
<td>50</td>
<td>Monitor-Analyze-Plan-Execute-Knowledge (MAPE-K)</td>
<td>Architecture, Blueprint</td>
<td>Autonomic Computing, Self-Management</td>
<td>Y</td>
</tr>
<tr>
<td>51</td>
<td>An innovative Paradigm for Autonomic Opportunistic Communication (HAGGLE)</td>
<td>Project, Architecture, Software</td>
<td>Autonomic Networking, Mobile Networks</td>
<td>(Y)</td>
</tr>
<tr>
<td>52</td>
<td>Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services (CASCADAS)</td>
<td>Project, Framework, Software</td>
<td>Component-based Applications, Self-Organisation</td>
<td>observe</td>
</tr>
<tr>
<td>53</td>
<td>BIOlogically-inspired autonomic NETworks and Services (Bionets)</td>
<td>Project, Service Framework, Architecture</td>
<td>Pervasive Computing, Embedded Systems</td>
<td>observe</td>
</tr>
<tr>
<td>54</td>
<td>Autonomic Network Architecture (ANA)</td>
<td>Project, Architecture, (Prototype) Software</td>
<td>Autonomic Networking</td>
<td>observe</td>
</tr>
<tr>
<td>56</td>
<td>SAHARA - Service Architecture for Heterogeneous Access, Resources, and Applications</td>
<td>Project, Architecture</td>
<td>Service Composition</td>
<td>observe</td>
</tr>
<tr>
<td>57</td>
<td>PROSPECT- A Prospect of Multi - Domain Management in the Expected Open Services Marketplace</td>
<td>Project, Methodology</td>
<td>Broadband Network, End-to-End Management</td>
<td>observe</td>
</tr>
<tr>
<td>58</td>
<td>Policy-based management on multidomain/multiprovider environments</td>
<td>Literature, Framework, Prototype</td>
<td>Policy-based Management, QoS</td>
<td>observe</td>
</tr>
<tr>
<td>59</td>
<td>Performance Service Oriented Network monitoring Architecture (perfSONAR)</td>
<td>Project, Architecture, Software</td>
<td>SOA, Monitoring Architecture, Network Performance Data</td>
<td>observe</td>
</tr>
<tr>
<td>60</td>
<td>OVF Open Virtualisation Format</td>
<td>Standard</td>
<td>Virtual Machine, Functional Description</td>
<td>(Y)</td>
</tr>
<tr>
<td>61</td>
<td>Libvrt</td>
<td>API, Software</td>
<td>Virtualization, OS</td>
<td>Y</td>
</tr>
<tr>
<td>62</td>
<td>OpenNebula</td>
<td>Software</td>
<td>Virtual Machine Manager</td>
<td>Y</td>
</tr>
<tr>
<td>63</td>
<td>Amazon Elastic Compute Cloud (EC2)</td>
<td>Service Offer</td>
<td>Cloud, Compute Capacity</td>
<td>N</td>
</tr>
<tr>
<td>64</td>
<td>XCalibre Flexiscale</td>
<td>Service Offer</td>
<td>Virtual Server</td>
<td>N</td>
</tr>
<tr>
<td>65</td>
<td>ElasticHosts</td>
<td>Service Offer</td>
<td>Virtual Server</td>
<td>N</td>
</tr>
<tr>
<td>66</td>
<td>ServePath GoGrid</td>
<td>Service Offer, API</td>
<td>Cloud, Compute Capacity</td>
<td>N</td>
</tr>
<tr>
<td>67</td>
<td>Zimory</td>
<td>Service Offer</td>
<td>Cloud, Compute Capacity</td>
<td>N</td>
</tr>
<tr>
<td>68</td>
<td>Nirvanix SDN</td>
<td>Service Offer</td>
<td>Cloud, Storage Capacity</td>
<td>N</td>
</tr>
<tr>
<td>69</td>
<td>Ganglia</td>
<td>Software</td>
<td>Distributed Monitoring</td>
<td>Y</td>
</tr>
<tr>
<td>#</td>
<td>Title</td>
<td>Classification</td>
<td>Keywords</td>
<td>Relevant?</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------</td>
<td>-----------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>70</td>
<td>Nagios</td>
<td>Software, Commercial</td>
<td>Monitoring</td>
<td>(Y)</td>
</tr>
<tr>
<td>71</td>
<td>Groundwork - Monitor Community Edition</td>
<td>Software</td>
<td>Monitoring</td>
<td>N</td>
</tr>
<tr>
<td>72</td>
<td>MonALISA</td>
<td>Software</td>
<td>Monitoring</td>
<td>observe</td>
</tr>
<tr>
<td>73</td>
<td>Zabbix</td>
<td>Software</td>
<td>Monitoring</td>
<td>N</td>
</tr>
<tr>
<td>74</td>
<td>Osmius – The Opensource Monitoring Tool</td>
<td>Software</td>
<td>Monitoring</td>
<td>observe</td>
</tr>
<tr>
<td>76</td>
<td>Java Message Service (JMS)</td>
<td>API, Standard</td>
<td>Messaging, Java</td>
<td>N</td>
</tr>
<tr>
<td>77</td>
<td>AMQP – Advanced Message Queuing Protocol</td>
<td>Protocol, Software</td>
<td>Messaging</td>
<td>(Y)</td>
</tr>
<tr>
<td>78</td>
<td>CO-operating Real-time senTient objects: architecture and EXPERimental evaluation CORTEX</td>
<td>Project, Concept</td>
<td>Collaborating Objects</td>
<td>observe</td>
</tr>
<tr>
<td>79</td>
<td>Real-time reconfiguration for guaranteeing QoS provisioning levels in Grid environments</td>
<td>Literature, Concept</td>
<td>QoS Provisioning</td>
<td>observe</td>
</tr>
<tr>
<td>80</td>
<td>Autonomic policy-based management</td>
<td>Literature, Architecture, Concept</td>
<td>Policy-based Management</td>
<td>observe</td>
</tr>
<tr>
<td>82</td>
<td>Rule-based and Ontology-based Policies</td>
<td>Literature, Concept</td>
<td>Rule-based Policy Model</td>
<td>observe</td>
</tr>
<tr>
<td>83</td>
<td>Model-based Performance Prediction</td>
<td>Literature, Model</td>
<td>Performance Prediction</td>
<td>Y</td>
</tr>
<tr>
<td>84</td>
<td>Performance Prediction Techniques for Component-based Systems</td>
<td>Literature, Concept</td>
<td>Performance Prediction, Component Model</td>
<td>observe</td>
</tr>
<tr>
<td>85</td>
<td>Measurement-based Approaches to Performance Prediction</td>
<td>Literature</td>
<td>Performance Prediction</td>
<td>observe</td>
</tr>
<tr>
<td>86</td>
<td>Performance Prediction Techniques for Web Services and Service-oriented Architectures</td>
<td>Literature</td>
<td>Performance Prediction, SOA</td>
<td>observe</td>
</tr>
<tr>
<td>87</td>
<td>UML Profile for Schedulability, Performance and Time (UML SPT)</td>
<td>Profile</td>
<td>Model-driven Performance Optimization, UML</td>
<td>N (or very limited)</td>
</tr>
<tr>
<td>88</td>
<td>Core Scenario Model (CSM)</td>
<td>Model</td>
<td>Model Transformation, UML, UML SPT</td>
<td>Y</td>
</tr>
<tr>
<td>89</td>
<td>Kernel Language for Performance and Reliability Analysis (KLAPER)</td>
<td>Model</td>
<td>Model-driven Performance Engineering</td>
<td>Y</td>
</tr>
<tr>
<td>90</td>
<td>Service Component Architecture (SCA)</td>
<td>Programming Model, Software</td>
<td>Application Development, SOA, Service Composition</td>
<td>observe</td>
</tr>
<tr>
<td>#</td>
<td>Title</td>
<td>Classification</td>
<td>Keywords</td>
<td>Relevant?</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------------------------------------</td>
<td>----------------</td>
<td>--------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>92</td>
<td>From Failure To Vaccine (FFTV)</td>
<td>Literature, Concept</td>
<td>Self-protecting Systems</td>
<td>N</td>
</tr>
<tr>
<td>93</td>
<td>Self-protective techniques</td>
<td>Literature, Concept</td>
<td>Self-protection</td>
<td>N</td>
</tr>
<tr>
<td>94</td>
<td>Performance Model Driven QoS Guarantees and Optimization in Clouds</td>
<td>Literature, Concept</td>
<td>Performance mode, Clouds</td>
<td>observe</td>
</tr>
<tr>
<td>95</td>
<td>A Game Theoretic Framework for SLA Negotiation</td>
<td>Literature</td>
<td>Game theory, SLA negotiation</td>
<td>observe</td>
</tr>
<tr>
<td>96</td>
<td>Turning Software into a Service</td>
<td>Literature</td>
<td>SaaS, SOA</td>
<td>observe</td>
</tr>
<tr>
<td>97</td>
<td>Optimal Web Service Composition Using Hybrid GA-Tabu Search</td>
<td>Literature, Algorithm</td>
<td>Service Composition, Genetic Algorithms, Hybrid Genetic Algorithms Service Composition, Genetic Algorithms</td>
<td>observe</td>
</tr>
<tr>
<td>99</td>
<td>Cloud computing state-of-the-art and research challenges</td>
<td>Literature</td>
<td>Cloud Computing, data centres, virtualization</td>
<td>Y</td>
</tr>
<tr>
<td>100</td>
<td>SLA-based profit optimization in Automatic Computing Systems</td>
<td>Literature, Concept</td>
<td>Service delivery, QoS, monitoring, NP-hard problem, tabu-search algorithm</td>
<td>tbd</td>
</tr>
<tr>
<td>102</td>
<td>BREIN: Business objective driven Reliable and Intelligent grids for real business</td>
<td>Models, Software</td>
<td>SLA Management, Negotiation, Brokering, Infrastructure Management</td>
<td>As BREIN is a project providing a complex framework, there is no definite answer. Please refer to the resp. section for an answer.</td>
</tr>
<tr>
<td>103</td>
<td>SLA-Negotiation in BEinGRID</td>
<td>Architecture</td>
<td>SLA Negotiation, SLA Management</td>
<td>observe</td>
</tr>
<tr>
<td>104</td>
<td>SLA-Negotiation in ASSESSgrid</td>
<td>Architecture</td>
<td>SLA Negotiation, SLA Broker</td>
<td>observe</td>
</tr>
<tr>
<td>105</td>
<td>SLA Monitoring and Evaluation in BEinGRID</td>
<td>Architecture</td>
<td>SLA Monitoring, SLA Evaluation</td>
<td>observe</td>
</tr>
<tr>
<td>106</td>
<td>IaaS resource advanced reservation</td>
<td>Models</td>
<td>IaaS, advanced reservation, virtual machine</td>
<td>Y</td>
</tr>
<tr>
<td>107</td>
<td>Online Scheduling Algorithms for Advance Reservations and QoS in</td>
<td>Models</td>
<td>Grids, advanced reservation</td>
<td>Y</td>
</tr>
</tbody>
</table>
3.1 **Shared Information and Data Model (SID)**

**Keywords:**
Information Model

**Abstract / Summary:**
The Shared Information and Data model (SID) is a set of comprehensive standardized information definitions, developed by the TeleManagement Forum (TMF), acting as the common language for building easy to integrate OSS (Operational Support System) and BSS (Business Support System) solutions.

The SID model focuses on what are called “business entity” definitions and associated attribute definitions. A business entity is a thing of interest to the business such as customer, product, service, or network, while its attributes are facts that further describe the entity. Together, the definitions provide a business-oriented perspective of the information and data that it is needed to run in an organization.

The adoption of the SID as the industry’s standard information model is growing rapidly, with many service providers, system vendors, equipment vendors and systems integrators using the SID as the basis for their development and integration. And the influence is widening as the principles are adopted by other industry forums through our industry liaison program.

**Described requirements**

**Functional:**
- The Data Model must support the description of the SLA agreed among customer, service providers and others.
- The Data Model must support the description of the agreed service, related guarantee terms, penalties and rewards.
- The Data Model must allow to represent SLA Template to be able to give to the customers the needed information in order to achieve agreements.
- The Data Model must allow the representation of all the participants in the creation and negotiation of agreements, like customers or service providers.

**Non Functional:**
- The SID model has to be extended in order to be able to describe all the entities and attributes required.
What is the novelty described in this document?

SID presents an standardized common vocabulary and a set of information/data definitions and relationships used in the definition of OSS and BSS architectures. There are described different entities divided in different layers that map on the different layers, the project is dealing to, like service, resource, customer or products.

Inside of this vocabulary there is a set of entities related to SLA and other concepts that have a close relationship with SLA.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

This specification is directly relevant to SLA@SOI and may be used for specify the common entities of the data model needed for the project. Because of they are not described all the entities needed, it is requested to extend SID to include those entities needed to the purpose of the project and not included inside it.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

There are some companies (service provider, system vendors, equipment vendors and system integrators) that has adopt SID for their development and integration. SID is a living document, but the implementations of this standard are internal implementations.

References:

3.2 Web Services Agreement (WS-Agreement)

Keywords:
SLA Model, SLA Negotiation

Abstract / Summary:

WS-Agreement [1] is a standardised specification for the establishment of SLAs between initiators and responders. It defines a protocol and the respective abstract model for linking agreements to services, irrespective of the domain-specific details of contract terms.

The specification contains a schema for specifying an agreement, a schema for specifying an agreement template, and a set of port types and operations for managing agreement life-cycle, including creation, expiration, and monitoring of agreement states.
Described requirements

Functional:

- The SLA representation language must support the description of the agreed service, related guarantee terms, penalties and rewards.
- The SLA representation language must be extensible with domain specific quality parameters.
- The SLA representation language must allow to represent SLA Template, i.e. SLAs customisable by the users by setting specific parameters.
- The negotiation protocol must allow at least one-round negotiation between a provider and a consumer.

Non Functional:

- The SLA representation language should be XML-based in order to support interoperability.
- The SLA negotiation protocol must be based on W3C web services standards in order to support interoperability.

What is the novelty described in this document?

WS-Agreement is the first standardised work on the topic of SLA establishment. It decouples service logic and SLA content from SLA signalling (the SLA envelope), therefore being suitable for applications in any domain in need of digital contracts.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

This specification is directly relevant to SLA@SOI and is already used for wrapping domain-specific guarantee terms. To this end WS-Agreement has to be extended because of the requirements of the negotiation between the customer and SLA@SOI framework on the remuneration for the use of the services.

Extensions to the protocol from a single-round negotiation which is now supported, to a multi-round negotiation that allows for small customisations and best results, will be needed later on.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

A large number of implementations of different maturity levels exists. A concise list can be found here [2]. Most of the implementations integrate WS-Agreement into their software and are therefore of no use for SLA@SOI. But a complete implementation of the specification in Java, WSAG4J [3], exists and is available for download. The implementation has been thoroughly tested as one of the two implementations needed to grant WS-Agreement “Recommendation” status and could be characterised as beta. No documentation except for one describing the API exists.
3.3 Web Services Agreement Negotiation (WS-Agreement Negotiation)

Keywords:
SLA Negotiation

Abstract / Summary:
The protocol for SLA negotiation defined by WS-Agreement [1] follows a simple “take-it-or-leave-it” approach that allows parties involved in the negotiation to only accept or reject a certain offer. Neither are modifications of any offer parameter allowed, not the re-negotiation of existing agreements.

To extend the usage of WS-Agreement and support more application scenarios, the WS-Agreement group at the Open Grid Forum proposed the WS-Agreement Negotiation protocol [2].

Described requirements

Functional:
- The protocol must support bilateral negotiation of agreement offers.
- The protocol must allow negotiation of agreement offers for new and renegotiation of existing agreements.
- It must provide a simple negotiation state machine.

Non Functional:
- It is a symmetric protocol.
- It must be built on top of the WS-Agreement specification.

What is the novelty described in this document?
WS-Agreement is the first standardised work on the topic of SLA establishment. It decouples service logic and SLA content from SLA signalling (the SLA envelope), therefore being suitable for applications in any domain in need of digital
contracts. Through the extension with the WS-Agreement protocol, its applicability is increased and a larger number of usage scenarios can be realised.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
This specification is directly relevant to SLA@SOI. To apply the protocol to SLA@SOI, adaptations and extensions of the Protocol Engine will be necessary.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
Two implementations are currently on the way: One by the European project SmartLM [3] and one by the German project SLA4D-Grid [4]. Both base on the draft specification and will be adapted to the final OGF Proposed Recommendation. Both will be open source and are currently prototypes.

References:


3.4 NextGRID Service Level Agreement

Keywords:
SLA Model

Abstract / Summary:
The NextGRID SLA [1, 2] aims at providing confidence in the robustness, reliability, security and performance of the available services, while allowing providers to operate those services in an efficient and ultimately profitable manner. This happens in a heterogeneous, loosely coupled and service oriented environment made up from multiple organisations, each owning and operating a segment of the infrastructure. Figure 1 shows that the end-to-end lifecycle of an SLA consists of six stages. The NextGRID SLA lifecycle consists of four stages:

1. SLA Publishing and Discovery;
2. SLA Negotiation;
3. SLA Operation; and
4. SLA Decommissioning.

Described requirements

Functional:
- Support for the abovementioned SLA lifecycle.
- Customer – provider symmetry: customer obligations are captured by the SLA.
- Handling of changes to operational conditions.

Non Functional:
- Strong security and trust.

What is the novelty described in this document?
The NextGRID SLA has a strong focus on the secure and trust-driven operation of SLA-enabled infrastructures as it is not captured by other well-known SLA definitions. In addition, this is the first SLA where customer obligations and provider promises are captured symmetrically.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
From a conceptual point of view, the NextGRID SLA is of interest to SLA@SOI. The SLA has been in depth developed based on industrial use cases from partners such as BT, SAP and T-Systems. Some of the use cases are similar to those in SLA@SOI, e.g. financial service provision and ERP planning. In addition, the financial service provision with SLAs has been implemented and evaluated [3].

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
None publicly available.

References:
3.5 Work by Eid, Alamri and El Saddik on “A reference model for dynamic web service composition systems”.

Keywords:
Service Management, Dynamic Service Composition

Abstract / Summary:
To enforce extensive software reuse and dynamic adaptation, dynamic service composition has experienced an increasing interest in research efforts but, the lack of a general conceptual reference model for dynamic web service composition systems and the widespread use of these systems in service-enabled applications constitute a problem of management for these systems. To capture the requirements and challenges of these composition systems, a survey of a representative set of these systems is presented. Besides, in this work, a reference model for describing the functional structure and evaluating the performance of dynamic web service composition systems based on existing dynamic web service composition platforms and prototypes, is developed.

Described requirements

Functional:
There exists a confusing set of systems at different development stages and goals. This work, must derive the functional requirements such systems must have in order to be suitable for general dynamic composition needs. The proposed model may be regarded as an approach towards understanding the functional characteristics of a typical dynamic web service composition system. Because the operational requirements and characteristics are unlikely to be identical across different applications – ranging from business-to-business applications to ubiquitous, mobile environments where available components are dynamic and expected users may vary – the modelling scheme incorporates the basic constituting functional components.

Non Functional:
The method of analysis and deduction of the reference model is based on a survey on different approaches in the context of industrial and in that one of the scientific/research area. The reviewed systems are: SeGSeC (Fujii and Suda, 2004), eFlow (Casati et al., 2000), Aurora (Marazakis et al., 1997), STONE (Minami et al., 2003), ICARIS (Mennie and Pagurek, 2000), SELF-SERV (Benatallah et al., 2002), Composer (Sirin et al., 2002), Ninja (Chandrasekaran et al., 2000), SWORD (Ponnekanti and Fox, 2002), SHOP2 (Wu et al., 2003), Theseus (Wu et al., 2003), Argos (Ambite et al., 2005; Ambite and Weathers, 2005), Proteus (Ghandeharizadeh et al., 2003), and Fusion (Vandermeer et al., 2003).

What is the novelty described in this document?
To the best of knowledge of the authors there has been no such model in the literature at the moment they wrote the work (2008). Much of the related work, in fact, has emphasised what has been done so far in web service composition. For instance, Rao and Su (2004) [2] presents a survey of automatic service
composition approaches and derives an abstract framework that identifies common characteristics and features.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

This work can be used to better evaluate and classify, the solution adopted for the dynamic service composition in SLA@SOI, in respect to an enough general framework and reference model build by abstracting from the specific approaches.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

No

**References:**


### 3.6 DIANE - An Integrated Approach to Automated Service Discovery, Matchmaking and Composition

**Keywords:**

Service Management, Dynamic Service Composition, (Semantic) Matchmaking

**Abstract / Summary:**

When trying to find a match for a certain request it may often happen, that the request cannot be serviced by a single offer but could be handled by combining existing offers. In this case automatic service composition is needed. Although automatic composition is an active field of research it is mainly viewed as a planning problem and treated separatedly from service discovery. In this work it is argued that an integrated approach to the problem is better than seperating these issues as is usually done.

Propose a language (DSD Diane Service Descriptions) for the specification of the requests, and a corresponding, ontology and fuzzy logic based, language for the description of service offers.

Propose an approach and an efficient algorithm to solve the problem of matching service requests that ask for multiple connected effects, integrating service composition into service discovery and matchmaking.
Described requirements

Functional:
1. Service description languages must offer a mechanism to integrate, inside the request, the specification of preferences.
2. Service composition must be automatically performed during service discovery and matchmaking.

Non Functional:
None.

What is the novelty described in this document?
Service composition is an integral part of service matchmaking in order to address dynamically, on the fly, situations where no single service does match a request. One of the main benefits from integrating the service composition into the matchmaking is that the requestor doesn’t only attempt to find some fitting composition but instead is able to find the composition that suits best to the given request’s precise preferences. To the best of knowledge of the authors this distinguishes DIANE composition approach from related work.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
This work propose essentially a technique (very interesting) to enable dynamic synthesis to compose unforeseen services. Anyway in the SLA@SOI use cases generally the kind of services to provide are well known.

Besides this work doesn’t specify any relationship with negotiation activity which should be involved in the activity of composition if it would be SLA-aware.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
No software available: only benchmark for the evaluation of Semantic Web Service Frameworks available at DIANE project: http://hnsp.inf-bb.uni-jena.de/DIANE/en/.

1 “The most closely related work with regard to discovery is the recently presented WSMO-MX Matchmaker by Kaufer and Klusch [2]. WSMO-MX is a hybrid matchmaker for WSML services that borrows the graph-matching approach from the DSD Matchmaker, but combines it with other concepts developed within other matchmakers which the DSD Matchmaker is lacking. What distinguishes the DSD Matchmaker most from WSMO-MX, as from all other discovery approaches is DSD’s concept of precise fine-grained preferences and ranking. The METEOR-S framework [3] provides dynamic binding of services, but works with composite service templates and does not attempt to dynamically synthesize service compositions as we do. Although METEOR-S stresses the importance to select component services optimized with regard to certain global optimization criteria like overall monetary cost, it is lacking fine grained user preferences as realized by DSD’s fuzzy sets.” [1]
3.7 Governance Interoperability Framework (GIF)

Keywords:
Information System, Service Oriented Architecture

Abstract / Summary:
The Governance Interoperability Framework (GIF) provides a standards-based approach to publishing and discovering functional and non-functional information (i.e. metadata) in a Service-Oriented Architecture (SOA) across multiple vendors.

The objective is to drive interoperability through the adoption of standards and common approaches to modelling. GIF represents a collection of APIs defined by standards organization, data mappings and classifications and, leverages UDDI and WS-Policy standards among others as building blocks. In cases where a particular standard does not exist, GIF proposes either an extension to existing standards or an approach to attaining integration in as “open” a manner as possible (by GIF partners).

The SOA components covered by GIF include a SOA governance system incorporating UDDI registries, XML intermediaries, and SOA management systems. There are two pillars of integration: control integration and business service data integration, both based on the Model-View-Controller (MVC) pattern.

The scope of the current GIF framework is the following:
Registry APIs are adopted from UDDI specification (inquiry, publication, security, subscription, custody and transfer ownership).

The UDDI data model provides for the categorization of published entities, through the categoryBag structure including representations of organizational units, contacts, relationships, consumer-producer relationships, Web Services Management (WSM) framework, WSM Metrics (Aver. Response Time, Aver. Message Count, Count of Failure, et), Constraints, Configuration, and Capabilities and Runtime Policy vocabulary (not released yet).

Described requirements

Functional:
• linkage of SLA specifications with management activities

Non Functional:
• interoperability between different WS- and related standards

What is the novelty described in this document?
The Governance Interoperability Framework is not novel in the scientific sense nor is a standard itself, but rather leverages existing standards to support SOA governance interoperability. Though SLAs have not yet been explicitly addressed there is already some work on specifying Web Service Metrics.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it? What the needed improvement?
The further development of GIF has to be carefully watched as extensions for SLA management are planned for the next phases.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
There is a commercial offering of management products around GIF provided by HP.

References:
[1] HP SOA Governance Interoperability Framework (GIF)
[https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-130-27%5E2804_4000_100]

3.8 CentraSite
Keywords:
SOA registry, SOA repository

Abstract / Summary:
CentraSite is a standards-based SOA registry and repository jointly developed by Fujitsu and Software AG. As the central "SOA Store" for the enterprise, CentraSite aims at greater visibility and control of integrated SOA based applications, better support on decision-making, and increased productivity. It serves as a Web services and SOA asset management platform, holding all an enterprise's metadata assets (DNA of enterprise applications), and offering reports on usage.

This shall be achieved through:
• Complete SOA lifecycle management
• Better reliability - understanding impact of changes in your SOA before they are made
• Improved transparency and enhanced collaboration across the organization
Run-time governance also involves service-level agreement (SLA) monitoring and reporting. By tracking the actual performance of a service and comparing it to the requirements specified in the SLA, the system can identify non-compliant services that require prompt action.

**Described requirements**

**Functional:**
- Compliance of service metrics with WS-Policy

**Non Functional:**
None.

**What is the novelty described in this document?**
ContraSite is not novel in the scientific sense. However, it is quite relevant in terms of integrated modelling and management of SOA. Actual SLA models are probably in a proprietary format and hierarchical SLAs are out of scope.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it? What the needed improvement?**

The further development of CentraSite has to be carefully watched, in particular how SLA management steps are included into overall governance processes.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

There is a commercial offering of CentraSite via both, Fujitsu and Software AG.

**References:**


### 3.9 TM Forum SLA Management Handbook

**Keywords:**
SLA Management

**Abstract / Summary:**

SLA Management Handbook series is a handbook to assist two parties in developing a Service Level Agreement (SLA) by providing a practical view of the fundamental issues. The parties may be an "end" Customer, i.e., an Enterprise, and a Service Provider (SP) or two Service Providers. In the latter case one Service Provider acts as a Customer buying services from the other Service Provider. For example, one provider may supply network operations services to the provider that supplies leased line services to its customers. These relationships are described as the Customer-SP interface and the SP-SP interface.
The perspective of the SLA Management Handbook series is that the end Customer, i.e., an Enterprise, develops its telecommunication service requirements based on its Business Applications. These requirements are presented to a Service Provider and the two parties begin negotiating the specific set of SLA parameters and parameter values that best serves both parties. For the SP, the agreed-upon SLA requirements flow down through its organization and become the basis for its internal management and control of its Quality of Service (QoS) processes. For the Enterprise Customers, the SLA requirements serve as a foundation or a component of its internal network services or business services.

This Document is divided into three Volumes.

**Described requirements**
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

**Functional:**
- What contents have to have an SLA.
- What KPI (Key Performance Parameters) are important to deal to in an SLA.
- Methods to calculate the important parameters of a system.
- What are the relationships among different KPIs.

**Non Functional:**
None.

**What is the novelty described in this document?**
TM Forum SLA Management Handbook describe the principles of what contents has to have an SLA, the KPI (Key performance parameters) are important to deal to, and the relationship among them.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
This specification is directly relevant to SLA@SOI because one of the purpose of the project is create agreements between customer and service providers to be able to consume services. From this point of view, this handbook exposes the principles of what contents have to have these SLA.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**
No information provided.

**References:**
3.10 TMForum Service Delivery Framework (SDF) Catalyst Project

Keywords:
Product Lifecycle Management, Service Provisioning, Event Management, SLA Model

Abstract / Summary:
The Syndicated Services Catalyst Project tries to demonstrate a new concept that standardizes Product Lifecycle Management, service provisioning, event generation, and subscription across Service Provider domains to facilitate service syndication and end-to-end management of the services.

A Syndicated Service is a self-contained service that has been created and established in a hosted environment by a Service Provider and is ready to be used (or "on-boarded") by another Service Provider. From these syndicated services, it is possible to create commercial agreements or SLAs associated with the syndicated services.

The Service Provider who can syndicate services must expose service access, usage, assurance, and billing capabilities. The Service Provider on-boarding the service will use these capabilities to unify and extend to syndication partners its own product management process including updates to fulfillment, assurance, and billing processes.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- There must be a catalog of products and services.
- There must be a method to manage the product’s lifecycle.
- There must be a method to discover resources.
- There must be a method for a service provider to syndicates their services.
- There must be possible the monitorization of the health of resources.
- The product must implement the track usage of resources.

Non Functional:
- There must be follow WS-eventing as a protocol that allows Web services to subscribe to or accept subscriptions for event notification messages.

What is the novelty described in this document?
This project presents new concepts used in the new environment of contracting services. It tries to demonstrate the profits of the syndication of the services in order to be used by other service provider or the events generation in order to improve the quality and the management end to end of services.

Other goal achieved by this project, is the improvement of the final user perception.
How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

This specification is directly relevant to SLA@SOI and may be taken into account because this project deals with some concepts that SLA@SOI deals too like the management end to end of the services, from the creation to the monitorization of the services.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

There is a proof of concept created by the members of the TM Forum in order to demonstrate their objectives. There is a prototype of internal use.

References:

3.11 Web Services Agreement Language (WSLA)

Keywords:
SLA Model

Abstract / Summary:
WSLA’s [1] centre of attention lies in providing a framework and a technology for monitoring and evaluation of Service Level Agreements. Its core language schema provides the means to define Quality of Service statements, as well as the consequences arising from it – thus providing the basis for electronic contracts. As opposed to WS-Agreement, it does not support means for publication and negotiation of contracts though.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- A description of the parties, their roles (provider, customer, third parties) and the action interfaces they expose to the other parties of the contract.
- A detailed specification of the service level parameters (SLA parameters) guarantees are applied to.
- Metrics to define how to measure an item, in the case of Resource metrics, or how to aggregate metrics to composite metrics.
- A representation of the parties’ obligations. Service level objectives for a formal expression of the guaranteed condition of a service in a given period. Action guarantees to represent promises of parties to do something, for example, to send a notification in case the guarantees are not met.

**Non Functional:**

- The SLA representation language should be XML-based in order to support interoperability.
- The SLA negotiation protocol must be based on W3C web services standards in order to support interoperability.

**What is the novelty described in this document?**

./.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

It could be of interest to read the WSLA specification, but most of its concepts have been transferred into WS-Agreement (IBM was involved in WS-Agreement, which initially has been seen as a successor of WSLA). In addition, WSLA is not actively maintained anymore although there is still some uptake.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

Not as far as the author’s knowledge is concerned.

**References:**


### 3.12 Web-based Enterprise Management (WBEM) Standards

**Keywords:**

WS-Management, Common Information Model

**Abstract / Summary:**

Web-Based Enterprise Management (WBEM) represents a set of management and Internet standard technologies developed by the Distributed Management Task Force (DTMF). The goal of these standards is to unify the management of distributed computing environments. [1]
WBEM comprises the following components [1][2]:

- **Common Information Model (CIM):** CIM provides a common definition of management information for systems, networks, applications and services, and allows for vendor extensions. CIM’s common definitions enable vendors to exchange semantically rich management information between systems throughout the network.
  - The CIM schemas provide the actual model descriptions. Schemas are sets of CIM classes that represent a particular management domain.
  - The CIM Specification defines the details for integration with other management models.

- **Managed Object Format (MOF):** The MOF is a formal description of the classes and associations used in CIM schemas.

- **CIM-XML:** A protocol that uses XML over HTTP to exchange Common Information Model (CIM) information.

- **CIM Query Language (CQL):** A query language that is used to select sets of properties from CIM object instances.

- **CIM Object Manager (CIMOM):** The database where the instances of the CIM classes are stored. A CIMOM is the central point for accessing management resources.

- **WS-Management:** Web Services for Management (WS-Management) provides a common way for systems to access and exchange management information across the entire IT infrastructure. By using Web services to manage IT systems, deployments that support WS-Management will enable IT managers to remotely access devices on their networks. WS-Management allows the management information in the CIM to be exposed in a Web services environment.

### Described requirements

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

#### Functional:

- To provide standards for creating a centralized and harmonized management view on distributed systems, composed of network, system and application components.

- To support designing and implementing manageability infrastructures for distributed systems.

#### Non Functional:

- Extensibility: Reference models are extensible and may therefore be adapted to new requirements.

- Scalability: Distributed architecture for manageability infrastructure (Manager-Agent-Model)

### What is the novelty described in this document?

- Comprehensive framework for designing and implementing manageability infrastructures for distributed systems with extensive reference data/information model.

- Covers the whole stack of distributed systems
- Fully object-oriented approach to information modelling with precise
  metamodel and well-documented semantics of reference elements

How may this work be applicable to SLA@SOI? What are the necessary
actions that should be taken to apply it?

The Common Information Model on the one hand serves well as basis for
SLA@SOI infrastructure and application/service landscape model (used for SLA
provisioning and monitoring) as well as application performance model (used for
SLA monitoring and adjustment). The WBEM framework (CIM provider + CIM
Object Manager) in combination with WS-Management on the other hand can be
used for implementing unified and harmonized manageability interface along with
the underlying infrastructure. Nevertheless, extensions are required for modelling
management information for service-oriented and component-based applications
as well as for an SLA-driven management of resources. So far, SLAs and their
relationship to the underlying resources are not well reflected.

Is there a readily available implementation of the described work? If so,
please state its licensing type and its maturity level (e.g.
prototype/alpha/beta/RC/stable).

There are several stable implementations available e.g. Open WBEM, Microsoft
Windows Management Instrumentation (WMI), and Solaris WBEM, which support
the development of the manageability infrastructure. All frameworks but MS-WMI
are open source.

References:
[1] DMTF: Web-Based Enterprise Management (WBEM), URL:
[2] P. Monday: Introduction to WBEM and the CIM, IBM developer works,

3.13 CIM - Common Information Model

Keywords:
Information Model

Abstract / Summary:
The Common Information Model (CIM) is a standard, issued by Distributed
Management Task Force (DMTF), that specifies how entities, such as physical
machines and software packages, in an IT environment are represented as a
common set of objects and relationships between them. The goal is to allow
consistent management of these entities, independent of their manufacturer or
provider. CIM provides this this standard by a set of extensible schemas that
allow 3rd parties and vendors add their specific needs. The CIM schema is broken
down into three different areas:

- Core Schema - not specific to any platform, is an information model that
captures notions that are applicable to all areas of management
- Common Schema - this schema consists of a number of schemas that
capture technology/platform independent notions that are common to
particular management areas for example devices, networks, systems, applications.

- Extension Schema - 3rd parties and vendors can extend the basic model class and associations with their own proprietary model to cover the management area particular to them.

CIM models can be represented in a number of different representations, the most familiar being Managed Object Format (MOF) and CIM-XML. MOF is a representation that follows the same approach as an interface definition language such as OMG IDL. CIM models can be queried using the CIM Query Language and such models are stored and accessed in what is known as a CIM Object Manager (OM). To access a CIM OM one can use the standard of WS-Management.

**Described requirements**

**Functional:**
CIM is a complete and encompassing set of standards that allow for the functional operation and management of managed entities.

**Non Functional:**
None.

**What is the novelty described in this document?**
CIM is an industry standard. It also is very comprehensive in the number of entities that are modelled with it. Indeed it is its size that could be seen as its disadvantage. The DTMF standards body who maintain the CIM standard are also actively pursuing modelling and management related activities in the area of virtualisation which is very appropriate to SLA@SOI. Such initiatives of interest include the Open Virtualisation Format (OVF) and the Virtualisation Management Initiative (VMAN). Of particular note, CIM provides a UML Profile for itself.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
This work could be used to expose the internal model of the infrastructure landscape to the infrastructure provider. By exposing it as such the infrastructure provider could then use a suite of CIM management tools to interact and manage the infrastructure landscape.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**
Several stable implementations are available, including:

2. OpenPegasus, see [http://www.openpegasus.org/](http://www.openpegasus.org/), published under an MIT license
3. OpenWBEM, see [http://www.openwbem.org/](http://www.openwbem.org/), published under a BSD style license
3.14 Web Services Distributed Management (WSDM) Standard

Keywords:
Web Services Management

Abstract / Summary:
The Web Services Distributed Management (WSDM) standard was submitted to the Organization for the Advancement of Structured Information Standards (OASIS), with version 1.0 ratified in March 2005, and version 1.1 ratified in August 2006 [1]. It seeks to unify management infrastructures by providing a vendor, platform, network, and protocol neutral framework for enabling management technologies to access and receive notifications of management-enabled resources. [2]

The WSDM standard specifies how the manageability of a resource is made available to manageability consumers via Web services. It consists of the following standards:

- The Management Using Web Services (MUWS) standard deals with the basic mechanisms and Message Exchange Patterns (MEPs) for managing any manageable resource using Web services as the platform for exchanging messages.
- The Management of Web Services (MOWS) standard addresses the management of a Web service itself that is a Web service is the manageable resource. MOWS may be viewed both as an application of the WSDM MUWS standard and as an extension of the WSDM MUWS standard.

