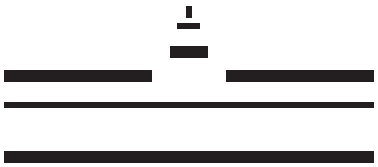
Bastian Baranski

# Service Level Agreements in Spatial Data Infrastructures

2012

Geoinformatik



Westfälische
Wilhelms-Universität
Münster

# Service Level Agreements in Spatial Data Infrastructures

Inaugural-Dissertation
zur Erlangung des Doktorgrades der Naturwissenschaften
im Fachbereich Geowissenschaften
der Mathematisch-Naturwissenschaftlichen Fakultät
der Westfälischen Wilhelms-Universität Münster

vorgelegt von
Bastian Baranski
aus Moers, Deutschland

2012

| | |
|---|---|
| Dekan: | Prof. Dr. Hans Kerp |
| Erstgutachter: | Prof. Dr. Edzer Pebesma |
| Zweitgutachter: | Prof. Dr. Achim Streit |
| Tag der mündlichen Prüfung: | 14. Juni 2013 |
| Tag der Promotion: | ..... |

# Abstract

The Spatial Data Infrastructure (SDI) development is driven by public authorities, like national mapping or environmental agencies. Most of the governmental SDI stakeholders from state, national and regional level are mandated by legal frameworks to promote the SDI development. One example for such a legal framework is the Infrastructure for Spatial Information in the European Community (INSPIRE) directive which aims at building an European SDI based on the member states' national SDIs. Not only SDI service providers but all kinds of Geographic Information (GI) experts will continue to be faced with a rapidly increasing volume of available geospatial data with a higher spatial, temporal, and thematic resolution. Furthermore, the on-demand access to all kinds of geospatial data becomes more and more relevant for many application domains, as for instance eGovernment applications providing citizens with basic access to geographic information and Spatial Decision Support Systems (SDSS) facilitating disaster management.

All these developments result in ambitious requirements regarding the reliability, performance and scalability of SDI services and corresponding applications. Some of the application domains may have challenging requirements regarding the service quality, while others may have lower requirements. However, in all application domains it is important for service providers to be aware of the minimum service quality that shall be delivered to individual customers or in an application context. Furthermore, it is always important for service providers to be in a position to actually deliver the expected and promised service quality level. These and related aspects can be summarized under the heading of Service Level Management (SLM). An integral part of SLM is the Service Level Agreement (SLA), which is a negotiated contract that formalizes business relationships between service providers and service consumers in order to measure, manage and enforce certain service quality guarantees.

This thesis develops a concept for the integration of SLAs in SDIs. The selected multi-step approach involves the development of an abstract SLA model and a web-based SLA management architecture. The aim of the abstract SLA model is to describe the domain-specific structure and content of SLAs that can be applied in SDIs from a conceptual point of view. The purpose of the web-based SLA management architecture is to enable the on-demand and online negotiation of SLAs in established SDIs without the need of prior offline communication between service providers and service consumers. The selected policy-based approach covers not only agreement negotiation and service consumption, but also the complete agreement life cycle including service monitoring and agreement evaluation. This thesis also describes an approach on how to realize common domain-specific service quality requirements in SDIs under the terms of

previously negotiated SLAs by means of an Hybrid Cloud approach. The presented Hybrid Cloud architecture helps SDI service providers to match the basic INSPIRE service quality requirements without investing in rarely used hardware in advance. The idea of the proposed Hybrid Cloud architecture is always to provide sufficient computational resources in order to achieve a constant average service response time, which should be independent of the number of users requesting a service.

The presented concepts for SLA representation and integration are implemented as a proof-of-concept. The 'WS-Agreement Application Profile for OGC Web Services' is a mapping of the abstract SLA model to an extended and particular version of the Web Services Agreement Specification (WS-Agreement). The 'Service Level Agreements for OGC Web Services (SLA4OWS)' framework enables SDI service providers to offer different service quality levels and pricing models for existing SDI services. The SLA4OWS framework utilizes Cloud Computing infrastructures to implement the offered service levels in an economical fashion. The '52°North Hybrid Cloud' is an implementation of the proposed Hybrid Cloud architecture and can be integrated in the SLA4OWS framework. By incorporating Amazon Elastic Compute Cloud (EC2) resources into the local Information Technology (IT) infrastructure, the proposed architecture offers potentially unlimited resources on-demand and nearly in real-time.

# Zusammenfassung

Viele öffentliche Behörden auf nationaler und regionaler Ebene sind durch rechtliche Rahmenbedingungen dazu verpflichtet die Entwicklung von Geodateninfrastrukturen (GDI) voranzutreiben. Ein Beispiel für einen solchen rechtlichen Rahmen ist die Infrastructure for Spatial Information in the European Community (INSPIRE). Bei INSPIRE handelt es sich um eine Initiative der europäischen Kommission mit dem Ziel eine europäische GDI, basierend auf den nationalen GDI der einzelnen Mitgliedsstaaten, aufzubauen. Auch unabhängig von rechtlichen Anforderungen steigt der Bedarf nach allen Arten von raumbezogenen Daten, die für immer mehr Anwendungsbereiche von großer Relevanz sind. Zu diesen Anwendungsbereichen gehören beispielsweise Anwendungen im Bereich E-Government, die Bürgern einen möglichst einfachen Zugang zu geografischen Basisinformationen geben sollen, aber auch alle Geographischen Informationssysteme (GIS) die im Katastrophenmanagement eingesetzt werden und speziellen Anforderungen gerecht werden müssen. Darüber hinaus sind die verfügbaren raumbezogene Daten in einer immer höheren räumlichen, zeitlichen und thematischen Auflösung verfügbar.

All diese Entwicklungen führen zu anspruchsvollen Anforderungen hinsichtlich der Zuverlässigkeit, Leistung und Skalierbarkeit von Dienste innerhalb einer GDI und den darauf aufbauenden Anwendungen. Einige der Anwendungsbereiche haben beispielsweise sehr anspruchsvolle Anforderungen an die Verfügbarkeit der zugrundeliegenden Dienste, während in anderen Anwendungsbereichen eine minimale Dienstgüte (Quality of Service, QoS) akzeptiert werden kann. Für Dienstanbieter ist es generell wichtig, welches QoS-Level den jeweiligen Nutzern in einem Anwendungskontext zur Verfügung gestellt werden soll. Des Weiteren ist es für den Dienstanbieter von großer Bedeutung wirklich in der Lage zu sein, die geforderten QoS-Level zu jedem Zeitpunkt zur Verfügung zu stellen. Diese und weitere Aspekte werden unter Begriff 'Service Level Management (SLM)' zusammengefasst. Ein elementarer Bestandteil von SLM ist das Service Level Agreement (SLA), ein ausgehandelter Vertrag welcher die Geschäftsbeziehung zwischen einem Dienstanbieter und einem Dienstnutzer regelt. Neben allgemeinen Informationen zum Vertrag und eventuell anfallenden Nutzungskosten gehören zu einem SLA unter anderem die so genannte Key Performance Indicators (KPIs) und Service Level Objectives (SLOs), welche die genau zu erbringende QoS festlegen.

Die vorliegende Arbeit entwickelt ein Konzept für die Integration von SLAs in GDIs. Der ausgewählten mehrstufigen Ansatz beinhaltet die Entwicklung eines abstrakten SLA-Modells und einer web-basierten SLA-Management-Architektur. Das Ziel des abstrakten SLA-Modells ist die konzeptionelle Beschreibung der Struktur und des

Inhaltes von SLAs speziell für die ausgewählten Anwendungsbereiche. Der Zweck der web-basierten SLA-Management-Architektur ist es, die (Online-) Aushandlung von SLAs in bereits existierenden GDIs zu ermöglichen, ohne dass eine vorherige (Offline-) Kommunikation zwischen Dienstanbieter und Dienstnutzer vonnöten ist. Der gewählte Policy-basierte Ansatz deckt nicht nur die Aushandlung von SLAs und die eigentliche Dienstnutzung ab, es wird der vollständige Lebenszyklus von SLAs unterstützt. Dazu gehört sowohl die permanente Überwachung der angebotenen Dienste als auch die permanente Evaluierung aller aktiven SLAs. Die vorliegende Arbeit beschreibt im Weiteren einen Hybrid Cloud-basierten Ansatz, mit dem Dienstanbieter die QoS-Anforderungen von INSPIRE unter Berücksichtigung aller zuvor ausgehandelten SLAs effektiv umsetzen können. Die Idee der vorgeschlagenen Architektur ist durch eine Kombination von lokaler Infrastruktur (Private Cloud) und externen Ressourcen (Public Cloud) immer genügend Rechenkapazität bereitzustellen, um konstante Antwortzeiten unabhängig von der Anzahl der Nutzer eines Dienstes zu realisieren.