Manageable resource, Endpoint Reference (EPR) and manageability consumer are the three main components of WSDM. Management information regarding the resource must be accessible through a Web service endpoint. To provide access to a resource, this endpoint must be able to be referenced by an EPR as defined in the WS-Addressing standard. The EPR provides the target location to which a manageability consumer directs messages to retrieve and change management information. On the other hand, the manageable resource may also direct notifications of significant events to a manageability consumer, provided the consumer has subscribed to receive notifications.

WSDM is built upon W3C and other OASIS standards like XML, SOAP, WSDL, WS-Addressing, WS-ResourceFramework and the Message Exchange Patterns (MEP).
Described requirements  
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

**Functional:**
- Provide a standard for management of and using web services which seamlessly integrates into existing Web Services (WS) standards.  
  o Provide Management capabilities as service, meaning they are offered in a standardized way (using WS endpoints) and are location-transparent, discoverable, composable etc.  
  o Offer reference design for manageable Web Services

**Non Functional:**
- Interoperability: Vendor, platform, network, and protocol neutral framework for enabling management technologies to access and receive notifications of management-enabled resources.

**What is the novelty described in this document?**
- Standards specifies in detail how to extend resources by manageability interfaces based on WS standards (Management Using Web Services, MUWS)
- Standards offers recommendations / reference models for Management of Web Services (MOWS)

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

WSDM-MUWS expresses how access to manageable resources can be provided via web services. Thus it supports the interoperability between manageable resources and manageability consumers. WSDM-MOWS enables the modelling of atomic web services as managed resource but there are no details on the composition of a manageable resource. In SLA@SOI this standard could be used for designing and implementing the unified manageability interface. However, WSDM specifies neither a comprehensive information model nor a (management) function model. Moreover, it does not include an instrumentation technology and does not specify how the underlying manageability infrastructure should be implemented. It only focuses on the interface design using WSDL. Thus, in SLA@SOI we would have to combine the WSDM standard with an existing or custom framework for implementing the manageability infrastructure as well as with existing or custom technologies for instrumenting our resources (WS and WS compositions).

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

With Apache Muse an initial Java-based and open source implementation of the standard specifications exists that supports the implementation of the unified manageability interface. Nevertheless, additional technologies, like WBEM-CIM, are required in order to implement the underlying manageability infrastructure.

**References:**
3.15 Java Management Extensions (JMX)

Keywords:
Resource Management, Java

Abstract / Summary:
The Java Management Extensions (JMX) technology was developed by Sun Microsystems and is part of the Java Platform Standard Edition since the Java 2 Platform, Standard Edition (J2SE) 5.0 Release.

JMX is used to manage resources such as applications, devices and services with the Java Programming Language. Because of its dynamic structure JMX can be used to monitor and manage resources as they are created, installed and implemented [1].

The architecture of the JMX Technology consists of three levels:

- The instrumentation of the managed resources is implemented by Java objects known as Managed Beans (MBeans). These beans follow the design patterns and interfaces defined in the JMX specification. MBeans can represent any resource that needs to be managed and are designed to be flexible, simple, and easy to implement. Furthermore, a notification mechanism is provided by the instrumentation level, which allows MBeans to create and distribute notification events to other consumers, like further MBeans or a management application.

- A JMX agent manages the MBeans. It directly manages the lifecycle of the MBeans and makes them available to remote management applications. The MBean server in which MBeans are registered is the core component of a JMX agent. Further parts are a set of services to manage the MBeans and at least one adaptor or connector for the communication with a management application.

- The management application outside the agent’s Java Virtual Machine is the third level of the JMX architecture. The multiple connectors of a JMX agent provide the same remote management interface through different protocols. Thus, a remote management application can connect to a JMX agent transparently through the network without consideration of the protocol.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- Provide a standard for management of and using Java Enterprise Edition.
Non Functional:
- Neutral to operating system
- Compliance to JEE specification
- Scalability: Distributed architecture for manageability infrastructure (Manager-Agent-Model)

What is the novelty described in this document?
- Comprehensive framework for instrumenting resources of a distributed system using
- Seamless integration with Java Enterprise Edition, which for instances eases instrumentation of JEE-based applications
- JMX management agent as an out-of-the-box component for managing MBeans (=manageability interfaces of single resources)

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Although JMX is a platform-dependent standard, the managed resources have to support Java technology. Thus, it would be applicable to the SLA@SOI open reference case, but not necessarily for every industrial use case. Moreover, JMX does not provide an information model. In this case, a complementary model has to be used, for instance the Common Information Model. Also, the instrumentation of the resources in terms of sensors and effectors is not supported by JMX. In summary, we could use JMX in SLA@SOI for implementing a manager-agent based manageability infrastructure design.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
Stable implementations of JMX are included in every available JEE application server, for instance the open source solutions Sun Glassfish or JBoss.

References:

3.16 OASIS Data Centre Markup Language (DCML)

Keywords:
Data Format, Data Model

Abstract / Summary:
OASIS’s Data Centre Markup Language (DCML) is a data format and corresponding data model for exchanging information among management systems. Systems exchange information in a well known vocabulary and a well defined format. The DCML framework specification defines the conceptual data model in which data centre elements are described, how this data model is extended to represent specific elements, how the conceptual model is encoded...
into DCML document instances, and processing rules for interpreting DCML document instances.

**Described requirements**

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

**Functional:**
- Representation of state of the environment itself, similar to and compatible with what would be described by a traditional management system.
- Specification of the blueprint or recipe that can be used by an automation system to construct and manage an environment.
- Description of set of rules, policies, best practices and standards that should be used in managing the environment.

**What is the novelty described in this document?**

DCML is designed to be used in a number of scenarios. First is by automation systems to codify descriptions of environments and the formerly manually interpreted rules governing the management of those environments. Second is by traditional and automated management systems alike to describe and exchange their views on and representations of managed environments. DCML is a data format and corresponding data model. It is not a protocol or application program interface. Therefore, it specifies how to represent information, not how to transfer, access, create or modify information. Future work in these areas is possible, but for now, DCML defines the first step in modern management system information sharing.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

Conceptually, the DCML attempt is directly relevant and applicable to SLA@SOI context because SLA@SOI targets hosted software in enterprise data centres. However, DCML has not gained any acceptance from the industry. There hasn’t been much activity in this regard and no latest developments have emerged in the DCML effort. Therefore, adopting DCML wouldn’t be optimal decision because it might become obsolete. However, concepts from DCML can be adopted to be incorporated into SLA@SOI’s landscape data model.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

As mentioned previously, DCML didn’t gain any acceptance from the industrial community. There are no tools / implementations available for DCML.

**References:**

3.17 OMG System Modelling Language (SysML)

Keywords:
Modelling, System Engineering Applications

Abstract / Summary:
Object Management Group’s System Modelling Language (SysML) is a modelling language for system engineering applications. SysML supports the specification, analysis, design, verification and validation of a broad range of complex systems. These systems may include hardware, software, information, processes, personnel, and facilities. The SysML language reuses and extends many of the packages from UML. The SysML profile specifies the extensions to UML. It references the UML4SysML package, thus importing all the metaclasses into SysML that are either reused as-is from UML or extended in SysML. The semantics of UML profiles ensures that when a user model “strictly” applies the SysML profile, only the UML metaclasses referenced by SysML are available to the user of that model. If the profile is not “strictly” applied, then additional UML metaclasses which were not explicitly referenced may also be available. The SysML profile also imports the Standard Profile L1 from UML to make use of its stereotypes.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- SysML model management constructs support models, views, and viewpoints. These constructs extend UML’s capabilities and are architecturally aligned with IEEE-Std-1471-2000 (IEEE Recommended Practice for Architectural Description of Software Intensive Systems).

Non Functional:
- Requirements driven. SysML is intended to satisfy the requirements of the UML for SE RFP.
- UML reuse. SysML reuses UML wherever practical to satisfy the requirements of the RFP, and when modifications are required, they are done in a manner that strives to minimize changes to the underlying language. Consequently, SysML is intended to be relatively easy to implement for vendors who support UML 2.1 or later versions
- UML extensions. SysML extends UML as needed to satisfy the requirements of the RFP. The primary extension mechanism is the UML 2.1 profile mechanism as further refined in Chapter 17, “Profiles & Model Libraries” of this specification.
- Partitioning. The package is the basic unit of partitioning in this specification. The packages partition the model elements into logical groupings which minimize circular dependencies among them.
- Layering. SysML packages are specified as an extension layer to the UML metamodel.
Interoperability. SysML inherits the XMI interchange capability from UML. SysML is also intended to be supported by the ISO 10303-233 data interchange standard to support interoperability among other engineering tools.

What is the novelty described in this document?
SysML supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. SysML offers systems engineers several noteworthy improvements over UML, which tends to be software-centric. For example, SysML's semantics are more flexible and expressive, SysML is a smaller language that is easier to learn and apply etc.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Although SysML is quite versatile in modelling a variety of systems, it doesn’t provide any specialized features that are suitable for SLA@SOI out of the box. However, conceptually some features can be investigated to be incorporated into the SLA@SOI landscape model. Another advantage of using this approach could be interoperability with other modelling approaches being adopted within SLA@SOI which are UML based.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
There are number of tools supporting SysML modelling activities. These include EmbeddedPlus SysML toolkit (add-in to IBM Rational Software Modeller / Architect), IBM Rhapsody (formerly TeleLogic), Enterprise Architect + MDG Technology for SysML from Sparx Systems, MagicDraw SysML plugin and TOPCASED-SysML.

References:

3.18 GLUE Specification

Keywords:
Information Model

Abstract / Summary:
The GLUE specification is provided by the GLUE working group and is currently available in version 2.0. The specification includes a conceptual information model for Grid entities described using natural language and enriched with a graphical representation using UML Class Diagrams [1]. In addition to this, the working
The following main entities of the GLUE information model are introduced by [1]:

- **Location**: describes geographical positions of domains and services.
- **Contact**: represents contact information for requests related to different areas (e.g., user support, security or sysadmin).
- **Domain**: a collection of actors that can be assigned with roles and privileges to entities via policies.
- **Service**: abstracted, logical view of actual software components that participate in the creation of an entity providing one or more functionalities useful in a Grid environment.
- **Endpoint**: a network location having a well-defined interface and exposing the service functionalities
- **Share**: a utilization target for a set of resources managed by a local manager and offered via related endpoints. The share is defined by configuration parameters and characterized by status information.
- **Manager**: a software component locally managing one or more resources. It can describe also aggregated information about the managed resources.
- **Resource**: provides a capability or capacity, managed by a local software component (manager), part of a logical service, reachable via one or more endpoints and having one or more shares defined on it.
- **Activity**: a unit of work managed by a service and submitted via an endpoint; when accepted by the endpoint, than it can be mapped to a share and can be executed by a local manager via one or more resources.
- **Policy**: defines statements, rules or assertions that specify the correct or expected behaviour of an entity.

Based on these abstract entities GLUE introduces a conceptual model for a computing service to share computational capacity in a Grid environment and a storage service to share storage capacity in a Grid environment.

**Described requirements**
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

**Functional:**
- Comprehensive conceptual information model for Grid environments covering all relevant aspects.

**Non Functional:**
- Easy-to-use

**What is the novelty described in this document?**

- Covers management information for complete Grid environments, which other (reference) information models have not sufficiently addressed so far, e.g., the Common Information Model.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
The GLUE information model covers the functional aspects of a Grid environment, but does not address information required for service-oriented applications. Moreover, it is limited to the modelling of conceptual information models. Regarding the implementation, it refers to the WBEM standards. Thus, in SLA@SOI GLUE could serve as a basis for modelling and implementing management capabilities for the virtualized infrastructure, but is less adequate for implementing the application-level manageability infrastructure.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

With GLUEMan [3] there is a framework available to manage information providers for GLUE 2.0. It also provides a client enabling to query GLUE 2 information and render them according to the available mappings. GLUEMan is based on Open Pegasus [4] an open-source implementation of the DMTF CIM and WBEM standards in C++.

**References:**


**3.19 Service Modelling Language (SML)**

**Keywords:**

System Modelling, Service Modelling

**Abstract / Summary:**

The Service Modelling Language (SML) is the successor of Microsoft’s System Definition Model (SDM). The SML specification was developed by an Industry Group and is currently being standardized in a W3C working group established to generate W3C Recommendations.

SML [1] provides a set of constructs for creating and validating models of complex services and systems. These models might contain policy, deployment, configuration information as well as service level agreements, etc. and are described in a technically uniform XML format.

In SML a model is realized as a set of interrelated XML documents. Thereby a distinction is draws between the definition documents and the instance documents. The definition documents include the abstract structure of the model as well as information a model validator requires to decide whether the model as a whole is valid or not. The instance documents support the description of the individual resources represented by the model.

The constraints described within the definition documents are captured in two ways:
• Schemas: A Schema is a constraint on the structure and content of the documents in a model. The schema language used in SML is XML Schema. Additionally a set of extensions to XML Schema is defined to support cross references between documents.

• Rules: Rules are Boolean expressions that constrain the structure and content of documents within a model. In SML Schematron and XPath is used to define rules.

To interchange SML models between different systems the Service Modelling Language Interchange Format standard is available. The purpose of SML-IF is to package the set of documents that form an SML model into a standard format so that it can be exchanged in a standard way [2].

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- Generic XML-based framework for specifying arbitrary management information, both in an abstract way and on the instance level

Non Functional:
- Interoperability: Vendor, platform, network, and protocol neutral framework
- Extensibility: Can be adapted to all kinds of management environments:

What is the novelty described in this document?
• Generic XML-based approach for describing and managing management information

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

The SML could be used in SLA@SOI for designing and implementing the various landscape models. However, SML is a domain-neutral language and does not define any management-specific features. Thus, it can be seen as a layer of model on top of existing reference information models (e.g. CIM), describing the desired state, interconnected relationships and management policies of the distributed system. A comprehensive application performance management, however, is not well supported. Moreover, the SML specifications are in an early development stage (last call version). Therefore, they are not fully mature yet and only little tool support is available so far.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

The Eclipse Foundation offers an open source reference implementation of the SML specification called COSMOS (Community-driven Systems Management in Open Source) [3]. However, the implementation is in a very early phase yet.


3.20 Common Model Library (CML)

Keywords:
System Modelling, Service Modelling, SML-based

Abstract / Summary:
The Common Model Library (CML) is specified by the CML working group, a consortium of 11 leading technology companies. The specification is based on the W3C Service Modelling Language (SML) and defines common expressions and the semantics for concepts, which enable information exchange between both management tools and managed resources [1].

Currently, the consortium is working on an initial draft of the specification i.e. there is no concrete specification available yet. According to [2], the CML is expected to include:

- A library of models expressed as SML compliant documents
- Common and shared modelling elements expressed as SML document fragments
- Guidelines for encoding models
- Patterns and best practices
- Semantic definitions
- Examples and scenarios
- Compliance or conformance suites, validated at group workshop meetings

CML will support a multitude of management disciplines including service desk, configuration management, performance management, systems management, SLA management, and more. It will also facilitate the sharing of information across a typical set of lifecycle phases, including planning, development, deployment, and operations phase.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- Reference library/model for a multitude of management disciplines
- Clear definition of semantics
- ITIL compliance

Non Functional:
- Interoperability: Vendor, platform, network, and protocol neutral in combination with SML

References:
- Extensibility: Can be adapted to all kinds of management environments
- Precise definition of semantics

What is the novelty described in this document?
- ITIL compliant reference model
- Service-orientation: Concepts of services and SLAs included in reference model

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
CML looks very promising and will probably provide a suitable information model for SLA@SOI manageability infrastructure. However, neither a ratified specification nor an implementation is available so far.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
Not available.

References:

3.21 ITIL Configuration Management Databases (CMDB)

Keywords:
Configuration Management, ITIL

Abstract / Summary:
Configuration Management Database (CMDB) is a repository of information related to the components of the Information Systems. The term CMDB stems from ITIL (Information Technology Infrastructure Library). In the ITIL context, a CMDB represents the authorized configuration of the significant components of the IT environment. A key goal of a CMDB is to help an organization understand the relationships between these components and track their configuration. The CMDB is a fundamental component of the ITIL framework's Configuration Management process. CMDB implementations often involve integration with other systems, such as Asset Management Systems.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)
Functional:
- The model repository should be able to accommodate models targeting different domains.
- There exist various repositories within an organization with specialized information sources. It is of vital importance that data from various sources should be able to collated and compiled to present a consolidated, consistent and coherent view of the enterprise landscape.
- The repository should be able to host different data models and must be bound to specific data model. The compliance to data models shall be left to the implementation efforts.

What is the novelty described in this document?
A key goal of a CMDB is to help an organization understand the relationships between these components and track their configuration. The CMDB is a fundamental component of the ITIL framework's Configuration Management process. CMDB implementations often involve integration with other systems, such as Asset Management Systems. These integrations may make use of either a real-time, federated design or an ETL (extract, transform, load) solution.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
SLA@SOI architecture contains a number of architectural components to store information about various types of landscape elements. For example, service landscape, software landscape, infrastructure landscape etc. CMDB concepts can be used to implement these architectural components. There are various open source implementations available which can be used readily with minor efforts. Additionally, using the CMDB would help the project align with ITIL concepts and adopt the processes and best practices prescribed within the ITIL framework.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

There are a very large number of CMDB implementations available in the market. Most of these implementations are predominantly commercial, but some open source implementations are also available. Some of the commercial implementations / products include IBM CCMDB, HP uCMDB, BMC Atrium CMDB, CA CMDB etc. The open source CMDB implementations include OneCMDB, RapidCMDB etc.

3.22 Making BPEL Flexible - Adapting in the Context of Coordination Constraints Using WS-BPEL

Keywords:
Business Processing, Constraints, BPEL
Abstract / Summary:
Adaptive processes need to use concurrent activities that must satisfy coordination constraints. The paper shows how these constraints can be added to BPEL processes, and how instrumentation can be used to provide a modified version of the process in which they are enforced. The authors use SWRL to extend a process definition with coordination constraints. The new version of the process uses the internal throw of fault variables to enforce the constraints.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
Enforcement of coordination constraints that are added to the process using SWRL (Semantic Web Rule Language).

Non Functional:
Processes that contain SWRL constraints are still standard BPEL processes, since they are introduced using the standard BPEL extension mechanism. This means the process can still be run on any standard BPEL engine. After the adaptive version of the process is created, the process is still 100% BPEL compliant. There is no need for special execution environments.

What is the novelty described in this document?
The use of SWRL as an instrumentation language, the use of deployment-time process modification, and the focus on coordination constraints.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The paper is of interest for our work on instrumentation, since it gives example of similar work, although the initial requirements are different from ours. However, it does not apply directly to SLA@SOI since we are interested in run-time instrumentation, while in this work instrumentation is done at deployment time.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
Nothing reported.

References:
3.23 Towards Aspect Weaving Applications

Keywords:
BPEL, AOP

Abstract / Summary:
In this paper the authors advocate that software need to accommodate new features in the context of changing requirements. This can be achieved using Aspect Oriented Programming techniques. In the paper the authors make the example of a BPEL engine extended with AOP, as well as that of processes themselves. They also advocate that both can benefit from using a domain-specific aspect language that is easier to manipulate than a general-purpose one. The implementation provided is based on SmartTools, a toolkit for creating semantic analysers. Indeed, the BPEL engine they use is based on proprietary technology.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
Provide a simple mechanism to evolve a system avoiding re-deployment.

Non Functional:
Study the use of domain-specific languages instead of general-purpose ones.
Study the use of AOP to provide

What is the novelty described in this document?
The main novelty is the use of domain-specific aspect languages. This paper is also one of the first to present an approach that combines AOP techniques with BPEL.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The paper is of interest since it clearly discusses the needs for instrumentation in dynamic applications. The implementation, however, is proprietary and cannot be of help in SLA@SOI.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

References:
3.24 A dynamic and reactive approach to the supervision of BPEL processes

Keywords:
BEPL, ActiveBPEL; Process Monitoring, WSCol, WSReL

Abstract / Summary:
In this paper the authors present a supervision framework for BPEL processes. The framework uses instrumentation to provide both monitoring and recovery capabilities. Monitoring activities are defined using the WSCoL language, while recovery is defined using the WSReL language. The authors provide an AOP extended version of ActiveBPEL, in which monitoring and recovery are enabled. The instrumentation is therefore added on-the-fly, so that a defined process does not need to be re-deployed should the monitoring and/or recovery requirements change.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
Provide a framework for the run-time monitoring and recovery of BPEL processes.
Possibility to modify the amount of monitoring and recovery activities being performed at run time.

Non Functional:
Proposal of two flexible languages for monitoring and recovery with a level of abstraction similar to that of the BPEL processes themselves.

What is the novelty described in this document?
Nothing reported.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The approach is very relevant to the SLA@SOI project. Indeed, some of the code can provide an inspiration for our implementation. Unfortunately, the code was developed for a very old version of ActiveBPEL, and since then many technical details have changed.
Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

Nothing reported.

References:

3.25 ServiceCom: prototype tool for service composition specification, construction and execution developed by Tilburg University

Keywords:
Service Life Cycle, Service Composition

Abstract / Summary:
The theoretical approach (see SOBI) the work is based on is focused on a number of aspects that have not been addressed thus far in other initiatives with regard to service composition.

These include a “life cycle” of service composition, an abstract service composition “information model”, a “methodology” for the dynamic and flexible development of compositions, and the use of “service components” for reuse and specialization of compositions [3].

Service composition life cycle
This framework defines a phased approach to the specification of service compositions, which is known as the service composition life-cycle. In this approach four broad phases are distinguished spanning composition definition, scheduling, construction and execution, as illustrated in the following figure:
The service composition development starts with an abstract definition and gradually makes it concrete so that executable service processes can be generated from these abstract specifications. Briefly, the main phases are:

- **Definition**: specifies the constituent activities of a composite service and the constraints under which they operate.
- **Scheduling**: the service composition system determines how and when services will run and prepares them for execution. During this phase the system may generate alternative composition schedules and present them to the application developer for selection.
- **Construction**: construct an unambiguous composition of concrete services out of a set of desirable or potentially available/matching constituent services. Similar to the scheduling phase the system may produce alternative construction schemes (e.g. varying in quality or price) from which the application developer can select.
- **Execution**: the service composition system prepares the constructed composed services for execution. This phase maps the resulting specification to a standard executable web service orchestration language (e.g. BPEL4WS).

**Information Model for Service Composition**

In this work, a model-driven approach [5] has been developed to facilitate the development and management of dynamic service compositions. The Information Model (IM) is an abstract meta-model that represents the building blocks of all possible service compositions. It models, using UML, the components required for a given composition and how they are related to each other. Relationships in the IM indicate how a composition is constructed. The required information is modelled as classes containing special purpose attributes so that the required information for service composition can be captured and described.

**Service Composition Methodology**

The dynamic service composition development and management is an automated process governed by rules and administrated by rule engines. Business rules can be used in the context of service composition to determine how the composition should be structured and scheduled, how the services and their providers should be selected, and how run time service binding should be conducted. SOBI project utilises the Object Constraint Language (OCL) to express the business rules that govern and steer the process of service composition.

**Service Component**

Aim of a service component is to raise the level of abstraction in web services by modularising synthesized service functionality and by facilitating web service reuse, extension, specialization and service inheritance. Service components represent modularized service-based applications that package and wire together service interfaces with associated business logic into a single cohesive conceptual module [4]. These modules can be extended, specialized, parameterised, customized, and generally inherited, to assist in the creation of new applications².

² Service components have a recursive nature in that they can be composed of published web services while in turn they are also considered to be themselves web services (albeit complex in nature). Service components package together a number of related service messages and functionality, provided by diverse service providers, into a self-contained software module (called the service component class). This module exposes a well-defined interface and contains the business (or composition) logic.
Service Com: a SOBI prototype
To support the service component mechanism a Service Composition Specification Language (SCSL) has been developed.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- The composition description language (SCSL in the prototype serviceCom) must support the specification of an abstract composite service in such a way it can be progressively made concrete following each step of the lifecycle. For this reason it must support:
  1. description of activity, binding, condition and composition type constructs used (among others) to define a web service composition
  2. description of different forms of activity choreographies; SCSL utilizes the composition type construct for this aim. Supported types include If, IfElse, ParaWithSync, ParaAlt, SeqAlt, SeqNoInt, SeqWithInt and WhileDo patterns.
- The framework must manage the entire life-cycle of service compositions (described by mean of SCSL), ranging from abstract service component definition, scheduling, and construction to execution.

Non Functional:
- Use of WSDL for the description of the web services.
- SOAP is used as protocol for message exchange between web services.

What is the novelty described in this document?
ServiceCom covers the complete life cycle of service composition, including the description, planning, building and executing of compositions. At the time it was developed, there was no other tools that provide such an integrated approach to service composition.

Besides this, other web service composition solutions, e.g. BPEL4WS, were either not flexible or too complicated as they lacked proper support for modularity and reusability. Being ServiceCom based upon the service component mechanism, it promotes reusability and specialization in the specification of compositions.

---

3 Activity constructs represent constituent services through the use of name and description characteristics. To bind activities to particular web service providers binding constructs can be attached to activity constructs. These bindings identify an operation on a port of a service in the WSDL interface of the provider. Alternatively, they may provide search criteria to enable the locating of appropriate providers during runtime. Condition constructs may be used in case of conditional compositions to govern the control flow within the service component.
Furthermore, in the building of service compositions reuse of class files is supported, thus shortening the building process increasing its efficiency.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

This work advocates a “phases based” approach to service composition development, which could be applied to the Service Composition at SLA@SOI; besides a mechanism of dynamic binding is supported by mean of “binding constructs” which can provide search criteria to enable locating of appropriate providers during runtime.

Anyway the work seems to be more oriented to propose a theoretical approach more than specify a concrete solution since the prototype serviceCom was not further developed.

Besides no aspect about agreement matter is taken into account and the choice of re-proposing, from scratch, a service composition description language (SCSL) isn’t completely sharable compared with approaches based on extensions of standard languages (e.g. see SeCSE aproach) since these last ones take advantage from the already available constructs (e.g BPEL includes workflow primitives to represent most of the workflow patterns, consequently, from this point of view, the language offers already a good expressive power.)

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

ServiceCom, a Java-based tool that implements the Service Component concept, supporting reusable web service composition specification, combination and execution was developed at Tilburg University.

**References:**


### 3.26 OWL-S (Ontology Web Language for Services)

**Keywords:**

...
Abstract / Summary:

The OWL Ontology Web Language for Services (OWL-S) [1][5], formerly DAML-S, is a markup language to describe the properties and capabilities of Web Services in such a way that the descriptions can be interpreted by a computer system in an automated manner. OWL-S allows applications to automatically discover, compose, and invoke services in a dynamic services-oriented environment.

The information provided by an OWL-S description includes:

- ontological description of the inputs required by the service
- outputs that the service provides
- preconditions and postconditions of each invocation

This set of features is known as IOPE (Input, Output, Precondition).

OWL-S is described by means of three elements: the service profile for advertising and discovering services; the process model, which gives a detailed description of a service's operation; and the grounding, which provides details on how to interoperate with a service, via messages.

Figure 2: Top level of the service ontology

The class Service provides an organizational point of reference for a declared Web service; one instance of Service will exist for each distinct published service. The properties presents, describedBy, and supports are properties of Service. The classes ServiceProfile, ServiceModel, and ServiceGrounding are the respective ranges of those properties. Each instance of Service will present a ServiceProfile description (what the system does), be describedBy a ServiceModel description (how to use the service), and support a ServiceGrounding description (how to access/invoke a service).

The Service Profile provides a way to describe the services offered by the providers, and the services needed by the requesters. An OWL-S Profile describes a service as a function of three basic types of information: what organization
provides the service, what function the service computes, and a host of features that specify characteristics of the service.

**Described requirements**
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

**Functional:**
- OWL-S must support automatic Web service discovery.
  
  With OWL-S markup of services, the information necessary for Web service discovery could be specified as computer-interpretable semantic markup at the service Web sites, and a service registry or ontology-enhanced search engine could be used to locate the services automatically. Alternatively, a server could proactively advertise itself in OWL-S with a service registry, also called middle agent, so that requesters can find it when they query the registry. Thus, OWL-S enables declarative advertisements of service properties and capabilities that can be used for automatic service discovery.

- OWL-S must support automatic Web Service invocation.
  
  OWL-S markup of Web services provides a declarative, computer-interpretable API that includes the semantics of the arguments to be specified when executing these calls, and the semantics of that is returned in messages when the services succeed or fail. A software agent should be able to interpret this markup to understand what input is necessary to invoke the service, and what information will be returned. OWL-S, in conjunction with domain ontologies specified in OWL, provides standard means of specifying declaratively APIs for Web services that enable this kind of automated Web service execution.

- OWL-S must support automatic Web Service composition and interoperation.
  
  With OWL-S markup of Web services, the information necessary to select and compose services will be encoded at the service Web sites. Software can be written to manipulate these representations, together with a specification of the objectives of the task, to achieve the task automatically. To support this, OWL-S provides declarative specifications of the prerequisites and consequences of application of individual services, and a language for describing service compositions and data flow interactions.

**Non Functional:**
- Use of the Web Services Description Language (WSDL) as a grounding mechanism. The OWL-S' concept of grounding is generally consistent with WSDL's concept of binding.

**What is the novelty described in this document?**

OWL-S (previously DAML-S) is the first well-researched Web Services Ontology, and has numerous users from industry and academy. OWL-S provides one important foundation for the efforts of the Semantics Web Services Language (SWSL) committee of the Semantic Web Services Initiative (SWSI). SWSI is a
collaborative international effort towards the development of Semantic Web Services technology.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

This work is directly applicable to SLA@SOI to service composition development. Semantic specifications of Web services can support greater automation of service selection and invocation, automated translation of message content between heterogeneous interoperaing services, automated or semi-automated approaches to service composition, and more comprehensive approaches to service monitoring and recovery from failure.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

The OWL-S 1.2 Release can be freely downloaded from [3]. A set of tools [4] has been developed related to OWL-S. For example, an OWL-S Matcher (an algorithm that outputs different degrees of matching for individual elements of OWL-S descriptions), a Semantic Web Service Composer or a WDSL2DAML-S Converter.

References:
[9] DAML Services- Tools

3.27 MoSCoE: (Modeling Web Service Composition and Execution)

Abstract / Summary:
MoSCoE [1-4] is a new incremental approach to service composition, based on the three steps of abstraction, composition and refinement.

Abstraction refers to the high-level description of the service desired (goal) by the user, which drives the identification of an appropriate composition strategy. In the event that such a composition is not realizable, MoSCoE guides the user through successive refinements of the specification towards a realizable goal service that meets the user requirements.
MoSCoE develops a new framework for (semi)-automatically realizing new services from pre-existing ones. The user provides a high-level specification of the desired service $G$ using UML state machines, a visual paradigm for representing dynamics of software systems. They provide the sequence of functions and relationships required to attain the goal service. MoSCoE translates this goal specification into a Finite State Automata (FSA), $A_G$, which reveals the exact sequence in which $G$ evolves. In addition, state machines in MoSCoE are semantically annotated by the client using appropriate domain ontologies from a repository. This is achieved by importing OWL ontologies into a UML model. MoSCoE assumes that these ontologies (and mappings between them) are specified by a domain expert using existing tools such as INDUS. The user also provides non-functional requirements such as performance and reliability criteria that need to be satisfied by the composite service.

The service providers in MoSCoE publish their component services by providing OWL-S and WSDL specifications. In particular, given $n$ component services, $C_1, \ldots, C_n$, their OWL-S process models are translated by MoSCoE into corresponding FSAs, $A_1, \ldots, A_n$.

The framework consists of two main modules: composition management module and execution management module.

The composition module: consists in finding a strategy $S$ that defines the sequence in which component service FSAs should be composed. However, if such a composition cannot be realized, MoCSoE identifies the cause(s) for the failure of composition. This information can be used by the user to minimally reformulate the goal to reduce the ‘gap’ between the desired functionality. The process can be iterated until a feasible composition is realized or the user decides to abort. The approach ensures that (i) if a composition is produced then in fact realizes the user-specified goal functionality; and (ii) the algorithm is guaranteed to find a composition that meets the user needs as captured in the goal specifications (whenever such a composition exists).

The execution module: considers non-functional requirements (e.g., performance, cost) of the goal (provided by the user) and analyzes each composition strategy. It selects a strategy that meets all the non-functional requirements of the goal, generates executable BPEL4WS code, and invokes the MoSCoE execution engine. This engine is also responsible for monitoring the execution, recording violation of any requirement of the goal service at runtime. In the event a violation occurs, the engine tries to select an alternate composition strategy.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
The framework must implement a model-driven iterative refinement and incremental approach to Web service composition.

The framework must use formal verification methods to guarantee the soundness and completeness (with certain restrictions) of the composition process.

Available services and user requirements change over time. Then, the framework must support rapid re-design and re-deployment of services, through appropriate reuse of parts of previously assembled services or incorporation of new components.

**Non Functional:**
- Composed services are described using UML state machines.
- Component services are published using OWL-S and WSDL descriptions.
- The framework must be used associating semantics to Web services using ontologies and techniques for specifying mappings between ontologies.

**What is the novelty described in this document?**

In respect to other approaches MoSCoE introduces some element of novelty briefly described below:

- In so many other approaches, for specifying the functional requirements of a composite service, the user is responsible for understanding and using service specification languages like OWL-S and BPEL4WS, or even complicated labeled transition systems which requires more detailed understanding about the behavior of the desired service. This makes the process of service composition tedious and error-prone. MoSCoE tries to fill this gap proposing an approach which allow specifying composition requirements using high-level language (UML based) which is intuitive and easy to understand.

- The existing approaches provide “single-step request-response” paradigm for Web service composition. In other words, a user submits a goal specification to a composition analyzer, which tries to find an appropriate strategy for realizing the composite service. In the event, such a composition is unrealizable, the whole process fails. As opposed to this, MoSCoE proposes to offer provisioning for iteratively refining the goal specification in an intuitive way, to build composite services.

- Individual Web services needed for realizing a desired functionality are often developed by autonomous groups or organizations. Consequently, semantic gaps, arising from different choices of vocabulary for specifying the functionality of the services, are inevitable. This framework offers support for assembling complex Web services from independently developed component services providing support for bridging such semantic gaps.

- Available services as well as user requirements change over time. Thus, environments for service composition need to support rapid re-design and re-deployment of services, through appropriate reuse of parts of previously assembled services or incorporation of new components.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

This work offers, without doubt, a lot of interesting aspects; anyway they aren’t directly applicable to SLA@SOI for the reasons below:
- The approach is “goal based” to support the maximum degree of dynamicity while in SLA@SOI we are oriented to a procedural approach (also known as template based) since the processes, we are interested in, are characterized by interactions more or less complex but static.

- The focus in our project is on the SLA awareness of the dynamic discovery and composition more than on allowing to specify composition requirements using high languages which are intuitive and easy.

- Not kind of negotiation support found: the refinement step should encompass also negotiation activity to consider the approach SLA aware.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

To our knowledge, there is no implementation of the MoSCoE architecture.

References:


3.28 MoDe4SLA

Keywords:
SLA Violation, Monitoring, Petri Nets

Abstract / Summary:

In [1], Bodenstaff et al. propose an approach called MoDe4SLA to analyse the cause of an SLA-violation of composed services. They suggest bilateral monitoring of the service offered and the services used. A dependency model specifies which services are called and how they are used. In the context of response time analysis, the authors use Coloured Petri Nets for this purpose. A cost is assigned to each service to identify its influence in a composed setting. Cost analyses are performed on the basis of the dependency model. Furthermore, monitoring and analysis of event logs allow the extraction of information about the degree of SLA fulfilment and about SLA violations. The results are visualised in the dependency model and help service providers to optimise their service offers.

What is the novelty described in this document?
The novelty of this paper lies in the integrated approach of SLAs analysis during development phase and monitoring of these dependencies using event logs during runtime.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

(How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it? What the needed improvement?)

The concept of root cause analysis for SLA-violations of composed services can be applied and extended in later stages of SLA@SOI to support SLA management and re-negotiation.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

No implementation is available online.

References:


3.29 Service Centric System Engineering
European Project – FP6

SeCSE approach to dynamic service discovery & composition-

Keywords:

Service Discovery, Service Composition

Abstract / Summary:

SeCSE propose an approach which covers both development and execution levels: first defining a methodology that allows designing dynamic compositions by describing, at design time, “abstract service compositions” using an extended process description language and second, by providing a runtime platform flexible enough to support and enable the autonomic and dynamic behavior of service compositions.

Described requirements

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

---

4 i.e. Each “task” of the process is not yet bound to a specific component service
Functional:
[The description language requirements]

1. Must give support for modeling of a service interaction.

The service interaction view does not focus on a particular service, but rather it outlines the interactions among services from a global and external viewpoint. A role should be associated with one or more resources capable of accomplishing it and this kind of correlation role-resource(s) should be more flexible with respect to the way exploited by current service composition languages.