Die entwickelten Konzepte zur Repräsentation und Integration von SLAs werden in der vorliegenden Arbeit prototypisch umgesetzt. Das 'WS-Agreement Application Profile for OGC Web Services' bildet das abstrakte SLA-Modell auf eine konkrete Version (Application Profile) der 'Web Services Agreement Specification (WS-Agreement)' ab. Das 'Service Level Agreements for OGC Web Services (SLA4OWS)' Framework ermöglicht es Dienstanbietern unterschiedliche QoS-Level und Preismodelle für existierende Dienste in einer GDI anbieten zu können. Die '52°North Hybrid Cloud' ist eine Open Source Implementierung der Hybrid Cloud-Architektur zur Realisierung der QoS-Anforderungen von INSPIRE. Sie kann als integraler Bestandteil des SLA4OWS-Framework verwendet werden, um Ressourcen von Amazon Elastic Compute Cloud (EC2) in die lokale Infrastruktur zu integrieren. So können die angebotenen QoS-Level nicht nur auf einer technischen Ebene sondern auch unter ökonomischen Gesichtspunkten realisiert werden.

# Acknowledgements

I would like to take the opportunity and thank all the people without whom this thesis would have been impossible.

First, I have to thank my supervisor Prof. Dr. Edzer Pebesma for considering this thesis promising enough to give it a try and for helping me to shape my ideas and visions in many fruitful discussions. A big thanks goes to Prof. Dr. Ulrich Streit not only for giving my the opportunity to work at the Institute for Geoinformatics but also for being a competent and inspiring contact person even after his retirement. In addition, I have to thank my second supervisor Prof. Dr. Achim Streit for helping me to bridge the gap between two disciplines by embedding me in his former division at Jülich Supercomputing Centre (JSC).

This thesis would have been impossible without a supportive and motivating work environment. In particular, I want to thank the members of the Sensor Web, Web-based Geoprocessing and Simulation Lab (SWSL) working group at the Institute for Geoinformatics for providing deep insights into various research topics and for helping me to keep the big picture behind this thesis in mind. A special thanks goes to everyone who was involved in the 52°North community for providing an open-minded network of great researchers and "coding monkeys". I would also like to thank the con terra GmbH - especially Christian Elfers - and the GeoMobile GmbH - especially Dr. Michael Gerhard - for giving me the opportunity and the freedom to work responsibly on this thesis beside my regular day jobs.

I would like to highlight a few people for being there and supporting me over all the years. I want to thank Rüdiger Gartmann and Dr. Roland Wagner for arousing my interest in the field of Geoinformatics during my time at the Fraunhofer ISST. A special thanks goes to Dr. Bastian Schäffer and Dr. Theodor Foerster for giving me constant support and valuable feedback over the last years.

Finally, I have to thank people from outside research. Thanks to my parents Helga and Rainer for their constant support and belief in me over all the years. Thanks to my friends for their support and their sense of humor. Most of all, I have to thank Iris for her unlimited support, patience and love.

# Publications

This thesis is based on ideas, fragments and figures that have appeared previously in the following publications.

### Journals

Díaz, L., Remke, A., Kauppinen, T., Degbelo, A., Foerster, T., Stasch, C., Rieke, M., Schaeffer, B., Baranski, B., Broering, A., and Wytzisk, A. (2012). Future SDI – Impulses from Geoinformatics Research and IT Trends. *International Journal of Spatial Data Infrastructures Research (IJSDIR)*, Volume 7

Baranski, B., Foerster, T., Schäffer, B., and Lange, K. (2011). Matching INSPIRE Quality of Service Requirements with Hybrid Clouds. In Wilson, J. P., Stewart Fotheringham, A., and O'Sullivan, D., editors, *Transactions in GIS*, volume 15, pages 125–142. Wiley Online Library

### Book Chapters

Schäffer, B., Baranski, B., Foerster, T., and Brauner, J. (2012). A Service-Oriented Framework for Real-time and Distributed Geoprocessing. *Geospatial Free and Open Source Software in the 21st Century. Lecture Notes in Geoinformation and Cartography*, pages 3–20

Foerster, T., Schäffer, B., Brauner, J., and Baranski, B. (2011). Geospatial Web Services for Distributed Processing - Applications and Scenarios. In Zhao, P. and Di, P., editors, *Geospatial Web Services: Advances in Information Interoperability*, pages 245–286. Hershey

Baranski, B. and Schäffer, B. (2010). Towards Service Level Agreements in Spatial Data Infrastructures. In Rajabifard, A., Crompvoets, J., Kalantari, M., and B., K., editors, *Spatially Enabling Society: Research, Emerging Trends, and Critical Assessment*, pages 149–162. Leuven University Press

Schäffer, B., Baranski, B., and Foerster, T. (2010b). Towards Spatial Data Infrastructures in the Clouds. In Painho, M., Santos, M. Y., and Pundt, H., editors, *Geospatial Thinking. Lecture Notes in Geoinformation and Cartography*, pages 399–418. Springer

### Conferences (Full Paper)

Baranski, B. (2012). The Service Level Agreements for OGC Web Services (SLA4OWS) Framework. In Löwner, M., Hillen, F., and Wohlfahrt, R., editors, *Geoinformatik 2012 -*

*Mobilität und Umwelt*, pages 383–388. Shaker Verlag

Foerster, T., Baranski, B., Schäffer, B., and Lange, K. (2010). Geoprocessing in Hybrid Clouds. In Zipf, A., Behncke, K., Hillen, F., and Schaefermeyer, J., editors, *Die Welt im Netz*, pages 13–19. Akademische Verlagsgesellschaft

Schäffer, B., Baranski, B., and Foerster, T. (2010a). Licensing OGC Geoprocessing Services as a Foundation for Commercial Use in SDIs. In *Second International Conference on Advanced Geographic Information Systems, Applications and Services*, pages 111–116. IEEE Computer Society

Baranski, B., Schäffer, B., and Redweik, R. (2010b). Geoprocessing in the Clouds. In *OSGeo Journal*, volume 8, pages 17–22. Open Source Geospatial Foundation (OSGeo)

Baranski, B., Deelmann, T., and Schäffer, B. (2010a). Pay-per-Use Revenue Models for Geoprocessing Services in the Cloud. 1st International Workshop on Pervasive Web Mapping, Geoprocessing and Services (WebMGS 2010). Como, Italy.

Chung, L., Fang, Y., Chang, Y., Chou, T., Lee, B., Yin, H., and Baranski, B. (2009). A SOA based debris flow monitoring system. In *Proceedings of the 17th International Conference on Geoinformatics, 2009*, pages 1–6. IEEE

Brauner, J., Foerster, T., Schäffer, B., and Baranski, B. (2009). Towards a Research Agenda for Geoprocessing Services. In J. Haunert, B. K. and Milde, J., editors, *Proceedings of 12th AGILE International Conference on Geographic Information Science*. Hanover, Germany: IKG, Leibniz University of Hanover

Baranski, B. (2008). Grid Computing Enabled Web Processing Service. In Bishr, M., Pebesma, E., and Bartoschek, T., editors, *Proceedings of the 6th Geographic Information Days. Ifgi Prints.*, volume 32, pages 243–256

**Conferences (Abstract)**

Baranski, B. (2009). Guaranteeing Quality of Service in a Spatial Data Infrastructure by using Service Level Agreements. Presented at GSDI 11 World Conference, Rotterdam, The Netherlands

**Professional Publications**

Baranski, B. (2011). WS-Agreement Application Profile for OGC Web Services. Open Geospatial Consortium (OGC), OGC 11-094 (Discussion Paper)

Baranski, B., Woolf, A., Shaon, A., and Kurzbach, S. (2009). OWS-6 WPS Grid Processing Profile Engineering Report. Open Geospatial Consortium (OGC), OGC 09-041 (Engineering Report)

# Contents

# List of Figures

# List of Tables

# List of Listings

# List of Abbreviations

| | |
|---|---|
| AdV | Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland |
| API | Application Programming Interface |
| BNF | Backus Naur Form |
| BPEL | Business Process Execution Language |
| CPU | Central Processing Unit |
| CRS | Coordinate Reference System |
| CSW | OGC Catalogue Service |
| DRM | Digital Rights Management |
| DSL | Domain Specific Language |
| EC2 | Elastic Compute Cloud |
| EPR | Endpoint Reference |
| EPSG | European Petroleum Survey Group Geodesy |
| ERM | Entity Relationship Model |
| EU | European Union |
| GAE | Google App Engine |
| GDI-DE | Geodateninfrastruktur in Deutschland |
| GeoDRM | Geospatial Digital Rights Management Reference Model |
| GeoXACML | Geospatial eXtensible Access Control Markup Language |
| GI | Geographic Information |
| GIS | Geographic Information System |
| GML | Geography Markup Language |
| GRAAP | Grid Resource Allocation Agreement Protocol |
| HTTP | Hypertext Transfer Protocol |
| ICT | Information and Communications Technology |
| IETF | Internet Engineering Task Force |

| | |
|---|---|
| INSPIRE | Infrastructure for Spatial Information in the European Community |
| IP | Internet Protocol |
| IR | Implementing Rules |
| ISO | International Organization for Standardization |
| IT | Information Technology |
| ITU | International Telecommunications Union |
| JEXL | Java EXpression Language |
| JSON | JavaScript Object Notation |
| KPI | Key Performance Indicator |
| NIST | National Institute of Standards and Technology |
| NSDI | National Spatial Data Infrastructure |
| OASIS | Organization for the Advancement of Structured Information Standards |
| OCCI | Open Cloud Computing Interface |
| OGC | Open Geospatial Consortium |
| OGF | Open Grid Forum |
| OGSA | Open Grid Services Architecture |
| ORM | OGC Reference Model |
| OS | Operating System |
| OSI | Open Systems Interconnection |
| OWS | OGC Web Services |
| PDP | Policy Decision Point |
| PEP | Policy Enforcement Point |
| QoS | Quality of Service |
| RAM | Random Access Memory |
| REST | Representational State Transfer |
| S3 | Simple Storage Service |
| SDI | Spatial Data Infrastructure |
| SDSS | Spatial Decision Support Systems |
| SLA | Service Level Agreement |
| SLA4D-Grid | Service Level Agreements for D-Grid |

| | |
|---|---|
| SLA4OWS | Service Level Agreements for OGC Web Services |
| SLM | Service Level Management |
| SLO | Service Level Objective |
| SOA | Service Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SOS | Sensor Observation Service |
| UML | Unified Modeling Language |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| URN | Uniform Resource Name |
| VermWertGebT | Gebührenordnung für das amtliche Vermessungswesen und die amtliche Grundstückswertermittlung in Nordrhein-Westfalen |
| VM | Virtual Machine |
| W3C | World Wide Web Consortium |
| WCS | Web Coverage Service |
| WFS | Web Feature Service |
| WMS | Web Map Service |
| WPS | Web Processing Service |
| WS-Agreement | Web Services Agreement Specification |
| WSDL | Web Services Description Language |
| WSRF | Web Services Resource Framework |
| XACML | eXtensible Access Control Markup Language |
| XML | Extensible Markup Language |
| XSD | XML Schema Document |

# Chapter 1

# Introduction

This thesis develops a concept for the integration of Service Level Agreements in Spatial Data Infrastructures. This chapter provides the motivation for the conducted research. Based on the motivation, the concrete objectives and research questions of this thesis are defined. The research approach selected for this thesis is described in the methodology section. After an overview about frequently used terms and definitions, this chapter ends with an overview of the thesis structure.

## 1.1 Motivation

Geographic Information System (GIS) [De Smith et al., 2007] have been under constant development in recent years. Emerging web services concepts and technologies such as the Service Oriented Architecture (SOA) paradigm [Papazoglou, 2003] encouraged the evolution from classical desktop- and data-centric GIS to distributed and loosely-coupled architectures composed of open and interoperable web services merged into the Spatial Data Infrastructure (SDI) concept [Groot and McLaughlin, 2000]. In the past, open standards based SDIs - for instance based on standards developed by the Open Geospatial Consortium (OGC) – focused on the development of standards for the retrieval, portrayal and processing of geospatial data through web services [Kiehle et al., 2006]. The development of such open and interoperable standards for geospatial data and services is one important key factor for the sustainable success of the SDI concept in the Geographic Information (GI) domain [Bank, 2004]. But the SDI concept collects not only (web service) technologies but also policies, human resources and institutional arrangements in order to facilitate the availability of and access to geospatial data [Nebert, 2004].

The development of SDIs is mostly driven by public authorities, like national mapping or environmental agencies [Masser, 2005]. The applications developed by the government and that run on such infrastructures are manifold. They range from eGovernment applications providing citizens with basic access to geographic information [Nogueras-Iso et al., 2004] to Spatial Decision Support Systems (SDSS) facilitating disaster management [Rajabifard et al., 2004] not only for public authorities but also for non-governmental organizations [DeCapua and Bhaduri, 2007]. Furthermore, most of the governmental SDI stakeholders from state, national and regional level are mandated by legal frameworks to promote the SDI development. One popular example for such a legal framework is the Infrastructure for Spatial

Information in the European Community (INSPIRE) directive [EU, 2007], which aims at building an European SDI based on the member states' national SDIs. However, the SDI development is fostered not only by legal frameworks but also by a great potential for enabling the market value of geospatial data as for instance shown in [Frick et al., 2002] and [Fornefeld et al., 2003].

The current GI research agenda will continue to be faced with emerging challenges as for instance the handling of ambiguous data quality in volunteered geographic information [Goodchild and Glennon, 2010], the integration and analysis of real-time data from both sensors and humans [Craglia et al., 2008], the seamless integration of heterogeneous data from multiple sources following the Linked Open Data concept [Ortmann et al., 2011], and shifting the classic map metaphor to the third, fourth, and fifth dimension in order to overcome the constraints of the two-dimensional thinking in GIS [Goodchild, 2010]. Even long-known open issues such as the missing semantic interoperability between geospatial information especially across different information communities [Bishr et al., 1999] are still not solved completely [Wytzisk and Sliwinski, 2004].

Not only SDI service providers but also all kinds of GI experts will continue to be faced with a rapidly increasing volume of available geospatial data with a higher spatial, temporal, and thematic resolution. The on-demand access to all kinds of geospatial data becomes more and more relevant for many application domains. The integration of map-based models for community representations in web-sites are an important incentive for citizens to participate in eParticipation and eGovernment solutions [Carver, 2003]. The producers and holders of public geographical information are obligated to realize the high economic potential of geospatial information [Fornefeld et al., 2008] by allowing third-parties to mash up governmental data and services to new applications for instance by means of subscription-based revenue models [Donker, 2009]. Many of the critical problems that arise in disaster management are inherently spatial and can be solved with spatial decision support systems [Densham, 1991]. Emerging technologies such as the high accurate GPS-based localization in smartphones in combination with haptic devices such as vibrating wristbands foster the development of novel context- and location-aware mobile assistant systems that can help elderly people or people with visual impairments to use mainstream map services without frustration [Anastassova et al., 2010]. These are just a few examples that show how not only the available volume of geospatial data but also the potential number of users will continue to increase.

Along with emerging laws and provisions such as the INSPIRE directive, these developments result in ambitious requirements regarding the reliability, performance and scalability of geospatial services and corresponding applications. Some of the application domains may have challenging requirements regarding the service quality, while others may have lower requirements. For example in disaster events the utilized geospatial services "should exhibit high levels of availability and resilience" and the answers to urgent questions "are expected in near real-time" [Onchaga, 2005], whereas in a user-centric approach the service delivery in eGovernment applications is focused on the overall user satisfaction which not only depends on strong service performance criteria but also on other "soft" factors [Alanezi et al., 2010].

In all application domains it is important for service providers to be aware of the minimum service quality that shall be delivered to individual customers or in an application context. From the service consumer perspective, in many application domains it is important to find a service that not only implements specific functionality and offers specific resources, but also guarantees a specific service quality level. Furthermore, it is always important for service providers to be in a position to actually deliver the expected and promised service quality level. This includes sufficient available computing capacity and the ability to (automatically) manage the computing infrastructure according to the service consumers' expectations and the varying overall system load. However, the permanent monitoring and reporting of service performance helps service providers to react quickly on service quality fluctuations in order to consequently meet the users' expectations, which is an essential skill in future and potential highly competitive GIS markets [Donker, 2009]. Furthermore, the permanent monitoring and reporting of service performance enables service consumers to check whether a service provider really provides the promised service quality level.

Scenarios in which the delivered service quality varies automatically according to the individual service consumers' requirements or the application context are known as multi-channel service delivery [Patricio et al., 2009] or differentiated service provisioning [Eggert and Heidemann, 1999]. The realization of multi-channel service delivery can achieve cost savings by higher levels of efficiency, but constantly providing reliable and high service quality levels also results in additional costs for service maintenance [EC, 2004]. Therefore, in some application domains it is worthwhile for service providers to act as value added service providers by means of revenue models that forward infrastructure and related costs to service consumers. From the service consumer perspective, in some application domains it may be important to perform a benefit-cost ratio analysis of different service offerings in order to find an adequate service level at acceptable costs.

All these aspects can be summarized under the term of 'Service Level Management (SLM)', which is "the disciplined, proactive methodology and procedures used to ensure that adequate levels of service are delivered to all IT users in accordance with business priorities and at acceptable cost" [Sturm et al., 2000]. An integral part of SLM is the Service Level Agreement (SLA), which is a negotiated contract that formalizes business relationships between service providers and service consumers in order to measure, manage and enforce certain service quality levels. This thesis approaches the problem of integrating SLAs in SDIs, which is a multi-step approach.

Firstly, it is important to create a mutual understanding about the service quality requirements of the service consumer and the conditions under which the service provider is able and willing to deliver the demanded service quality. Such a mutual understanding can be established by SLA negotiation. This thesis aims at developing a SLA model to document the domain-specific service quality expectations and conditions of service delivery. This thesis also aims at developing a web-based SLA management architecture in order to enable the on-demand and online negotiation of SLAs in SDIs without the need of prior offline communication between service providers and service consumers.