Possible form of association role-resource(s) are:
- A role could be associated with exactly one service; in this case, the requirements to satisfy that role are directly achieved by a resource instance (i.e., a specific service).
- A role could be associated with a syntactical interface that represents a resource type and not directly a resource instance, i.e., without specifying the precise service conform to it (e.g., a syntactical interface could be described in WSDL). The meaning is that the role can be associated with any service that is closed to the syntactical interface.
- A role could be associated with a SeCSE service specification, rather than a precise syntactical interface. This kind of correlation would increase the degree of flexibility offered by languages such as BPEL or WS-CDL, where the role representation is strongly dependent on the WSDL expressive power. A SeCSE service specification expresses the concept of resource type at a higher-level of abstraction with respect to a simple WSDL interface.

Based on what discussed above, we argue that the language supporting service composition must allow the definition of the interaction models from a global viewpoint among the roles defined above.

2. Must give support for the modelling of a service process view (e.g. orchestration)

A service process view is the behavior that the service performs internally to realize the function it provides, i.e., the private flow of the business process that implements the service (Dijkman and Dumas, 2004).

---

5 For instance:
- it identifies all the services.
- It identifies the global role of each service. Each service could be both simple and composed. It identifies the interactions between the participant services where an interaction could involve more than two services, e.g., the one-to-many send service interaction pattern.

6 A SeCSE service specification could individuate a class of services that are syntactically different (i.e., different syntactical interfaces) but semantically equivalent. The role could be covered by any service that is compliant with the aforementioned service specification.
The description language must allow a service integrator to define both internal tasks\(^7\) and communication tasks of the process.

3. **Must give support for the creation of an abstract service composition**

An abstract service composition assumes the form of a process that performs the final business’ objective by combining different functionalities represented by tasks. An abstract service composition cannot be executed till the binding between tasks representing the functionalities and the service that performs the functionalities is not realized. The mapping of a task onto a service can happen at design-time or can be deferred until run-time.

The separation between the abstract service composition and its concrete realisation will improve the reusability of the abstract definition of the composition and also the flexibility of implementation (Yushi et. al., 2004).

These considerations lead to the following requirements for the description language of the composition: the service integrator should be able to define a flow of tasks defined in terms of abstract input and output parameters, i.e., the input and output parameters that appears in the process definition\(^8\).

4. **Must give support for different realization approaches**

As said above the composition language should allow a service integrator to define abstract service composition.

In order to obtain a really executable business process (i.e., an orchestration) representing the workflow-based service composition, the service integrator should be able to specify the way to realise the various tasks included into the abstract service composition. The term realisation has the meaning of establishing a bond between a task and a service capable of performing the functionality (i.e., the operation) the task represents. A service integrator could express, for each communication task that requests a service to perform the functionality:

\(^7\) For instance, considering a service-centric-system with two services S1, S2. S1 and S2 collaborate in a peer-to-peer way. The collaboration is described by the service interaction viewpoint. Moreover, S1 is realized defining a process. The viewpoint could focus on the internal representation of service S1 (i.e., the private process of S1) in terms of:
- the “internal tasks” of the private process, i.e., tasks that do not appear to the external partner S2;
- the “communication tasks” that allow the communication between S1 and the partner S2 (send communication tasks) or along the opposite direction (receive communication tasks). These tasks make possible the interactions that the service interaction view involving S1 and S2 defines.

\(^8\) The term binding refers to the creation of a bond between a communication task of that process and one (a combination of) service(s) capable of satisfying that task, i.e., a service capable of receiving the message (or invocation) the communication task send to it, in order to perform the requested functionality. Moreover, a task can be defined using input and output parameters of the functionality to be performed. It’s important to note that the format of input and output parameters of the task could be different from the actual format of input and output parameters of the service performing the functionality so that there is the need of a *mediation* to solve the problem of syntactical and semantic heterogeneity between source data.
• Conditions, i.e., constraints, under which the realisation of the communication task has to occur i.e., the task can be bound to a discovered service when a given condition is true or an event occurs.
• The functionality the communication task wants to request (send) to the external service and the role the service has to play in order to perform the functionality.
• Other realisation constraints, typically based on QoS preferences or that depend on users’ profile, which should be useful in order to choose the best service to be bound to.

This way, the service integrator could express ‘realization preferences’ that differ from task-to-task on the basis of the overall business goal of the process and other criteria such as performance.

[The binding dynamism degree requirements]

1. **Must give support to pre-execution binding**
   That is to say just before execution and based on global QoS optimization criteria.

2. **Must give support to run-time binding**
   The composed service is progressively bound during each execution, according to the functional/QoS characteristics of the service component to select.

**Non Functional:**
- The service composition language:
  A. is an extension of ws-bpel (because it is a consolidated standard established by Oasis).
  B. annotations are based on XML schemas to support easy integrability with ws-bpel.

**What is the novelty described in this document?**

With reference to the functional requirements described in the section above a comparison with ws-bpel language is reported below.

**Support for the modelling of a service interaction view**

BPEL, does not permit to define an interaction model from a global viewpoint: this language takes into account only the viewpoint of a single participant, i.e., the process being described.

**support for the modelling of an orchestration**

BPEL includes workflow primitives to represent most of the workflow patterns. Consequently, from this point of view, the language offers a good expressive power.
The term abstract process in BPEL, it is used to indicate the behavioral interface of a process-based composition.

The term abstract service composition that appears in the requirements section has a different meaning: an abstract service composition of this type will become deployable and executable when the tasks the process collects will be bound to a service, following the before mentioned realization approaches.

BPEL does not support a notion of abstract process according to the second meaning of the term.

In BPEL each communication task (i.e., receive, invoke activity) is statically bound to a WSDL operation and the portType offering the operation. The only consented form of dynamic binding is related to the access point, e.g., the URL to the service that is conform with the WSDL interface can be delayed till the execution of the communication task by a previous assignment activity.

Considering the invoke communication task, it must declare the exact name of a WSDL operation. Within a communication task is not possible to refer to a conceptual operation or express a declarative goal that will be achieved calling one or a combination of external services.

At last, the input and output parameters of the communication task are WSDL messages completely conforming to the input and output parameters of the service to invoke: a BPEL engine cannot provide adaptation mechanisms to solve possible heterogeneity of this typology.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

This work is direct relevant to SLA@SOI since it proposes a complete solution to the dynamic service discovery and composition which is also an objective of our project. Besides it’s of particular interest since the solution is based on a “procedural approach” (also known in literature as “template based approach”) which is a choice applicable also in SLA@SOI since in all foreseen use cases the process of the composite services are well known, that is, the interactions are more or less complex but static.
On the other side the language, the work proposes, is not directly applicable since in SLA@SOI the description of the queried service, for each abstract task, is already maintained inside a SLA template. In fact the SLA model\(^9\) conceives SLA templates as an enriched kind of service description which produces the advantage that, while primarily enabling ‘SLA aware’ search, it none-the-less remains consistent with more traditional views of service discovery based on purely functional Service properties.

So we need only a mechanism to refer a SLA template from the process description without extending the description language.

Last but not least the language proposed in the work should be extended since in this version it does not support negotiation rules which can be triggered by an event asserting that the SLA template, for the requested service, owns negotiable parameters.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

There is an implementation of the SecSE development environment available as an open source reference implementation (BSD license) at the web address http://sourceforge.net/projects/secse/ in the Sourceforge.NET web site.

**References:**


**3.30 SCENE: To offer a language for composition design that extends the standard BPEL language with rules used to guide the execution of binding, re-binding, negotiation, and self-reconfiguration operations.**

**Keywords:**

BPEL, Dynamic Business Processes

---

\(^9\) See “A5.1a SLA Model & Template Definition.doc” at: [https://svn.fzi.de/svn/sla_at_soi/project/Workpackages/A5-SLAFoundations/trunk/Deliverables/Internal/](https://svn.fzi.de/svn/sla_at_soi/project/Workpackages/A5-SLAFoundations/trunk/Deliverables/Internal/)
Abstract / Summary:

SCENE [1] offers a language for composition design that extends the standard BPEL language with rules used to guide the execution of binding and re-binding self-reconfiguration operations.

A SCENE composition is enacted by a runtime platform composed by a BPEL engine executing the composition logic, an open source rule engine, Drools, responsible for running the rules associated to the composition, WS-Binder[3] that is in charge of executing dynamic binding and re-binding, and by a Negotiation component that can be used to automatically negotiate SLAs with component services when needed [2].

Described requirements

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
1. the language used to develop the composition should support the designer in defining the constraints and conditions that regulate selection, negotiation, and replanning actions at runtime.
2. the runtime platform should be flexible enough to support the selection of alternative services, the negotiation of their service level agreements, and the partial replanning of a composition.

Non Functional:
- The service composition language:
  A. is an extension of ws-bpel (because it is a consolidated standard established by Oasis).
  B. annotations are based on XML schemas to support easy integrability with ws-bpel.

What is the novelty described in this document?

The challenge nowadays is to make such compositions able to dynamically reconfigure themselves in order to address the cases when the component services do not behave as expected and when the execution context changes. The authors argue that the problem has to be tackled at two levels: on the one side, the runtime platform should be flexible enough to support the selection of alternative services, the negotiation of their service level agreements, and the partial replanning of a composition.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

See considerations made for the SeCSE approach to which this work is very near in the intentions and approach (see sota_A33_Secse.doc).

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
No specific implementation available; In order to demonstrate that the approach enables the development of context-aware service composition, focus was addressed on a real case study in the automotive and telecommunication domains, where the evaluation results may be useful to understand, supervise and improve the approach to support dynamic compositions.

References:


3.31 EVEREST (EVEnt REaSoning Toolkit)

Keywords:
Event-based Monitoring

Abstract / Summary:
Everest is a solution for run-time monitoring of a software system based on events.

The properties to be monitored are expressed in Event Calculus (EC, a temporal first order logic). A core monitoring engine is able to check the satisfaction of rules expressed in EC against events captured by suitable (application-specific) event captors.

The core monitoring engine has been used for the monitoring of service-based systems, i.e. BPEL processes, distributed systems with mobile peers, and systems for ambient intelligence.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

A distinction can be made on the basis of the properties that can be monitored by the Everest core monitoring engine.

Functional properties
Mainly defined for service-base systems, i.e. correct order of service invocations in a business process. Other functional properties to be monitored can be specific
to the considered use case (specific EC predicates should be defined to express rules).

**Non Functional properties:**
Response time, availability, throughput for service based systems. An extension of the core monitoring engine exists for expressing and monitoring security properties (authorization, authentication, confidentiality)

**What is the novelty described in this document?**
The novelty of the Everest approach lies in its applicability to different contexts. Generic software systems can be

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
The Everest core monitoring engine has been developed at City University London and will be used as one of the core monitoring engine within the SLA@SOI framework.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**
There is a prototype implementation of the core monitoring engine. Prototypes are available also for all the extensions to the monitoring engine, e.g. pattern-based security related monitoring rules, distributed monitoring for ambient intelligence.

The prototypes are not licensed.

**References:**
3.32 **Dynamo (and related work)**

**Keywords:**
Process Monitoring, BPEL

**Abstract / Summary:**
Dynamo represents a solution for assumption-based monitoring of service-based systems (BPEL processes). In the first version of the prototype, the monitor was able to check the BPEL process execution against a set of assumptions describing functional pre- and post-conditions on the execution of activities in the BPEL process. Such assumptions were expressed as annotations, made at design-time, of the BPEL process using the language WS-CoL. This approach has been extended in several ways: the language for expressing properties to be monitored has been extended to accommodate also QoS (time-related properties); aspect oriented programming techniques have been adopted to separate the code for executing the BPEL process from the monitoring-related executable code.

Prototypes have been implemented as extensions of ActiveBPEL engine.

**What is the novelty described in this document?**
In contrast to event-based (non-intrusive) monitoring, assumption-based (intrusive) monitoring does not need external components for performing monitoring, since monitoring is executed directly by the (extended) BPEL engine.

However, assumption-based may also lead to the degradation of the performance of the BPEL engine, if monitoring activities can not be tailored on the basis of the current service load.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
With a proper technical adaptation, this approach can be used within SLA@SOI, since it has been developed specifically for service-based systems.

Part of the BPEL engine instrumentation implemented in Dynamo, i.e. event capturing for BPEL processes, will be re-used within SLA@SOI for capturing events at the software service layer of execution.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**
There is a prototype implementation of the core monitoring engine. Prototypes are available also for all the extensions to the monitoring engine, e.g. time-related properties, monitoring of conversational Web service specified with WS-CDL.

The prototypes are not licensed.
References:


3.33 Work by Ardissono et. al on “Fault Tolerant Web Service Orchestration by Means of Diagnosis”

Keywords:
Failure Monitoring, Failure Management

Abstract / Summary:
The authors present a monitoring approach for the purpose of diagnosis of failures in service-based systems. In the context of composite service-based systems, the goal of the diagnosis is “to identify the service responsible for the problem, the faulty activities, and the other services involved in failure”. The presented approach deals with a particular case of the system diagnosis, namely consistency-based diagnosis, where the goal is to assign certain behavioral model to the components and observe its consistency with respect to the real execution. Furthermore, the authors present an approach to the diagnosis-aware fault handling, where the global hypothesis on the occurred problem drives the recovery actions within the application components.

The proposed solution relies on a distributed architecture, where each component service is associated with the corresponding local diagnoser, which interacts with the service through a special diagnosis interface. Such diagnoser collects the local information regarding the actions executed by its service and the messages it interacts with the other services, and derives local hypothesis about the problem and the other involved services. This hypothesis is then transferred to the global diagnoser component that is in charge of requesting other local diagnosers in order to collect the most complete information regarding the occurred problem.

What is the novelty described in this document?
The novelty of the approach lies in the focus on faults diagnosis in service-based systems execution and in the implementation of a fully distributed architecture for performing diagnosis.
How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

Though diagnosis of systems faults is not one of the main focus of SLA@SOI monitoring framework, runtime predictive monitoring may benefit from information from the diagnosis of faults in previous executions of service/business process.

From the architectural point of view, the distributed architecture involving several local diagnosers and a global diagnoser appears to be similar to the architecture envisaged for the SLA@SOI monitoring, with several local monitors coordinated by a global monitor.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

A proof of concept implementation of the approach exists, as it is used for evaluation issues.

References:


3.34 Work by Lazovik et.al. on “Business process management”

Keywords:

Business Process Management

Abstract / Summary:

In this approach the authors propose an architecture, where the planning-based adaptation of the business process is interleaved with the execution and monitoring of the process and the corresponding assertions. The adaptation requests are specified in a XSRL query language that defined functional constraints and preferences of the user. The assertions are specified in the assertion language XSAL, which allows for defining the behavioral policies on the execution of process activities in the domain.

Using these specifications and the process model, the framework tries to exploit various providers, in order to better satisfy the query, taking into account the domain assertions. While executing the adapted process, the framework monitors the relevant events and if the violation of the plan or assertion is detected, tries to dynamically modify the plan taking into account new situation and assertions.

What is the novelty described in this document?
The novelty of the approach is twofold. First, it gives an important role to the user in the specification of properties to be monitored, by providing a framework for expressing rules for adaptation and monitoring. Second, this work mixes the intrusive and non-intrusive approaches to runtime monitoring of service-based systems. Specifically, activities for planning monitoring are interleaved with the BPEL code for the execution of the service-based business process. At runtime, events are collected to detect the violation of the monitoring properties specified by the planner.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

This work has been developed in the context of service-based systems and, therefore, could be theoretically applied in the context of SLA@SOI. Extensions are required to deal with hierarchical and multi-layer specification of SLAs.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype,alpha/beta/RC/stable).**

A proof of concept implementation of the approach exists, as it is used for evaluation issues.

**References:**


### 3.35 Colombo (IBM)

**Keywords:**

SOA Development, SOA Deployment, SOA Execution Platform

**Abstract / Summary:**

Colombo is a platform for developing, deploying, and executing service-oriented applications and system that incorporates the tools and facilities for checking, monitoring, and enforcing service requirements expressed in WS-Policy notations. WS-Policy notations define the quality-of-service assertions that can be attached to a particular service, operation, or a message type. The concrete assertions are defined in a certain domain-specific language, e.g., WS-Transactions or WS-Security that define the properties of the transaction protocols and security characteristics respectively.

Apart from checking the compliance of policies at deployment-time, it is necessary to verify them at run-time, when, e.g., service invocations calls/bindings take place or messages are sent/received.

The Colombo platform comes with the module that manages the policy assertions. Besides the evaluation of the assertions attached to particular service-related
entity, the framework provides means for policy enforcement, e.g., it may approve the delivery of a message, reject the delivery, or defer further processing.

**What is the novelty described in this document?**

The main novelty of this approach is represented by policy enforcement, i.e., the ability of framework to take control actions, such as message dropping, in case monitoring properties are violated at runtime.

The limitation of the framework is twofold. First, Colombo is developed to deal with service-based system, and does not consider infrastructure related services. Second, monitoring properties can be expressed only with domain-specific languages belonging to the WS-* star, which may be not enough expressive for complex behavioural or QoS-related monitored properties.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

SLA@SOI should keep into consideration this approach, especially for what concerns the enforcement of control actions.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

Colombo programming can be implemented in multiple languages; Java and BPEL realizations of this programming model are available as a research prototype, which is not licensed.

**References:**


**3.36 Work by Momm et.al. on “Model-driven Development of Monitored Processes”**

**Keywords:**

Automatic Business Process Development, Monitoring, BPEL, BPMN

**Abstract / Summary:**

The work discusses the problem of how to develop automated SOA-based business processes with integrated monitoring information for process controlling. Automated BPEL-based business processes are often developed in a top-down manner, starting with a visual notation of the process (e.g. in BPMN) and then translating the visual model into an executable BPEL process model. If the BPEL process is to be monitored, then also process metrics have to be specified during process development.

The presented solution utilizes a model-driven approach to developing monitored business processes. The authors have created a metamodel which allows
modelling of process performance metrics (PPIs) based on BPMN process elements. The BPMN process model with the corresponding PPI model is transformed to a BPEL process model which contains additional activities for publishing events needed for the calculation of the PPIs. These events are sent to an external monitoring tool by invoking its Web service interface. For measuring the duration of the activity, for example, two additional BPEL invoke activities would be inserted, before and after the activity, respectively. These activities would invoke corresponding operations on the monitoring tool.

The benefit of the approach is that events needed for monitoring are automatically determined and corresponding activities for event publishing are automatically generated. The authors demonstrate the approach based on a case study in the context of the management of examinations.

What is the novelty described in this document?
The main novelty of the model is represented by the application of model-driven techniques to the modelling of monitorable business processes.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The model-driven approach used in this work is extremely relevant to SLA@SOI, as it may be used to model monitoring information at the different layers of the SLA@SOI SLA management framework.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
A proof of concept implementation of the approach exists, as it is used for evaluation issues

References:

3.37 Work by Roth et.al on “Probing and Monitoring of WSBPEL Processes”

Keywords:
Business Processes, Event Tracing, BPEL

Abstract / Summary:
The work deals with the problem on how to extract events from a BPEL process in order to enable auditing in an interoperable way. The BPEL specification does not specify how the execution of the BPEL process should be logged in an audit trail. Each BPEL engine vendor implements a different event model and auditing mechanism. Thus, a monitoring tool would have to provide adapters for each BPEL engine it wants to support.
The presented solution extends a BPEL process model definition with special auditing activities which log state changes to an external monitoring web service. The extended BPEL process does not use any proprietary elements and is BPEL standard compliant. Therefore, the extended BPEL process can run on any process engine and send events to the monitoring tool. First, the authors introduce five different strategies for auditing BPEL processes: (i) instrumentation of web service requests of the BPEL process on protocol level and (ii) on application server level, (iii) utilizing the auditing service of a process engine used for enacting the BPEL process, (iv) using probes in the operational systems that track state changes of the business process, or (v) including the auditing mechanism as a partner within the BPEL process. They employ the fifth strategy and show how to transform a BPEL model into an auditable model which can be used for process monitoring purposes. For every audited activity, a new scope is created which hosts and executes all the necessary steps for pre- and postauditing. For the monitoring service which is invoked by the extended BPEL process, an interface is presented. Finally, the authors propose some extensions to the BPEL specification for supporting their approach.

The benefit of the approach is that the extended BPEL process does not use any proprietary elements and is BPEL standard compliant. Therefore, the extended BPEL process can run on any process engine and send events to the monitoring tool. The authors have implemented a prototype consisting of a tool which extends a BPEL process adding auditing activities and a monitoring service which is invoked by the BPEL process at process execution time.

**What is the novelty described in this document?**

The novelty of the work is represented by the opportunity to design a BPEL-compliant process from which events required for monitoring can be extracted. On the other hand, however, this represents also a limitation, since it tightly couples monitoring to BPEL-enabled service-based systems.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

This work should be kept into consideration within SLA@SOI, since it deals with the extraction of events for monitoring in a BPEL process. However, the current approach to monitoring in SLA@SOI uses event captors (instrumentation at the service and infrastructure layer), which is already decoupled from the service process execution context (i.e. BPEL process).

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

A proof of concept implementation of the approach exists, as it is used for evaluation issues.

**References:**


Keywords:
Business Process Analysis, Agents

Abstract / Summary:
Jeng et al. (2003) have developed an agent-based architecture for providing continuous, real-time analytics for business processes. The architecture includes an integration layer, an event processing container (EPC), a BI agent layer, and a dashboard layer.

The integration layer extracts events from business applications and workflow systems. These events are provided to the EPC which is a robust, scalable and high-performance event processing environment. It is able to handle a large number of workflow events in near real-time. The incoming process events are transformed on-the-fly into metrics that are stored in the process data store. Furthermore, the EPC also publishes information to the BI Agent Layer for analytical processing. The BI Agent Layer is based on the decision cycle involving the five sub processes: sense, detect, analyze, decide, and effect. The sensing agents retrieve the events and metrics from the EPC or process data store, and provide them to reactive, deliberate, and proactive agent layers, which analyze and respond to situations and exceptions in a business environment based on business policies. Response agents generate action outputs unto the business environment by following the directives delivered from the agent layers. The authors demonstrate the approach and the implementation based on a use case on supply chain management for microelectronics manufacturing.

What is the novelty described in this document?
The novelty of this approach is represented by the development of an agent-based infrastructure for collecting monitoring-related data during service provisioning.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Although this work does not deal explicitly with SLA monitoring, but with the collection of provisioning data (statistical profiles of service utilization etc.), the work should be taken into account within SLA@SOI for what concerns service instrumentation and, more generally, the instrumentation required at different layers of the service landscape to retrieve data relevant for monitoring.
Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

A proof of concept implementation of the approach exists, as it is used for evaluation issues.

References:

3.39 VieDAME (and related work)

Keywords:
Business Processes, Design, BPEL

Abstract / Summary:
The work presents for designing and executing highly dynamic BPEL-based business processes. In particular, the authors present an extension to a BPEL-compliant engine introducing service adaptation and monitoring.

In case of dynamic service substitutions, the adaptation layer is able to dynamically mediate between different service interfaces. This mechanism is implemented by intercepting and appropriately modifying SOAP messages used for invoking activities in business processes.

From the point of view monitoring, the framework adopts aspect-oriented techniques for retrieving events relevant for monitoring, such as service calls and response.

The proposed framework also involves a component that performs dynamic service substitution of services unavailable at runtime. Dynamic service substitution is based on heuristic algorithms, which account for both functional and non-functional (i.e. QoS) equivalence of services’ specifications.

What is the novelty described in this document?
The main novelty of this work is represented by the integration of several advanced technologies for modern service-based systems, such as service dynamic substitution, adaptation and monitoring.

For what concerns service monitoring, in particular, the aspect-oriented capturing of monitoring events in BPEL-engines does not represent a novel contribution, since it already considered in other approaches reviewed in this state-of-the-art analysis.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
SLA@SOI should take into account the mechanism adopted to capture monitoring-related events at runtime proposed in this approach.
Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

A proof of concept implementation of the approach exists, as it is used for evaluation issues.

References:

3.40 Work by Jurca, R., Binder, W., and Faltings, B. on “Reliable QoS Monitoring Based on Client Feedback”

Keywords:
System Monitoring, QoS Properties

Abstract / Summary:
The work presents an approach for including client feedback in the monitoring of software systems (service-based). The work focuses on the monitoring of QoS properties (e.g. availability and response time) and proposes a QoS monitoring mechanism in which the feedback from clients is taken into account to define penalties on SLA provisioning for service providers. The authors demonstrate that, because of the SLA penalty mechanism, service providers get incentives in fulfilling the QoS properties they have declared in SLAs.

What is the novelty described in this document?
The main novelty of this approach is to consider client-generated information for enhancing service runtime monitoring. Moreover, this approach is also the first work which consider incentives as a driving mechanism to manage SLAs in service-based systems.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
SLA@SOI, especially, in its first year, is focusing mostly on the service provider’s perspective on SLA monitoring. Therefore, the work presented in this paper should be taken account since, by focusing on client-side generated ratings, introduces a new perspective for SLA monitoring.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

A proof of concept implementation of the approach exists, as it is used for evaluation issues.

References:
3.41 Work by W.M.P. Van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and E. Verbeek, on "Conformance checking of Service Behavior"

Keywords:
Service Monitoring, Status Check, Petri Nets

Abstract / Summary:
This work presents an approach for checking and quantifying the actual behaviour of a service with data concerning the actual execution of the service. A model of service behaviour, expressed in the form of Petri Net, is created from the log of several executions of the service. New execution of the same service may be then checked against the created Petri-Net model.

For what concern the quantification of service behaviour compliance and conformance, the work proposes two metrics, namely fitness and appropriateness.

What is the novelty described in this document?
The main novelty of this approach is represented by the definition of metrics for defining the compliance between predicted and actual service execution.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Although this work belongs to the area of process monitoring, rather than monitoring of service-based systems, SLA@SOI should take into account two main aspects of this work: (i) the need for quantifying the compliance of service execution according to pre-specified models of service provisioning and (ii) the mechanism used for extracting relevant data for building service compliance modelling.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
A proof of concept implementation of the approach exists, as it is used for evaluation issues.

References:
3.42 Assumption-based monitoring of service compositions

Keywords:
Service Composition, BPEL, Monitoring

Abstract / Summary:
The approach aims at the checking of the assumptions under which the state-full services are supposed to participate to a specific service composition. The interaction protocol of business services participating to the composition must be specified in Abstract BPEL. The approach gives the possibility to express boolean, statistical and temporal properties holding for a business process instance. Such a property specification relies on Run-Time Monitoring specification Language (RTML), an expressive monitoring language based on past temporal logic. The approach supports also the possibility to aggregate historical information about properties over all the execution instances of a given business process. An architecture which clearly separates the business logic of a web service from the monitoring functionalities has been defined. Translation of the property to the corresponding monitor program (in Java) deployable on the identified architecture is also supported.

Described requirements

Functional:
[2] The approach gives the possibility to express boolean, statistical and temporal properties on a give business process instance.
[3] The approach gives the possibility to aggregate historical information about properties over all the instances of a business process.
[4] The monitor program (in Java) corresponding to the property, based on the monitoring specification and the composition model, is generated using specific automata-theoretic techniques.
[5] The deployed monitor programs run in parallel to the business services it should mimic observing their behavior by intercepting the input/output messages.

Non Functional:
[1] Separation of the business logic of a web service from the monitoring functionalities.
[2] The interaction protocol of business services part of the composition must be specified in Abstract BPEL.
[6] The run-time environment is implemented as an extension to a BPEL engine.

What is the novelty described in this document?
It is possible to express boolean, statistical and temporal behavioral properties holding for a process instance. It is also possible to aggregate historical
information about monitoring properties over all the instances of a given business process.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

The approach is relevant for SLA@SOI project since it leverage the level of the business properties that can be monitored. As far as now the approach is bound with BPEL, it should be generalized in order to support general business services.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

There is a prototype readily available and it is downloadable from http://www.astroproject.org.

**References:**


[3] Astro is a research project in the field of web services and service-oriented applications. http://www.astroproject.org/

### 3.43 Monitoring privacy-agreement compliance

**Keywords:**
Right Management, Monitoring

**Abstract / Summary:**

The approach aims at monitoring at run-time the compliance of the privacy agreement defining the users privacy rights and their possible handling by the service provider. The authors state that the approach is able to handle the usage control management of the private user information going beyond the check of the user credential. The proposed solution, based on the modeling and monitoring of the requirements on the privacy data, mashes-up the privacy agreement model with run-time compliance monitoring technique. An architecture enabling the whole approach is also defined.

Within the proposed approach the privacy requirements, defined in terms of data right (user allowed actions on data) and data obligation (provider required actions on data), are modeled in a tailored version of the WS-Agreement framework handling privacy aspects. From the privacy requirements a set of monitor private units expressed in Linear Temporal Logic are extracted, translated in automata and eventually deployed on the proposed architecture as monitors.

**Described requirements**
**Functional:**

[7] The privacy agreement specification relies on a convenient and clean formalism for expressing the privacy management requirements based on an extended version of the WS-Agreement framework.

[8] The privacy property which require to be monitored are defined using linear temporal logic.

[9] The monitor collects the information about the privacy data use (from the service logs) and evolves consequently updating the status of the observed privacy unit accordingly.

[10] The run-time monitoring approach exploits automated techniques for the extraction and execution of monitor programs.

**Non Functional:**

[3] Separation of the business logic of a web service from the data privacy concern.

[11] An architectural framework enabling the privacy agreement monitoring has been defined.

**What is the novelty described in this document?**

Unlike most existing approaches, the one investigated here focuses on the specification and monitoring of privacy related properties.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

Although the approach seems to be really interesting, it is not really close to the monitoring approaches we foresee within the SLA@SOI project. It could be interesting to consider this approach as a possible extension to SLA@SOI monitoring functionalities in order to support also privacy-related properties.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

The approach still in development and it doesn't seem to have a released environment.

**References:**


**3.44 Performance monitoring for utility computing**

**Keywords:**

SLA Monitoring

**Abstract / Summary:**
The approach aims at the monitoring of service-level agreements formally defined between the service provider and the service consumer. The solution is specifically designed to address the utility computing domain which is characterized by the provisioning of different resources with specific quality properties. The service agreement are managed on the basis of the contract pattern, while the terms contract axiomatization, where the effects of critical events on the contract state and evolution are defined, is based on event calculus. Within this work is proposed an architecture, as long as a proposed reference implementation, supporting both the contract life-cycle managing, analysis and reporting and the presentation of the results of the SLA monitoring.

Described requirements

Functional:
[12] The service agreement are managed on the basis of the contract pattern
[13] The contract axiomatization is based on event calculus

Non Functional:
[14] The authors propose an architecture, as long as a proposed reference implementation

What is the novelty described in this document?
The novelty of this approach is related to the fact that it has been applied and evaluated on a real world deployment scenario: SLA within utility computing.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The approach doesn't seem relevant for SLA@SOI project due to the fact that it doesn't rely on standard languages for the specification of service contracts (e.g. WS-Agreement).

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
This approach has been implemented as a Java-based prototype of a generic EC reasoning component. The implementation includes query interpreters for CTXML and ecXML.

References:
3.45 Cremona

Keywords:
SLA Lifecycle, SLA Management, WS-Agreement

Abstract / Summary:
The paper presents an architecture (named Cremona), as long as the reference implementation, of a framework handling the creation, the management and the monitoring of a service level agreements represented as WS-Agreement documents. Cremona provides a layered model for definition, creation, negotiation, binding, and monitoring of contracts.

Described requirements

Functional:
[15] The clients service level agreement requirements are specified thought WS-Agreement.
[17] The framework monitoring module support the observation, the detection and the prediction of contract violations.
[18] The framework monitoring module come with a management interface allowing the user to track the current situation and to take agreement-level actions.

What is the novelty described in this document?
This framework is one of the few examples of an integrated solution managing all the different aspects of the WS-Agreement document life cycle.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The Cremona framework is relevant for SLA@SOI project since it aims at the management of the WS-Agreement document life-cycle.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
There is a prototype but it is not clear what the current development status is.

References:
3.46 Automated SLA monitoring

Keywords:
SLA Monitoring

Abstract / Summary:
The approach aims at the automated monitoring of the Service Level Agreement. The proposed solution addresses the problem by describing the SLA in a formal language, by enabling the distributed measurement of the relevant SLA data and finally by defining a full flagged monitoring engine. The formal SLA specification relies on an annotation language which gives the possibility to identify the measured parameter, to identify the role (service provider or service consumer) engaged in the measurements, the relevant events that should take place, and the instructions for evaluation of the values. The distributed measurement is achieved providing some instrumentation facilities enabling the distributed collection of the necessary data (e.g. message sniffing and/or log analysis). The monitoring engine coordinates the monitoring actions and manages the monitor life-cycle according to the duration periods specified in the SLA.

Described requirements

Functional:
[19] The approach proposes an annotation language for formal modeling the SLA between the service provider and the service consumer.
[20] The approach is based on the distributed measurement of the relevant SLA data.
[21] The engine supports distributed monitoring, in case that the measurements must be performed both at the consumer and at the provider side.

Non Functional:
[4] The implementation of the engine comes also with management and audition facilities.

What is the novelty described in this document?
This approach is clear and comprehensive but outdated: the proposed language for the specification of SLA cannot compete against the de facto standard WS-Agreement.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The approach is not interesting for SLA@SOI project (see previous point).

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
It seem that the development has been closed.
3.47 Query-based business process monitoring

Keywords:
Business Process Monitoring, BPEL

Abstract / Summary:
The approach aims at providing a business process monitoring framework enabling the design of complex monitoring tasks that deal with both events and flow. In the BMon approach the monitor is defined through a query language allowing the description of the execution pattern. The BMon query language - expressed with nested directed acyclic graphs - describes sequential, parallel, repetition or alternatives execution of activities and allows the indication of the granularity levels to be employed for different components of a monitoring task. Bmon comes with a convenient graphical tool allowing the specification of the monitor in the query language. The monitor it is translated in BPEL and it is deployed on the same engine as the monitored business process. A reporting facility enabling the violation notification has been also implemented.

Described requirements

Functional:
[22] Monitoring query language enabling the specification on complex monitoring tasks that deal with both events and flow.
[23] The framework comes with a convenient graphical tool allowing the specification of the monitor in the query language.
[24] The framework comes with a convenient reporting facility enabling the solicit monitor property violation notification.
[25] The monitor is eventually emitted in BPEL language and deployed on the same engine as the monitored business process runs.

Non Functional:
[5] The generated monitors are extremely efficient and incurs only very minimal overhead.

What is the novelty described in this document?
The interesting feature of this approach is the possibility to specify complex behavioural properties on the execution of the BPEL process.
How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

The approach is relevant for SLA@SOI project since it leverage the level of the business properties that can be monitored. As far as now the approach is bound with BPEL, it should be generalized in order to support general business services.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

A reference implementation for demonstration purpose is available.

References:

3.48  Model-driven development for BPM

Keywords:
Model-Driven Development, IBM Toolset

Abstract / Summary:
The approach aims at the use Model Driven Development methodologies to enable efficient monitor development and deployment. Within the MDD context an application development is driven by the specification of a set of model describing different aspects of the system. The model are via refined in order to fill the gap between business model and the technology platform. The proposed framework defines a model capturing the required KPIs based on incoming events and required notifications, as well as the technical environment consisting in a database storing the KPI values and a dashboard viewer. The framework uses and extends the IBM business observation metamodel and introduces a data warehouse metamodel and other platform-specific and transformational models. The observation model captures the monitoring tool in terms of how metrics are to be computed and which actions to take in certain situations. The data warehouse model deals with storage of metrics, and their visualization in dashboards. Eventually the model are transformed to code deployable on the defined technical environment. The framework has been developed within the IBM toolset context.

Described requirements

Functional:
[26] The KPI are specified in terms of incoming events and required notifications the via UML diagram based on the observation model metamodel
[27] Automated translation from the observation model and data warehouse model in code actual deployable on the technical environment.
What is the novelty described in this document?
The novelty of this approach is the idea of applying Model Driven Development to the monitoring of business metrics.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The approach seems to be relevant for SLA@SOI project. The SLA terms should be modelled in terms of the observable and data warehouse.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
The framework is implemented as a stable release and is shipped within the IBM toolset context.

References:

3.49 Intelligent Business Operation Management (iBOM)

Keywords:
Business Metrics, Monitoring, KPI, Prediction

Abstract / Summary:
This aim of this approach is to provide a business platform (iBOM) that can be used by the analyst not only to define and monitor business metrics, but also supports the analysis of the causes of undesired KPI values, and supports predictive monitoring of future deviations from the specified KPI values.

To support this intelligent analysis of business metrics, the approach combines business activity monitoring with data mining approaches based on decision trees. In order to obtain process informations, the framework requires to define abstract processes which model the steps of the internal process in terms of events which are to be extracted from existing systems. The abstract process model is used as an input to the monitoring engine in order to obtain process informations and display the process status according to the occurring events. In order to enable the business user to specify business metrics, IT engineers are required to specify business metric templates. These templates can be used by
business users, that instantiate them with concrete values, in order to specify the properties to be monitored.