Secondly, when a SLA is created it is important to monitor and report whether the service provider is really fulfilling the promised service quality levels. Therefore, the developed SLA management architecture must cover not only the agreement negotiation but also the whole agreement life cycle including service monitoring and agreement reporting.

Thirdly, for service providers it is also important to be in a position to actually deliver promised service quality levels to specific service consumers or in an application context. Therefore, this thesis aims at describing an approach on how to realize common domain-specific service quality requirements in SDIs under the terms of previously created agreements. The approach shall be no replacement for advanced resource management concepts, but it can act as a "best practice" for SDI service providers that meets the domain-specific requirements and capabilities of many SDI service providers.

## 1.2 Objectives

This thesis aims at developing a concept for the integration of SLAs in SDIs. In particular, it aims at the following objectives:

1. Evaluating the domain-specific requirements of different SDI stakeholders for the integration of SLAs in SDIs.
2. Formalizing an abstract SLA model that documents the domain-specific service quality requirements of service consumers and the conditions under which the service provider is able and willing to deliver the demanded service quality.
3. Developing a web-based SLA management architecture for the seamless integration of the abstract SLA model in SDIs.
4. Developing a concept for managing the infrastructure of SDI service providers under the terms of previously created SLAs in order deliver promised service levels to individual service consumers or in an application context.

The following issues are related but outside the scope of this thesis:

- Developing methods for creating and implementing multilateral SLAs.
- Developing models for describing the Quality of Service (QoS) history of web services.
- Developing methods for realizing QoS-aware service discovery and service chaining.
- Developing methods for measuring, quantifying and ensuring data quality.
- Developing methods for managing network traffic or implementing QoS at the Open Systems Interconnection (OSI) layers level.
- Developing methods for realizing resource reservation in distributed computing systems.

The concepts in this thesis are developed with a focus on, but are not limited to SDI services that are based on standards developed by the OGC.

## 1.3 Questions

Based on the motivation and the objectives, the following main research question is addressed in this thesis:

1. *How can SLAs be integrated into SDIs?*

The main research question can be divided into the following sub-questions:

a) *What are the domain-specific requirements for the integration of SLAs in SDIs?*

   Answering this research question aims at the first objective, which lays the foundation for the conducted work in this thesis.

b) *What is the domain-specific content of SLAs that can be applied in SDIs?*

   Answering this research question is related to the design of the abstract SLA model, which is the second objective of this thesis.

c) *How can service consumers and service providers negotiate SLAs for SDI services?*

   Answering this research question is related to the design of the web-based SLA management architecture, which is the third objective of this thesis. The research question covers related aspects such as the SLA-aware service discovery in SDIs.

d) *How can the complete SLA life cycle be integrated into SDIs?*

   Answering this research question is also related to the design of the web-base SLA management architecture. The research question covers related aspects such as the monitoring and accounting of SDI services.

e) *How can SDI service providers deliver promised service levels to individual customers or in an application context?*

   Answering this research question aims at the fourth objective, which aims at guidelines for SDI service providers to manage their computing infrastructure under the terms of previously created agreements.

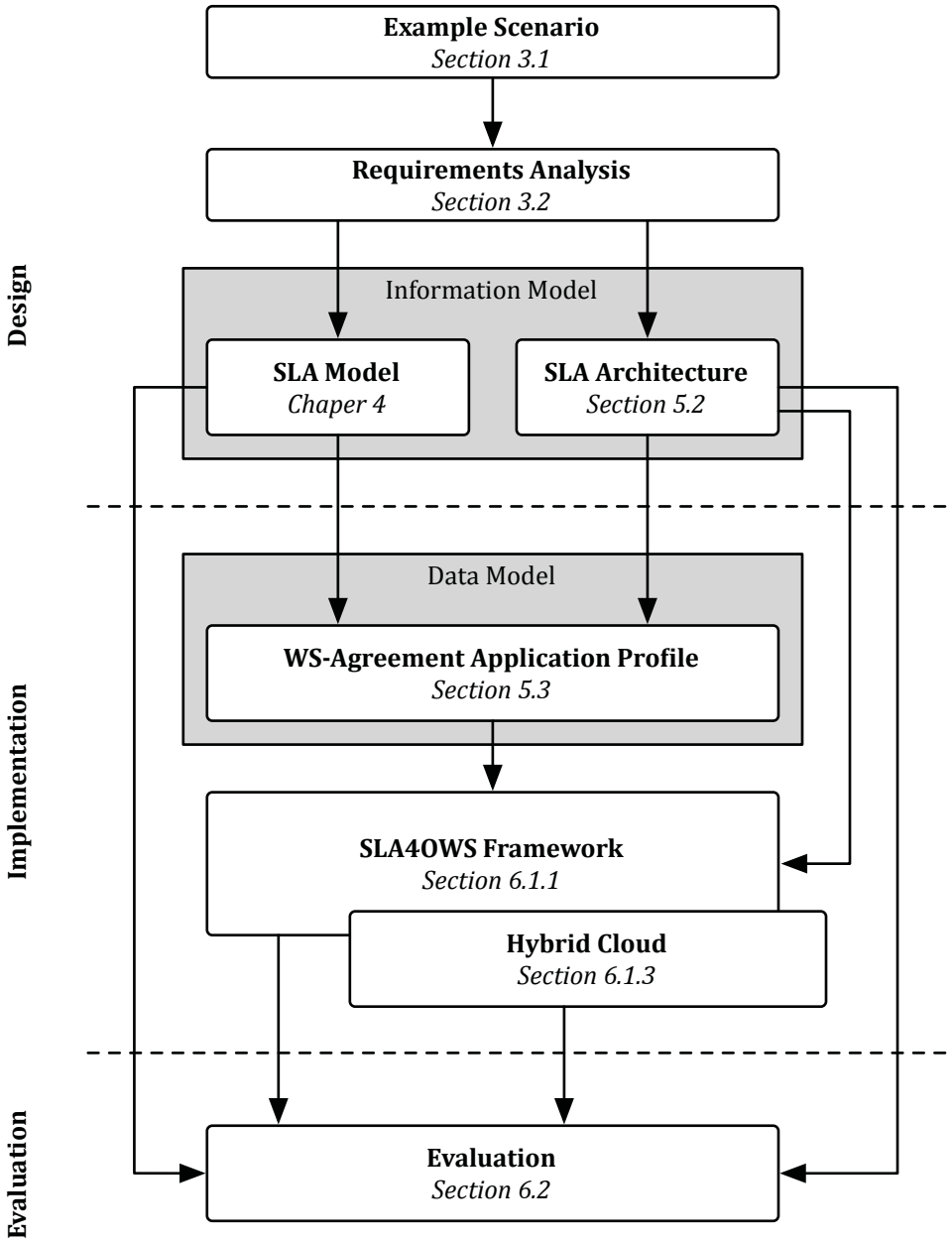
## 1.4 Methodology

To answer the research questions, this thesis presents a design, an implementation and an evaluation of a web-based SLA management architecture for the integration of SLAs in SDIs. The dependencies between these tasks are shown in Figure 1.1.

### Design

In the design phase, the requirements for the application of the SLA concept in SDIs are derived and generalized from an example scenario. The example scenario covers aspects from four application domains in which SDI services are utilized for different purposes. Therefore, the evaluated requirements cover relevant organizational, conceptional and technical aspects reflecting specific requirements and conditions of

Figure 1.1: Methodology



different SDI stakeholders. Based on the evaluated requirements, a concept for the integration of SLAs in SDIs is developed. The development of such a concept is a two-fold challenge.

To integrate the SLA concept in SDIs, first of all an abstract SLA model and a web-based SLA management architecture are developed. The abstract SLA model describes the domain-specific structure and content of agreements. The web-based SLA management architecture enables service providers and service consumers to create agreements, which document the service quality requirements of the service consumers and the conditions under which the service provider is able and willing to deliver the demanded service quality. Furthermore, the web-based SLA management architecture realizes the SLA enforcement and manages the complete SLA life cycle. Both the abstract SLA model and the web-based SLA management architecture are independent of any concrete technology. They serve as the foundation for the implementation phase.

That is followed by the development of a concept for managing the infrastructure of SDI service providers under the terms of previously created agreements in order deliver promised service levels to individual customers or in an application context. According to the objectives of this thesis, the concept for managing the infrastructure does not encompass the low-level implementation of service quality as for instance realizing high-performance web services and high-availability databases. It rather describes a state of the art and best-practice approach for SDI service providers to deploy reliable and scalable SDI services in a cost-efficient manner.

### Implementation

The abstract SLA model and the web-based SLA management architecture are implemented as a proof of concept using state of the art methods and technologies. Furthermore, several SLA documents are developed with regard to particular characteristics of the four application domains of the example scenario.

### Evaluation

Based on the implementation of the design, the proposed concept for the integration of SLAs in SDIs is evaluated against the objectives and the requirements of this thesis. The evaluation provides an elaboration of all relevant assets and drawbacks of the abstract SLA model and the web-based SLA management architecture.

## 1.5 Terminology

For the purposes of this thesis, the following terms and definitions apply.

**Agreement**

> A Service Level Agreement (SLA) is a negotiated contract between service providers and service consumers about the delivery of a service. The subject of an agreement may be any kind of service, but this thesis focuses on web services that are based on OGC standards. An agreement may contain not only functional

but also non-functional web service characteristics such as service response time and availability. Furthermore, an agreement may contain concrete guarantee terms reflecting for instance promised service quality levels and service usage limitations. For each of these guarantee terms, an agreement should define the obligated contracting party. In some cases an agreement may be a legally binding contract, in others it is simply a statement of good will from the service provider (to deliver promised service quality) and the service consumer (to use the service according to promised conditions).

#### Business Value

An agreement typically contains service descriptions and guarantee terms, but also values that represent the strength of an agreement in domain-specific terms such as service usage costs and penalties for contract violations.

#### Key Performance Indicator

An agreement should contain a set of measurable and exposed service characteristics, the so-called Key Performance Indicators (KPIs). These domain-specific service properties are key elements of an agreement and permanently measured during agreement runtime.

#### Quality of Service

The International Telecommunications Union (ITU) defines the term 'quality of service' as "the collective effect of service performances, which determine the degree of satisfaction of a user of the service" and notes that "the quality of service is characterized by the combined aspects of service support performance, service operability performance, service integrity and other factors specific to each service" [ITU, 1994]. In the context of GI, the quality attribute not only covers classic web service characteristics such as the service response time and availability, but also for instance the accuracy, resolution and completeness of geospatial information.

#### Service Consumer

The service consumer is an entity that creates agreements in order to obtain guarantees from the service provider regarding the availability of certain services and associated service levels.

#### Service Level Objective

An agreement should contain a set of domain-specific guarantee terms that reflect the quality of service aspect of the agreement. The so-called Service Level Objectives (SLOs) define concrete service quality goals by defining valid value ranges for the KPI measurements.

#### Service Provider

The service provider is an entity that offers agreements in order to provide services not only along with defined service levels but also under stated business conditions.

**Template**

An agreement template is a document that is used by service providers to advertise their services along specific service levels. Templates are the entry point for an agreement negotiation process in which service providers and service consumers clarify the expectations and the responsibilities regarding a successful delivery of the service. Some templates are defined by service providers in a "take it or leave it" proposition, others can be modified by service consumers before finally committing a (potentially legally binding) contract. Templates have the same content as an agreement, but may also contain additional information and creation constraints to describe the range of agreements the service provider is willing to accept.

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY and OPTIONAL in this thesis are to be interpreted as described in [Bradner, 1997].

## 1.6 Overview

The remainder of this thesis is structured as follows.

**Chapter 2 - Research Context**

This chapter describes the context of this thesis. It introduces the concepts and state of the art of Spatial Data Infrastructures, Service Level Agreements and Cloud Computing. This chapter also describes related work in all of these areas.

**Chapter 3 - Requirements Analysis**

This chapter defines the domain-specific requirements for the integration of SLAs in SDIs. It describes an abstract scenario that covers aspects from four application domains in which SDI services are utilized for different purposes. Based on the abstract scenario and the application domains, the organizational, conceptional and technical requirements for the integration of SLAs in SDIs are derived and generalized.

**Chapter 4 - Agreement Formalization**

This chapter formalizes an abstract SLA model that is independent of any specific technology. The abstract SLA model defines the domain-specific structure and content of SLAs that can be applied in SDIs and that are based on standards developed by the OGC. The description of the domain-specific content covers aspects such as functional service descriptions, service quality guarantees and pricing models.

**Chapter 5 - Service Level Management Architecture**

This chapter presents the design of a web-based SLA management architecture that allows the integration of the abstract SLA model in SDIs. The web-based SLA management architecture covers not only agreement negotiation and

service consumption, but also the infrastructure management under the terms of previously created agreements.

### Chapter 6 - Implementation and Evaluation

This chapter presents an implementation and demonstration of the abstract SLA model and the web-based SLA management architecture. Based on the implementation, this chapter evaluates the presented concept for the integration of SLAs in SDIs. The evaluation provides an elaboration of the advantages and limitations of the concept with respect to the research questions, the objectives and the requirements.

### Chapter 7 - Conclusion and Outlook

This chapter summarizes and discusses the results of this thesis. Furthermore, this chapter provides directions for future research.

# Chapter 2

# Research Context

This chapter describes the context of this thesis. It introduces the concepts and state of the art of Spatial Data Infrastructures, Service Level Agreements and Cloud Computing. This chapter also describes related work in all of these areas.

## 2.1 Spatial Data Infrastructures

Geographic Information System (GIS) [Goodchild, 1991] have been under constant development in recent years. Emerging web services concepts and technologies such as the Service Oriented Architecture (SOA) paradigm [Papazoglou, 2003] encouraged the evolution from classical desktop- and data-centric GIS to distributed and loosely-coupled architectures composed of open and interoperable web services merged into the Spatial Data Infrastructure (SDI) concept [Groot and McLaughlin, 2000].

### Definition

The Spatial Data Infrastructure (SDI) concept can be described as a "collection of technologies, policies and institutional arrangements that facilitate the availability of and access to spatial data" [Nebert, 2004].

Masser [Masser, 2005] maintains that the SDI

> *(...) supports ready access to geographic information. This is achieved through the coordinated actions of nations and organizations that promote awareness and implementation of complementary policies, common standards and effective mechanism for the development and availability of interoperable digital geographic data and technologies to support decision making at all scales for multiple purposes. These actions encompass the policies, organizational remits, data, technologies, standards, delivery mechanisms, and financial and human resources necessary to ensure that those working at the (national) and regional scale are not impeded in meeting their objectives.*

The most frequent organizational approach to SDIs is hierarchical, ranging from local to state, national and regional levels [Rajabifard and Williamson, 2001]. The developed policies and institutional arrangements are focusing on various aspects

such as the organization of human resources and leadership, the commitment of licenses for data sharing, the definition of interoperable standards for geospatial data access, and the negotiation of cost-sharing agreements between organizations [Budhathoki and Nedovi-Budi, 2006].

The SDI development is particularly driven by public authorities, like national mapping or environmental agencies [Masser, 2005]. Many countries are in the process of establishing their SDI in order to "better manage and utilize their spatial data assets" [Rajabifard, 1999]. The innovators in that field are for instance the United States National Spatial Data Infrastructure (NSDI) [FGDC, 2005], the Canadian Geospatial Data Infrastructure (CGDI) [CGDI Architecture Working Group, 2001] and the Geodateninfrastruktur Deutschland (GDI-DE) [GDI-DE, 2010]. At the same time, some countries are finding it necessary "to cooperate with other countries to develop multinational SDIs to assist in regional decision-making that has an important impact across national boundaries" [Rajabifard et al., 2000]. The most sophisticated example from Europe is the Infrastructure for Spatial Information in the European Community (INSPIRE) directive [EU, 2007] which aims at building an European SDI based on the member states' national SDIs. The INSPIRE directive is legally valid since 2007 and the INSPIRE roadmap [EU, 2011b] specifies future deadlines and milestones for the implementation of services and datasets through the European Union (EU) member states.

## Standardization

The development of interoperable and open standards is an important key factor for the successful sharing of geospatial resources [Wytzisk and Sliwinski, 2004]. Considerable progress in defining geospatial standards has been made by the Open Geospatial Consortium (OGC)[1], an international industry consortium in which more than 450 government, academic, and private sector organizations intend to collaborate. The OGC was founded in 1994 as a non-profit organization with the mission to facilitate the adoption of free and openly available standards for spatial data products and services, and to accelerate the market assimilation of interoperability research through a collaborative consortium process.

The OGC Technical Committee (TC) publishes several types of documents as for instance Best Practices Documents, Engineering Reports and Discussion Papers. The Best Practices Documents contain discussion of best practices related to the use and/or implementation of an adopted OGC document. The Engineering Reports are the primary output of OGC Interoperability Program Initiatives (testbeds, pilot projects and interoperability experiments). The Discussion Papers present technology issues being considered in the Working Groups of the OGC TC. The two major OGC publication types are Abstract Specifications and Implementation Specifications. The Abstract Specifications provide the conceptual foundation for most of the OGC specification development activities. The Implementation Specifications are technical documents that detail service interfaces or data encodings.

[1] http://www.opengeospatial.org

The OGC Reference Model (ORM) [OGC, 2003] provides a framework that describes the OGC Standards Baseline. The ORM embraces approved Abstract Specifications and Implementation Specifications as well as relevant Best Practices in order to provide guidelines for defining SDI architectures for specific use cases. The OpenGIS Web Services Architecture [Whiteside, 2005] describes the most fundamental aspects of OGC Web Services (OWS) as for instance the organization of service components into multiple tiers, and the use of open standards for service interfaces and service communication. In the ORM the clients and services communicate through HTTP GET and POST, using standard Extensible Markup Language (XML) encoding formats for transferring data and messages. The OGC Web Services Common Standard [Whiteside and Greenwood, 2010] specifies many of the aspects that are common to all OWS as for instance the mandatory `GetCapabilities` operation, which allows any client to retrieve metadata from a server.