**Described requirements**

**Functional:**
- [28] providing visibility into processes which are not executed by a process engine, but run implicitly across diverse systems
- [29] enabling the business user to define KPIs in an intuitive way
- [30] enabling monitoring of business metrics
- [31] enabling prediction of business metric deviations

**What is the novelty described in this document?**
The approach supports not only the monitoring of KPI values, but also the analysis of the causes of undesired KPI values and predictive monitoring of future KPI values. However, the explanation of the causes of the deviations is not enough mature to be useful to the business user.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
The approach may be relevant to SLA@SOI for what concerns predictable system engineering (in particular for predictive monitoring).

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**
The approach doesn't seem to have a released implementation.

**References:**

**3.50 MAPE-K (Monitor-Analyze-Plan-Execute-Knowledge).**

**Keywords:**
Autonomic Computing, Self-Management

**Abstract / Summary:**
MAPE-K is an autonomic architecture proposed by IBM [1], based on the autonomic element (AE). The AE has to know the environment and how to influence it in order to keep it in optimal conditions, without the need of any external operation.
The autonomic architecture implements an intelligent control loop. For a system component to be self-managing, it must have an automated method to collect the details it needs from the system; to analyze those details to determine if something needs to change; to create a plan, or sequence of actions, that specifies the necessary changes; and to perform those actions. When these functions can be automated, an intelligent control loop is formed.

As shown in the following figure, the AE is composed of four parts that share knowledge. They form the autonomic element architecture MAPE-K (Monitor-Analyze-Plan-Execute-Knowledge):

![Figure 4: IBM Architecture](image)

- The monitor function provides the mechanisms that collect, aggregate, filter and report details (such as metrics and topologies) collected from a managed resource.
- The analyze function provides the mechanisms that correlate and model complex situations (for example, time-series forecasting and queuing models). These mechanisms allow the autonomic manager to learn about the IT environment and help predict future situations.
- The plan function provides the mechanisms that construct the actions needed to achieve goals and objectives. The planning mechanism uses policy information to guide its work.
- The execute function provides the mechanisms that control the execution of a plan with considerations for dynamic updates.
- Knowledge: The autonomic element employs knowledge to interpret the information from the environment and to perform the appropriate actions. It forms a space of understanding among all the blocks. It is defined using semantic technologies.

**Described requirements**

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)
Functional:
- The resulting system must have self-managing capabilities.
- When a system component is first brought into a system, it should configure itself internally.
- It should be able to monitor its internal and external states.
- It should optimize its internal behavior in response to changing external conditions.
- It should heal itself internally if problems occur.
- It should protect itself from external probing, attack, and corruption.
- System constituent components (resources, managers and knowledge sources) must be able to discover each other, identify other components with which to communicate, and coordinate with those other components to achieve their mutual goals.

Non Functional:
- An enterprise service bus is used to connect various autonomic computing building blocks.
- Policies are used to translate high-level directives into specific actions to be taken by resources.

What is the novelty described in this document?
This work presents a high-level architectural blueprint to assist in delivering autonomic computing in phases. The architecture reinforces that self-management uses intelligent control loop implementations to monitor, analyze, plan and execute, leveraging knowledge of the environment.

The architectural approach allows the creation of self-managing resources and to compose them to form self-managing systems. The desired self-management behaviors of self-managing resources—self-configuration, self-healing, self-optimization and self-protection—are recommended best practices in their design.

The IBM MAPE-K architecture is not related to a specific technology. Instead its purpose is to work with existing computing technologies, as well as with new technologies that will emerge in the future.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The IBM architecture offers interesting aspects about autonomic computing that can be directly applied in those subsystems of the SLA@SOI framework that are going to be build following autonomic principles.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
This is an architecture definitions, without an IBM available implementation.

References:
3.51 HAGGLE (An innovative Paradigm for Autonomic Opportunistic Communication)

Keywords:
Autonomic Networking, Mobile Networks

Abstract / Summary:
Haggle is a full Future and Emerging Technologies (FET) Integrated Project funded under the Situated and Autonomic Communication (SAC) program of the Information Society Technologies priority area of the European Union's Framework Programme 6 (FP6). The goal of the SAC program is to promote research in the area of new paradigms for communication/networking systems that can be characterised as situated (i.e. reacting locally on environment and context changes), autonomously controlled, self-organising, radically distributed, technology independent and scale-free.

Four projects integrate the SAC Area: BIONETS (BIOlogically-inspired autonomic NETworks and Services), ANA (Autonomic Network Architectures), HAGGLE (An innovative Paradigm for Autonomic Opportunistic Communication) and CASCADAS (Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services).

HAGGLE project, launched on 1/1/2006 with a duration of 4 years, aims to solve the connectivity and networking problems in mobile ad hoc environments while introducing a new application-driven message forwarding approach. They propose a radical departure from the existing TCP/IP protocol suite, completely eliminating layering above the data-link, and exploiting and application-driven message forwarding, instead of delegating this responsibility to the network layer. The project defines an innovative system that uses best-effort, context aware message forwarding between ubiquitous mobile devices, to provide services even when connectivity is local and intermittent. The Haggle approach is more oriented to the human way of communicating, rather than to other technological aspects of communication. It introduces a new autonomic communication paradigm, based on advanced local message forwarding and sensitive to realistic human mobility. It relies on a communication architecture that uses opportunistic message relaying, and is based on privacy, authentication, trust and advanced data handling.

The Haggle project introduces a Pocket Switched Networking (PSN) approach where two types of applications are defined:

(a) known-sender where one node needs to transfer data to a user defined destination. The destination may be another user (who may own many nodes), all users in a certain place, users with certain role (e.g. “police”), etc. The key point is that, often, the destination is not a single node but is instead a set of nodes with some relationship, e.g. the set of nodes belonging to a message recipient.
(b) known-recipient where a device requires data of some sort, e.g. the current news. The source for this data can be any node which is reachable using any of the three connectivity types, including via infrastructure (e.g. a news
webpage), neighbours (e.g. a recent cache of a news webpage) or mobility (e.g. the arrival of a mobile node carrying suitable data).

In both classes described above, the endpoints of a network operation are no longer described by network-layer addresses, but are instead a set of desirable properties. As a result, general network operations no longer have single source and destination nodes.

Current applications perform very badly in the PSN environment, since they are typically designed around some form of infrastructure which is not always available. The root cause of this is the fact that applications are provided with a networking interface that only understands streams of data directed at anonymous numeric endpoints (namely TCP/IP). This forces developers to implement protocols for naming, addressing and data formatting internally in the applications themselves, e.g. SMTP, IMAP and HTTP.

Instead, Haggle is an unlayered architecture which internally comprises four modules: delivery, user data, protocols and resource management:

![Haggle unlayered architecture](image)

**Figure 5: Haggle unlayered architecture**

As compared to the current layered network architecture, there are a number of high-level differences:

- User data is not isolated from the network, allowing it to be shared with other suitable nodes without an application being involved in each transfer.
- The application does not include network protocol functionality, making it easy for it to be agnostic as to the delivery method, and making the application code simpler.
- Haggle performs delivery using user-level names, allowing it to make use of all suitable protocols and network interfaces for delivery of a given data item.
- Haggle includes a resource management component which mediates between the other three components using user-specified priorities to ensure efficient resource user.

**Described requirements**
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

**Functional:**

- The architecture must support communication in the presence of intermittent network connectivity, by exploiting autonomic opportunistic communications (i.e. in particular in the absence of end-to-end communication infrastructure).
- Privacy, authentication, trust, and advanced data handling must be guaranteed.
- Haggle nodes have to handle the connectivity, have to be able to forward the messages and to manage the information (as in the memory) and the resource.
- Encoding/decoding information must be supported.
- Storing of information in each node must be optimized, being a trade off between the expected utility of the information (improvement of data availability and performance in terms of delay), and the cost of storing it in the local memory.
- The Haggle node should use those subset of data passing through it to be aware of the environment it is operating in.
- Messages should be efficiently distributed through the network, minimizing the consumed resources.
- Routing should be context aware, taking fast reactions to current events in order to ensure the communication between members.
- Nodes should learn from the past events, and use this knowledge for the next actions (forwarding) to be taken.
- Self-protecting capabilities: protection of the message content and protection of the information about nodes involved in the forwarding, and in particular the destination.

**Non Functional:**

**What is the novelty described in this document?**

Several projects aim to develop a radically new network architecture. Compared to other projects, HAGGLE presents a revolutionary paradigm for autonomic communication, based on advanced local forwarding and sensitive to realistic human mobility. The project implements a simple and powerful architecture oriented to opportunistic message relaying, and based on privacy, authentication, trust and advanced data handling.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

The Haggle project focuses on new autonomic networking architectures, whereas SLA@SOI focuses on management of service level agreements. Nevertheless there are some approaches introduced in Haggle's mobile networking principles that can be used in SLA@SOI. For example, Haggle has identified the need of the monitoring of context and resources, and adaptation of algorithms and processes. While in Haggle these mechanisms are designed to make an efficient and effective use of the resources, in SLA@SOI monitoring and adjustment modules will be in charge of detecting SLA violations and implementing corrective and proactive actions.
Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

A first release of the Haggle software architecture and supporting documentation can be found at Sourceforge.net ([http://sourceforge.net/projects/haggle/](http://sourceforge.net/projects/haggle/)) under GNU General Public License (GPL). A new version of the Haggle software architecture is being developed in C++ and will be released in the first half of 2009.

The current implementation has been tested to work with Microsoft Windows XP Service Pack 2 and with all type of WiFi Adapters.

References:


### 3.52 CASCADAS (Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services)

**Keywords:**
Component-Based Applications, Self-Organisation

**Abstract / Summary:**

CASCADAS is a full Future and Emerging Technologies (FET) Integrated Project funded under the Situated and Autonomic Communication (SAC) program of the Information Society Technologies priority area of the European Union's Framework Programme 6 (FP6). The goal of the SAC program is to promote research in the area of new paradigms for communication/networking systems that can be characterised as situated (i.e. reacting locally on environment and context changes), autonomously controlled, self-organising, radically distributed, technology independent and scale-free.

Four projects integrate the SAC Area: BIONETS (BIOlogically-inspired autonomic NETworks and Services), ANA (Autonomic Network Architectures), HAGGLE (An innovative Paradigm for Autonomic Opportunistic Communication) and CASCADAS (Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services).

CASCADAS project was launched on 1/1/2006, with a duration of 3 years. Its overall objective is to develop and validate an autonomic component-based framework to enable composition, execution and deployment of innovative
services capable of flexing and coping with unpredictable environments by dynamically self-adapting to situation evolutions.

Particularly the project development activities aims at prototyping a toolkit based on distributed self-similar components (Autonomic Communication Elements) characterised by autonomic features (self-configuration, self-optimization, self-healing, self-protection, etc). The Autonomic Communication Element (ACE) is the basic component abstraction over which the CASCADAS vision is built. Services are being created and executed (in a distributed way) by the self-aggregation of ACEs.

The ACE Component Model

The ACE forms the core of the CASCADAS Framework, and its component model enables the design of applications in a self-similar manner. Central to this, is the concept of organ. An organ is an ACE internal operative component and, as the name suggests, behaves as an organ in the human body. As such, it is capable of adapting its own execution to unforeseen events and, more in general, to the context in which the ACE is operating. Each organ is responsible for a specific type of tasks, and the interaction among them allows the constitution of an ACE as a standalone component. Assembling an ACE from a set of organs, each of which with clearly defined tasks, leads to a well structured modularized component model depicted in the following figure:

![Figure 6: The ACE Component Model](image)

ACEs’ behaviour is specified in one or more plans contained in the ACE’s self-model, initially created by the developer but capable of being modified autonomously by the ACE itself based on self-awareness information.

The Facilitator is the organ that provides an ACE with capabilities for autonomously adapting its behaviour to changes in the context in which it is executed. According to these changes, the Facilitator modifies the behaviour of the ACE by adapting the existing capabilities or adding new ones.
The *Executor* organ governs the evolution of the ACE according to the actions taken as per self-model. Execution of plans may involve the use of specific functionalities contained in the ACE. To this extent, the Executor may query the *Functionality Repository* organ to obtain the needed ones. The purpose of this repository is store the functionalities deployed while also translating Executor requests into calls to functionalities through an invocation interface exported.

The *Gateway* organ is in charge of handling interactions with the external world. Two different communication protocols are used: GN-GA (Goal Needed-Goal Available) is used for initial service discovery through a publish-subscribe paradigm, and a connection-oriented communication protocol for all other communications, where (one-to-one or one-to-many) communication channels are established through a contracting technique.

The *Manager* organ is in charge of handling internal communication among the organs and taking responsibilities for the ACE’s lifecycle management. With respect to this activity, at any time an ACE can be in one of the four following life cycle states: inactive, running, prepared to move, and destroyed.

Supervision features are brought by a dedicated organ called *Supervision*. This organ is originally disabled, and its activation determines the nature of the ACE as supervisable (thus effectively enabling supervision). The organ is based on *Checker Objects*, which are delegated monitoring activities of specific points in the ACE and are consulted every time a monitoring event occurs. The event can be either accepted or denied, and this latter case may lead to an exception in the supervised organ.

**Described requirements**

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

The requirements of the ACE model are described in the deliverable D6.1 Part A ([3]).

**Functional:**

- ACEs must tolerate very lightweight ACEs with lightweight interfaces, suitable for lightweight devices.
- ACEs must support interoperability between different devices.
- ACEs must support for interoperability between the network-level, the service-level (i.e., the ACE level), as well as the user level.
- ACEs must tolerate execution over unreliable devices and unreliable network links.
- ACEs must support for dynamic and spontaneous aggregation and composition, even in absence of centralized control.
- ACEs' aggregation model should support self-similarity, in which a group of ACEs can be accessed as a single entity.
- Support for transparent aggregation from the outside.
- ACEs should be capable of handling both users level events as well as network level and device level events.
- ACEs should be able to communicate with each other accordingly to various means (point-to-point, any cast and multi cast, local multi cast, probability multi cast).
- ACE should be able to include mechanisms to support the building and maintenance of various structures of overlay networks.
- Such overlay must be lightweight.
Such overlays must be scalable.

- ACE should support mechanisms for dynamic life cycle management (grow, swarm, & shrink...)
- ACEs should be able to provide services with various and tunable QoS, and also should support for QoS evaluation.
- There must generally be some support for autonomic control system in ACEs and in distributed ACEs aggregates
- There must be general support for pervasive supervision tools
- There must be support for fast reactions when something changes
- There must be support for retrieving device specific information (for monitoring and evaluation), and provide access to device specific configuration functions (for effection/actuation)
- Profiles for specific devices (i.e. meta-information) should be made available in order to interpret monitored data and to turn abstract reconfiguration activities into device specific actions.
- Network components (i.e. the “controlling ACEs”) should at least provide a heard-beat function that allows to determine whether a specific network element is still functional.
- There should be a way to monitor whether a new device enters or leaves a certain network segment, i.e., a method to keep track of the actual network topology.
- The existence of situation dependent self-adaptation requires that an ACE configuration has to maintain an internal image of its own structure (probably in implicit/distributed and divided into small parts of knowledge within each of the configuration elements).
- Pervasive supervision systems must protect themselves from attacks (or, which is the same, must be able to exploit the WP4 security mechanisms).
- There must be support for dynamic system monitoring
- There must be support for dynamic system reconfiguration.
- Monitoring activities must be lightweight.
- To enable self-organized adaptation of work and topologies, ACEs must be able to move from node to node.
- Cloning functionality, whereby a new ACE is created which inherits identity and, if required, internal state from its parent, is necessary to enable self-organized dissemination and distribution of services in a large-scale network.
- There must be support for the exchange of personal and confidential data over the network.
- There must be support to control integrity of data.
- There must be support to specify privacy levels for data and services.
- There must be support for large-scale networks, and this any Mechanism should be inherently scalable to very large networks.
- There must be support for mechanism to avoid resources exhaustion.
- Security mechanism should work without the assumption of centralized authorities/services available.
- There must be support to associate “levels of trust” to information and services.
- Security mechanisms may be subject themselves to attacks, and must thus be subject to observability and external control.
- There must be support for recognizing adversarial and free-riding behaviors.
- Security mechanisms should properly adapt to the current status in a situation-aware way, by properly exploiting knowledge networks.
- Knowledge networks must support for a virtual view of environment to facilitate the concept of interest to adapt to changing conditions.
There must be support for distribution of knowledge across a dynamic network.
There must be support for self-similar knowledge aggregation and for access to knowledge at different granularity levels.
There must be support to represent and manage knowledge related to the user and the social level (user and social context profiling).
There must be support to represent and manage knowledge related to the ACE level (profiling of ACEs and of their dynamic and aggregated status).
There must be support to manage in an integrated (cross-layer) way user-level, ACE-level, and network-level, knowledge.
There must be support for construction and management of aggregated distributed knowledge.
To reach a high level of integration, knowledge networks should provide standard semantic mechanisms to organize and compose heterogeneous information and heterogeneous services.
It may be necessary to protect selected sensible parts of knowledge networks from attacks.
There must be support for spatial knowledge and spatial representation of situations.
There must be support for semantic knowledge and shared ontologies to facilitate interoperability.

Non Functional:
- ACEs should be able to implement complex and stateful communication protocols (e.g., negotiations).
- ACEs should support dynamic interfaces (i.e., should be able to dynamically adapt the provided functionalities).
- ACEs does not necessarily have a clearly identifiable name/identifier, or a specific stakeholders, and must be able to interact in anonymous way.
- Reconfiguration process must be fast and without service interruption.
- Self-organization mechanisms, without being undermined in their basic “autonomous” nature, should in any case somehow tolerate observability and external control by supervision and security mechanisms.
- Identification and authentication mechanisms should be lightweight and standardized.
- There must be support for dynamic instantiation of security mechanisms.
- Support for identification of network faults and automatic reconfiguration.
- To achieve a proper reconfiguration meaningful context information are to collected with high time granularity.
- Knowledge networks should provide also for producing and organizing new knowledge, inferred from existing one (e.g., for the sake of prediction).

What is the novelty described in this document?
CASCADAS project presents a new model of distributed components, called ACEs (Autonomic Communication Elements), able to autonomously self-organize with each other towards the provisioning of specific user communication services, and able to self-adapt such provisioning to social and network contexts.

The essence of the innovation stands in exploiting highly distributed resources (even commodity servers of low-cost) running autonomic S/W solutions based on
distributed self-aggregating, self-organising components (ACEs). The overall self-
similar architecture (both pizza-box servers and clusters of servers have the same
functional architecture) supporting a distributed replication of data. This will allow
high levels of availability also starting from low-cost commodity H/W.

The project vision aims at validating a so-called Open Autonomic Service
Environment defined as a highly distributed platform for composing, executing
and providing situation-aware and dynamically adaptable communication and
content services.

How may this work be applicable to SLA@SOI? What are the necessary
actions that should be taken to apply it?

CASCADAS provides a framework to develop autonomic component-based
applications. It is supposed to dramatically reduce the costs associated to the
development and configuration of systems and services. In this sense, it could be
used in the implementation of those subsystems of SLA @ SOI requiring
autonomous characteristics.

Is there a readily available implementation of the described work? If so,
please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

The most important result of the project is an Open Source toolkit with a set of
well-integrated abstractions, algorithms, tools, and application demonstrations.
The toolkit is available in sourceforge: http://sourceforge.net/project/showfiles.php?group_id=225956, under GNU
General Public License (GPL). The status of the software is production/stable.

References:
[3] Description of Application Scenarios and of the Services to be Provided
[4] Acetoolkit:
http://sourceforge.net/project/showfiles.php?group_id=225956

3.53 BIONETS (BIOlogically-inspired autonomic NETworks and Services)

Keywords:
Pervasive Computing, Embedded Systems

Abstract / Summary:
BIONETS is a full Future and Emerging Technologies (FET) Integrated Project
funded under the Situated and Autonomic Communication (SAC) program of the
Information Society Technologies priority area of the European Union's
Framework Programme 6 (FP6). The goal of the SAC programme is to promote
research in the area of new paradigms for communication/networking systems
that can be characterised as situated (i.e. reacting locally on environment and context changes), autonomously controlled, self-organising, radically distributed, technology independent and scale-free.

Four projects integrate the SAC Area: BIONETS (BIologically-inspired autonomic NETworks and Services), ANA (Autonomic Network Architectures), HAGGLE (An innovative Paradigm for Autonomic Opportunistic Communication) and CASCADAS (Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services).

BIONETS project was launched on 1/1/2006, with a duration of 4 years. Its goal is to provide a biologically-inspired open networking paradigm for the creation, dissemination, execution, and evolution of autonomic self-evolving services able to adapt to localised needs and conditions while ensuring the maintenance of a purposeful system.

The project addresses problems in pervasive communication/computing environments characterised by an extremely large number of embedded devices. These embedded devices will possess computing and (basic) communication capabilities, having the potential to form a massively large networked system, orders of magnitude larger than the current Internet. Overall, the complexity of such environments will not be far from that of biological organisms, ecosystems, and socio-economic communities.

Traditional communication approaches are ineffective in this context, since they fail to address the main challenges of such environments: Heterogeneity (huge differentiation at the device and service level), Scalability (billions of nodes, a multitude of users and services) and Complexity (management of a large-scale heterogeneous mobile network, provisioning of consistent and secure service operations).

BIONETS overcomes these issues via an autonomic and localized peer-to-peer communication paradigm. Services in BIONETS are also autonomic, and evolve to adapt to the surrounding environment, like living organisms evolve by natural selection. Network operations will be driven by the services, providing an ad hoc support when and where needed to fulfill users requests. Security issues will be considered as a fundamental part of the services themselves, representing a key ingredient for achieving a purposeful autonomic system. The network will become just an appendix of the services, which, in turn, become a mirror image of the social networks of users they serve.

The overall structure of BIONETS networks is depicted in the following figure:
The heterogeneity issue has been addressed by introducing two-tier SOCS (Service Oriented Communication Systems) network architecture. The upper layer consists of so called U-Nodes (User Nodes) which are basically devices running services. U-Nodes form islands of connected devices and may exchange information when getting into mutual communication range; decisions are taken by the service itself.

The lower layer consists of tiny devices (T-Nodes) with sensing/identifying capabilities (i.e. sensors, tags or RFIDs), gathering data from the environment. They do not communicate among themselves, but they simply answer to polI messages sent by U-Nodes which are interested in getting the actual value of the random field they are sensing.

The resulting network topology is an “archipelago of connected islands of nodes”.

Described requirements

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Requirements fot BIONETS project are defined in “Deliverable 1.1.1 Application Scenario Analysis, Network Architecture Requirements and High-Level Specifications”. In short, network architecture must show self-organization capabilities: self-configuration, self-management, self-optimization, self-healing, self-protection, self-adaptation, self-awareness and self-localization.

Functional:

- System needs to scale with the high number of nodes, to manage with heterogeneity and complexity.
- When a node is starting up, it needs to be aware of its internal components, external components and nodes which contribute into the system.
- Configuration of components and systems shall follow high-level policies of them. The rest of system should adjust automatically and seamlessly into the situation.
- The roles of each node need to be discovered, allocated dynamically.
The environment need to be continuously monitored to keep the system in the living state.

Components and systems modelling shall seek opportunities to improve their own performance and efficiency. System shall automatically detect, diagnose and repair localized software and hardware problems. Each node should automatically detect the meaningful changes happening in the environment. The system shall automatically defend against malicious attacks or cascading failures. The system should use early warning to anticipate and prevent system-wide failures. The system shall automatically detect the meaningful changes happening in the environment, and adapt the system behaviour accordingly.

System shall be automatically situation aware and detect the environment conditions to be able to act as a stand alone situation in a proactive and self-aware way.

System shall automatically localize the node itself and the other nodes and discover the information stored in them.

Non Functional:
None.

What is the novelty described in this document?

BIONETS will design an innovative framework for the creation, dissemination and evolution of autonomic services, able to survive and evolve in a resource-constrained, dynamically changing and heterogeneous environment, without relying on a centralized control.

Evolution in an autonomic communication system should happen in an on-line and distributed way: the system is continuously undergoing selfoptimisation, i.e., transforming itself to comply with significant changes in requirements or in the environment. Evolution goes beyond short-time reactive adaptations that are pre-programmed in the system and for which well-known techniques can be employed. This is a challenging research topic: currently little has been done in evolving complex interacting systems during their own operation.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

The BIONETS Service Framework (SerWorks) includes capabilities to react in an autonomic way to changes in the context/environment. These capabilities include different evolution and adaptation strategies, service life-cycle handling and mobility support. This could be directly used in the SLA@SOI Adjustment module.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

The net result of BIONETS will be the provisioning of a digital ecosystem for autonomic services, able to fulfill users demands and needs in a transparent and efficient way by exploiting the unique features of pervasive communication/computing environments.
The resulting architecture, named SerWorks, is divided into three frameworks. The upper layer of the architecture is the Service Framework, which includes the application/service-level logic and the functions supporting their distributed and autonomic execution and management. The middle layer includes the Interaction Framework, whose purpose is to provide multiple concurrent interaction models supporting the communication among the distributed services and the realization of a shared data space. The lowest layer is the Networking Framework, which provides the basic communication capabilities in the disappearing network context, supporting means for establishing secure communications over opportunistic networks.

The Service Framework provides a runtime environment for the execution of service logic running on U-nodes and T-nodes. On the other hand, the service framework includes capabilities to react in an autonomic way to changes in the context/environment. These capabilities include different evolution and adaptation strategies, service life-cycle handling and mobility support. The necessary control logic for performing such functions is abstracted into service mediators, architectural elements complementary to the service cells.

Architecture specifications can be downloaded from BIONETS Web Site. There is no current available implementation of the SerWorks architecture.

References:

3.54 ANA (Autonomic Network Architecture)

Keywords:
Autonomic Networking

Abstract / Summary:
ANA is a full Future and Emerging Technologies (FET) Integrated Project funded under the Situated and Autonomic Communication (SAC) program of the Information Society Technologies priority area of the European Union’s Framework Programme 6 (FP6). The goal of the SAC program is to promote research in the area of new paradigms for communication/networking systems that can be characterised as situated (i.e. reacting locally on environment and context changes), autonomously controlled, self-organising, radically distributed, technology independent and scale-free.

Four projects integrate the SAC Area: BIONETS (BIOlogically-inspired autonomic NETworks and Services), ANA (Autonomic Network Architectures), HAGGLE (An
ANA project was launched on 1/1/2006, with a duration of 3 years. Its main goal is to explore novel ways of organizing and using networks beyond today’s Internet technologies. ANA aims at designing and developing a novel autonomic network architecture that enables flexible, dynamic, and fully autonomous formation of network nodes as well as whole networks.

The resulting **autonomic network architecture** will allow dynamic adaptation and re-organisation of the network according to the working, economical and social needs of the users. This is expected to be especially challenging in a mobile context where new resources become available dynamically, administrative domains change frequently, and the economic models may vary.

The Autonomic Network Architecture (ANA) project has two complementary objectives that iteratively provide feedback to each other: a scientific objective and a technological one. The scientific objective of this proposal is to identify fundamental autonomic network principles that enable networks to scale not only in size but also in functionality.

A functionally scaling network is considered as a synonym for an evolving network which includes the various self-x attributes essential to autonomic communication such as self-management, self-optimization, self-monitoring, self-repair, and self-protection. The hypothesis is that, due to these self-x attributes, such functional scaling will naturally lead to networks that are not only richer in functionality but which also scale in size.

The technological objective of ANA is to build an experimental autonomic network architecture, and to demonstrate the feasibility of autonomic networking within the coming 4 years.

**Described requirements**

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Requirements for ANA project are described in ANA Deliverable D1.2 Requirements Analysis, available at [http://www.ana-project.org/deliverables/2006/D.1.2.-Requirements.pdf](http://www.ana-project.org/deliverables/2006/D.1.2.-Requirements.pdf):

**Functional:**

- It is possible to design highly scalable networks with the ANA architecture that are not restricted by address spaces or other means.
- Multiple realms must be supported concurrently. To achieve this, a simple bootstrap like process that allows to bootstrap multiple different realms needs to be developed.
- Realms can be designed in such a way that they are extensible and can easily include new solutions without breaking the network architecture and design.
- Realms must be able to host other (partial) realms and provide means for inter-realm communication.
A monitoring facility has to be part of the architecture. It must be able to potentially monitor all data and events of interests.

Information and knowledge management is needed to exchange and use monitoring and other types of meta-data as the basic element of solutions with self-* properties.

The provision of monitoring infrastructure and knowledge supports the need for more resilient network.

The information and knowledge plane enables also easier self-configuration and self-organization, e.g. through better service discovery solutions.

The provision of the information and knowledge plane also leads to better solutions for self-optimization, e.g. for the case of structuring communication into uni-cast, any-cast, and multi-cast communication for efficient resource utilization from networks viewpoint.

Mobile users and mobile terminals can be better supported than it is possible in today’s Internet.

Non Functional:
- The monitoring facility must also provide means for storing monitoring data.

What is the novelty described in this document?

Several projects aim to develop a radically new network architecture. In Europe, ANA is timely positioned to explore the design space of autonomic networking, and so are other projects of the EU-funded initiative in Situated and Autonomic Communications (SAC). In the US, the GENI initiative aims at developing a generic world-wide research infrastructure upon which next generation network architectures could be built and tested.

Compared to other projects, ANA will focus on the “autonomic” behavior of future network architectures: this is where ANA will provide the majority of its innovations.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

The ANA project focuses on new autonomic networking architecture, which is out of the scope of the SLA@SOI project. Nevertheless, there are some approaches introduced in ANA that can be used in SLA@SOI. For example, the concept of monitoring as a dynamic and adaptive system, where the decision module starts and controls the monitoring system, and has the ability to dynamically adapt the monitoring parameters under consideration to the needs of a given module when necessary (i.e. when additional information is required to detect traffic anomalies, the data capturing is updated “on-the-fly”).

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

A key objective of the ANA project is to demonstrate and validate the architectural principles with a real implementation. The ANA software was developed in C for Linux systems.

The first version of the ANA prototype is now publicly available for download at http://www.ana-project.org/web/software/start.
3.55  **FOCALE: A Novel Autonomic Networking Architecture**

**Keywords:**
Distributed Computing, Resource Orchestration, Ontology, Machine Learning

**Abstract / Summary:**
Foundation, Observation, Comparison, Action, and Learning Environment (FOCALE) is a semantically rich architecture [1] for orchestrating the behaviour of heterogeneous and distributed computing resources. This approach uses the fusion of information models, ontologies, and machine learning and reasoning algorithms to enable semantic interoperability between different models.

The basic building block of FOCALE is the Autonomic Communication Element, shown in the following figure:

![Autonomic Computing Element](image)

**Figure 8: FOCALE - The Autonomic Computing Element**

This approach assumes that any Managed Resource (which can be as simple as a device interface, or as complex as an entire system or network) can be turned...
into an Autonomic Component. By embedding the same Autonomic Manager in each ACE, we provide uniform management functionality throughout the autonomic system. This is then expanded, first to a uniform Autonomic Management Domain (a set of Autonomic Elements communicating through an Event Service) and then to an Autonomic Management Environment), which consists of a set of Autonomic Management Domains.

The Autonomic Computing Element has the following components:

- Foundation, a set of finite state machines (FSMs) that define the desired behaviour of the system. DEN-ng information/data models and ontologies are used to build the foundation.
- Model-Based Translation Layer (MBTL): a mediation layer between the Managed Resource and the Autonomic Manager. It translates vendor-specific sensed data into a form that the Autonomic Manager can process; similarly, it enables the Autonomic Manager to define actions to be taken that are translated into vendor-specific commands.
- The Observe, Compare, Reason and Act elements form the control loop. Reasoning is based on ontologies.
- Learn: One or more machine learning algorithms may be employed to gain experience from the environment, and to aid the reasoning process.

The policy server serves two purposes: (1) to control the action of each of the ACE components, and (2) to interface to the outside world (humans and machines, such as ACEs).

Described requirements

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Requirements behind the design of the FOCALE architecture are:

**Functional:**
- Ensure that Autonomic Enabled Management Components can also be efficiently managed.
- Provide a means to orchestrate the behavior of Autonomic Components.
- Ensure that all Autonomic Components adjust their functionality in unison, according to policy.
- Provide a means to reason about the environment and recommend or take appropriate actions, so that the underlying business goals are not violated and, hopefully, optimized.

**Non Functional:**
None.

What is the novelty described in this document?

FOCALE is a novel autonomic networking architecture, which introduces different techniques to seamlessly manage and integrate heterogeneous and distributed computing resources using two levels of control loops. The use of knowledge engineering techniques is proposed to manage legacy as well as new (as in inherently autonomic) components and orchestrate their behaviour. FOCALE
solution uses Ontologies as a shared vocabulary between semantically equivalent components and Policies.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

The FOCALE architecture offers interesting aspects about autonomic computing that can be directly applied in those subsystems of the SLA@SOI framework that are going to be build following autonomic principles.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

This is an architecture definitions, without an available implementation.

References:

3.56 SAHARA - Service Architecture for Heterogeneous Access, Resources, and Applications

Keywords:
Service Composition

Abstract / Summary:
SAHARA project [2] develops an architecture for the creation, placement, and management of services for composition across independent providers. Its goal is to enable end-to-end service composition with desirable, predictable and enforceable properties spanning multiple potentially distrusting service providers.

SAHARA classifies service composition into two models based on the type of interaction between composed component service providers: Cooperative Model, where the Service providers interact in a distributed fashion, with distributed responsibility, to provide an end-to-end composed service; and the Brokered Model, where a single provider, the broker, uses the functionalities provided by underlying service providers and encapsulates these to compose the end-to-end service.

In either case, the end-user subscribes to only one provider. However, the difference lies in the way the responsibility for the composed service is apportioned.

SAHARA has performed in-depth evaluations of several mechanisms for service composition across the domains:
• Measurement-based adaptation: dynamically selection of service providers, and service instances based on current network and server loads.
• Utility-based resource allocation mechanisms:
  o Auctions: Auctions allocate resources to consumers based on their bids, which represent the value of the good to them (i.e. the current resource load).
  o Congestion pricing: assigns scarce resources to consumers using the abstraction of price as a means to moderate demand. During high demand, such a market ensures that the price increases. Only those consumers with the greatest need and having sufficient currency will obtain the needed resource. During low demand, the price drops, and access to the resource with be cheap and plentiful.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
• The model must support composition across independent service providers.
• The model must allow third-parties to discover components and to broker new services from constituent pieces. An essential need for brokers is to be able to verify the performance and behavior of the assembled components. If a component does not meet its performance or behavioral specification, it must be “composed out”, and a new instance from a different provider “composed in”.
• The architecture must allow a distributed policy management between service providers. By negotiating policy changes at various points in the topology using a map, policy agents can improve load balancing and fault tolerance.
• The model must support the establishment and monitoring of trust relationships between inherently untrusting entities. Typically this is done by a AAA (Authentication, Authorization and Accounting) server, but in a multidomain scenario several AAA servers may be involved.
• The model must include a pervasive monitoring and measurement infrastructure is to verify whether the provided service adheres to the desirable properties advertised by its provider. Such properties can be specified in a bilateral Service Level Agreement (SLA) between provider and requester.
• The system must support efficient and dynamic resource allocation across providers.

Non Functional:
None.

What is the novelty described in this document?
SAHARA is a new service architecture, where systems are organized not as monolithic structures deployed by a single business entity, but rather as a dynamic confederation of multiple—sometimes cooperating and sometimes competing—service providers.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

Since SLA@SOI framework must be extended to multidomain/multiprovider scenarios, the SAHARA architecture and service composition models should be carefully watched.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

A prototype for secure authentication for WLAN roaming can be downloaded from [2].

**References:**


**3.57 PROSPECT- A Prospect of Multi-Domain Management in the Expected Open Services Marketplace.**

**Keywords:**

Broadband Network, End-to-End Management

**Abstract / Summary:**

PROSPECT is an ACTS (Advanced Communications Technologies & Services) project of the EU Research Programme. PROSPECT has tested the integrated end-to-end management of Integrated Broadband Communication (IBC) services, using multiple service providers in a broadband network environment.

The main objective of the Prospect project was as follows [3]:

'To realise and validate, via a commercial/business end-user trial, the integrated end-to-end management of IBC services, in a multi-service provider and broadband network environment. The focus will be on the design and implementation of solutions for
interconnecting management systems for competing and co-operating service providers of pan-European services.*

PROSPECT's technical approach is based on the goal of executing a trial which can demonstrate and validate the management of co-operating and competing services in support of commercial / business end-users service requirements.

PROSPECT made use of new concepts (TINA, ODPRM, CORBA, interworking between CORBA and TMN world) for service management. The trials showed a tele-educational service scenario, composed of a number of multi-media tele-services from independent service providers.

Faced with the problem of integrating multiple service management systems over several business scenarios, the project opted for a component-based system architecture. Reusable components, defined with open interfaces, facilitated the composition of the trial systems. The project developed both functional units, for instance a subscription management component, and generic technology gateways, for example a CORBA/CMIP gateway component.