Based on these supporting documents, fundamental Implementation Specifications are for instance the Web Map Service (WMS) specification [de la Beaujardiere, 2006] and the Web Feature Service (WFS) specification [Vretanos, 2010]. Over time, the OGC standards development process reached a certain level of acceptance and maturity, so that some Implementation Specifications became relevant for other organizations and initiatives. The WMS specification has been approved for instance by the International Organization for Standardization (ISO) Technical Committee 211 (ISO/TC 211) as an official ISO specification [ISO, 2005]. Furthermore, the INSPIRE directive defines a family of so-called Network Services that can be traced back to corresponding OGC Implementation Specifications.

## 2.2 Service Level Agreements

Service Level Management (SLM) is "the disciplined, proactive methodology and procedures used to ensure that adequate levels of service are delivered to all IT users in accordance with business priorities and at acceptable cost" [Sturm et al., 2000]. There are several main reasons for implementing SLM in an Information Technology (IT) organization [Sturm et al., 2000].

- **Client Satisfaction**

  Discussing the users' requirements helps to understand the client's service requirements and act on behalf of them.

- **Managing Expectations**

  Discussing and documenting the users' requirements helps to avoid ever rising levels of users' expectations.

- **Resource Regulation**

  Documented users' requirements can be used as indicators for ongoing system capacity requirements in order to avoid capacity problems.

- **Internal Marketing**

  Established SOA government processes help to ensure consistent service levels that can be used as a pro-active marketing tool.

- **Defensive Strategy**

  Determine the concrete service levels that should be delivered helps to calculate appropriate IT cost of providing these (or higher) service levels in the future.

The key element for enforcing SLM is the Service Level Agreement (SLA).

## Definition

In the most basic form, a SLA is a negotiated "contract or agreement that formalizes a business relationship, or part of the relationship, between two parties" [Lee and Ben-Natan, 2002]. The SLA is the result of a formal negotiation between service providers and service consumers [McConnell and Siegel, 2004], and it helps to "identify expectations, clarify responsibilities, and facilitate communication between a service provider and its customers" [Karten, 1998]. In the IT domain, the SLA is an "agreement between the computing service provider and the user quantifying the minimum acceptable service to the user" [Hiles, 2002]. Depending on the negotiation process the SLA either can be a *statement of good will* from the service provider (to deliver promised service quality) and the service consumer (to use the service according to promised conditions) without any legally binding, or a *legally binding contract* that formally defines concrete rights and obligations for the service provider and the service consumer [Berger, 2005].

The application of SLAs is not new in the mainstream IT. They "emerged in the early 1990s as a way for IT departments and service providers within private (usually corporate) computer networking environments to measure and manage the quality of service (QoS) they were delivering to their internal customers" [Lee and Ben-Natan, 2002]. The International Telecommunications Union (ITU) defines the term 'quality of service' as "the collective effect of service performances, which determine the degree of satisfaction of a user of the service" and notes that "the quality of service is characterized by the combined aspects of service support performance, service operability performance, service integrity and other factors specific to each service" [ITU, 1994].

## Content

The SLA typically contains the following parts that are derived from [Berger, 2005].

### Agreement Context

An agreement should contain basic elements such as a the agreement scope, details about the groups that negotiated the agreement and information about the contract period. The agreement scope may consists of a short human-readable description of the service and the aim of the agreement. The information about the contracting parties may contain common contact details and for instance a banking account. The information about the contract period may contain a fixed start and end date, or information about an automatic contract renewal after a specific period of time under stated conditions. In some cases an agreement must contain legal elements as for instance information about the place of jurisdiction,

or liability and warranty clauses.

#### Service Provisioning

An agreement must contain a detailed description of the service to which the agreement pertains. Such a detailed service description shall cover all important aspects as for instance a formalized description of the actual service that is provided, steps that must be performed before and during service provisioning or general technical requirements as for instance the utilized IT infrastructure. In some cases it is useful do describe the limitations of the agreement, which may contain service aspects that are not covered by the agreement (but can be expected at first glance) or some restrictions for the service consumer (e.g. the number of employees that make use of the service).

To describe the service quality, which is expected by the service consumer and promised by the service provider, the agreement must define service level indicators, corresponding measurement procedures and concrete service level objectives. The Key Performance Indicators (KPIs) of an agreement define a set of quantifiable measurements regarding specific aspects of the service quality. In the context of web services, such KPI can be for instance the web service availability over a specific period of time (in percent) or the average web service response time (in milliseconds). The agreement must contain information on how to measure the designated KPIs, always with the service user's perspective in mind. Finally, the agreement must contain Service Level Objectives (SLOs) that define the minimum service quality level that will be considered acceptable by the service consumer and that is promised by the service provider. In order for SLM to be successful, the KPIs and SLOs of an agreement must be attainable, meaningful, understandable, measurable, controllable, affordable and mutually accepted [Sturm et al., 2000].

An agreement may define costs for using the service. In that case, the agreement must contain a pricing model that define fixed or variable service usage fees, discounts or sliding scale fees. The pricing model may also contain penalties for not meeting the stipulated service quality levels. Furthermore, the agreement must contain information about the settlement period and the method of payment.

#### Agreement Management

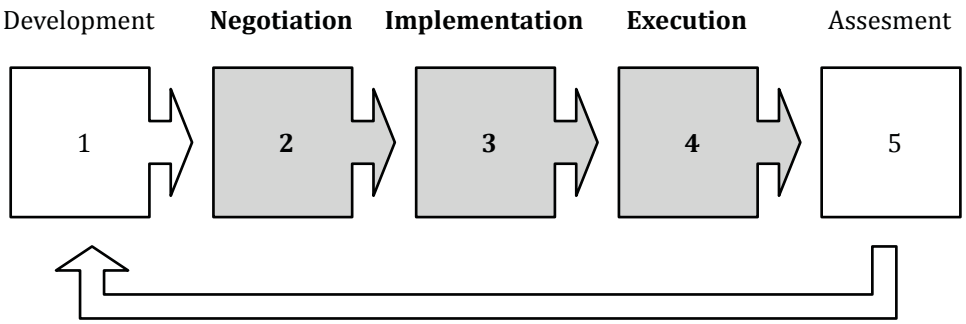
An agreement should contain information about administrative tasks as for instance periodically reviews of the agreement, continuous reports on service level indicators and corresponding service level objectives, or how to arbitrate conflicts between contracting parties.

Furthermore, it is important that all contracting parties understand their respective roles and responsibilities in the agreement context. Therefore, each obligation in the agreement must be attached by such a responsibility information.

### LIFE CYCLE

The typical agreement life cycle is described in [Lee and Ben-Natan, 2002] and consists of the following five phases (Figure 2.1):

Figure 2.1: Agreement Life Cycle (adapted from [Lee and Ben-Natan, 2002]).




### Development

The development phase covers service and template development. When the service development is finished and the service provider understands every technical, organizational and financial impact of prospective service offerings, the development of one or more templates for a particular service can start. It is possible to define more than one template with different service quality levels for the same service. This gives potential service consumer "the opportunity to weigh competing priorities within his or her own company" [Sun et al., 2005]. In general, these templates have the same structure and content as an agreement. They are used to advertise a service offering and they are the entry point for the negotiation phase.

### Negotiation

The negotiation phase starts with template discovery in order to find an adequate service offering that matches the individual requirements of a service consumer. Some of the templates are defined in an "take it or leave it" proposition [Lee and Ben-Natan, 2002], others can be modified by the service consumer before finally committing to a potentially legally binding contract. However, the outcome of the negotiation is an agreement, that only contains terms and conditions that are approved by all contracting parties.

### Implementation

The implementation phase implies all tasks that are necessary to provide an individual service instance in compliance with the corresponding agreement. The service provider must perform the initial setup of the service, install new monitoring capabilities, prepare periodical reports, and arrange organizational responsibilities in case of SLA violations.

### Execution

The execution phase reflects the normal day-to-day business of the service provider. This phase covers all tasks that must be performed on an ongoing basis during the whole agreement period in order to operate the service in compliance

with the corresponding SLA. This includes measurement of KPIs, evaluation of SLOs, adjustment of the service setup and reporting SLA compliance to all contracting parties.

**Assessment**

The assessment phase can be performed either periodically during or only once after the agreement period. The customer-focused assessment analyzes the service provider's performance from the customer's viewpoint and focuses on SLA compliance and customer satisfaction. The provider-focused assessment aims at "optimizing the use of the SLA by the service provider in order to improve profitability through achieving better compliance or reducing penalty exposure by changing the commitment contained within the SLAs" [Lee and Ben-Natan, 2002]. The assessment phase provides the input for an optimized agreement development phase in the future.

This thesis focuses on the negotiation, implementation and execution phases.