Management services implemented by the project comprised of management systems for composite services such as the Tele-Education Service, for information services such as the WebStore service, and for communication services such as the Virtual Private Network service. While the systems for the composite service and the information services were implemented using CORBA technology, the management systems for the communication services used both CORBA and TMN platforms. This reflects the situation in the expected open service market, where enterprises must collaborate across heterogeneous technological domains.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

**Functional:**
- To provide a management architecture for creation, deployment, co-operative provisioning and usage of Integrated Broadband Communication (IBC) services.
- To realise and validate, via a commercial / business end-user trial, the integrated end-to-end management of IBC services, in a multi-service provider and broadband network environment.
- To gather commercial end-user and service provider requirements and experiences in using IBC services and evaluate the ability of the integrated end-to-end management system to meet their needs.
- To empower the end-users to manage their services by providing user access to the end-to-end service management infrastructure.
- To set up the service configuration and status monitoring of client and server systems.
- To implement inter-domain management interfaces between service providers and network operators to negotiate resource and bandwidth allocations.
- To provide inter-domain management interfaces between service providers and end-users to agree on service profiles and to monitor service usage and quality of service contracts.

**Non Functional:**
- PROJECT architecture must use a TMN framework
- The architecture must ensure the co-existence between CORBA and TMN systems.
- To provide IDL/GDMO interaction translation components

**What is the novelty described in this document?**

The main benefit of the Prospect system to the service provider stakeholder is the automation of service management processes, which directly leads to reduced costs in operating the service. The service customer also benefits from the automation of service management processes, and, additionally, from on-line access to all service management functions, which speeds up the customer-provider interactions and makes them more accurate.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

SLA@SOI should have a deep look into PROSPECT project, since both share some goals: the automation of inter-domain delivery and management of services. Nevertheless, they have some differences: PROSPECT gives the end users the power to manage their services by providing them with access to the end-to-end service management infrastructure, which SLA@SOI does not. Also when dealing with SLA violations, PROSPECT only foresees its impact on customer’s bill, while SLA@SOI will implement an integrated adjustment module.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

The Prospect Methodology was presented at a joint NMF/NIM chain workshop in Dublin. The methodology was well received and included as background information in the NMF submission to ITU-SG 4 representing results from ACTS.

The detailed work on Accounting Management in Prospect was passed onto VITAL where it influenced their further modelling of TINA Accounting Management. This VITAL work was fed back to TINA-C and now appears in the TINA Service Component Specification.

**References:**


3.58 **Policy-based management on multidomain/multiprovider environments.**

**Keywords:**
Policy-Based Management, QoS

**Abstract / Summary:**
In the context of fierce competition in open liberalized markets, Service Providers (SP) are therefore currently investigating opportunities to provide their customers with differentiated service level agreements (SLAs) which state the obligations entered into by both network provider and customer. However, to satisfy the customer needs, it is mandatory to take its requirements into account in a flexible way, even if the management of the end to end communication is more challenging since the service can span heterogeneous provider domains. And yet needs to be managed on an end-to-end basis. Then policy-based management becomes a gaining approach to deploy management strategies.

Policies are a set of pre-defined rules (defined actions to be triggered when a set of conditions are fulfilled) that govern resources, including conditions and actions that are established by the administrator with parameters that determine when the policies are to be implemented in the network. In the case of a Service Provider, policies are defined based on one hand the high-level business objectives of the SP and on the other hand on the SLA agreed with its customers and partners SP. Policies allow changing the behavior of a system without changing its implementation, creating adaptable systems whose behavior can be altered dynamically.

Reference [2] investigates the possibility to merge policy based management with mobile agents in order to handle QoS of communications spanning over a number of ISP domains. In this environment, mobile agents will act on behalf of users or third party service providers, to obtain the best end to end service based on a negotiation process between ISP policy management systems. Generally, an agent can be regarded as an assistant or helper, which performs routine and complicated tasks on the user's behalf. In the context of distributed computing, an agent is an autonomous software component that acts asynchronously on the user's behalf.

The framework is comprised of a Policy Enforcement Point (PEP) that is a policy decision enforcer component and a Policy Decision Point (PDP) which is the decision-making component. It defines a set of agents depending on their respective roles in the architecture:
- **Local PEP Agent:** collects information related to the entire domain
- **PEP Mobile Agents:** sent to the PDP system when the PEP agents notes a change in the system. The sent agent contains all the information needed
to identify the source of the request (customer) and the destination of the call (calling party) as well the parameters related to this event (for example QoS parameters).

- Local PDP Agent Manager: retrieve related information and policies from the policy DB and tries to trigger any policy rule that can be triggered regarding the information carried by the PEP mobile agent. If any configuration related to new policies are defined, it creates a PDP Mobile Agent, it send to remote PEP to perform the new configuration. It takes also into account interdomain interactions, when a decision needs to be negotiated with remote domain PDP Agent Manager. When interacting with remote domain, it creates a PDP Domain Mobile Agent.

- PDP Mobile Agent: When a PDP has taken a decision it sends a PDP Mobile Agent to enforce policies directly in the PEP component in all the network elements that are concerned by this new decision.

- PDP Domain Mobile Agent: When a PDP has to take a decision related to an interdomain connection, it has to identify the set of remote domain need for the connection and send a Domain Mobile Agent to negotiate the term of services needed by the customer. It is the only multidomain agent in the framework.

Reference [1] detects a need for a common (general-purpose) policy language that allows the configuration of services in different areas using the same core language. It presents a policy ontology where specific policy languages are created as simple extensions of a generic policy model. In addition, by explicitly representing (and mapping) the domain the policies are applied to, domain concepts can be used directly inside policies. As a result, policies can be more targeted towards different areas and domains.

The ontology can be divided into three main parts:

- The **generic policy model** defines the common structure of policies and how they can be combined into policy sets. It contains generic concepts that are common to different types of policy languages. The generic model is based on a combination of concepts found on previous policy languages (XACML, PCIM, Ponder) and the rules format of ILOG Jrules.

- **Specific policy language extensions** are created on top of the generic policy model. They extend this model by introducing concepts that are important in an specific area. For example, the security policy language defines security related properties like roles and signatures.

- By explicitly representing the **domain model**, policies can be targeted to a specific domain as well. If the domain model concepts, are mapped to concepts in the generic policy models, authors can use these high-level concepts when specifying policies. For example, policies can refer to concepts like telecom operator, service provider and end user (and their properties).

All types of policies can be described as **condition-action rules**, where the action describes what should happen, and the conditions describe under which circumstances. Multiple policies can be combined using policy sets. A policy set is a combination of policy elements, where a policy element is either a policy or another policy set. Each policy set uses a policy combination algorithm that describes how the (possibly conflicting) policies should be combined.

**Described requirements**
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:

- **Common policy language:** There is a need for a common policy language that can be used for expressing policies in different areas (like security, QoS, etc.) and in various application domains (like telecom, healthcare, etc.).
- **Ease of use:** Different types of users, ranging from system administrators to end users, must be capable of specifying their preferences using these policies. Therefore, a policy solution should simplify policy specification by offering not only concepts at a higher level of abstraction, closely related to the problem domain, but also user-friendly formats for specifying policies.
- **Extensibility:** It should be straightforward to extend existing policy languages with new concepts, add new domain concepts, create new policy languages, add new policy representation formats, etc.

Non Functional:

None.

**What is the novelty described in this document?**

Non reported.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

Although the work of the Reference [2] refers to network environments, its ideas could be extrapolated to multidomain services in general, being so helpful in the SLA@SOI project.

The authors of reference [1] have implemented a prototype in the context of a telecom Service Enabling Platform (SEP), and have extended the general policy model to a security policy language and an SLA/SLO policy language. Since it provides a general policy language, SLA@SOI can take advantage of this work when extending the model to a multiprovider/multidomain environment. However, this ontology focuses on event-driven policies. It has not been determined yet whether it can be applied to other types of policies, such as state-based policies. Furthermore, this work does not address the problem of policy refinement, where higher-level policy concepts are mapped to implementation-specific classes. Although the policy ontology can be used to translate high-level concepts to domain concepts, the translation of these domain concepts to implementation classes is currently hidden in the implementation of our policy engine (e.g. services and their operations are defined using a WSDL interface). And finally, in the prototype the different ontologies and mappings are specified using an implementation- or tool-dependent format (Java and JRules). It might be better to specify them in an independent format (e.g. OWL) and explicitly import them into the tools.
Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

References [1] and [2] both have implemented prototypes.

References:

3.59 perfSONAR (performance Service Oriented Network monitoring ARchitecture.)

Keywords:
SOA, Monitoring Architecture, Network Performance Data

Abstract / Summary:
GÉANT2 [1] is the high-bandwidth, academic Internet serving Europe’s research and education community. Connecting over 30 million researchers with a multi-domain topology spanning 34 European countries and links to a number of other world regions, GÉANT2 is at the heart of global research networking. GÉANT2 is co-funded by the European Commission under the EU’s Sixth Research and Development Framework Programme.

The perfSONAR project [2] is carried out by a consortium of organisations (GEANT 2, ESnet, Internet 2 and RNP) and aims to design and build network performance monitoring middleware (the perfSONAR web services) and visualisation tools that assists the analysis of local or remote interconnected network domains. perfSONAR tries to make it easier to solve end-to-end performance problems on paths crossing several networks.

The goal of perfSONAR is to enable ubiquitous gathering and sharing of this performance information in order to ease management of advanced networks, facilitate crossdomain troubleshooting and to allow next-generation applications to tailor their execution to the state of the network. This system has been designed to accommodate easy extensibility for new network metrics and to facilitate the automatic processing of these metrics as much as possible.

perfSONAR is a services-oriented architecture composed of three layers, as shown in the following figure:
The Measurement Point Layer is responsible for performing active or passive measurement tests via multiple Measurement Points (MPs), i.e. existing network monitoring tools. The MP is wrapped into a higher level abstraction called Measurement Point Service, belonging to the Service Layer, which hides the implementation details of the MP.

The Service Layer is composed of multiple services that control the monitoring infrastructure, receive, store and exchange measurement and network topology data. Services interact with each other without human intervention (e.g. measurement data retrieved by a Measurement Point Service is fed into a Measurement Archive Service and manipulated by a Transformation Service) and with the upper User Interface Layer.

The end users interact via the visualisation tools at the User Interface Layer only.

The following services have been defined in the perfSONAR framework:

- **Measurement Point (MP) service**: collect and publish the data that the network domains’ measurement tools have collected. The measurement data can be collected on demand or in regular intervals according to a defined schedule. The data can be published to a client, transferred to other web services or stored in an archive (a database or a file system).
- **Measurement Archive (MA) service**: stores the measurement data. The Measurement Archive web services can also be used to write measurement data that other web services provide to data stores.
- **Lookup service (LS)**: registers information regarding active services and their capabilities. Each network domain that joins the perfSONAR infrastructure needs a Lookup web service with which all other web services in this domain register. The Lookup services of multiple network domains communicate with each other and inform clients which web services are available. Clients just need the URL of the Lookup web service.
- **Topology service (TS)**: stores network topology information
• Authentication service (AS): provides authentication and authorisation services required in users - services interactions
• Transformation service (TrS): performs manipulation (aggregation, statistics) on available data sets
• Resource Protector (RP) service: arbitrates the use of limited measurement resources

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
• perfSONAR must provide standardised access to multi-domain network information.
• The system must provide easy identification of information sources.
• The system must keep track of network performance.
• The system must support on-demand tests to provide additional evidence about the network behaviour.
• perfSONAR must provide consistent end-to-end service monitoring.
• The system must support the adding or removing of components and Measurement Points during the system operation.
• The system should be self-configurable, allowing components and Measurement Points to autonomously announce their existence and capabilities.
• The system should be decentralized, allowing each administrative authority to limit the system capabilities in accordance with locally-specified policies and procedures.
• The system must be scalable, that can a) incorporate multiple networks and overlay virtual communities, b) handle varying numbers of users and servers, and c) handle varying information volumes as well as differing types of monitoring data and tools.
• The system must be secure: it cannot be exploited for uses other than performance monitoring, which is not particularly vulnerable to attack.
• It must be a safe system that does not overly congest the networks it is trying to monitor.
• It must be a fault-tolerant system that fails gracefully in the presence of module failures.
• The system must have self-diagnosing capabilities, providing clear and timely exception messages in the case of failure.

Non Functional:
• The architecture must be based on Web Services (WS) technology, which allows defining the interaction between services through well defined, language independent interfaces.
• The perfSONAR approach must not have any dependencies from the lower networking technologies, then permitting new services to be easily added.
• The protocol must follow the Open Grid Forum (OGF) Network Measurement Working Group schema [5].

What is the novelty described in this document?
perfSONAR is a Service Oriented monitoring architecture for retrieving, storing, processing and presenting network performance metric measurement data in a multi-domain network environment.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

perfSONAR is focusing on network level metrics, while SLA@SOI deals with integrated services (from the business to the infrastructure level). But our project should have a deep understanding on the architecture for multi-domain monitoring, and should study if the main ideas in perfSONAR can be directly applicable in SLA@SOI.

The framework has been build flexible enough to cater for new metrics and for different type of technologies.

In order for the described architecture to be truly useful in a multi-domain environment, there is the need to harmonise the type of collected measurements and the procedures for their composition in the TrS

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

The perfSONAR infrastructure is open and any tool can take advantage of it. Many perfSONAR services and monitoring applications have already been implemented as standalone measurement tools. Weathermaps, looking-glasses, IPPM measurements, and many other monitoring applications have already been implemented using the perfSONAR framework.

Latest version perfSONAR MDM 3.1 was released [3] in January 2009. It can be downloaded from perfSONAR web site [2]. The software has been tested to work on Red Hat Enterprise Linux 4.x or 5.x, and Debian 4.0.

References:

3.60 OVF Open Virtualisation Format

Keywords:
Virtual Machine, Functional Description

Abstract / Summary:
The Open Virtual Machine Format (OVF) describes an open, secure, portable, efficient and extensible format for the packaging and distribution of (collections of) virtual machines. Open Virtual Machine Format Specification have been submitted by Dell, HP, IBM, Microsoft, VMware, and XenSource to the Distributed Management Task Force (DMTF) for further development into an industry standard. Of particular note as described within the OVF whitepaper[http://www.vmware.com/resources/techresources/1003]:

“The OVF format has several specific features that are designed for complex, multi-tier services and their associated distribution, installation, configuration and execution workflow:

1. It directly supports the configuration of multi-tier applications and the composition of virtual machines to deliver composed services.
2. It permits the specification of both VM and application-level configuration.”

OVF has direct relationships with CIM as it uses CIM schemas and vocabularies to help describing its schema. There may also be a relationship to the Open Virtual Appliance Specification (OVA), however a precursory glance at the specification would hint that it was consumed by the OVF effort, in part. The specification was made available [1] on the Xen-CIM mailing list in 2006. V 1.1.0 of the specification was published in January 2010. ANSI has since ratified the V 1.1.0 as standard INCITS 469-2010 [2].

**Described requirements**

**Functional:**

OVF is a standard that allow for the functional description of virtual machines

**Non Functional:**

It does not include a means to describe non-functional aspects of a virtual machine.

**What is the novelty described in this document?**

It describes an industry standard that could become the default standard for describing virtual machines.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

The OVF standard and associated schema could form the basis of the document that is contained in a provisioning request. However, although describing functional properties of a virtual machine it does not describe non-functional properties. To be used in SLA@SOI and for it to take advantage of the SLA@SOI framework, OVF would need to be extended to account for non-functional properties.
Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

1. VMware describe their OVF Tool - http://www.vmware.com/resources/techresources/1013

References:


3.61 Libvirt

Keywords:
Virtualization, OS

Abstract / Summary:
SLA@SOI aims to provide a framework whereby the life cycle of physical and virtual infrastructure (such as virtual machines, network, etc) can be managed by means of an Application Programming Interfaces (APIs).

By interfacing with a hypervisor running on top of physical hardware introducing abstraction layers for resource provisioning, scheduling and management, it is possible to treat these resources as collection of services that realise the SOI approach.

The initial building blocks of the SOI approach is the provision of a unified interface to virtualization, independent to particular virtualization implementations. Currently within the open source community Libvirt[1] is one of the most prominent and promising projects. Sponsored by Redhat, Libvirt is an open source API that provides a generic way to interact with different types of open source virtualization technologies (such as Xen[2] and KVM[3] among many others) for the management of the lifecycle of virtual machines.

Libvirt is used by Redhat and many other linux distribution vendors as a means of providing an up-to-date and abstract interface to any type of supported underlying hypervisor. Although Redhat acquired the company behind KVM, the current default virtualization technology in Redhat based systems is Xen based. In Ubuntu, the default virtualization technology is KVM and mainline Xen kernel integration has been left out as a community effort.
From an open source strategy perspective Libvirt allows Redhat to abstract away from the particular implementations or vendors and use the most suitable virtualization approach available. For the future, it is expected that Redhat will integrate KVM by default rather than Xen.

Described requirements

Functional:
- Unified Virtualization API
- Integration with main open source linux distributions
- Management features
- Support for latest virtualization in hardware features (such as Intel VT for example [7])
- Integration with standards (such as CIM[6] for example)
- Networking features (Virtual Networks, Network Bridge)
- Virtualization as OS resources: Users, processes, quotas (KVM)

Non Functional:
- Active community and commercial/enterprise support available
- GUI tools for virtualization management
- Easy to use for quick prototyping
- Free and open source

What is the novelty described in this document?

Unified approach to virtualization

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

SLA@SOI requires the usage of current virtualization technologies and exploits their features to provide the holistic SLA framework. By building on solid foundations such as libvirt, SLA@SOI would be able concentrate on adding value to the higher levels of the SOI stack.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

Libvirt, published under the GPL license, is readily available in mainstream linux distributions

References:
3.62 OpenNebula

Keywords:
Virtual Machine Manager

Abstract / Summary:
From the OpenNebula’s website [1]:

“OpenNebula is an open source distributed VM manager that enables the dynamic placement of VMs on a pool of physical resources. OpenNebula extends the benefits of virtualization platforms from a single physical resource to a pool of resources, decoupling the server not only from the physical infrastructure but also from the physical location.”

The approach of OpenNebula is based on extending some of the well known grid computing concepts (such as grid federation, global grids [2]) and applying them to virtual resources (virtual machines) rather than the more traditional approach of Grid computing of presenting physical machines as addressable resources.

OpenNebula also presents potential service consumers with a unified interface to several providers supporting virtualization, from small players using standard open source solutions to virtualization based commercial infrastructure providers such as Amazon’s EC2[3] or Elastic Hosts[4]

With the introduction of virtualization and the multi-provider federated approach, one of the challenges is to transport the physical virtual machine images to and from the providers. With an extensible plugin-based architecture, it is possible to add or enhance different OpenNebula components. For example Haizea extends the basic scheduler capabilities to include virtual image management (transfer, lifecycle) and resource allocation.

The development of OpenNebula is partly supported by the EU FP7 funded project Reservoir.

Described requirements

Functional:
- Client/Server based
- Front-End (Server) / Execution nodes (Hypervisors)
- Very lightweight implementation
- Basic "SLA" support (Elasticity Rules)
- Communication among Frontend-Clients via ssh
- xmlrpc based external API
- Basic scheduling (provisioning) algorithm. More advanced scheduling supported via Haizea.
- Integration with libvirt via shell scripts
- MAD (Middleware Access Drivers) as a means to abstract the underlying technology (hypervisor, file transfer, etc)
Shell scripts/Model/Patterns
- Front end in C++, Exec Nodes/Drivers in Ruby/bash scripts/libvirt/shell
- Drivers for EC2, Elastic Host

Non Functional:
- Active project
- Free and open source

What is the novelty described in this document?
Multi-provider, federated virtual machine management

Relevance to SLA@SOI:
SLA@SOI is currently not addressing aspects of federation infrastructure or the scheduling approach (image transfer). OpenNebula could present interesting techniques to implement these.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
At the time of writing, version 2.2.1 of OpenNebula is available under the Apache License, Version 2.0, whilst the release of version 3.0 is imminent.

References:
[1] OpenNebula, see http://www.opennebula.org/
[6] Reservoir, see http://www.reservoir-fp7.eu/

3.63 Amazon Elastic Compute Cloud (Amazon EC2)

Abstract / Summary:
Amazon Elastic Compute Cloud (Amazon EC2) [1] is a web service that provides resizable compute capacity in the cloud. It provides customers complete control of their computing resources hosted on Amazon's computing environment. EC2 has matured into perhaps the most significant and substantial implementation of a hosted service oriented infrastructure (SOI) in the marketplace today. EC2 adopts a “Paying for What You Use” model.

Described requirements

Functional:
EC2 provide web service interfaces to launch instances with a variety of operating systems, load them with the custom application environment, manage the network's access permissions and run the image using as many or as few systems as desired.

EC2 provides a means to create and upload the virtual machine image (known as Amazon Machine Image – AMI) containing the applications, libraries, data and associated configuration settings. Or users may use the pre-configured and templated images provided by EC2.

The configurations are categorised using a “T-shirt” model, (small, large and extra-large), with standard, micro, high memory, high CPU, cluster compute and cluster GPU instances available. The standard instance ranges from small instance of 1.7GB memory, 1 EC2 compute unit, 160GB storage on a 32 bit platform to extra large instances of 15GB memory, 8 EC2 compute units, 1690 GB of instance storage on a 64 bit platform.

High-memory instances offer up to 68.4GB of memory. High-CPU instances are available containing up to 20 EC2 compute units [2].

The operating systems currently available include Red Hat Enterprise Linux, Windows Server 2003/2008, Oracle Enterprise Linux, OpenSolaris, openSUSE Linux, Ubuntu Linux, Fedora, Gentoo Linux and Debian.

Non Functional:

The EC2 services are offered under a common Service Level Agreements (SLA). Amazon states that it will maintain the level of service with at least 99.95% uptime during the service year. In the event that this SLA is violated, the customers will be eligible to receive a service credit.

What is the novelty described in this document?

Using SLA@SOI terminology, Amazon EC2 provides only a single SLA template that offers different choices with different hardware configurations. There is no means by which Amazon EC2 can perform SLA negotiations in any automated fashion and it is very much a “what you buy is what you get” scenario.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

The adoption of SLAs by EC2 is neither comprehensive, seamless nor automated and thus SLA@SOI as envisioned has the opportunity to demonstrate significant progress in this area.

From an SLA@SOI perspective, it should not be assumed that the infrastructure or platform providers are only internal. Adopting EC2 as a 3rd party provider would make this project more relevant, potentially creating a SLA layer for those infrastructures providers that do not have a good SLA foundation.
Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

Amazon EC2 is a stable, commercially available service, with extensive support for developers/customers [3].

Eucalyptus is an open-source hosting framework with an EC2 compatible interface. It offers hosting alternatives for customers of EC2.

References:
[1] Amazon EC2, see http://aws.amazon.com/ec2/
[4] Eucalyptus, see http://eucalyptus.cs.ucsb.edu/

3.64 XCalibre Flexiscale

Keywords:
Virtual Server

Abstract / Summary:

XCalibre Flexiscale provides computing infrastructure resources in a “Pay As You Go” model. It provides a true utility pricing model with no lock-in. It claims to provide all the power or storage resources needed in less than a minute.

Described requirements

Functional:

Flexiscale provides self-provisioning of Virtual Dedicated Servers (VSD) via the Control Panel or API, which allows customers to start/stop/delete VDS and to change memory/cpu/storage/IPS. The API presents a SOAP/XML web services interface for integration.

The operating systems currently available from Flexiscale include Windows Standard Server 2003, CentOS, Debian and Ubuntu. The memory offer ranges from 0.5 GB to max 8GB. The persistence storage is based on a fully virtualised high-end SAN/NAS back-end. XEN is used as the virtualisation platform.

Flexiscale uses standard ISO boot images, and also allows the customer to provide their own boot images. The latest API version FlexiScale API 0.5 Reference available does not provide a means to upload images yet.

Non Functional:
The Flexiscale services are offered under a common Service Level Guarantee. Flexiscale states that nodes will be available for 100% of each calendar month, excluding any live-recovery. A live-recovery commitment describes an automatic restart within 15 minutes of a node failure.

**What is the novelty described in this document?**

Using SLA@SOI terminology, Flexiscale provide only a single SLA template that offers different choices with different hardware configurations. There is no means by which Flexiscale can perform SLA negotiations in any automated fashion.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

The adoption of SLAs by Flexiscale is neither comprehensive, seamless nor automated at this stage and thus SLA@SOI as envisioned has the opportunity to demonstrate significant progress in this area.

From an SLA@SOI perspective, it should not be assumed that the infrastructure or platform providers are only internal. Adopting Flexiscale as a 3rd party provider will make this project more relevant, potentially creating a SLA layer for those infrastructures providers that do not have a good SLA foundation.

However, due to the limited features exposed by the Flexiscale API, deployment of virtual images in an automated does not appear possible.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

Xcalibre Flexiscale is a commercially available service, with extensive support for developers/customers [2].

**References:**

[2] Flexiscale API, see [https://api.flexiscale.com](https://api.flexiscale.com)

### 3.65 ElasticHosts

**Keywords:**

Virtual Server

**Abstract / Summary:**

ElasticHosts offers a flexible and scalable infrastructure at competitive prices that allow customers to instantly add capacity for growth or peaks on demand. ElasticHosts provides two pricing schemes, by monthly subscription and by hourly burst used. Customers have complete control to choose the operating system, applications and configuration of your virtual servers, and can even upload and boot your own custom disk images.
ElasticHosts virtual servers are designed, configured and supported for all forms of web hosting, unlike Amazon EC2 which is a general purpose solution not well-tailored for web hosting.

**Described requirements**

**Functional:**
ElasticHosts provide a web interface to manage the virtual servers. The web interface is complemented by a HTTP API and a command line tool. The HTTP API allows users to create drives, upload and download drive images and create and control virtual servers on ElasticHosts infrastructure. The API works in a REST style, and ElasticHosts also provides a simple command line tool and a drive upload tool for Unix or Windows Cygwin users to control the infrastructure from users’ own scripts without writing any code.

ElasticHosts provide pre-installed images of Debian and Ubuntu. ElasticHosts also provide pre-installed software provided by CohesiveFT. You may also upload your own installation media or upload the raw hard drive image from an existing physical or virtual server running any supported operating system. ElasticHosts support p2v, v2v and v2p migration to and from the ElasticHosts cloud.

ElasticHosts provide different configuration sizes ranging from basic to high-end configurations with up to 20000 core-Mhz, 8 GB memory, 2TB disk. ElasticHosts is using Linux KVM for the virtualisation platform.

**Non Functional:**
The ElasticHosts virtual servers are physically located in the UK and US. UK hosted customer images are within the jurisdiction of EU data protection laws.

ElasticHosts offer a service level agreement that includes 100% availability in every calendar month, but with various exceptions. These include previously announced maintenance, force-majeure, and issues with the customer’s software. Credits are awarded in the event of downtime exceeding 15 minutes.

**What is the novelty described in this document?**
ElasticHosts provides arbitrarily sized VMs, under various payment models, and with true static IP addresses.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
ElasticHosts does not provide automated negotiation, automatic detection of SLA violations, and claims must be submitted manually. SLA@SOI as envisioned has the opportunity to demonstrate significant progress in these areas.
From an SLA@SOI perspective, it should not be assumed that the infrastructure or platform providers are only internal. Adopting ElasticHosts as a 3rd party provider will make this project more relevant, potentially creating a SLA layer for those infrastructures providers that do not have a good SLA foundation.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

ElasticHosts is available for hosting at the time of writing, with online support for developers and customers [2].

References:
[1] ElasticHosts, see http://www.elastichosts.com/products/

3.66 ServePath GoGrid

Keywords:
Cloud, Compute Capacity

Abstract / Summary:
ServePath GoGrid is a multi-tier, cloud computing platform that allows customers to manage their cloud hosting infrastructure. GoGrid is ServePath’s cloud hosting service with custom-build dedicated servers to create a hosted server network that can scale to meet seasonal or sudden spikes of internet traffic while providing the high I/O and CPU performance demanding database servers require. GoGrid provides two pricing models, pay-as-you-go and pre-paid plan (monthly subscription), paying only for the deployed RAM and data transfer customers use.

Described requirements

Functional:
GoGrid provides a REST-like API query interface to programmatically control the cloud hosting infrastructure. The GoGrid API specification is available under a Creative Commons Sharealike license. Tools based on this OpenSpec standard can be used to control the GoGrid platform or any other compatible cloud computing providers.

Each GoGrid account includes free hardware load balancing.

GoGrid supports a variety of preconfigured Linux and Windows operating systems including Windows Server 2008, Windows Server 2003, Centos and Redhat Enterprise Linux. Each server image is provided with preinstalled software such as Apache, IIS, MySQL. GoGrid has started to provide a means for users to create or upload custom images – this feature is currently in Beta.
GoGrid provides various sizes of dedicated servers, with from 4 to 8 cores, 8 to 24 GB of memory and various hard disk and RAID configurations.

**Non Functional:**
GoGrid claims [1] to provide a common “10,000%” Guaranteed and 100% Uptime Service Level Agreement covering Server Uptime, Persistent Storage, Network Performance, Cloud Storage, Server Reboot, Support Response Time, Domain Name Services, Physical Security, and 24 x 365 Engineering Support. Customers will be eligible to receive a service credit if the SLA is violated.

**What is the novelty described in this document?**
Using SLA@SOI terminology, GoGrid provides only a single SLA template. There is no means by which GoGrid can perform SLA negotiations in any automated fashion. SLA Monitoring is not automatic – violations have to be manually escalated within 48 hours of the start of the failure.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
The adoption of SLAs by GoGrid is not comprehensive and not automated at this stage and thus SLA@SOI as envisioned has the opportunity to demonstrate significant progress in this area.

From an SLA@SOI perspective, it should not be assumed that the infrastructure or platform providers are only internal. Adopting GoGrid as a 3rd party provider will make this project more relevant, potentially creating a SLA layer for those infrastructures providers that do not have a good SLA foundation.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**
GoGrid is a commercially available service, with online support for developers/customers [2].

**References:**
[1] GoGrid, see http://www.gogrid.com/

**3.67 Zimory**

**Keywords:**
Cloud, Compute Capacity

**Abstract / Summary:**
Zimory [1] technology claims to enable the most efficient use of infrastructure resources – harnessing heterogeneous virtual servers across one or many data centers to build an internal computing cloud.

Zimory Enterprise Cloud combines virtual servers from multiple customers into a single homogeneous computing cloud. Depending on customers’ current resource requirements, they can move applications quickly within a data center as well as between various locations.

Zimory Public Cloud brings together supply and demand. Customers can use Zimory to sell server capacities they do not need for a certain time period to other companies. With Zimory Public Cloud customers get additional revenue and reduce the operating cost of their data center. Customers may also rent short-term capacity from other providers.

Zimory is based on a native pay-as-you-go charge model.

**Described requirements**

**Functional:**

Zimory does not provide any obvious APIs or Web Services. It appears that customers access their platform via their Cloud Portal control panel.

Zimory currently support VMWare and XEN for the virtualization technology. Zimory supports standard linux operating system distributions including Ubuntu and CentOS. Zimory allow users to upload XEN and VMWare images through the Cloud Portal control panel in a manual manner.

Zimory limits the number of vCPUs to no more than 2x the number of physical cores. You can currently have up to 4 vCPUs in a single deployment. Zimory offer the memory from a range of 128MB to maximum 4GB. The virtual disk storage size is limited to 80GB.

**Non Functional:**

Zimory have press releases referring to service level agreements (SLAs) in three levels, Gold, Silver and Bronze. A Gold SLA cloud delivers the strongest quality standards. This includes availability and security standards. The providers offering these resources are compliant with all relevant security certifications. A Silver SLA offers high availability and security standards. The providers are known brands. A Bronze SLA delivers the usual quality and availability standards of hosting providers. It does not contain certifications and additional security offerings.

**What is the novelty described in this document?**

Using SLA@SOI terminology, GoGrid provides only a single SLA template. There is no means by which GoGrid can perform SLA negotiations in any automated fashion.
How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

The approach to SLAs provided by Zimory is not explicit. Zimory broadly describe three levels of SLA but the details of what these levels actually represent is not clear at the time of writing.

The adoption of SLAs by Zimory is not holistic at this stage and thus SLA@SOI as envisioned has the opportunity to demonstrate significant progress in this area.

From an SLA@SOI perspective, we should not assume that the infrastructure or platform providers are only internal. Adopting Zimory as a 3rd party provider will make this project more relevant.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

Zimory is commercially available.

References:
[1] Zimori, see http://www.zimory.com/

3.68 Nirvanix SDN

Keywords:
Cloud, Storage Capacity

Abstract / Summary:
Nirvanix's Global Storage delivery Network (SDN) provides intelligent, policy based Cloud Storage for businesses that require data security, availability and Enterprise level support [1]. The SDN is powered by Nirvanix's patent pending Internet Media File System (IMFS), a clustered file system that includes all of Nirvanix's globally distributed storage nodes under one namespace. This allows for significant scalability and data availability by leveraging geo-location intelligence that stores and delivers data to a requester based upon proximity to the closest node.

Nirvanix Applications includes the Nirvanix CloudNAS which claims to offer a fast, secure, and easy way to gain access to the benefits of Cloud Storage.

Described requirements

Functional:
Nirvanix SDN is available via the Nirvanix Web Services API, SOAP/REST interfaces via HTTP or HTTPS [2].

Nirvanix provides unlimited storage on demand worldwide.
Nirvanix supports dynamic load-balancing of traffic for best-user experience during peak loads.

Nirvanix supports intelligent routing of files to the closest global location for accelerated performance.

Non Functional:

Nirvanix provides 99.9%-100% uptime guarantee depending on business needs. Customers are eligible to receive a service credit if Nirvanix fail to meet the minimum level of service.

What is the novelty described in this document?

Using SLA@SOI terminology, Nirvanix provide only a single SLA template. There is no means by which Nirvanix can perform SLA negotiations in any automated fashion.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

The adoption of SLA by Nirvanix is not holistic and not automated at this stage and thus SLA@SOI as envisioned has the opportunity to demonstrate significant progress in this area.

From an SLA@SOI perspective, we should not assume that the infrastructure or platform providers are only internal. Adopting Nirvanix as a 3rd party provider will make this project more relevant.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

Nirvanix is commercially available, and offers online developer/customer support [2].

References:

[1] Nirvanix, see http://www.nirvanix.com/

3.69 Ganglia

Abstract / Summary:
Ganglia [1] is a scalable distributed monitoring system designed for high-performance computing systems (clusters and grids). It is built upon open and widely used technologies, such as XML, for data representation and XDR (External
Data Representation) for efficient data transport. Round-robin database (RRDtool) [2] is used for storing historical data at different time scales. It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency.

The Ganglia monitoring system has been ported to many operating systems and processor architectures and is currently deployed in thousands of clusters around the world.

The core of the Ganglia monitoring system consists of three modules:

**gmond** (Ganglia monitoring daemon) is a multi-threaded data collector daemon, installed on every cluster node that needs to be monitored. It collects basic performance metrics (CPU, memory, storage, network and process). Relevant changes are either multicasted or sent to a number preconfigured servers. Each gmond daemon also listens for multicasts from other gmonds in the same cluster and answers requests for an XML description of the the cluster state (this is typically requested by gmetad). The entire cluster state is stored in a multi-threaded in-memory hash table. The state of the cluster is very compact making it possible for gmonds to capture entire state without excessive need of memory. The hash table only stores the current state of the cluster. Therefore, it is up to other components of the monitoring system to store historical data. Although gmond uses thresholds to determine relevant changes, it is not possible to receive notifications via e-mail, sms, etc.

The state of the node is transmitted in compact External Data Representation (XDR) [3]. On the other hand, XML over a TCP connection is used for the description of the entire cluster state (this information is only polled by other services).

In order to detect node availability heartbeat messages are sent periodically. These messages are also used for automatic discovery of new nodes and detection of restarts. The advantage of using heartbeat messages is that there is no need for a configuration of the network structure in advance. Thus, it is suitable for dynamic clusters. On the other hand, since there is no network structure, the Ganglia monitoring system cannot detect nodes that became unavailable before the collection process started.

**gmetad** is a daemon that collects monitoring information from other gmetad and/or gmond services and stores their state in an indexed round-robin databases. It provides simple query mechanism for collecting historical information. gmetad services can be configured in and arbitrary hierarchy.

**web front-end** is a graphical web interface to the historical data stored by gmetad. The presentation of data is hierarchal making it easy to see the state of the cluster/Grid as a whole and drill down to specific nodes.

**Described requirements**

**Functional:**
- supports different monitoring metrics (CPU, network, disk, memory etc.) and new metrics can be added.
- ability to monitor servers, devices and applications
- automatic discovery of nodes (heartbeat messages)
- simple configuration

**Non Functional:**
- negligible per-node overhead
What is the novelty described in this document?
The designed architecture enables some important features needed for good quality of monitoring in distributed systems. The Ganglia approach avoids pooling clients for data collection which is a great improvement on monitoring systems for decentralized systems.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The monitoring system is an important part of the SLA@SOI project, because many decisions are made on collected data. Ganglia is already used as one of the supported monitoring engines of the SLA@SOI infrastructure monitoring component.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
A stable version of Ganglia can be downloaded from http://ganglia.info/ under the BSD License. There is extensive online documentation [4].

References:

3.70 SLA Evaluator (GT4 & .NET versions)

Keywords:
Monitoring

Abstract / Summary:
SLA Evaluator is a web service that evaluates whether metrics collected from execution hosts where a service/application is running deviate from target metrics and guarantees expressed in an SLA about the infrastructure that is expressed in WS-Agreement. The system is intended to provide a Grid provider with the capability of monitoring SLAs established with end-user. It also supports the configuration and initiation of an SLA.

Described requirements:
2.0, and WSE 3.0 for its deployment. The described properties of the framework are as follows:

- It produces notifications of violations of WS-Agreement guarantee terms by execution hosts (grid) metrics, and initiations/terminations of SLAs.
- It enables the configuration, initiation and termination of SLAs.
- It supports monitoring against single grids.
- Notifications are expressed in WS-Notification.
- The GT4 version is available on Windows and Linux under an Apache licence.
- The .NET version is available on Windows.

**Functional properties:**
SLA Evaluator does not support the monitoring of SLA guarantee terms for functional properties or properties of software services.

**Non Functional properties:**
Supports monitoring of infrastructure metrics.

**What is the novelty described in this document?**
No particular features making SLA Evaluator novel against competition have been identified.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
SLA Evaluator could potentially be plugged into the SLA@SOI monitoring platform. This would require the development of a reasoning component gateway (RCG) that would provide an adaptor of its interface to that of the SLA@SOI monitoring platform and a parser to translate SLAs expressed in SLA(T) into the SLA Evaluator's internal language. The RCG should also include a translator of WS-Notifications into the monitoring events language of the SLA@SOI monitoring platform.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**
Beta level system.

**References:**
3.71 Nagios

Keywords:
Monitoring

Abstract / Summary:
Nagios is a host, service and network monitoring program. A central Nagios server periodically performs checks by polling each device across a network connection which could be more resource and bandwidth intensive. Nagios was not designed to be a Grid monitoring system, although it can be used to monitor the operation of grid services in the same way as other network services.

Described requirements
Nagios watches specified hosts and services, alerting the user when things go bad and when they get better.

Functional properties
By default it does not perform checking of functional properties, but allows users to easily develop their own service checks through plugin mechanisms.

Non Functional properties:
Nagios monitors network services (SMTP, POP3, HTTP, NNTP, PING, etc.) and host resources (processor load, disk usage, etc.).

What is the novelty described in this document?
Nagios is mature and well supported. It allows users to create their own plugins for extending its features and many plugins are now available.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Nagios is used as one of the supported monitoring engines of the SLA@SOI infrastructure monitoring component. A wrapper named IMonitoringEngine has been provided and integrated inside monitoring component, which allows different monitoring engines to be used a backbone monitoring system for an SLA@SOI infrastructure monitoring.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
It is a commercial tool and can be downloaded from its website [1].

References:
[5] Nagios, see http://www.nagios.org/
[7] Nagios forge, see http://www.nagiosforge.org
3.72 **Groundwork - Monitor Community Edition**

**Keywords:**
Monitoring

**Abstract / Summary:**
GroundWork Monitor Community Edition is a monitoring solution that enables users to maintain network visibility and control [1]. It uses the results from 15 other open source projects (Nagios, Ganglia, Cacti, RRDtool etc) to capture, store, evaluate and present the status, events and performance of monitored devices. Nagios is the core component of the GroundWork monitor.

Since GroundWork Monitor Community Edition includes both Nagios and Ganglia it is suitable for monitoring both:
- many aspects of a smaller number of different devices (Nagios features [2])
- a limited number of aspects of a large number of identical devices (Ganglia features [3])

**Described requirements**
Groundwork watches specified hosts and services, alerting users when things go bad and when they get better.

**Functional properties**
By default GroundWork does not perform checking of functional properties, but allows users to easily develop their own service checks trough plugin mechanisms.

**Non Functional properties**
GroundWork monitors network services (SMTP, POP3, HTTP, NNTP, PING, etc.) and host resources (processor load, disk usage, etc.)

Recommended Minimum Hardware for a small installation, up to 150 devices: 2 GB RAM, 1 CPU, 2.8 GHz P4 or better, 80 GB disk. Software is available for: Red Hat Linux Enterprise Linux 5, Red Hat Linux Enterprise Linux 4, CentOS 5, Novell SuSE Linux Enterprise, Server (SLES) 10, Ubuntu 6.06 Server LTS, Debian 4

What is the novelty described in this document?
As many commercial monitoring tools nowadays available on market, it does not have particular features that distinguish itself from other network monitoring programs. However, notice that, since GroundWork Monitor Community Edition uses the results from 15 other open source projects, it takes advantage of individual features provided by the different tools is based on.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

GroundWork Monitor Community Edition might be used as one of the monitoring engine within the SLA@SOI framework. It might be used for monitoring especially non functional properties.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

GroundWork Monitor Community Edition can be downloaded from its website. Source code is available under the GPL V2.0 license.

**References:**

[3] Ganglia home page, see http://ganglia.info

### 3.73 MonALISA

**Abstract / Summary:**

MonALISA [1], which stands for Monitoring Agents using a Large Integrated Services Architecture, has been developed over the last four years by Caltech and its partners with the support of the U.S. CMS software and computing program.

The MonALISA framework provides a distributed monitoring service system using JINI/Java and WSDL/SOAP technologies. Each MonALISA server acts as a dynamic service system and provides the functionality to be discovered and used by any other services or clients that require such information. Services are registered with a lookup service. The client application connects directly with each service it is interested in for receiving monitoring information.

The framework provides monitoring information from large and distributed systems to a set of loosely coupled higher level services, that in turn analyse the information, and take corrective action to improve the overall efficiency of operation of the Grid, e.g., load balancing. The services are managed by an efficient multi threading engine that schedules and oversees their execution.

It can integrate existing monitoring tools and procedures and to provide this information in a dynamic way to any other services or clients. Internal data collection engine uses SNMP for monitoring compute nodes and network links, routers and switches. External modules can be loaded for interfacing with existing monitoring tools, e.g., Ganglia. Collected values are stored in a relational database, locally for each service. Older data are compressed by evaluating mean values on larger time intervals.
Described requirements

Functional:
- Monitoring all aspects of complex systems including:
  - System information for computer nodes and clusters.
  - Network information (traffic, flows, connectivity, topology) for WAN and LAN.
  - Monitoring the performance of Applications, Jobs or services.
  - End User Systems, and End To End performance measurements.
  - Secure, remote administration for services and applications.
  - Graphical User Interfaces to visualize complex information.

Non Functional:
- Global monitoring repositories for distributed Virtual Organizations.

What is the novelty described in this document?
The MonALISA monitoring system provides modules for visualisation techniques presenting network topology or statistics and for interfacing with other existing systems.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The MonALISA framework is based on Dynamic Distributed Service Architecture and is able to provide complete monitoring, control and global optimization services for complex systems as SLA@SOI.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

MonALISA is well-established and can be downloaded from http://monalisa.cern.ch/monalisa.html and appears to have a strong academic community supporting it. The MonALISA license is available from http://monalisa.cacr.caltech.edu/monalisa__Download__ml_license.html.

References:
[1] MonALISA, see http://monalisa.cern.ch/monalisa.html
  CHEP 2003, La Jola, California, March 2003
[3] A Distributed Agent-based Architecture for Dynamic Services
  Harvey B. Newman, Iosif C. Legrand, Julian J. Bunn
  CHEP 2001, Beijing, Sept 2001
  CHEP 2004, Interlaken, Switzerland, September 2004
3.74 Zabbix

**Keywords:**
Monitoring

**Abstract / Summary:**
ZABBIX is an enterprise-class open source distributed monitoring solution, fully featured and extensible.

ZABBIX is software that monitors numerous parameters of a network and the health and integrity of servers. ZABBIX uses a flexible notification mechanism that allows users to configure e-mail based alerts for virtually any event. This allows a fast reaction to server problems. ZABBIX offers excellent reporting and data visualisation features based on the stored data. This makes ZABBIX ideal for capacity planning. It also allows definition of simple SLAs and tracks their state and history.

ZABBIX supports both polling and trapping. Its flexible architecture (2 or 3 layers) with optional Master agents and Proxies allows low network load and efficiency in massive setups, up to several thousand nodes. All ZABBIX reports and statistics, as well as configuration parameters, are accessed through a web-based front end. A web-based front end ensures that the status of the network and the health of the servers can be assessed from any location. Properly configured, ZABBIX can play an important role in monitoring IT infrastructure. This is equally true for small organisations with a few servers and for large companies with a multitude of servers.

ZABBIX is free of cost. ZABBIX is written and distributed under the GPL General Public License version 2. It means that its source code is freely distributed and available for the general public. Both free and commercial support is available and provided by ZABBIX Company.

ZABBIX consists of several major software components:
- ZABBIX Server – is a main component, where data is collected and stored
- ZABBIX Proxy - is a lightweight process, which collects data collection on behalf of ZABBIX Server.
- ZABBIX Agent – collects data on local machine
- ZABBIX web interface - allows easy access to the monitoring data and then configuration of ZABBIX from anywhere and from any platform.

**Described requirements**

**Functional:**
- centralized or decentralized inquiry
- supports all common monitoring metrics (CPU, network, disk, file checksums etc.), also SNMP and IPMI based and supports custom extensions.
- Real-time SLA reporting
- Network monitoring: supports monitoring of Cisco, Juniper, 3Com, Nortel, Foundry and other routers as well as firewalls from Netscreen, Fortinet, Cisco or Checkpoint. Any network devices (routers, hubs, printers, etc.) having SNMP support can be easily monitored by ZABBIX.
- Web interface for configuration

Non Functional:
In the ZABBIX manual are mentioned hardware and software requirements for ZABBIX. The server requires following system resources and hardware configurations:

<table>
<thead>
<tr>
<th>Name</th>
<th>Platform</th>
<th>CPU/Memory</th>
<th>Database</th>
<th>Monitored hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>Ubuntu Linux</td>
<td>PII 350 MHz, 256MB</td>
<td>MySQL MyISAM</td>
<td>20</td>
</tr>
<tr>
<td>Medium</td>
<td>Ubuntu Linux 64 bit</td>
<td>AMD Athlon 3200+, 2GB</td>
<td>MySQL InnoDB</td>
<td>500</td>
</tr>
<tr>
<td>Large</td>
<td>Ubuntu Linux 64 bit</td>
<td>Intel Dual Core 6400, 4GB, RAID10</td>
<td>MySQL InnoDB or PostgreSQL</td>
<td>&gt;1000</td>
</tr>
<tr>
<td>Very large</td>
<td>RedHat Enterprise</td>
<td>Intel Xeon, 2x CPU 8GB, Fast RAID 10</td>
<td>MySQL InnoDB or PostgreSQL</td>
<td>&gt;10000</td>
</tr>
</tbody>
</table>

What is the novelty described in this document?
ZABBIX is a monitoring system that provides many useful features that are common to other systems, but there are all packed together. Also ZABBIX provide excellent user guide which is easy to read and understand so installing and configuring ZABBIX monitoring system should not be a problem. ZABBIX allows users to define an SLA and have ZABBIX keep track of the expected SLA and actual SLA. IT Services are defined as groups of triggers and can be configured to calculate the minimum of a group or maximum of a group.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
ZABBIX monitoring system would be easy to install in the SLA@SOI platform because of it’s well produced description and documentation. Centralised monitoring and network management with ZABBIX should also be very useful.
Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

ZABBIX stable version 1.8 can be downloaded under GPL. See http://www.zabbix.com/ for download and http://www.gnu.org/copyleft/gpl.txt for full licence.

References:
[1] ZABBIX, see http://www.zabbix.com/

3.75 Osmius – The Opensource Monitoring Tool

Abstract / Summary:

Osmius [1] is a very general purpose open source distributed monitoring solution. With its 3 layer architecture, low network load and basic SLA support it is very similar to Zabbix. It is more extendable and offers Java Management API, but on the other side it is still under development.

Described requirements

Functional:
- Monitoring all aspects of complex systems including:
  - System information for computer nodes and clusters.
  - Network information (traffic, flows, connectivity, topology) for WAN and LAN.
  - Common operationgs systems, Databases, Web servers, SNMP, LDAP, etc.
  - Monitoring the performance of Applications, Jobs or services.
  - End User Systems, and End To End performance measurements.

Non Functional:
- Secure, remote administration for services and applications.
- Graphical User Interfaces to visualize complex information.
- Aggregation of old data to save storage space
- Extendable with Osmius Development Framework
- Management Java API
- Flexible architecture allowing large scale deployment

What is the novelty described in this document?

Osmios is a flexibly designed general purpose monitoring system. Its extensibility and manageably allow a seamless integration into existing information systems.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The Osmius framework is based on Dynamic Distributed Service Architecture and is able to provide complete monitoring, control and global optimization services for complex systems as SLA@SOI. Its basic SLA support and extensibility features allow a seamless integration.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

Osmius is relatively new, but has already received some appreciation. It is an opensource project and can be downloaded from [1]. It is licenced under GPL v2.

References:
[1] OSMIUS, see http://www.osmius.net/

3.76 XMPP – Extensible Messaging and Presence Protocol

Keywords:
Messaging

Abstract / Summary:
The XMPP Standards Foundation describe XMPP as a “set of open XML technologies for presence and real-time communication developed by the Jabber open-source community in 1999, formalized by the IETF in 2002-2004, continuously extended through the standards process of the XMPP Standards Foundation, and implemented in a wide variety of software, devices, and Internet services” [1].

XMPP could provide a scalable, distributed messaging platform for the management of infrastructure in SLA@SOI.

Described requirements

Functional:
- Comprehensive support for near-real-time messaging, presence and request-response services.

Non-Functional:
- Authentication and confidentiality features are built into the specification.
- XMPP is highly extensible, as illustrated by RFC 3921 [2]

What is the novelty described in this document?
XMPP provides a mature, distributed, comprehensive and highly extensible messaging protocol. Particularly interesting extensions include Adhoc-Command [3] and IO-Data XEPs [4].
How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

XMPP could potentially be extended for use as the messaging protocol for management of infrastructure.

- XMPP’s open protocol and XEP mechanism ensure flexibility
- The PubSub concept is powerful and allows the transport of opaque payloads
- XMPP has widespread support from other messaging solutions – including JMS and many ESBs
- Multi User Chatroom (MUC) and direct one-to-one messaging provide a powerful means of debugging, offering “command line” interaction with XMPP nodes
- XMPP appears to scale well
- S2S is very attractive from the point of view of enabling resource federation.

Some potential challenges have been identified with XMPP that require further investigation:

- There appears to be no “all-in-one” decentralised (as in peer-to-peer or super-peer type) implementation of XMPP, although the specification allows for such interactions (S2S). Vertebra[5] may satisfy this need - it builds upon ejabberd [6] to accomplish this.
- XMPP PubSub XEP implementations have inconsistencies (core features and XEP coverage)[7][8] and there are limited Java client libraries available.
- Routing and workflow management features appear to be lacking in XMPP.

XMPP is already used in the SLA@SOI messaging library which is used overall in the framework.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

There are many open-source implementations of XMPP available in a variety of languages. These include clients, servers and libraries. Mature implementations include OpenFire [9] L and eJabberd [6], both published under GNU GPL Version 2. See [9] for a comprehensive list.

References:

[1] XMPP Standards Foundation, see [http://www.xmpp.org](http://www.xmpp.org)
3.77 Java Message Service (JMS)

Keywords:
Messaging, Java

Abstract / Summary:
The Java Message Service (JMS) API is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients. JMS is a part of the Java Platform, Enterprise Edition, and is defined by a specification developed under the Java Community Process as JSR 914 [1].

Enterprise messaging provides a reliable, flexible service for the asynchronous exchange of critical business data and events throughout an enterprise. The JMS API adds to this a common API and provider framework that enables the development of portable, message based applications in the Java programming language.

Described requirements

Functional:
- **JMS provider** which implements JMS interfaces and provides administrative and control features.
- **JMS clients** which are the programs or components that produce and consume messages.
- **Messages** which are the objects that communicate information between JMS clients.

Non Functional:
- Authentication and confidentiality

What is the novelty described in this document?
JMS API is a messaging standard that allows application to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
JMS provides API that can be used to implement messaging bus that SLA@SOI framework uses for communication between its components.
Model of particular interest that JMS provides is publish/subscribe model which supports publishing messages to a particular message topic. Subscribers may register interest in receiving messages on a particular message topic.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

The Reference Implementation (RI) of the Java Message Service (JMS) API is available as part of the Java 2 SDK, Enterprise Edition, version 1.3 [2] under license listed on page [3]. Maturity level is stable.

**References:**

### 3.78 AMQP – Advanced Message Queuing Protocol

**Keywords:**
Messaging

**Abstract / Summary:**
AMQP [1] is an open-standard wire-level messaging protocol for message oriented middleware. It was initially designed to support enterprise messaging requirements and addresses message orientation, queuing, routing (including publish-subscribe), reliability and security.

It could be adopted as the messaging protocol for the management of infrastructure in SLA@SOI.

**Described requirements**

**Functional**
AMQP can support point-to-point, fanout, publish-subscribe and request-response messaging paradigms.

**Non-Functional:**
Non-functional requirements supported include security, clustering, and federation. It includes the ability to resume interrupted transfers and supports transport through firewalls.

**What is the novelty described in this document?**
By defining the protocol at the wire-level in terms of octet-streams, AMQP supports complete interoperability between arbitrary systems (hardware, operating systems, software languages and libraries).
How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

AMQP could be used as the transport layer for higher level messaging standards such as Web service SOAP calls, guaranteeing the transfer of these messages.

Among its relevant features is its OS, hardware and software agnostic, it is an open standard, it can complement JMS, and it supports the Publish/Subscribe paradigm as well as Quality of Service.

Beside XMPP also AMQP implementation has been provided for the SL@SOI messaging library, which is a messaging system used overall in the framework.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

An open-source AMQP implementation – Qpid [2] - is being incubated by the Apache Foundation – it may or may not be promoted to a fully supported project. It includes a JMS compatible messaging server and a broker both written in Java, and APIs in C++, Java, C#, Ruby and Python. Qpid is published under the Apache License V2.0.

ZeroMQ is a particularly high-speed messaging layer that supports AMQP and exposes Java APIs among others. It is licensed under LGPL V3.0.

References:
[1] AMQP, see http://www.amqp.org
[3] ZeroMQ, see http://www.zeromq.org

3.79 CORTEX (CO-operating Real-time senTient objects: architecture and EXperimental evaluation)

Keywords:
Collaborating Objects

Abstract / Summary:
CORTEX is an IST Programme RTD Research Project, launched in April 2001 with a duration of 3 years. Its overall objective is to investigate the theoretical and engineering issues necessary to support the use of sentient objects in order to build large-scale proactive applications. A sentient object is a mobile, intelligent software component that is able to sense its environment via sensors and react to sensed information via actuators.

The programming model has to address any issues arising in environments built of networked components that will act autonomously in response to a myriad of
events and which have to affect and control the surrounding environment in order to operate independently from the human control.

Sentient objects are the basic building blocks of applications developed following the CORTEX programming model. Sentient objects must be able to discover and interact with each other and with the physical world in ways that demand predictable and sometimes guaranteed quality of service (QoS), encompassing both timeliness and reliability guarantees.

There are three major entities in the sentient object model:
- **Sensor**: entity that produces software events in reaction to a real-world stimulus detected by some world hardware device.
- **Actuator**: entity that consumes software events, and reacts by attempting to change the state of the real world in some way via some hardware device.
- **Sentient object**: entity that can both consume and produce software events, and lies in some control path between at least one sensor and one actuator.

The following picture gives a more detailed view of the sentient objects organisation in a CORTEX model application.

The most important feature of a sentient object is that it implements the control logic. This control logic works on stimulus coming from the external environments. The importance of the external environment events and states make context-awareness a key factor for the sentient object.

Three main components implement context-awareness in the sentient object:
- **Sensory Capture**: integrates the different events coming from sensors and filters them to limit noise and errors coming from the environment.
- **Context Representation**: transform raw data in a format that is useful for the sentient object
- **Inference engine**: implements the reasoning capability of the sentient object which has to be able to take the appropriate decision based on the incoming inputs. Such engine is based on a knowledge-base built by a set
of production rule (CORTEX adopts CLIPS as declarative language to specify rules and integrates it with Context Based Reasoning mechanisms).

Coordination mechanisms implemented by sentient objects are based on interaction through the environment, inspired to the behaviour of colonies of insects.

CORTEX ad hoc network target imposes limitations to the design of the event service, considering the lack of a network infrastructure. The event model adopted by CORTEX is STEAM which addresses a number of core issues for publish/subscribe framework. The key characteristic of the model is that it doesn’t require any event broker: brokering functions are implemented both at consumer and producer side.

The key hypothesis which has driven STEAM model is that in a pervasive environment with high mobility, entities are most likely to interact if they are in close proximity. So the rule is: closer consumers are located to a producer the more likely they are to be interested in the events propagated by the producer. This rule limits the forwarding of the event messages, reducing the usage of the communication resources.

Event filters are the main tools to control the propagation of the messages. The novelty of the approach is that subject and proximity filters are applied at the producer side, whilst content filter are applied at consumer side. The significant advantage of this approach is that consumers haven’t to forward content filter to producers when they change their geographical area. This simplifies the dynamic reconfiguration requirements as far as subscription and content filter is regarded.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:

- The programming model must support the development of applications constructed from mobile sentient objects.
- The model needs to take into account the provision of incremental real-time and reliability guarantees.
- The model will develop means to express QoS properties in the model, where QoS is taken as a metric of predictability in terms of timeliness and reliability.
- The middleware generated by the CORTEX must cope with applications that have some or all of the following characteristics:
  - Sentience – the ability to perceive the state of the surrounding environment, through the fusion and interpretation of information from possibly diverse sensors;
  - Autonomy – components of these applications will be capable of acting in a decentralised fashion, based solely on the acquisition of information from the environment and on their own knowledge;
  - Large scale - typical applications may be composed of billions of interacting hardware and software components;
- Time criticality - these applications will typically interact with the physical environment, and will have to cope with its pace, regardless of adverse conditions due to scale and technology shortcomings;
- Safety criticality – typical applications will interact with human users, whose well-being will frequently rely on them;
- Geographical dispersion - unlike current embedded systems, typical applications will integrate components that are scattered over buildings, cities, countries, and continents;
- Mobility – furthermore, they must possess the ability to move between hosts possibly of different networks, while remaining in continuous operation
- Evolution – these applications will have to cope with changing conditions during their lifetimes. Not only must the applications be designed to evolve, but their underlying support must also be adaptable.

Non Functional:
- Use STEAM as event model.

What is the novelty described in this document?

CORTEX is a highly innovative project that is both technically and scientifically challenging. Its major innovation is the novel paradigm of rapidly composable communities of co-operating sentient objects. CORTEX foresees that sentient objects will pervasively be in the future included in almost everything of our daily life. The models, architecture and middleware proposed by the CORTEX project constitute a key enabling technology to realise the vision of ubiquitous computing and drive new business opportunities for the widest community of technology and service providers.

CORTEX supports proactive applications, comprising mobile sentient objects with autonomous behaviour resulting from interactions with other objects and with the physical environment, i.e. driven by sensor inputs, as well as from the internal state of the objects.

Co-operating communities of sentient objects can thereby interact to achieve zones of QoS coverage where predictable behaviour is ensured and graceful degradation and failure modes are possible.

CORTEX has also made a number of significant and novel contributions to the field of distributed systems. More specifically, sentient objects can be mobile and thus break traditional assumptions about object naming associated with fixed locations (e.g. unique identifiers based on network addressing). CORTEX has delivered an object naming abstraction and associated brokerage services that maintain high performance addressing of sentient objects in highly dynamic and ad-hoc configurations.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

The properties of the sentient object, which represents the CORTEX main building block, seems to be very close to the concepts needed for the adjustment functionality in SLA@SOI. In particular, the autonomous behaviour,
reasoning capability across the components, the interaction with the environment and the support of proactive applications in order to control the environment.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

A demonstrator has been built in WP4 to consolidate the results of the work in architecture definition, and to provide a final evaluation of the project's findings. The implementation is not available from the CORTEX web page.

References:

3.80 Real-time reconfiguration for guaranteeing QoS provisioning levels in Grid environments.

Keywords:
QoS Provisioning

Abstract / Summary:
This paper presents an approach for QoS provisioning in SOA, capable of guaranteeing the high QoS demands of real-time business applications, using real-time monitoring information and recommendations based on the past experience. It is customizable for each application since it uses application-specific policies.

This approach takes into account all the parameters included in the SLA template, and it proposes an optimal solution in terms of QoS performance that is also economically fair for the service provider.

The authors propose that the SLAs should not only include information about a service’s quality during reservation, initialization and its usage, but also about the handling of SLA violations and corresponding penalties, mechanisms for SLA monitoring and rules for SLA enforcement and maintenance of the agreed QoS level. This sophisticated approach on the SLAs requires links to specific policies and declarations in the SLAs about which policy should be deployed when the threshold of a metric is reached. This threshold is of major importance since it shows that a violation is about to occur.

The SLAs and consequently the policies, must explicitly define the rules and the actions needed for the QoS provisioning process, especially in real-time aware business environment.

The proposed framework is depicted in the following figure:
Figure 9: Real-time reconfiguration framework

Its main components are:

- **SLA Repository**: store the SLAs for a specific service provider.
- **Policy Repository**: each SLA is associated with policies that include management rules, either for the overall SLA or for particular SLA terms. These rules are defined by the service provider and implemented so as to guarantee the required QoS level or address the specific runtime requirements of each application. Policies can be updated at runtime.
- **Real-time Monitoring**: collects real-time information from each resource of the service provider.
- **Recommendation Components**: The recommendation mechanism consists of two components, the Recommendation and the Recommendation Data Base (DB).
  - The Recommendation DB stores historical information from all previous job executions: job type, resources used, SLA violations occurred, previous decisions of the Supervisor and the policies that were used.
  - The role of the Recommendation Component is to assess the historical data of the Recommendation DB based on the current QoS level status and the monitoring information, and provide a metric for each solution about how the environment responded in the past under approximately the same conditions.
- **QoS Supervisor**: the QoS Supervisor is the main component of the proposed framework and initializes the QoS provisioning processes for each job. It takes a concrete decision on the parameters and resources that should be adjusted to maintain the quality level for each SLA term, or the highest possible for the overall job.
  - The Event Assessment is a sub-component of the QoS Supervisor that filters and prioritizes the real-time monitoring metrics. Its operation affects the real-time performance of the framework and the result of the QoS provisioning in case the parameters are not prioritized properly or are not linked with the correct SLA terms.

The monitoring parameters are forwarded to the event assessment that filters some of them. Only the most important ones are fed into the QoS Supervisor. This component identifies the related SLA and the available policies and requestes recommendation about how these policies performed in the past and the probability for a SLA violation. The SLA includes specific compensations for the overall job failure.
The recommendation mechanism uses the past experience to rate the quality provisioning capabilities of a service instance, and recommend the best policy that should be applied to guarantee the QoS level. For each policy, the recommendation reflects how good this policy performed in the past under the same or similar conditions. In order to guarantee the quality level, the Supervisor proposes the reconfiguration of the system that is defined in the appropriate (for the current conditions) policy, by knowing how this reconfiguration performed in the past and if SLA violations occurred.

The proposed framework was applied to a 3D rendering scenario. In this use case, based on the historical information, the supervisor could know the cost due to changing the QoS level in each policy and the estimated probability for an SLA violation based on the past experience recommendation. When the SLA violation probability became high, the system was reconfigured, applying one of the QoS provision policies. Adjusting the QoS level affects the cost and the SLA violation probability, and the best option for the service provider is to minimize both of these factor, in order to reduce the penalty for the service provider, and to allow the system to recover from this situation to the agreed one, as stated within the SLA.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- The approach must guarantee the high QoS demands of real-time business applications.
- The system must propose actions using real-time monitoring information and recommendations based on the past experience.
- The system must respond with the appropriate set of actions in order to guarantee the highest QoS level with a reasonable SLA violation probability.

Non Functional:
- XML-based language is used to express QoS requirements.
- The implementation follows SOA principles.

What is the novelty described in this document?
This approach handles QoS not only in resource level but it is SLA-driven and fully pluggable using application specific policies. Additionally, the real-time capabilities of the presented QoS provisioning approach are enhanced with an event assessment process, and a recommendation mechanism that exploits the historical records of each job.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Although this work focuses mainly on the infrastructure level, specially on grid computing, the approach of a supervisor system taking decisions based on the current and past situations can be directly applied in the SLA@SOI adjustment functionality. Specially interesting is the way that the system learns, and how a decision can be taken in order to guarantee the highest QoS level with a reasonable SLA violation probability and minimizing the cost in the Service Provider side.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

None reported.

References:

3.81 Autonomic policy-based management

Keywords:
Policy-Based Management

Abstract / Summary:
Policy-based management is an approach to simplify systems management through the use of rules. Several works are focused on studying the application of autonomic communication techniques to policy-based management. In this summary, two papers are compiled: Policy-based management of middleware for distributed sensor applications [1], and Autonomic policy-based management using web services [2].

Distributed software systems have become very heterogeneous, dynamic and large-scale, comprising a high variety of hardware devices, as well as diverse network infrastructures. Consequently, the manual management of the distributed system and its installed software has become extremely complex.

The first paper [1] seeks for an architectural solution for a middleware that supports autonomic management of distributed software systems through a policy-based approach. This middleware architecture suggests the introduction of a closed control loop on top of a highly customizable middleware architecture.

Such a policy-based management approach consists of several steps, which form some kind of closed loop: information gathering about the execution context, interpretation of policy files, matching their conditions and triggering any related
events, and the enforcement of their associated actions on the current execution context.

The middleware consists of several customizable component frameworks:

- An execution environment, responsible for keeping track of the installed configuration, implementing the core methods for managing the component frameworks in a device-specific language, and device-specific tuning of the underlying execution environment (e.g. memory management and concurrency).
- A distribution framework, to hide the physical distribution of the environment to the other middleware levels and the application.
- A customizable service framework, providing common middleware functionality, e.g. services responsible for data aggregation, persistence, or advanced group communication.
- The adaptation architecture for managing the entire middleware configuration (i.e. gathering information about the entire system, achieving safe adaptations). This adaptation architecture forms the basis for the control loop.

The control loop is an integral part of the architecture. The loop itself consists of three main parts: a monitoring framework (able to detect changes in the environment and even to detect overload situations before they actually lead the run-time system to crash), a policy management framework (to interpret and enforce the policies), and a reconfiguration framework (decides whether the action can be executed or not (e.g. when the load is below a certain threshold), and will prepare for the actual reconfiguration.

The second reference [2] proposes an architecture that uses Web services and automatic Web services composition as a complementary technique to policy refinement in order to automate policy-based management. The following figure shows this architecture:
The network devices are located on the bottom layer. Some of them will offer Web services interfaces for configuration, monitoring, and/or notifications.

The policy engine is the core of the architecture. It contains the registry for management Web services, the repository where policies are stored and the component that makes policy decisions.

The decision component uses the services stored in the registry to monitor the devices. Based on the monitoring information, policies stored in the repository are enforced. These policy decisions lead to changes in the device configurations. They are realized by calling the appropriate configuration Web services of the devices. In order to find the appropriate Web services for monitoring and configuration, policies must be mapped to Web service calls.

**Described requirements**

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)
Functional:

- Common policy language: There is a need for a common policy language that can be used for expressing policies in different areas (like security, QoS, etc.) and in various application domains (like telecom, healthcare, etc.).
- Ease of use: Different types of users, ranging from system administrators to end users, must be capable of specifying their preferences using these policies. Therefore, a policy solution should simplify policy specification by offering not only concepts at a higher level of abstraction, closely related to the problem domain, but also user-friendly formats for specifying policies.
- Extensibility: It should be straightforward to extend existing policy languages with new concepts, add new domain concepts, create new policy languages, add new policy representation formats, etc.

Non Functional:

None.

What is the novelty described in this document?

None reported.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

A policy-based management could be directly applicable in the design of the adjustment functionality in SLA@SOI. The determination of the corrective and preventive actions to execute in the dynamic context of the run-time services management could be better achieved using policies.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

None reported.

References:


3.82 SWRL: Semantic Web Rule Language

Keywords:

Semantic Wb, Rule Language, OQL, RuleML
Abstract / Summary:

SWRL (Semantic Web Rule Language) is a proposal for a Semantic Web rules-language, combining the Ontology Web Language (OWL) [2] and the Rule Markup Language (RuleML) [3].

Semantic Web Services attempt to describe services in a knowledge-based manner in order to use them for a variety of purposes, including: discovery and search; selection, evaluation, negotiation, and contracting; composition and planning; execution; and monitoring. Both rules and ontologies are necessary for such service descriptions and play complementary roles: while ontologies are useful for representing hierarchical categorisation of services overall and of their inputs and outputs, rules are useful for representing contingent features such as business policies, or the relationship between preconditions and postconditions.

The proposal extends the set of OWL axioms to include Horn-like rules. It thus enables Horn-like rules to be combined with an OWL knowledge base. It provides an extension of the OWL model-theoretic semantics to provide a formal meaning for OWL ontologies including rules written in this abstract syntax.

The proposed rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold.

Both the body and the head consist of zero or more atoms. An empty antecedent is treated as trivially true (i.e. satisfied by every interpretation), so the consequent must also be satisfied by every interpretation; an empty consequent is treated as trivially false (i.e., not satisfied by any interpretation), so the antecedent must also not be satisfied by any interpretation.

Atoms in these rules can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values. OWL DL becomes undecidable when extended in this way as rules can be used to simulate role value maps.

Described requirements

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- It must support Horn-like rules.

Non Functional:
- SWRL must support XML as syntax for rules.
What is the novelty described in this document?

SWRL is the first standardised work on the interoperation, semantically and inferentially, between the leading Semantic Web approaches to rules (RuleML Logic Programs) and ontologies. It enables to "build rules on top of ontologies": it enables rules to have access to ontological definitions for vocabulary primitives (e.g., predicates and individual constants) used by the rules.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

This specification is relevant to SLA@SOI since it can be used in the adjustment module in order to determine the corrective and preventive actions to execute.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

There are several implementations [4] that support SWRL:

- SWRLTab, an extension to Protege that supports editing and execution of SWRL rules.
- R2ML (REWERSE Rule Markup Language).
- Bossam, a forward chaining rule engine.
- Hoolet, an implementation of an OWL-DL reasoner.
- Pellet, an open-source Java OWL DL reasoner.
- KAON2, an infrastructure for managing OWL-DL, SWRL, and F-Logic ontologies.
- RacerPro, supports processing of rules in a SWRL-based syntax by translating them into nRQL rules.

References:

[1] SWRL proposal: http://www.w3.org/Submission/SWRL/
[2] OWL Web Ontology Language Overview: http://www.w3.org/TR/owl-features/

3.83 Rule-based and Ontology-based Policies

Keywords:
Rule-Based Policy Model

Abstract / Summary:

The agent-based approach has become an important software engineering methodology for the development of applications in complex environments, due to their ability to operate autonomously without constant human supervision.
Policies [1] can help in dynamically regulating the behavior of agents and in maintaining an adequate level of security, predictability, and responsiveness to human control. By changing policies, agent behavior can be continuously adjusted to accommodate variations in externally imposed constraints and environmental conditions without modifying the agent code or requiring the cooperation of the agents being governed.

With the Internet becoming ubiquitous, there is a need to develop adequate based-techniques for controlling agent behaviour within pervasive environments, characterized by their dynamicity, unpredictability and heterogeneity. Resources are not pre-determined, interacting agents are not always known a priori and, if agents roam across different network localities, they have different resource visibility and availability, depending on their location and on other context-dependent information, such as local security policies and resource state.

In these new complex pervasive scenarios, policies cannot be defined based on entity’s identities/roles, as in traditional solutions, or be specified a priori to face any operative run-time condition, but require continuous adjustments to adapt to the current situation. The authors propose the adoption of a semantic context-aware paradigm to policy specification. Reference [2] describes Proteus, based on these two principles: context-awareness to allow operations on resources to be controlled based on context visibility, and semantic technologies, which allow the high-level description and reasoning about context/policies.

Proteus is centered around the concept of context, defined as any characterizing information about controlled system entities and about their surrounding world. For each context, policies define operations on resources. In particular, policies can be viewed as one-to-one associations between contexts and allowed/obliged actions. Entities (actors/ resources) should and/or can perform only those actions that are associated with the contexts currently in effect (active context), i.e., the contexts whose defining conditions match the operating conditions of the requesting entity and of the environment as measured by specific sensors embedded in the system.

The policy activating contexts are defined as those contexts relevant to specific policies: the activation of a context either causes the activation of permissions or determines the actions that should be performed. Activating contexts of interest are determined by the defined policies (authorization activating contexts by authorization policies, obligation activating contexts by obligation policies).

The context-awareness is achieved through different kinds of adaptations:

- Policy adaptation: it consists of “instructing” the system such that, even though an activating context has changed, it should be still considered active if certain context conditions hold. Policy adaptation automatically prolongs the validity of an active policy even in presence of changes in its corresponding activating context.
- Action adaptation: it represents the ability to find alternative permitted/obliged actions in case the permitted/obliged actions as determined by the current state of the world cannot be performed. Finding alternative set of actions provides a powerful means of allowing an entity to continue to operate.
- Context adaptation: it consists of identifying an alternative context where permitted/obliged actions can be performed. Context adaptation can be
useful in case of dynamic policy conflicts, such as when an entity is obliged to perform an action that it is not allowed to. Instead of changing the set of permitted/obliged actions, Proteus tries to identify a different activating context where the actions can be performed.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- Policy-based control must deal with the dynamic context changes typical of pervasive scenarios, where it is not possible to exactly predict all the interactions an entity may be involved in and the kind of resources it may wish to have access to.
- Policy language must have the ability to model and represent the contexts in which agents operate and to which policies are associated, at a high level of abstraction.
- Policy language must have the ability to define what actions are permitted or forbidden to do on resources in specific contexts (authorizations or permission/prohibition policies).
- Policy language must have the ability to define the actions that must be performed on resources in specific contexts (obligations).