## Standardization

There are a number of organizations working on SLM and SLA standards. The IT Infrastructure Library (ITIL) covers SLM from an organizational perspective and provides best-practices for service design [Lloyd, 2008] and service operation [Cannon, 2007]. The Distributed Management Task Force (DMTF) provides an extension of the Common Information Model (CIM) [DMTF, 1999] in order to "allow the definition and association of policies, rules, and expressions that enable common industry communications with respect to service management" [Sturm et al., 2000]. The Internet Engineering Task Force (IETF) has issued several Request for Comments (RFC) documents that target application management [Kalbfleisch et al., 1999] and differentiated services [Blake et al., 1997] at the TCP/IP network level.

There is one standard - the Web Services Agreement Specification (WS-Agreement) [Andrieux et al., 2005] - that provides not only an XML Schema document for specifying the structure of agreement templates and agreements, but also a web service interface for managing the complete life cycle of agreements. The goal of WS-Agreement is to "standardize the terminology, concepts, overall agreement structure with types of agreement terms, agreement template with creation constraints and a set of port types and operations for creation, expiration and monitoring of agreements, including WSDL needed to express the message exchanges and resources needed to express the state" [Andrieux et al., 2005]. The WS-Agreement specification is currently developed by the Grid Resource Allocation Agreement Protocol (GRAAP) Working Group (WG)[2] of the Open Grid Forum (OGF)[3] and historically can be traced back to a number of approaches for general-purpose SLA languages that provide the foundation for the WS-Agreement specification.

The SLA language described in [Ludwig et al., 2002] provides an XML-based

[2] https://forge.ogf.org/sf/projects/graap-wg
[3] http://www.ogf.org

representation and a runtime system for SLAs. The presented approach includes a description on how parameters are measured and computed from raw metrics, how guarantees are defined with respect to those parameters and how SLA compliance is verified. The Web Service Level Agreement (WSLA) Framework [Keller and Ludwig, 2003] describes an approach for specifying and monitoring SLAs for web services. The WSAL framework is developed by IBM and focuses on a flexible, formal language to accommodate a wide variety of SLAs and the (web) services that are required to implement the WSLA framework. The SLAng language [Lamanna et al., 2003] also describes an XML-based representation of SLAs in order to realize end-to-end quality of service between network services, storage services and middleware services. An extensible an flexible XML-based SLA representation as a foundation for automated SLA management systems is presented in [Sahai et al., 2002]. The approach of [Rodosek and Lewis, 2001] provides a user-centric language for agreements and a method of mapping such a user-centric language to a network-centric language in order to develop a framework for dynamic service provisioning.

However, the WS-Agreement specification was investigated and developed in several research projects. The SLA@SOI project [Wieder et al., 2011] aimed at developing standards for a domain-independent syntax for machine-readable SLA specification and negotiation, systematic multi-layer SLA management, monitoring and accounting [Kearney et al., 2010]. The focus of this project was the development of standardized interfaces for adaptive infrastructures with harmonized access to different virtualization technologies and the development of advanced technologies for SLA enforcement on an infrastructure level. The Service Level Agreements for D-Grid (SLA4D-Grid) project [Kuebert et al., 2010] aimed at designing and realizing a SLA layer for the Germany's national Grid Computing infrastructure D-Grid[4]. The SLA layer offers individual users, whole D-Grid communities, and the providers of D-Grid resources service usage under given guarantees, Quality of Service (QoS) requirements and pre-defined business conditions. The European CoreGRID Research Network published a comprehensive report that provides an overview and comparison of SLA use in six of the European Commissions FP6 Projects [Parkin et al., 2008]. An outcome of several research projects is the WS-Agreement for Java (WSAG4J) framework [Wäldrich, 2011], an Open Source implementation of the WS-Agreement specification that automates typical SLA management tasks like SLA offer validation, service level monitoring, persistence, and accounting.

The WS-Agreement specification defines neither domain-specific (web) service descriptions, expressions and metrics for KPIs and SLOs, nor how and where to measure such properties. For good reasons, these domain-specific objectives are out of the scope of the WS-Agreement specification. For this purpose, there are a number of other approaches available.

The World Wide Web Consortium (W3C) for instance describes briefly the general requirements for web services and gives an overview about the most relevant quality aspects of web services [Lee et al., 2003]. The Organization for the Advancement of Structured Information Standards (OASIS) provides the Quality Model for Web Services

[4] http://www.d-grid.de

[Kim and Lee, 2005], a conceptual model "for Web services quality management and quality factors in the process of developing and using Web services". The Web Service Modeling Ontology (WSMO) Working Group[5] published a comprehensive report that provides an overview about most relevant approaches for describing non-functional web service properties [Toma and Foxvog, 2008]. The object-oriented Quality of Service Specification Language (QML) [Frolund and Koistinen, 1998] can be used for QoS specification in Unified Modeling Language (UML) class models and interface designs of distributed systems. There are a number of concrete approaches for the integration of QoS in web services available. Some of them provide ontologies for QoS in web services as XML Schema documents (or in other formal languages) as for instance [Aagedal and Ecklund, 2002], [Zhou et al., 2005] and [Papaioannou et al., 2006], others also provide an architecture for the integration of QoS information in the web service discovery process as for instance [Tian et al., 2003], [Ran, 2003] and [Yu-jie et al., 2005].

## 2.3 Cloud Computing

Cloud Computing is a recent trend in mainstream IT [Driver, 2008] and several companies such as Amazon, Google, Microsoft and Salesforce have made significant effort in this direction. The cloud metaphor describes an approach in which the storage and computational facilities are no longer located on single computers, but distributed over remote facilities operated by third party providers [Foster et al., 2008]. This section gives a short introduction to Cloud Computing and briefly describes how Cloud Computing can help SDI service providers to facilitate reliable and scalable SDI services in an economical efficient manner.

### Definition

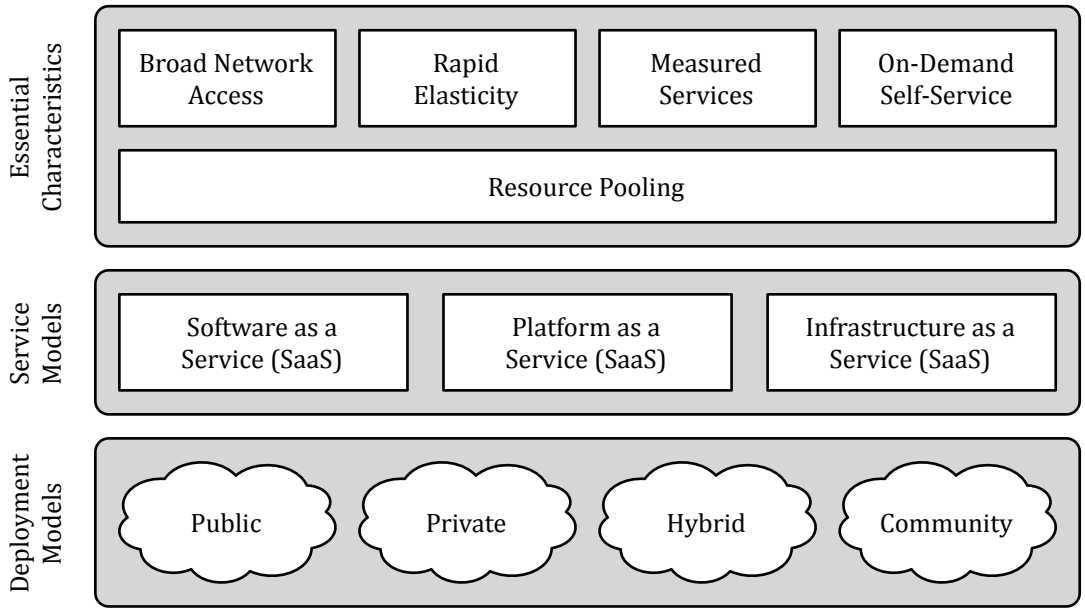
There are various definitions for the Cloud Computing paradigm available, as for instance presented in [Armbrust et al., 2009] and [Buyya et al., 2009]. Following the National Institute of Standards and Technology (NIST) definition of Cloud Computing [Mell and Grance, 2009], the Cloud Computing model is composed of five essential characteristics, three service models and four deployment models (Figure 2.2):

#### Characteristics

In Cloud Computing environments, users can allocate computational resources without requiring human interaction with a resource provider (On-Demand Self-Service). Examples of such resources include storage, processing, memory, network bandwidth, and virtual machines. These resources and their capabilities are available over the network via standardized mechanisms and simple web service interfaces (Broad Network Access). The resource providers' physical and virtual resources are pooled to serve multiple users and their dynamically changing demands (Resource Pooling). From the user perspective, the availability of resources in the cloud appears unlimited. They can be acquired from the resource provider in large quantity at any time in order to scale applications, services and storage depending on an application context (Rapid Elasticity).

[5] http://www.wsmo.org

Figure 2.2: NIST Cloud Computing Definition (adapted from [Brunette and Mogull, 2009])




Finally, the resource usage in Cloud Computing environments can be monitored and reported, providing transparency for both the users and the resource providers (Measured Service).