Non Functional:
- Proteus makes use of Web Ontology Language (OWL)-based ontologies.
- Proteus uses Semantic Web Rule Language (SWRL) to encode rules.

What is the novelty described in this document?
This work points out the importance of policy approaches for governing agents in pervasive environments specified in a way that is both context-based and semantically-rich. Two approaches have been used in previous research: an ontology-based approach that relies heavily on the expressive features of Description Logic (DL) languages, and a rule-based approach that encodes policies as Logic Programming (LP) rules. These papers analyze the emerging directions for the specification of semantically-rich context-based policies, and describe a hybrid approach that exploits the expressive capabilities of both DL and LP approaches.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The adjustment functionality in SLA@SOI will have to implement a set of policies in order to determine the corrective and preventive actions to execute. A semantic-based approach could help to develop policies in the dynamic context of the run-time services management.

Although Proteus focuses mainly on security issues, its conclusions may be extrapolated to other fields.
Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

Context and policy ontologies, instantiation and aggregation rules are available at [3].

References:


[3] Proteus at the Laboratory of Advanced Research on Computer Science, University of Bologna: http://www lia.deis.unibo.it/research/Proteus/.

3.84 Model-based Performance Prediction

Keywords:
Performance Prediction

Abstract / Summary:

Over the last fifteen years a number of approaches have been proposed for integrating performance evaluation and prediction techniques into the software engineering process. Efforts were initiated with Smith’s seminal work pioneered under the name of Software Performance Engineering (SPE) [1]. Since then a number of meta-models for describing performance-related aspects [2] have been developed by the SPE community, the most prominent being the UML SPT profile [3] and its successor the UML MARTE profile [4], both of which are extensions of UML as the de facto standard modeling language for software architectures. Other proposed meta-models include SPE-MM [5], CSM [7, 6] and KLAPER [8]. The common goal of these efforts is to enable the automated transformation of design-oriented software models into analysis-oriented performance models making it possible to predict the system performance. A recent survey of model-based performance prediction techniques was published in [9]. A number of techniques utilizing a range of different performance models have been proposed including standard queueing networks [10, 11, 12, 13], extended queueing networks [14, 15, 8], layered queueing networks [17, 18], stochastic Petri nets [19, 20], queueing Petri nets [22, 21], stochastic process algebras [23], Markov chains [16], statistical regression models [24, 25] and general simulation models [26]. In recent years, with the increasing adoption of component-based software engineering (CBSE), the performance evaluation community has focused on adapting and extending conventional SPE techniques to support component-based systems. Since component-oriented technologies are used as foundation for building modern SOA applications, their performance is essential for managing Quality-Of-Service (QoS) in SOA environments. Techniques for component-based performance prediction [27] are surveyed in
detail in the next section. The UML SPT and UML MARTE profiles as well as KLAPER are reviewed in more detail in separate sections.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- Automated transformation of design-oriented software models into analysis-oriented performance models making it possible to predict the system performance

Non Functional:
- Different techniques provide different trade-offs in terms of
  o Modelling expressiveness
  o Modelling overhead
  o Model analysis overhead
  o Model intuitiveness and ease of use
  o Prediction accuracy
  o Support for parameterization

What is the novelty described in this document?
The considered publications provide an overview of the state-of-the-art in model-based performance prediction.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Existing techniques for model-based performance prediction will be used as a basis for implementing the performance prediction functionality in A6.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
The considered techniques have been implemented as research prototypes which normally are not publicly available, however, one could request a copy from the authors.

References:


3.85 Performance Prediction Techniques for Component-based Systems

Keywords:
Performance Prediction, Component Model

Abstract / Summary:
A number of performance prediction methodologies and tools for component-based systems have been proposed [1]. Here we briefly review the most important ones. One of the first attempts towards compositional performance analysis of component-based systems was presented in [2]. The authors argue that in addition to descriptions of functional behavior, performance-related properties must be integrated in component models. An approach mainly based on formal techniques is sketched and illustrated using an example. The authors...
admit that an engineering approach to predictability is a necessary ingredient to ensure predictable components. Another approach to integrate component technology with analysis models was presented in [3, 4]. The authors describe a prediction-enabled component technology called PECT which aims to enable the prediction of assembly-level system properties based on certified component descriptions. This is achieved by imposing a set of restrictions on component designs and implementations which permit compositional analysis methods to be applied to component assemblies. While this is a step in the right direction, the proposed component specifications and analysis methods are rather simple and do not cover all the information needed for accurate performance prediction.

In [5, 6], a compositional methodology for automated performance analysis of component-based systems, called CB-SPE (Component-Based SPE) is presented. CB-SPE is a generalization of the conventional SPE method [7, 8, 9] adapting its concepts and steps to component-based architectures. Annotations based on the UML SPT profile [10] are used to augment component specifications with performance-related properties parametrized with respect to platform parameters. A disadvantage of this approach is that it does not support nesting component sub-models and each time a component is replaced with another the modeling steps have to be repeated. Furthermore, components are modeled at the object-level without considering concurrency within a component. In [11, 12], the authors propose a language called Component-Based Modeling Language (CBML) based on XML and UML2, which is designed to describe layered queueing models with embedded components. Component sub-models support parametrization, however, no explicit context model is defined for capturing variations in input parameters, deployment and configuration. A model assembler tool generates a solvable layered queueing model from a system definition with component bindings. Hierarchical component specifications with multiple levels of nesting are supported.

In [13, 14], a compositional method for performance analysis of component-based systems is proposed in which the effect of input parameters on component behavior and resource usage is modeled explicitly. Application developers can explore possible execution scenarios with different parameter initializations and find the worst-case scenarios where the predicted performance does not satisfy the requirements. The proposed approach, however, is still rather limited since it does not consider stochastic parameter specifications and does not provide a comprehensive component context model taking into account system configuration and deployment aspects.

In [15], it is argued that current component models do not reflect the influence of the deployment context on the component behavior sufficiently. The authors advocate an explicit context model as part of the component specification that captures the dependencies of functional and extra-functional properties on the component’s connections to other components, the execution environment, the allocated hardware and software resources and the usage profile. Context dependencies are specified by means of parametric contracts which can be considered as an adaptation mechanism, adapting a component’s “provides-and-requires-interfaces” depending on its context [16, 17]. A modeling notation based on extensions to the UML SPT profile [10] is proposed in [18] allowing component developers to explicitly specify the influence of parameters on the component resource demands as well as on their usage of external services. A parametric contract in the form of a so-called service effect specification is specified for each component service describing its behavior and control flow in an abstract and
parametric manner. Parameters can be characterized by specifying a probability
distribution over their value (for primitive types), sub-types (for Object types),
the number of elements (for collection types), the byte-size (for binary data) or
the parameter structure (for composite types). Annotated UML models are
transformed to stochastic regular expressions that are used for performance
analysis. The authors show how their approach can be integrated into the CBSE
process model by Cheesman and Daniels [19] to explicitly include early-cycle
model-based performance analysis [20].

In [21], the above approach is extended by introducing constructs for modeling
internal parallelism inside a component. Service effect specifications now support
forking of threads and can be parametrized in terms of the number of CPUs and
CPU cores. Stochastic regular expressions extended with an operator for
parallelism are used for performance analysis. The prediction accuracy, however,
is still rather limited. A major limitation of the proposed analysis method is that it
does not consider resource contention by multiple concurrent requests. The above
works were combined in the Palladio Component Model (PCM) [22] aimed as a
meta-model for specification of performance-relevant information in component-
based architectures. PCM is designed with the explicit capability of dividing the
model artifacts among the developer roles involved in the CBSE process.

Described requirements

(i.e. with reference with the addressed topic area, what are the specific
functionalities, and quality features of the system described in this
work?)

Functional:
- Component-based performance models support the composition of
  predictive performance models from models of the system
  components. Novel component meta-models support parameterization
  of the component models to take into account the influence of the
  component context (execution environment and workload).

Non Functional:
- Different techniques provide different trade-offs in terms of
  o Modelling expressiveness
  o Modelling overhead
  o Model analysis overhead
  o Model intuitiveness and ease of use
  o Support for parametrization
  o Support for modeling context dependencies

What is the novelty described in this document?

With the increasing adoption of component-based software engineering, novel
techniques for performance prediction of component-based software systems are
gaining in importance. The considered papers cover a range of techniques
offering different trade-offs in terms of model expressiveness and analysis
overhead. The Palladio Component Model provides a novel modelling approach
that supports explicit parameterisation with respect to the usage profile, the
executions environment and component connections to external services.
How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

Modern service-oriented architectures are typically built on top of component-based software platforms, e.g., Java EE, SCA, .NET. Performance prediction techniques for such systems can be used as a basis for performance prediction of SOA applications.

The PCM is especially suited as a starting point and a foundation on top of which the design-time prediction service of WP A6 can be built. In SLA@SOI, we extend the capabilities of the PCM to allow for reliability modelling and prediction, as well as being embedded into an automated SLA negotiation process.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

The considered techniques have been implemented as research prototypes which normally are not publicly available. However, one could request a copy from the authors.

References:


3.86 Measurement-based Approaches to Performance Prediction

Keywords:
Performance Prediction

Abstract / Summary:
In [1, 2], it is argued that while the use of performance models in the early stages of system development could be of help to identify bottlenecks in the system design, models often fail to capture important execution aspects that can only be determined at run-time. The authors propose a performance analysis method based on early cycle empirical testing. Application-specific performance tests are derived from architecture designs and executed on the middleware platforms chosen for building the system. The approach, however, provides limited automation and does not consider integrating empirical measurements with performance models. Moreover, since “fake” components are used in place of unavailable system components, the practicality and reliability of the approach is highly questionable.

In [3, 4], the authors use statistical regression techniques to model the relationship between performance-relevant parameters of software components (e.g., use of service calls, input parameters) and their resource demands. Regression models are extracted by running a set of relevant use cases on the components of interest and measuring their performance. The aim is to reuse models, fitted to measurements of existing components, to assess the performance of adapted versions of these components. The proposed method, however, can only be applied if the adapted components are sufficiently “similar” to existing ones.

Another measurement-based approach to performance prediction of component-based applications is proposed in [5, 6]. A simple benchmark is used to extract a performance profile of the underlying component-based middleware (e.g., Java EE or .NET) used to build an application. A generic performance model is then constructed that reflects the interactions between the key performance factors in applications deployed on the selected platform. A significant drawback of this approach is that application-specific behavior is not modeled explicitly and only very rough estimates of the system behavior can be obtained.

In [7, 8], the authors describe a technique for automated analysis of the system architecture and extraction of performance models based on traces obtained during operation. A limitation of the technique is that components having internal parallelism with forking and joining of the execution flow are not supported. Furthermore, a number of requirements are placed on the tracing tools which make it difficult to apply the technique in large distributed systems spanning multiple administrative domains.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)
Functional:
- Performance prediction based on measurement and monitoring data collected from a running system.

Non Functional:
- Techniques differ in
  - type of measurement data required
  - measurement/monitoring overhead
  - prediction accuracy

What is the novelty described in this document?
The considered publications describe novel techniques for performance prediction based on measurement and monitoring data collected from a running system. The extraction of a representative performance model capturing the major performance influences is a complex task that poses a lot of challenges.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Measurement based techniques for performance prediction are important both in the context of design-time performance prediction (6.1 – 6.4) and in the context of run-time performance prediction (A6.5).

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
The considered techniques have been implemented as research prototypes which normally are not publicly available, however, one could request a copy from the authors.

References:

3.87 Performance Prediction Techniques for Web Services and Service-oriented Architectures

Keywords:
Performance Prediction, SOA

Abstract / Summary:
A number of approaches for introducing QoS support in Web services have been proposed, for example [1, 2, 3, 4, 5]. In [1], an extension of UDDI enabling Web service discovery based on QoS requirements was presented. Similarly, in [2, 3], extensions of WSDL-based Web service descriptions were introduced to support QoS-related information. These studies, however, do not address the issue of how the service provider guarantees its QoS claims. An approach to dynamically select a service provider that best meets the consumer's needs is presented in [4]. An agent framework coupled with a QoS ontology is used, however, the framework does not support the ability to reserve the resources required for providing a selected QoS, and therefore again no QoS guarantees are provided.

In [3], a lightweight extension to WSDL (Web Service Description Language) introducing QoS characteristics was proposed. The latter, however, is far too simple and can only be used to model services at a very high-level considering each service as a black box. A much more detailed and fine-grained meta-model is needed in order to cover all aspects relevant for predicting the service performance. In [6, 7, 8], several methods for dynamic selection of services with the goal to optimize the overall QoS of a composition are proposed. In [6] workflow patterns are used, whereas the approaches in [7] and [8] use genetic algorithms and heuristic methods, respectively.

A different approach to QoS brokering and service selection is presented in [9], where analytic queueing models are used to predict the QoS of alternative services that could be selected under varying workload conditions. Service consumers provide to a QoS broker their utility functions and cost constraints on the requested services. The service provider that maximizes the customer's utility function subject to its cost constraints is selected. In [10, 11], a service discovery system enabling service compositions from semantic descriptions stored in a knowledge base is proposed. A recursive algorithm builds service compositions by adding services in each iteration. The search works backwards, since services are added that produce a certain output regardless of their input parameters. A valid
service composition produces a set of queried output parameters and input parameters necessary for the composed services.

An approach to modeling the performance of composite SOA services composed by means of BPEL (Business Process Execution Language) [12] was presented in [13]. Some further approaches based on simulation were proposed in [14, 15, 16]. These approaches, however, only consider static service compositions. Several larger efforts in the Web services arena have focused on describing, advertising and signing up to Web services at defined levels of QoS, for example, HP’s Web Services Management Framework (WSMF), IBM’s Web Service Level Agreement (WSLA) framework, the Web Services Offerings Language (WSOL) and the WS-Policy. These efforts consider QoS in its broader meaning (not limited to performance properties) and specifically target Web service management activities. Performance properties are defined at a very high level and their enforcement at the network and infrastructure layers is not dealt with.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
Performance models and evaluation techniques specifically targeted at Web Services and SOA-based systems.

Non Functional:
As discussed above, the considered methods provide different trade-offs in terms of
- level of abstraction at which the system is modelled
- modelling expressiveness
- predictive power
- support of open industry standards

What is the novelty described in this document?
Most of the techniques adapt existing performance prediction approaches to Web Services and SOA. However, currently available techniques consider services at a very high level without modelling their internal component architecture.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Performance prediction techniques for Web services and SOA are a central part of the SLA@SOI project. While the techniques considered here are limited in terms of predictive power, they may serve as a basis for building more complex models that enable more accurate performance prediction.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
The considered techniques have been implemented as research prototypes which normally are not publicly available, however, one could request a copy from the authors.

References:

[12] Andrea D’Ambrogio and Paolo Bocciarelli. A Model-driven Approach to Describe and Predict the
3.88 **UML Profile for Schedulability, Performance and Time (UML SPT)**

**Keywords:**
Model-Driven Performance Optimization, UML

**Abstract / Summary:**
The UML SPT profile is an extension to the UML, which enables developers to add performance related information to UML models. The aim is to reuse existing design documents and augment them with performance annotations. These annotated models are meant to be input for model transformations, which map them to performance models, such as queueing network models, stochastic Petri nets, or stochastic process algebras. Results from solving these models are fed back into the UML models. For this purpose, the SPT profile includes corresponding annotations to save performance metrics from analysis tools. Using UML SPT, software developers can specify scenarios consisting of several steps, which produce load onto the modelled resources. Furthermore, they may specify the workload of a scenario (e.g., the number of concurrent users). Several scientific approaches use the UML SPT profile. For example, Marzolla has implemented a simulation for UML models annotated according to the SPT profile.

**Described requirements**
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

**Functional:**
- Software architecture has to be modelled in terms of UML
- Static and dynamic models of the software system are available

**Non Functional:**
- Stochastic information (e.g., resource consumptions and branching probabilities) on the system under study are available.
What is the novelty described in this document?

UML SPT has been the first attempt to put model-driven performance engineering to practice. The annotation of existing, UML-based design documents with performance-relevant data should ease the use of performance prediction approaches during the software development process.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?

Some of the concepts realised in the scope of the UML SPT profile are relevant for the core model of non-functional properties in SLA@SOI. However, UML SPT is to be replaced by its successor MARTE (see below). Therefore, UML-SPT has only a limited relevance for SLA@SOI.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

The UML Profile is available and stable. However, there are only few, academic UML tools that can deal with UML SPT annotations. These tools are merely prototypes.

References:

3.89 Core Scenario Model (CSM)

Keywords:
Model Transformation, UML, UML SPT

Abstract / Summary:
The Core Scenario Model (CSM) is meant to simplify the design of model transformations for UML SPT. As UML designers may use different kinds of diagrams for expressing the performance properties of their systems (e.g., activity diagrams, sequence diagrams, collaboration diagrams, etc.), CSM aims at providing a common intermediate model as the target for mapping the different annotated UML diagrams. Model transformations to the performance domains (e.g., to queueing network models, stochastic Petri nets, stochastic process algebras) then only have to be defined for the intermediate model instead of for each UML diagram. Opposed to UML SPT, CSM explicitly models control flow as sequences, alternatives, loops, and forks.
Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- Software architecture has to be modelled in terms of UML
- Static and dynamic models of the software system are available

Non Functional:
- Stochastic information (e.g., resource consumptions and branching probabilities) on the system under study are available.

What is the novelty described in this document?
CSM integrates the information available in various representations of an UML specification into a single core model. Using such a core model eases the design and development of transformations from UML to analytical performance models.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The concept of a single core model for non-functional properties has already implemented in the context of SLA@SOI. Using a single core model, eases the communication between different parties involved in the service life-cycle.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
The CSM meta-model is available and stable. However, there are only few, academic UML tools that can deal with the model. These tools are merely prototypes.

References:
[1] Dorin B. Petriu and Murray Woodside

3.90 Kernel Language for Performance and Reliability Analysis (KLAPER)

Keywords:
Model-Driven Performance Engineering

Abstract / Summary:
The Kernel Language for Performance and Reliability Analysis (KLAPER) further elaborates the idea of a central core model for model transformations in software
performance engineering. As an intermediate model, KLAPER reduces the number of model transformations for \(N\) input models in the software design domain to \(M\) target models in the performance domain from \(N*M\) to \(N+M\). In doing so, KLAPER integrates performance specifications based on different notations, such as annotated UML models. There are several model transformations from KLAPER to Markov chains or extended queueing networks. KLAPER has been extended for reconfigurable architectures.

**Described requirements**

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

**Functional:**
- Software architecture has to be modelled in terms some architecture description language
- Static and dynamic models of the software system are available
- A transformation of the architecture model to KLAPER has to be implemented

**Non Functional:**
- Stochastic information (e.g., resource consumptions and branching probabilities) on the system under study are available.

**What is the novelty described in this document?**

KLAPER reduces the complexity of transformations from architectural models to analytical models by introducing an intermediate transformation language.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

The concept of a single core model for non-functional properties has already implemented in the context of SLA@SOI. Using a single core model, eases the communication between different parties involved in the service life-cycle.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

The KLAPER meta-model is available and stable. However, there are only few, academic transformations implemented for the model. Especially the transformation of UML SPT to KLAPER is not fully implemented which hinders the practical use of KLAPER for SLA@SOI.

**References:**


3.91 Service Component Architecture (SCA)

Keywords:
Application Development, SOA, Service Composition

Abstract / Summary:
Service Component Architecture (SCA) provides a programming model for building applications and solutions based on a Service Oriented Architecture. It is based on the idea that business function is provided as a series of services, which are assembled together to create solutions that serve a particular business need. These composite applications can contain both new services created specifically for the application and also business function from existing systems and applications, reused as part of the composition. SCA provides a model both for the composition of services and for the creation of service components, including the reuse of existing application function within SCA composites.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
- Software systems must be divided into service components that are hierarchically structured in composites
- Services must be offered by components

Non Functional:
- Within a single domain, the used technology must be provided by a single vendor
- Communication across domain must use standardised protocols

What is the novelty described in this document?
SCA represents the first complete solution for component-based and service-oriented software development. Its concept of separating communication from implementation makes the development of service components straightforward in the technology of your choice.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
SCA is a promising service and component technology. It stands a good chance to replace today’s component technologies such as Java EE and .Net in the future.
Therefore, SCA is one of the key technologies to look at when reasoning on service-oriented architectures.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

Various open source implementations are available. One worth mentioning here is Apache Tuscany. Its current version 1.4 is stable and available under Apache License, Version 2.0.

References:
[1] SCA Project
   http://www.osoa.org/display/Main/Service+Component+Architecture+Home
   http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications
   Pearson Education
   http://tuscany.apache.org/home.html

3.92 FFTV (From Failure To Vaccine)

Keywords:
Self-protecting Systems

Abstract / Summary:
FFTV is a technique for developing self-protecting systems. FFTV observes values at relevant program points, e.g., method calls or variable assignments. When FFTV detects a software failure, it uses the collected information to identify the execution contexts that lead to the failure, and automatically enables mechanisms for preventing future occurrences of failures of the same type. Thus, failures do not occur again after the first detection of a failure of the same type.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

A distinction can be made on the basis of the information collected by FFTV. It gathers information about both data exchanged and interactions performed during system executions.

Functional properties:
- Data: FFTV collects data regarding both simple and complex data types. The former are also known as primitive data types, e.g., float, integer, char. The latter can array of char or complex
objects. FTTV infers model over the collected data using DAIKON inference engine.

- **Interactions**: FTTV collects interaction sequences performed during system executions and infers interaction patterns for further checking.

**Non Functional properties:**

FTTV it does not take into account non functional properties.

**What is the novelty described in this document?**

Failure prevention techniques can be roughly classified as *failure-specific* and *general* techniques. Failure specific techniques are based on design-time prediction of failures that are likely to occur at run-time. General techniques are based on mechanisms that handle catastrophic events that do not depend on the specific application.

FTTV is not an alternative to failure-specific and general mechanisms, but it complements the set of problems dealt by existing techniques.

The novelty of the FTTV lies on the ability to prevent specific problems that are difficult to predict at design-time, but can be identified at run time by learning from program executions.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

FTTV can address different classes of problems by capturing different information and distilling different models. For instance, FTTV can detect problems caused by unexpected data values passed between services or assigned to service state variables.

The FTTV methodology (observation, model inferring, and runtime model checking) could be applied to SLA@SOI at both infrastructure level and service level. In both cases, services/resources usage data can be collected by monitors that, after inferring behavioral models, can check them at runtime in next executions.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

A prototype fully implementing FTTV approach has not been released yet. So far, FTTV has been validated using semi-automatic code instrumentation and manually defined oracles for detecting ad-hoc violations.

**References:**


3.93 Self-protective techniques

Keywords:
Self-Protection

Abstract / Summary:
Self-protecting techniques have been widely studied for hardware systems [1, 2]. The many techniques for design for testability, Built-In Self Test and Built-In Self Repair are effective for hardware devices, but focus mostly on manufacturing faults, and are based on fault models not shared with software systems, and do not apply well to many relevant software problems.

Classic failure prevention techniques for software systems can be classified in two main groups: failure specific and general techniques. Failure specific techniques support developers in defining proper procedures to handle problems that can be predicted at design-time. Main approaches are based on the development of defensive code and design of checking mechanisms, e.g., defensive programming [3], assertions [4], self-checking systems [5] and exception handlers [5].

General techniques aim to preserve system functionalities in case of catastrophic events. The main techniques have been studied for safety critical applications by the fault-tolerance community [7]. Common fault tolerance approaches include redundancy, service relocation, and transactional services [8], to add dynamic recovery mechanisms, and rejuvenation features [9], to prevent aging problems, i.e., systems with a state that degrades over time potentially leading to failures.

Failure specific techniques can handle problems that can be predicted at design-time, but they cannot deal with failures difficult to predicted before deployment, such as environmental problems, for instance problems that derive from specific context of use, or configuration problems, for instance problems that derive from the integration of the application with other systems, or domain dependent problems, for instance unexpected ranges of values in a specific domain.

General techniques can deal with general catastrophic events, but cannot effectively cope with application specific problems that do not produce catastrophic effects.

Several techniques to prevent failures and design healing strategies complement the aforementioned techniques.

For instance, Zachariadis, Mascolo and Emmerich defined a framework for designing mobile systems that can automatically adapt to different environmental conditions. Cheng, Garlan and Schmerl defined a language that supports the definition of domain-level objectives dynamically enforced on a target system [10]. Modafferi, Mussi and Pernici defined a plug-in to add recovery mechanisms to Ws-BPEL engines [11]. Several researchers outlined the idea of using external models to support adaptation and healing mechanisms, and implementing healing and prevention procedures by changing architectures and using advanced services such as transactions [12, 13].

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)
The presented techniques can be applied in many contexts, e.g., distributed systems, business oriented applications, and service oriented applications. Techniques that might interest SL@SOI are those who use checking mechanisms.

**Functional properties:**
Since failures are mostly caused by unexpected/erroneous interactions among system’s components or wrong exchanged data, techniques that use checking mechanisms deal mostly with functional properties.

**Non Functional properties:**
None.

**What is the novelty described in this document?**
The considered publications provide an overview of the state-of-the-art in self-protective techniques.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
Existing techniques for self-protective can be taken into account when a composite service is designed, developed and deployed. These techniques can contribute to increase the overall system reliability.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**
Some of the presented techniques represent what is broadly used nowadays in real systems to avoid failure consequences, e.g., redundancy, service relocation, and transactional services. Others are on going researches and some implementations exist only as prototypes.

**References:**
3.94 Performance Model Driven QoS Guarantees and Optimization in Clouds

Keywords:
Performance model, Cloud

Abstract / Summary:
The paper proposes methods for using optimisation techniques in cloud-environments taking into account QoS constraints. Control algorithms are described to provide a QoS level aiming for guarantees as agreed upon in pre-established service level agreements.

The main actors in the described cloud environment are the cloud-provider and application-providers. The cloud provider is the operative of the public or private cloud environment. The application-provider is the operative and responsible for cloud-based applications that make use of the cloud-provider’s infrastructure. Application-provider and cloud provider are bound to each other by an agreement guaranteeing functional and non-functional properties regarding the cloud-service. The paper assumes that the application-provider will always try to minimise the resource usage for deployed applications since payment is directly dependent on resource-consumption. This implies a payment-model as applied by Amazon Web Services. The authors define a measure for the application-provider’s profit and argue that this is the primary metric to be maximised. They further assume that a cloud-provider’s profits will be maximised automatically if all customers i.e. application-providers maximise their profits. This assumption is based on the fact that efficient application providers will need fewer resources and as such the cloud-provider can serve additional customers. The cloud-provider will as such save money on bulk reductions for its customers. The customers benefit from overall lower costs themselves through efficient usage of resources.
The authors proceed by defining a formal model based on different layers in a cloud and a simplified metamodel for a service-architecture. Deriving from this model a performance model is generated [20]. This performance model is based on layered queuing networks [5], [6], [15], [19]. The paper goes on to define workload and QoS-requirements based on node throughput, user-count, maximal response-time and mean think-time for an arbitrary class of an application. Based on this a feedback-control loop is derived to adjust QoS to the needed values.

The paper goes on to introduce the applied network flow model to be used in conjunction with the previously established layered queuing network. The network flow model divides all nodes into hosts, server tasks, services and classes of users. These different categories are brought into relation with each other by appropriate flows.

Based on the described model an objective function is developed that describes an application-provider’s profit. Further the objective function for the cloud-provider is developed by summing all application’s profit functions.

The paper continues with a case study comparing incremental optimisation with a full optimisation for different numbers of application instances. The maximal number of application instances is 50, leading to a profit of 1278 in the incremental and to a profit of 1649 in the full optimisation case. The calculation time for the full optimisation is 295 seconds on a state of the art pc.

The paper concludes by stressing the feasibility of this approach for maximising the profits for application operators as well as cloud operators.

**Described requirements**

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

The requirements to apply the proposed optimisation techniques are virtual infrastructures that are capable of allocating virtual machines to new hosts dynamically.

**Functional properties:**

The paper refers to virtual machines hosting services and user-classes that fit defined application classes. Further the throughput values for virtual machine nodes, physical hosts, application instances and user nodes need to be known.

**Non-Functional properties:**

Non-functional properties such as response-time, think-time and maximum response time are taken into account.

**What is the novelty described in this document?**

The novelty of this paper is the combination of layered queuing networks with linear programming techniques to optimise the provisioning of applications in a cloud-environment taking into account delays in the queuing network.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
Key concepts of this paper might directly be relevant to SLA@SOI especially for provisioning and adjustment. Further key concepts from the paper might be helpful for negotiation time optimisation to evaluate templates.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

The paper states only that a proof of concept has been developed for evaluation within the CERAS cloud-environment and that this implementation will be extended in future.

**References:**


### 3.95 A Game Theoretic Framework for SLA Negotiation

**Keywords:**

Game theory, SLA negotiation

**Abstract / Summary:**

The paper establishes a formal model using techniques from game theory to describe automatized negotiations for service level agreements. The basis for this work in game theory are signalling games as described in [1].

First the formal is introduced by describing the actors as provider and customer for a service. Between both a set of initial contracts will exist. A contract is a tupel \((p, q)\). Further a customer belongs to a customer-type from a set of customer-types. Further there is a probability distribution for the customer-types. Two utility functions are introduced. The utility function for the provider is based on a contract \((q, p)\) as parameter, while the customer's utility depends on the contract and additionally the customer-type.
During the game the customer first chooses a contract \((q', p')\) such that the Euclidean distance to another contract \((q, p)\) is below a threshold \(\underline{\text{ }}\). This contract is then modified to the customer’s own advantage and provided to the provider. The provider then extracts information from the provided contract and updates its belief on the respective customer-type. The provider will then based on the received offer create a counter-offer the will again be within a certain vicinity of the received contract.

Some constraining assumptions are made to simplify the formal problem. The first one is that a client will accept any contract, which will in result for her in a utility greater or equal to zero. The analogue assumption is made for the provider. The game consists the formally of two action spaces, one for the customer and one for the provider, and of a global pay-off function. Action-spaces can be described informally as contracts that are central within the contract space and have a short distance to all other contracts. The pay-off function depends on both action-spaces and on the client type and is a matrix of real values.

Further the provider’s and the customer’s strategies and the equilibrium are defined. Equilibrium depends on the current beliefs, the customer’s and the provider’s strategy. Equilibrium is reached if neither the customer nor provider are going to change their strategies since there is not incentive for doing so with respect to optimising the own utility. Further given the customer’s actions the provider does not change her beliefs about the client in equilibrium.

The paper continues with the discussion of a use case and a simulation including a brief evaluation.

The described method has mostly two drawbacks that it is limited to negotiations between only one provider and one customer. The other drawback is that the method depends critically on the set of initial contracts. If the initial contracts are optimal for the clients there may be no need for them to bargain.

### 3.96 Turning Software into a Service

**Keywords:**

SaaS, SOA

**Abstract / Summary:**

This paper provides a holistic overview of the Software as a Service (SaaS) paradigm. The authors observe that the demand for software services over time is changing from the traditional possession and supply-led models towards subscribe and demand-led models. In this context, a new ‘Service Technology Layer’ is identified that has been previously missing in the service models. This new layer leads to a whole new stack of protocols that can be complemented largely by the contemporary Web Services stack, however some vital features are missing. The paper identifies these gaps and describes their importance. Key service oriented functions proposed are briefly explained below:

Service Description: A mechanism by which providers may describe what they can offer and the terms of negotiation if need be. This mechanism must also take into account the description of client’s needs, in order to match with provider’s offers. Service description must also allow for quality of service features and legal provisions of contract making.
Service Discovery: The paper encourages semantic approaches to be incorporated with service registries so that searches may be conducted for discovering a required service beyond the constraints of SQL or XML filters.

Service Negotiation: It is observed that although several frameworks exist that cater for negotiation needs (e.g., ebXML, DAML-S and WSEL), yet these do not allow for automated negotiations.

Service Delivery: The paper defines service delivery as a combination of invocation, provision and suspension. Provision is the process by which the service provider provides the service as per the agreement reached with the client. Invocation is the normal usage of the service by the client and suspension allows for ending the contract for service usage if guarantees or bounds for provision are violated. WSCI and WSEL are viewed favourably.

Service Composition: The authors note that in a truly service oriented world, larger, customized or complex services will be constructed at run time by composing lower level services that perform the required tasks. The paper points to existing technologies like WSCL, CS-WS and WSC which may be used to allow service providers converse or compose using choreography.

Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
The paper emphasizes on the need for describing the semantics of services, which may later be used to discover appropriate services with a high rate of customer satisfaction.

Non Functional:
The work does not describe a system as such, however it identifies areas including non functional aspects that must be considered while developing a service oriented system.

What is the novelty described in this document?
The paper is considered one of the earliest to address the SaaS area in particular and identifies critical gaps that exist in existing service related technology stack.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The paper brings to light various frameworks that address several concepts also faced in SLA@SOI, e.g., negotiations, creating legal automatic agreements, provisioning, discovery techniques, service composition at runtime.
Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

Nothing reported.

References:

3.97 Optimal Web Service Composition Using Hybrid GA-Tabu Search

Keywords:
Service Composition, Genetic Algorithms, Hybrid Genetic Algorithms.

Abstract / Summary:
This paper addresses the ‘service composition’ problem in service oriented systems by using a hybrid Genetic algorithm and Tabu-Search technique, which the authors develop after considering research results available from previous literature [2,3]. Genetic algorithms were chosen primarily due to their ability to find one or more optimal solutions from the available search space, keeping in view global QoS constraints – an objective not met by the local optimization techniques employed traditionally for such problems. However, the authors observe that the approach is feasible only when the combinatorial size is extremely large. For smaller combinatorial size, Integer programming can outperform Genetic algorithms. The work presents a fitness function that expresses end user’s preference or favouritism for QoS attributes. The fitness function maximizes or minimizes the QoS attributes with the objective to reach the ideal service composition.

It is observed that the quality or fitness of the initial population in the Genome matters a lot to the overall result the algorithm produces. Nevertheless, in each population, the mutation operator is used such that an individual that apparently violates the desired constraints is ‘proportionally’ selected, to maintain the diversity of each population.

Recently, the use of Genetic algorithms along with local optimization techniques like Tabu-Search or simulated annealing has gained popularity. The paper under discussion adopts the same hybrid approach. Employing Tabu-Search increases the search space for the Genetic algorithm by keeping it to run into the trap of local optimality and also assists to maintain population diversity by using Tabu lists to contain sometimes the fittest chromosome and sometimes the least fit one. In the end, the authors prove with evaluation results that their hybrid GA-Tabu search algorithm outperforms the GA-only approach for large combinatorial sets in terms of highly fit service compositions created as a result.
Described requirements
(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
The presented algorithm is proven to generate high quality service compositions, when large set of candidates are to be searched.

Non Functional:
It is unclear if the presented algorithm can be executed within the bounds of usual request-response styled applications. Therefore, the performance of the algorithm in terms of execution time needs to be further explored.

What is the novelty described in this document?
The authors claim to have improved existing formulas for describing fitness value and penalty technique.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The work provides good insight into the use of hybrid GA algorithms for the purpose of service composition. The SLA@SOI project may evaluate this technique for its Planning and Composition (POC) module.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
Nothing reported.

References:
[4]
3.98 Non-functional Properties In Web Services

Keywords:

Abstract / Summary:
This paper introduced the classifications of services: functional, behavioural, and non-functional. The importance and the definition of non-functional properties are discussed latter. [1,2] Especially non-functional can be used for service selection. We are talking about outsourcing while making planning and optimization, in the other words, the selection of the services can be based on the non-functional properties of the services, like availability, price, reputation, penalty, security trust, most notably QoS and so on.

Thus, non-functional properties of the services are the basis of negotiation and final agreement setting.

What is the novelty described in this document?
It defined and modelled the relative comprehensive non-functional properties of the services, which give us a clear picture about base on what a selection of service could be.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
For infrastructure planning and optimization (IPOC), it is a good reference and supplies more metrics that IPOC might consider about. Especially like modelling of penalty, reputation, price, security trust, and some notably QoS.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
Not reported.

References:

3.99 Cloud computing state-of-the-art and research challenges
Keywords:
Cloud Computing, data centres, and virtualization

Abstract / Summary:
State of the art of cloud computing.
Features: pay as you go pricing model, scalable, risks, easy access.

Its definition [1]: Cloud computing is a model for enabling convenient, on-
demand network access to a shared pool of configurable computing resources
(e.g., networks, servers, storage, applications, and services) that can be rapidly
provisioned and released with minimal management effort or service provider
interaction.

The connection with other techs like grid, utility computing, virtualization tech,
autonomic computing.

Architecture/ business model
Types of cloud:
Private cloud: a more performance, reliability, and security based cloud
environment.
Public cloud: public cloud offering the resources as services to public with lacking
of fine-grained control over data, network and security settings.
Hybrid cloud: combination of the both above.

Research challenges:
1, automated service computing: allocate and de-allocate the resources to lower
the implementation cost, in SLA@SOI, service manager has to do this job.
2, virtual machine migration
3, Service consolidation
4, Energy management
5, Traffic management and analysis
6, Data security: this might be a aspect that relates to cloud provider splits its
service into some sub-services providers by outsourcing.
7, Software frameworks
8, Storage tech and data management

What is the novelty described in this document?
This paper describes the current state of the art of cloud computing, especially
about the research challenges that still exist in this area.

How may this work be applicable to SLA@SOI? What are the necessary
actions that should be taken to apply it?
Not directly, but quite useful to our own researches.

Is there a readily available implementation of the described work? If so,
please state its licensing type and its maturity level (e.g.
prototype/alpha/beta/RC/stable).
Not reported.

References:

3.100 SLA-based profit optimization in Automatic Computing Systems

Keywords:

Abstract / Summary:
Automated service computing: allocate and de-allocate the resources to lower the imple cost.

This paper design a set of dispatching and control policies for the dispatcher in service oriented environment. The objective is to maximize the provider's profits associated with multiple classes of SLAs. It introduces some Utility Functions (cost functions) [1, 2, 3, 4] to describe the optimization problem. It shows that the overall problem is NP-hard, and develop meta-heuristic solutions based on the tabu-search algorithm[5].

What is the novelty described in this document?
This is about automated service provisioning topic. In order to reduce the operating cost, data centers map the requests from customers (SLA) to its infrastructure framework. This paper elaborated a set of dispatching and control policies that could be used by data center to allocate the workloads to right servers with optimal profit and the lowest implementation cost.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Generally this is a good idea that service manager might reference, how to map the SLA request with its own infrastructure. For planning and optimization, NP-hard, and develop meta-heuristic solutions based on the tabu-search algorithm are basic and relevant topics.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
Stable

References:

3.101 SLA-aware Virtual Resource Management for Cloud Infrastructures

Keywords:

Abstract / Summary:
This is about automated service provisioning topic. By introducing a utility function (constraint programming), a trade-offs solution is discussed between the SLA fulfilment and the operating cost.

There are two main steps: mapping the application or workload to the corresponding VMs and allocating the VMs to corresponding physical machines.

This paper supposes that there would be two modules: Local Decision Module (LDM) and Global Decision Module (GDM). LDM is used for mapping the service level to utility value and as well as the resource level to utility value. And further it will communicate the GDM. GDM is responsible for determining the VM allocation and placing the VMs on PMs in order to minimize the number of active PMs. [1]

What is the novelty described in this document?
Each Physical Machine has one or more VMs, the core idea of this paper is to minimize the number of active PM by creating, upgrading, downgrading and migrating VMs.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Generally this is a good idea that service manager might reference, how to map the SLA request with its own infrastructure.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

stable

References:


3.102 BREIN: Business objective driven RELiable and Intelligent grids for real busiNess

Keywords:
SLA Management, Negotiation, Brokering, Infrastructure Management

Abstract / Summary:
BREIN is a European project that focuses on bringing closer together e-business models and concepts, and Grid computing. In the process, BREIN applies various techniques from the area of artificial intelligence, intelligent systems, semantic web, etc.

With regard to SLA@SOI, BREIN is highly relevant from a conceptual point of view. BREIN has researched the topic of SLA management to a large extent, looking into negotiation, evaluation, monitoring etc. Here, we are mostly concerned with the architecture of BREIN, and the differentiating factors in comparison to SLA@SOI.

What is the novelty described in this document?
BREIN introduces novelty by bringing together Grid computing (specifically: the concept of Dynamic Virtual Communities) with business models and concepts, thus making Grid computing a technology applicable to real-world e-Business. From an SLA management point of view, BREIN takes a wide look at the topic and addresses topics highly relevant to SLA@SOI, such as negotiation, evaluation and monitoring.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
Although BREIN introduces a number of novelties, applying the results directly on SLA@SOI is difficult due to a number of different design choices, as well as various differentiating factors.

To start with, BREIN is specifically focused on Grid services. In contrast, SLA@SOI does not assume any Grid middleware – rather, we explore principles and technologies applicable to different areas, with Cloud computing being the foremost one.

BREIN is using semantics to describe service capabilities. As a matter of fact, the use of semantic technologies is prevalent in BREIN, affecting all sides of the project – from advertisements and matchmaking, to negotiation. SLA@SOI preferred to avoid semantic technologies, and approached the problem by means of a new and very complete SLA model, that bears semantics itself. In this model, term dictionaries can be appended for domain-specific purposes. The combination was preferred in order to have a more lightweight approach, and also to avoid the dependency on specific semantic technologies.

In terms of business, BREIN is focused in the business models of Grid and it has developed some components that are oriented to Billing and Trade. However SLA@SOI focused on the SLA aspects that can be modelled and it considers all the SOI terms from business to infrastructure. However we will study BREIN billing component in order SLA@SOI obtain an advantage of it.

With regard to SLA negotiation, in SLA@SOI negotiation strategies are decoupled from negotiation protocols. This allows to have protocols reused with different strategies (i.e. different Planning/Optimization components). In addition, protocols can be (re)configured, even at runtime. This allows for maximum flexibility and software reuse.

In SLA@SOI we pursued technology independence from the very beginning. While BREIN is WS-Agreement specific, we are not bound to XML, Web Services, or any other technology. Our architecture allows for translators (syntax converters) that people can introduce to apply new interoperability requirements.

From an architectural point of view, in SLA@SOI we currently have a fully-modular, hot-pluggable OSGi-based architecture, where components can come and go at runtime. That means, for instance, a new resource pool can be added to the system or disappear, and the system is technically in a position to handle this without a need to restart or reconfigure. The same applied for a different Planning/Optimization component that implements a different strategy, etc.

From an infrastructure point of view, the BREIN Virtualised Resource Management and monitoring Toolkit is targeted solely on the Xen virtualisation platform whereas SLA@SOI needs to support heterogeneous infrastructure. BREIN allows certain infrastructure monitoring data to be polled from a resource monitoring service whereas SLA@SOI monitoring infrastructure needs to support dynamic runtime configurations: SLA infrastructure monitoring requirements can be different for each SLA, and arbitrary monitoring frameworks need to be supported. For reasons of scalability SLA@SOI also foresees the need to support subscription-based monitoring feeds and decentralised management of infrastructure monitoring.

Finally, the SLA@SOI SLA Manager (SLAM) is completely autonomous; there are no fixed cardinalities between SLAMs and providers, which is another differentiating point in relation to BREIN.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
3.103 SLA-Negotiation in BEinGRID

Keywords:
SLA Negotiation, SLA Management

Abstract / Summary:
BEinGRID provides an SLA negotiation component that adheres to WS-Agreement’s recommended architecture and provides opportunities for customisation.

What is the novelty described in this document?
BEinGRID’s SLA negotiation component provides an adoption of WS-Agreement including an adhering component architecture [1]. Internally the component separates a negotiation manager, negotiation strategies as well as local SLA and SLA template registries. The negotiation manager handles negotiation sessions and chooses appropriate negotiation strategies dynamically. These negotiation strategies influence how the negotiation component reacts to offers during a negotiation. These strategies are confined by WS-Agreement’s take-it-or-leave it negotiation protocol, since no WS-Negotiation adoption is available. The SLA registry is being used for existing SLAs and the SLA template registry serves as source for initiating negotiations.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The advantage of BEinGRID’s negotiation component is that it fully adopts WS-Agreement. This can be used to learn the methods and possibly re-use parts of the WS-Agreement related code base.

Besides that sla@soi’s generic SLA manager (GSLAM) provides a finer grained separation of concerns, which is embodied by individual sub-components that are partially domain independent.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
stable

References:
All mentioned components are available as open source and they are mature software components ready to be customised for usage.

References:

3.104 SLA-Negotiation in ASSESSGrid

Keywords:
SLA Negotiation, SLA Broker

Abstract / Summary:
ASSESSGrid provides a component for negotiating SLAs, which can be used in various scenarios.

What is the novelty described in this document?
ASSESSGrid’s negotiation manager provides an implementation of WS-Agreement as well as a Java-based component using Globus Toolkit 4.0 WS-Core features [1]. Further it extends WS-Agreement’s standard protocol by a “getQuote()” operation, which provides the component with advanced negotiation capabilities. The negotiation manager further provides interfaces for management and monitoring tools. The component provides support for three scenarios: Direct SLA negotiation, intermediary negotiation and negotiation via meta-providers. In direct SLA negotiation a customer is in direct contact negotiating an SLA with a provider. In the intermediary negotiation scenario a broker negotiates with multiple providers on behalf of a customer. Finally in the meta-provider scenario the negotiation manager creates a virtual market place for a customer by channeling the offers from multiple providers for one customer.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
BEinGRID provides a full implementation of WS-Agreement, which has not yet been achieved in sla@soi. For this the source code of negotiation manager should be analysed for the WS-Agreement adoption. If possible available code for that matter should be re-used.

Further the negotiation manager expands WS-Agreement’s standard protocol by the getQuote() operation, which should be also evaluated as an extension for sla@soi’s custom negotiation protocol.

Besides these points sla@soi surpasses ASSESSGrid’s negotiation manager in its capabilities.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
stable
All mentioned components are available as open source and they are mature software components ready to be customised for usage.

References:

3.105 SLA Monitoring and Evaluation in BEinGRID

Keywords:
SLA Monitoring, SLA Evaluation

Abstract / Summary:
BEinGRID provides a component for gathering monitoring data which is then evaluated towards existing SLAs. The notification takes place using a publish/subscribe system, which allows for human and automated recipients.

What is the novelty described in this document?
BEinGRID contains a component called SLA Runtime Monitor, which monitors SLAs and detects violations [1]. In addition this component tries to predict upcoming SLA violations during run-time. The architecture provides a resource monitor for low-level monitoring data, which is then evaluated by the Violation Evaluatator component. This evaluator component has SLAs to evaluate monitoring data against, by using heuristics. Events regarding an SLA violation will be broadcasted to subscribed actors such as human operators and automatic decision controllers. Broadcasts are being executed using a custom publish/subscribe system using WS-Notification. In case of a violation the automatic decision controller is equipped to react and adjust run-time states to prevent an SLA violation or recover from it.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The capabilities provided by the SLA Runtime Monitor are also available in sla@soi in form of multiple components: Low Level Monitoring System, Monitoring Manager and the Provisioning and Adjustment Component. These Components even surpass the features provided by BEinGRID’s monitoring capabilities.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
stable
The mentioned components are available as mature open source software.

References:
### 3.106 IaaS Resource Advanced Reservation

**Keywords:**
IaaS, advanced reservation (AR), virtual machine

**Abstract / Summary:**
In small clouds resources could be limited, and in this case requests for resources have to be prioritized and queued. By preempting, suspending and resuming VMs, AR is actualized and VMs can be efficiently scheduled on multi-host environments. For suspending and resuming VMs, authors estimate the transfer times accurately and schedule the necessary processes or actions with preparation overheads that have to be finished before the starting of other reservation. Thus, when a service provider lacks computational resources, the satisfaction of a reservation request is at the cost of suspending other VMs. And the suspension should also be in compliance with the agreed QoS terms (e.g., availability, reliability etc.) of its VM; otherwise, service provider has to be responsible to that inconsistency. [1]

**What is the novelty described in this document?**
A framework, named Haizea, is implemented for supporting advanced reservation. And this framework is successfully integrated with Opennebula in both simulation testing mode and run-time mode.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**
Generally this is a good reference for implementation of advanced reservation in IaaS scenario.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**
*stable*

**References:**

Keywords:
Grids, advanced reservation, QoS, computational geometry

Abstract / Summary:
By using computational geometry as a means, we can represent the reserved grid resources into computational geometry context. In there, some strategies can be deployed for gaining better resource utilization and guaranteeing QoS terms.

What is the novelty described in this document?
This framework is a good reference for representing the advance reservation into a 2-D diagram, therefore starting time and end-time represent x-axis and y-axis. Besides, there are a lot of corresponding researches for handling the structure of computational geometry so we can extend this work from Grids to Cloud IaaS.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
We can extend this work from Grids to Cloud IaaS.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).
stable

References:

3.108 Virtualized e-Learning on the IRMOS Real-time Cloud

Keywords:
SaaS, PaaS, Resource Provisioning, Cloud

Abstract / Summary:
This paper addresses one of the core issues faced by SaaS providers using the Cloud as a platform for service provisioning and delivery. It highlights the challenge to deploy applications with optimal provisioning of physical resources so that QoS level guarantees can be delivered to the users of application. The paper argues that this is the most important capability required by the SaaS providers in order to calculate exact pricing for the delivered service. The work includes developing a QoS performance model for the software services to offer based on the ISONI cloud test bed using in the project for experimental verification of the ideas. The solution is found in the direction of PaaS suite of services termed the IRMOS Framework Services (FS) that act as mediator between the SaaS and Cloud provider. Using FS, services are modelled and important non functional QoS properties are benchmarked to collect performance results. The performance modelling approach takes into account monitoring data to better understand performance of various QoS metrics when more or less physical resources are available. The major contribution of this work lies in the capabilities of its FS suite that maps high level requirements on application to low level system resources considering several possible combinations while minimizing the cost of the delivered solution.

Described requirements

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

Functional:
An approach to correlate workload inputs on software application and their QoS output considering various hardware settings is developed. Application performance modelling follows a black box approach which does not require much information about the application architecture and dependencies.

Non Functional:
The approach is tested for an e-Learning application but whether it is easily portable to any other application is not too obvious.

What is the novelty described in this document?
The authors claim to have delivered a suite of PaaS services which IaaS providers can install on their Clouds and which then act as a mediator for external SaaS providers for application benchmarking.

How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?
The work provides several similarities to what the SWPOC does for an ORC usecase where a SaaS provider negotiates QoS guarantees with a customer while negotiating (behind the scenes) with an IaaS provider which may be also a Cloud provider. We have used Palladio Component Model (PCM) based performance modelling approach to collect results in a similar manner for a host of infrastructure possibilities available from IaaS provider’s templates. The SWPOC uses an internal Translation component to obtain these results prior to negotiation. Unlike mentioned work, our work on SLA@SOI SWPOC does not rely on benchmarking by requiring Cloud providers to provide such PaaS services.
although we appreciate the idea. Instead, we rely on Service Evaluation component that is capable to predict performance given fixed infrastructure parameters for a VM and workload for software application to be installed on that VM(s). Thereby, we are directed more towards independent third party mediation to provide such performance modelling and prediction capabilities rather expecting from IaaS / Cloud providers to offer it as it wouldn’t be realistic considering the current state of Cloud services offered by current Cloud Computing players. Nevertheless, the work comes close to ours and share several design aspirations and lessons learned, despite difference in adopted approaches. Another differentiator for our work on SLA@SOI is to be able to identify technically and objectively feasible solutions within extremely scarce time constraints since our clients are agents representing customer, SaaS and IaaS providers that engage in automated SLA negotiations.

Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).

Since work conducted is part of IRMOS (Interactive Realtime Multimedia Applications on Service Oriented Infrastructures) project which is European Community’s Seventh Framework Programme FP7 under grant agreement n.214777, it is expected to be available publicly but nothing concrete on this is reported in this paper.

References:


3.109 A Negotiation Protocol Description Language for Automated Service Level Agreement Negotiations

Keywords:
Internet of Services (IoS), SLAs, Negotiation Protocol

Abstract / Summary:

This paper presents a negotiation protocol description language particularly focusing on SLA negotiations. As per the given approach, a protocol is defined as an XML document that reuses and extends the XSD from WSAG. The elements essential for negotiation protocols are identified and expressed in the protocol document with additional support to annotate negotiation rules as well. The work however does not attempt to define any set of rules that would enforce any single mechanism for example to processing offers. It is argued to keep the approach as
generic as possible paving a way for protocol-generic negotiation infrastructures. Some of the important protocol parameters formally addressed in this work include negotiation type, party role, guarantee terms, temporal attributes and more. The work also advocates a strict separation between negotiation protocol description and the negotiation strategy.

**Described requirements**

(i.e. with reference with the addressed topic area, what are the specific functionalities, and quality features of the system described in this work?)

**Functional:**

An example usage of the developed approach is illustrated by implementing an English auction protocol.

**Non Functional:**

Although of interest from design perspective, it is not clear whether the approach sufficiently accommodates for a host of negotiation scenarios possible in SOA and/or IoS based domains.

**What is the novelty described in this document?**

The authors claim to have delivered a generic approach for protocol description and hence moving towards a protocol-generic execution that would broaden interoperability among different IoS based applications.

**How may this work be applicable to SLA@SOI? What are the necessary actions that should be taken to apply it?**

This work was considered while designing and refining our own generic negotiation protocol approach in SLA@SOI. Additionally, an informal contact was also made with the first author and ideas were openly shared and debated. It was exciting to discover that both approaches have derived some common lessons and set forth similar targets albeit obvious differences in the novelty of individual approaches.

**Is there a readily available implementation of the described work? If so, please state its licensing type and its maturity level (e.g. prototype/alpha/beta/RC/stable).**

Nothing reported in this paper.

**References:**

4 ITIL & Service Management Analysis

4.1 Introduction

SLA@SOI aims at developing novel reference architecture for multi-level holistic SLA management [1] for service oriented infrastructures. Numerous approaches and frameworks have been developed over the years to tackle the service level management problem. IT Infrastructure Library (ITIL) being one of the frameworks that attempts to specify a set of concepts, best practices and policies for service providers to carry out effective service management. ITIL is the most widely used framework within the IT organization around the globe and is considered the most sophisticated approach for service management. This document examines both frameworks to identify and reason about the points where both frameworks converge and bear overlapping activities as well as the diverging points between these two frameworks.

While the previous chapter provides a general view on the state of the art in Service / SLA management, in this chapter we specifically compare the ITIL framework and SLA@SOI (as per reviewer’s comments in the interim review). We do not provide here a detailed overview of SLA@SOI framework, for this purpose M12 deliverables, specifically WP A1 deliverable, should be consulted.

This chapter is organized into two main sections. Section 2 introduced ITIL framework in a reasonable depth. Specifically the service lifecycle adopted in ITIL is presented in details in section 2.1. Section 3 delves into the comparison of SLA@SOI and ITIL frameworks. Section 3.1 takes a look at the service lifecycles in both frameworks and identifies the converging and diverging points between the two lifecycles. Section 3.2 provides an overview of the overlapping points between SLA@SOI and ITIL framework and section 3.3 looks at the points where both frameworks diverge from each other. Section 4 provides a summary of this document.

4.2 Information Technology Infrastructure Library (ITIL)

The Information Technology Infrastructure Library (ITIL) is a set of concepts and policies for managing information technology (IT) infrastructure, development and operations.

ITIL is the most widely accepted approach to IT service management in the world. ITIL provides a comprehensive and consistent set of best practices for IT service management, promoting a quality approach to achieving business effectiveness and efficiency in the support and maintenance of information systems. ITIL is published in a series of books, each of which covers an IT management topic. ITIL gives a detailed description of a number of important IT practices with comprehensive checklists, tasks and procedures that any IT organization can tailor to its needs. The current version of ITIL v3 was published
in May 2007 and comprises of five key volumes which also reflects the various phases of the service lifecycle in ITIL. These five volumes cover the following:

1. Service Strategy
2. Service Design
3. Service Transition
4. Service Operations
5. Continual Service Improvements

ITIL is composed of a common sense approach to service management. The following list defines the key characteristics of ITIL that contributes to its global success [2]:

**Non-proprietary:** ITIL service management practices are applicable in any IT organization because these are not tied to any technology platform, or industry types. ITIL is owned by UK Government and is not tied to any commercial proprietary practice or solution.

**Non-prescriptive:** ITIL offers robust, mature and time tested practices that have applicability of all types of organization. It continues to be useful in public and private sectors, internal and external service providers, small, medium and large enterprises and within any technical environment.

**Best Practice:** ITIL service management practices represent the learning experience and thought leadership of world’s best in class service providers.

**Good Practice:** Not every practice in ITIL can be considered ‘best practice’, and for good reason. For many, a blend of common, good and best practices are what give meaning and achievability to ITSM. In some respects, best practices are the flavour of the day. All best practices become common practices over time, being replaced by new best practices.

### 4.2.1 ITIL Service Lifecycle

The ITIL Service Lifecycle contains five elements (as shown in figure 1), each of which relies on service principles, processes, roles and performance measures. The Service Lifecycle adopts a hub and spoke design with Service Strategy at the hub, Service Design, Transition and Operations as revolving lifecycle stages and anchored by Continual Service Improvement. Each part of the lifecycle exert influence on the other and relies on the others for input and feedback. In this way, a constant set of checks and balances across the service lifecycle ensures that as business demand changes with business need, services can adapt and respond effectively to them [2].

At the heart of the service lifecycle is the key principle – all service must provide measurable value to business objectives and outcomes. ITIL service management focuses on business value as its prime objective. Each practise revolve around ensuring that everything a service provider does to manage IT services for the
business customer can be measured and quantified in terms of business value [2].

**Figure 11: ITIL Service Lifecycle**

In the following each of the five ITIL phases (figure 1) is shortly introduced and the management practices defined in each phase are outlined according to [2]. Note, even though the practices are defined in a certain phase, most of them are an issue in multiple phases.

**Service Strategy**

Service Strategy represents the core of the Service Lifecycle and sets the stage for developing a service provider's core capabilities. According to [3] Service Strategy defines an IT organization’s high-level approach to providing services. This includes identifying the market for its services, the identification of service offerings as well as the strategic assets that will constitute those services, and adding the envisioned services to the service portfolio.

Service Strategy comprises the following management practices [3]:

- **Demand Management:** Promoting reduced demand for services as needed by the IT organization. This may include reducing user access, providing user incentives to reduce demand during peak hours, etc.
Financial Management: Managing the accounting, charging, and collection of fees for IT services.

Risk Management: Identifying, evaluating, and determining acceptable responses to risks.

Service Portfolio Management: Managing the list of planned, existing, and retired services.

**Service Design**

Service Design turns Service Strategy into the blueprint for delivering the business objectives. Guidance is provided for the design and development of services and service management practices. Service Design covers design principles and methods for converting strategic objectives into portfolios of services and service assets. This includes the changes and improvements necessary to increase or maintain value to customers over the lifecycle of services, the continuity of services, achievement of service levels, and conformance to standards and regulations [4].

Service Design comprises the following management practices: [4]

- Service Catalogue Management: Provide and manage a widely-accessible catalogue of IT services which directly enable processes that are part of the business.
- Service Level Management: Assure that an agreed level of service is provided to IT customers.
- Capacity Management: Provisioning of a central point for performance and capacity-related analyses and planning. Capacity means the maximum throughput that can be provided by a configuration item (CI). Performance describes the measurable results of a resource, CI, or service.
- Availability Management: Provide a focal point for availability-related analyses and planning. Availability is defined as the ability of an item to perform its function when required.
- Service Continuity Management: IT services should continue to operate according to an agreed-to plan after a major outage.
- Information Security Management: Alignment of IT security with business security through an information system containing information security standards, policies, and procedures.
- Supplier Management: Manage internal or external service providers in support of service level targets.
- Application Management: The management and control of applications through their entire lifecycle, from creation to retirement.
- Data and Information Management: The control, organization, and disposition of data and information within the organization. This includes collection as well as disposal.

**Service Transition**

Service Transition contains the development and improvement of capabilities for transitioning new and changed services into live service operation. It provides guidance on how the requirements of Service Strategy encoded in Service Design are effectively realized in Service Operation while controlling the risks of failure and disruption [5].

Service Design comprises the following management practices: [5]

- Service Asset and Configuration Management: Control and track all CIs to promote integrity in the infrastructure. An asset is a capability or resource that is used in the delivery of a service. There are many types of assets, e.g. management assets, organization assets, process assets, knowledge...
assets, etc. Asset Management is the process that deals with inventory of all service assets. Configuration Management is the process that ensures that CIs within the IT infrastructure are identified, information about the CIs is maintained, and all updates are properly controlled.

- Change Management: Management of all changes to the IT infrastructure in a controlled manner.
- Release and Deployment Management: Build, test, and deploy capabilities to provide services.
- Service Validation and Testing: Assure that a new or changed service will meet customer requirements and will be fit for purpose and fit for use.
- Transition planning and support: Plan service transitions that appear in each stage of an IT service’s lifecycle.
- Knowledge Management: Ensure that the right information is provided to the right roles at the appropriate time (disjointed facts about events, context for data, etc).
- Evaluation: Determine the impact of a proposed service change, e.g. on the performance, whether as a result of a Request for Change, or a new Service Design Package, or testing.
- Communications and Commitment Management: Providing effective communication to all affected parties concerning a change.
- Organizational and Stakeholder Change Management: Managing process and cultural changes among IT stakeholders.
- Stakeholder Management: Resolving the needs and concerns of stakeholders of IT services. Stakeholders may represent a variety of interests, including customers, users, regulatory organizations, business units, partners, and others.

**Service Operations**

Management of the day-to-day operation of services is embodied in Service Operation. Service Operation provides guidance on how to maintain stability in service operations, allowing for changes in design, scale, scope and service levels. Organizations are also provided with knowledge allowing them to make better decisions in areas such as managing the availability of services, controlling demand, optimizing capacity utilization, scheduling of operations and fixing problems. [6]

- Event Management: Identification and resolving of system events that represent failures within CIs.
- Incident Management: Restore service operation to a user as quickly as possible.
- Request fulfilment: Processing of service requests and requests for information. A service request is a standard (pre-approved) change that is straightforward and virtually risk-free.
- Problem Management: Diagnose root causes of incidents, request changes that will resolve those root causes, and reduce the number of future incidents.
- Access Management: Provisioning of rights for a user to access a service.
- Facilities and Data Center Management: Management of the physical location where IT resources are housed.
- Information Security Management: Enforcement of information security policy during service operations.
- IT Operations Management: Operation and management of specific types of technology resources. Support for these resources requires specific types of expertise. Examples are Network Management, Database Administration, Middleware Management, etc.
Continual Service Improvements

Continual Service Improvement aims at creating and maintaining value for customers through better design, transition and operation of services. Principles, practices and methods from quality management, change management and capability improvement are combined. Guidance on Service Measurement, demonstrating value with metrics, developing baselines and maturity assessments are the main topics of Continual Service Improvement [7].

The strength of the ITIL Service Lifecycle rests upon continual feedback throughout each stage of the lifecycle. This feedback ensures that service optimization is managed from a business perspective and is measured in terms of the value, which business derives from services at any point in time through the Service Lifecycle. The following figure shows some examples of the continual feedback system within the Service Lifecycle.

4.3 ITIL vs. SLA@SOI: A Comparison

This section aims at comparing both SLA@SOI and ITIL frameworks. We will try to cover areas where both frameworks adopt similar approaches as well as contrasting aspects of both frameworks. This section is divided into three subsections; first subsection addresses the service lifecycle issues, second and third subsection discuss the converging and diverging points of both frameworks respectively.
4.3.1 **SLA@SOI vs ITIL: Service Lifecycle**

Service Lifecycle is present in both SLA@SOI and ITIL frameworks. Both lifecycle share some commonalities and some differences. ITIL Service lifecycle was discussed in section 2.1. Figure 1 represents the ITIL service lifecycle and figure 2 given below shows the service lifecycle followed in SLA@SOI.

In this subsection, we will use SLA@SOI’s lifecycle as reference point and will discuss how lifecycle followed in SLA@SOI compares against ITIL’s service lifecycle. Before we delve into the comparison discussion, one thing worth mentioning is that despite overall conceptual equivalence of both lifecycles, some phases do not exist explicitly in SLA@SOI’s lifecycle; however, the activities performed within those phases are still present as will be described later.

![Figure 12: SLA@SOI Service Lifecycle](image)

- There is no explicit Service Strategy phase in SLA@SOI; however the tasks performed in this phase (e.g. demand management, risk management, service portfolio management) in ITIL are applied in the initial phases of SLA@SOI’s lifecycle e.g. in Service offering phase. Activities like demand management and service portfolio management are the typical activities which will be performed during the service offering phase. Moreover, some activities from the Service Design phase of ITIL’s lifecycle also appears under the service offering phase of SLA@SOI lifecycle e.g. Service Catalogue Management, Supplier management etc.
- Service Negotiation Phase: This phase is out of scope of the service lifecycle of ITIL framework. ITIL assumes the agreements to have happened out of band and doesn’t prescribe any methodologies and protocols for SLA negotiation.
On the other hand, SLA / Service negotiation is an important phase of the SLA@SOI service lifecycle. SLA@SOI aims at developing an automated and standards based SLA/Service negotiation framework especially taking into account complex, multi-round negotiations.

- Service Provisioning Phase: Again, there is no explicit phase within ITIL’s service lifecycle; however, some of the activities performed in other phases like Service Design and Service Transitions are covered in the Service Provisioning phase of the SLA@SOI lifecycle. During the Service Provisioning phase, service providers engage in activities to procure, install, configure, validate and test the resources required to deliver the services according to the agreed.
- Service Operations phase is equivalent to the ITIL’s service operations. This phase involves the control loop where services are continuously monitored and analyze for potential SLA violations. In this regard, the activities from ITIL’s service operations like Event management, incident management, IT operations management are applicable equally well in the SLA@SOI service operations phase.

The points where SLA@SOI and ITIL lifecycles differ explicitly are:

- SLA@SOI service lifecycle does not have an explicit Continual Service Improvement phase; however, service operations, service provisioning and service negotiation phases make up for this deficiency. This shown in the figure 2 with cyclic arrows around the service operations and service provisioning phases. Continuous monitoring, analysis and adjustment activities serve the purpose to carry out continuous service improvements.
- Service Decommissioning phase as present in SLA@SOI lifecycle doesn’t explicitly exist within ITIL framework’s service lifecycle. Services provisioned for a fixed time period needs to be phased out and necessary activities like cleaning up databases, archiving required data, finalizing legal issues etc are covered within the decommissioning phase.

4.3.2 SLA@SOI vs ITIL: Convergence

This section discusses the points where both framework have overlapping areas, in other words both frameworks converge on these issues. These converging issues include:

Service Definition

Both SLA@SOI and ITIL have the same notion of a service. Actually, SLA@SOI adopts the definition of service as crafted by ITIL framework. The definition can be found in the WP A1 deliverable in the glossary section. This helps us take a broader view and address and larger set of services.

Broad Applicability

Like ITIL, SLA@SOI has a broader focus and aims at managing a wider variety of services. This is evident from the heterogeneous use cases to be developed within SLA@SOI. Some of these use cases include ERP Hosting, Enterprise IT services, eGovernment services, Financial Grids services etc.


**Service Lifecycle**

Both ITIL and SLA@SOI adopts a comprehensive and conceptually equivalent service lifecycle. The lifecycle spans the complete spectrum of activities involved in the service delivery and management.

**Non-proprietary**

As mentioned previously ITIL aims at developing a non-proprietary solution without being tied to any technology. SLA@SOI has similar objectives and plans to develop standards based and open framework without any dependency on a technology or vendor.

**Configuration Management Database (CMDB)**

Like ITIL, SLA@SOI employs a Configuration Management Database concept. CMDB is used throughout the service lifecycle. All the activities consume information from and produce information into the CMDB.

**4.3.3 SLA@SOI vs ITIL: Divergence**

Previous subsection presented the overlapping points between ITIL and SLA@SOI. The objective of this subsection is to present the points where both frameworks have dissimilarities or in other words areas where ITIL and SLA@SOI diverge from each other. These diverging points include:

**Architecture vs Best Practices**

ITIL provides a set of concepts, best practices and policies where SLA@SOI aims at delivering a reference architecture and an open source proof-of-concept implementation of the reference architecture which can be used by various service providers. Despite this being a difference between SLA@SOI and ITIL, it is worth pointing out that SLA@SOI aims at enabling service providers to adopt the best practices through its reference architecture implementation.

**SLA/Service Negotiation**

SLA@SOI plans to provide a sophisticated automated SLA negotiation apparatus. ITIL on the other hand doesn’t tackle the SLA negotiation topic and assumes the presence of established SLAs between the customers and service providers. Similarly renegotiation of SLAs for ongoing services is also out of scope of ITIL but SLA@SOI plans to address this issue.

**Multi-provider Focus**

ITIL aims at providing best practices for a single service provider whereas SLA@SOI also addresses the cross domain scenarios involving multi-providers and multiple organization domains. These multi-provider and multi-domain scenarios going to be typical scenarios in the emerging cloud ecosystem; therefore, it becomes utmost important to have manageability infrastructure in place to sustain and manage these scenarios.
**SLA Translation**

One of the main objectives of SLA@SOI is to design algorithms and methodologies to translate high level business SLAs to infrastructure SLAs and requirements. This translation enables accurate service provisioning to sustain the QoS guarantees agreed in the SLAs with the customers. SLA negotiation and planning is out of scope of ITIL framework, hence, ITIL framework and best practices don’t include any SLA translation methodologies or best practises.

**Predictive Management**

ITIL framework provides concepts, best practises and policies for service management; however, it only addresses reactive service management. SLA@SOI on the other hand has strong focus on prediction services facilitating a service provider to engage in predictive service management before any SLA violation. The prediction services address both software services as well as the infrastructure (predictive workload management). This predictive management capability gives SLA@SOI framework and critical edge.

**Federated CMDB**

Although both SLA@SOI and ITIL frameworks adopt a CMDB driven approach, SLA@SOI still has a different perspective. In SLA@SOI, there is no single centralized CMDB but multiple individual databases storing information like infrastructure, software services, SLAs etc. independently. SLA@SOI's approach resembles the federated CMDB [8] approach whereby all these independent data sources are federated into a single logical data source.

### 4.4 Conclusions on ITIL

SLA@SOI and ITIL framework both aims at enabling service providers to carry out effective service management activities within their organizations. Both approaches bear some similarities as well as possess some distinct capabilities. In this document we examined ITIL and compared it with SLA@SOI framework to identify how both approaches converge and diverge. First, we compared the service lifecycle present in both approaches. Service lifecycles from both SLA@SOI and ITI are conceptually equivalent; however the phases are labelled differently. Additionally, some of the lifecycle phases in ITIL don’t have a counterpart in SLA@SOI lifecycle but the activities carried out within these phases are still covered in other phases of the SLA@SOI lifecycle. Secondly, we discussed the converging points between SLA@SOI and ITIL. Finally, the diverging points were examined especially focusing on the area which are covered by SLA@SOI but are not within the scope of ITIL. It could be undertaken in the future to identify how some complementary approaches can be identified for the areas where both approaches diverge from each other.

### 4.5 Self-managing systems

SLA-driven system management is the primary approach discussed in this paper. It actually tries to derive all kinds of management decisions from the requested or agreed service level agreements. However, there are also other management functions which are partially related to SLA management. As reference structure for these functions we use the 4 main categories of self-managing systems [95], namely self-configuring, self-healing, self-optimizing, and self-protecting.
Configuration management is closely related to SLA management. Possible configuration options are captured in the service construction model and are taken into consideration during the planning phase. Once, an SLA has been agreed and is to be provided, the configuration parameters derived during planning phase are used to set up the system. The same holds for replanning/adaptation cycles.

Self-healing is in the first place independent from SLA management as the detection and recovery from low-level unhealthy situations can be done completely independent from agreed SLAs. However, the detection of SLA violations and the automated re-planning could be also understood as self-healing process. Furthermore, low-level unhealthy situations might be used for predicting possible future SLA violations.

Self-optimizing as very closely related to SLA management and simply cannot be done without taking into account the respective constraints of the contracted SLAs.

Self-protecting is in the first place independant from SLA management. However, certain self-protecting mechanisms can be made part of an SLA.
5 References

[1] "MIL-STD-1629A PROCEDURES FOR PERFORMING A FAILURE MODE, EFFECTS


[34] XMPP, available at http://www.xmpp.org


[37] SecSe, available at http://www.secese-project.eu/

International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM’97) (1997)


[80] JOpera. Available at http://www.jopera.org


[83] Werner Vogels, “Eventually Consistent”, Queue, v.6 n.6, October 2008


Appendix A: Abbreviations

- BCM: Business Continuity Management
- BLO: Business Level Objective
- BNF: Backus-Naur Form
- BPEL: Business Process Execution Language
- CARE: Condition, Action, Resource and Environment
- CIM: Common Information Model
- DMTF: Distributed Management Task Force
- FLEA: Formal Language for Expressing Assumptions
- FMECA: Failure Modes, Effects, and Critical Analysis
- FTA: Fault Tree Analysis
- GIF: Governance Interoperability Framework
- IaaS: Infrastructure as a Service
- ICT: Information and Communication Technology
- ISM: Infrastructure Service Manager
- JSON: JavaScript Object Notation
- KPI: Key Performance Indicator
- LDAP: Lightweight Directory Access Protocol
- MOF: MetaObject Facility
- MTTF: Mean Time to Failure
- MTTR: Mean Time to Repair
- OCCI: Open Cloud Computing Interface
- OGF: Open Grid Forum
- OVF: Open Virtual Machine Format
- ROPE: Risk-aware Process Evaluation
- PaaS: Platform as a Service
- PCM: Palladio Component Model


Qos  Quality of Service
SaaS  Software as a Service
SLA  Service Level Agreement
SOA  Service Oriented Architecture
TIP  Threat Impact Processes
UDDI  Universal Description Discovery and Integration
USDL  Universal Service Description Language
WSDM  Web Services Distributed Management
WSLA  Web Service Level Agreement
XML  Extensible Markup Language