### Service Models

Cloud Computing replaces the classic multi-tier architecture of web services [Ramirez, 2001] and creates a new set of layers [SUN, 2009]. The Cloud Computing characteristics can be used to give users the ability to run their web- or desktop-based applications in the cloud without managing the hardware infrastructure (Software as a Service, SaaS). Resource providers can offer runtime environments in the cloud, where users can deploy their applications created using particular programming languages and tools that are supported by the provider (Platform as a Service, PaaS). Furthermore, resource providers can offer complete access to virtual machines where users have total control over operating systems, storage and deployed applications (Infrastructure as a Service, IaaS).

### Deployment Models

When a resource provider makes his resources available for instance in a pay-as-you-go manner to the general public, it is called a Public Cloud. In the case that the Cloud Computing infrastructure is provisioned "for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations)" [Mell and Grance, 2009], it is called a Community Cloud. When Cloud Computing technologies are used to manage an internal data center and when such a data

center is not made available to the general public, it is called Private Cloud. In a so-called Hybrid Cloud, a Private Cloud is combined with resources of a Public Cloud in order to handle tasks that cannot be performed in the local data center for instance due to general hardware limitations and/or temporarily heavy workload.

Historically, the Cloud Computing model overlaps with some concepts of Distributed Computing [Attiya and Welch, 2004] and Grid Computing [Foster, 2003]. Both Grid Computing and Cloud Computing environments establish a network infrastructure for scaling applications by providing sufficient storage and computational resources. Grid Computing is mostly applied by scientific communities [Hartig, 2009] for large-scale computations as for instance executed in bioinformatics and biology, astronomy and high-energy physics [Deelman et al., 2004]. Whereas Cloud Computing enables especially small and medium-sized companies to deploy their web-based applications in an instant scaleable fashion without the need to invest in large computational infrastructures for storing large amounts of data and/or performing complex processes [Myerson, 2009]. As a consequence, national and international Grid Computing infrastructures as for instance the Worldwide LHC Computing Grid [Shiers, 2007] are typically funded by the government and operated by international joint research projects, whereas Public Cloud infrastructures are typically operated by large-sized enterprises under economic aspects, enabling smaller companies to use their infrastructure.

## Standardization

There are a number of organizations and companies working on Cloud Computing standards as for instance the Cloud Security Alliance (CSA)[6], the Open Cloud Consortium (OCC)[7], the Distributed Management Task Force (DMTF)[8], and the Open Grid Forum (OGF). Achievements of these standardization efforts are for instance the Open Virtualization Format (OVF) [DMTF, 2009] which defines file formats for virtual machines and the Open Cloud Computing Interface (OCCI) family of specifications [Nyrén et al., 2011] [Metsch and Edmonds, 2011a] [Metsch and Edmonds, 2011b] which defines an Application Programming Interface (API) for managing Cloud Computing environments. Several other organizations such as the Internet Engineering Task Force (IETF) and the OASIS have developed related standards that are essential for a successful integration of Cloud Computing technologies in real-world applications. Such supporting standards are for instance the Web Authorization Protocol (OAuth) [Leiba, 2012] for identity management and the complete WS-* family of specifications for web services such as WS-Security [Nadalin et al., 2006], WS-Policy [Bajaj et al., 2006], and WS-Agreement [Andrieux et al., 2005]. The web service protocols for managing Amazon Simple Storage Service (S3)[9] and Amazon Elastic Compute Cloud (EC2) are de facto industry standards and reimplemented by many projects and products as for instance Apache Deltacloud[10] and Eucalyptus[11].

[6] https://cloudsecurityalliance.org
[7] http://opencloudconsortium.org
[8] http://www.dmtf.org
[9] http://aws.amazon.com/s3
[10] http://deltacloud.apache.org
[11] http://www.eucalyptus.com

Supplementary policies and recommendation for establishing secure and trustful Cloud Computing environments under the terms of national and regional law are developed from several organizations and governmental agencies. The Federal Office for Information Security (BSI) promotes IT security in Germany and describes recommendations for setting up a solid security architecture for Cloud Computing environments [BSI, 2012]. The EU announced the EU Data Protection Directive [EU, 1995], a comprehensive privacy framework that aims at the protection of personal data of EU residents. The directive does not mention Cloud Computing explicitly, but it covers basic requirements for the cross-border movement and processing of personal data, which is an inherent characteristic of Cloud Computing environments.

These examples show many stakeholders and considerable momentum in Cloud Computing development. Especially in comparison to the adoption rate of standards in the Geographic Information (GI) domain, most of the developed Cloud Computing standards are less widespread outside the research community and therefore the standardization process still needs to mature [BMWI, 2012].

## Opportunities

The Cloud Computing characteristics and technologies can help SDI service providers in several ways. The application of the Cloud Computing model in the internal data center enables IT companies to increase the utilization rate of their existing hardware significantly [Zhang et al., 2010]. By outtasking software and data to facilities operated by third parties, service providers do not need to operate their own large-scale hardware infrastructure anymore [Leimeister et al., 2010]. In contrast to classical long-term outsourcing contracts, the on-demand usage together with pay-as-you-go revenue models enable service providers to transform fixed costs into variable costs, and to decrease up-front investments for deploying and maintaining cutting-edge IT services [Marston et al., 2011]. Furthermore, cloud resources can be allocated nearly in real-time and most Cloud Computing providers offer advanced mechanisms for scaling the deployed applications automatically for instance in case of high amounts of users requesting a service. This allows service providers to handle peak loads very efficiently, which is hard to realize with classic service deployment models [Tu et al., 2004]. In nearly all cases the total cost of ownership (including initial investment in hardware, software licenses, energy, fail-safety and technical engineers) for realizing highly elastic solutions (consume 100 server-hours today and no server-hours tomorrow) can also be reduced significantly [Armbrust et al., 2010].

There are still a number of open issues for adopting Cloud Computing in existing IT infrastructures, which are exemplified for instance in the so-called "Open Cloud Manifesto" [CCUCG, 2009]. Especially the absence of standardized and widely-adopted APIs for managing Cloud Computing infrastructures can be seen as one major obstacle. The vendor-specific APIs bind particular solutions (software, data) to specific vendors and therefore complicate the migration of applications between different Cloud Computing providers (vendor lock-in, data lock-in). Besides data backup and recovery responsibilities, the outsourcing of confidential data from data owners to third party infrastructures is problematic especially in the GI domain [Hillmann-Köster, 2011]. In

most cases, Public Cloud infrastructures as a deployment platform for data are not suitable. Private Cloud infrastructures maintained within an entity or a so-called "Trusted Cloud" [BMWI, 2010] can help to overcome these obstacles. Other critical obstacles to growth of Cloud Computing are for instance the service availability, data auditability, network bottlenecks and software licenses [Armbrust et al., 2010].

## 2.4 Related Work

The SLM and SLA aspects have been a common product in support of services offered by telecommunications service providers for many years, but the integration of SLAs in SDIs has not been widely investigated yet. There is already significant effort regarding geospatial digital rights management with a focus on authentication and access control. The aspect of geospatial licenses is already examined with a focus on identifying appropriate and harmonized SDI license models for data sharing. Furthermore, the performance and scalability of SDI services is widely examined and several approaches for integrating geospatial services and data into Grid Computing and Cloud Computing infrastructures have been developed. None of these efforts from different fields of application approaches neither the on-demand and online negotiation of SLAs in SDIs without the need of prior offline communication between service providers and service consumers, nor the automated SLA management in SDIs including agreement monitoring, agreement evaluation, agreement reporting and infrastructure management. However, the following sections provide a brief overview about most relevant related work.

### Geospatial Rights Management

The Geospatial Digital Rights Management Reference Model (GeoDRM) [Vowles, 2006] defines a "conceptual model for digital rights management of geospatial resources". The GeoDRM describes roles and relationships in digital rights management, and identifies necessary functionalities that must be provided by more detailed specifications in this area. These necessary functionalities cover aspects such as the development of machine-readable rights expression languages, methods for establishing trust between partners, and authentication and authorization methods that are fundamental for license enforcement.

The OGC Geospatial eXtensible Access Control Markup Language (GeoXACML) specification [Matheus and Herrmann, 2011] defines a geo-specific extension to the eXtensible Access Control Markup Language (XACML) [Moses, 2005] in order to support geospatial data types and geospatial authorization decision functions. The establishment and the administration of a GeoXACML-based access control system for SDIs is demonstrated in [Herrmann, 2010]. Access control and web services security have been addressed also in many OGC Interoperability Program Initiatives as for instance described in [Wagner, 2006b], [Gartmann and Leinenweber, 2009] and [Herrmann and Matheus, 2009].

Several other publications show a notable interest in this area. A more fine-grained view on the GeoDRM roles and relationships is presented for instance in

[Wagner, 2009]. In [Schäffer et al., 2010a] a security enabled architecture is presented, in which standard geoprocessing services can be enhanced in order to support ad hoc license agreements directly in process, without any prior offline negotiated agreements between service provider and service consumer. In [Gartmann and Schäffer, 2011] the "use of OASIS SAML for license encoding, the implementation details for the use of this license encoding for expressing access permissions to SOAP services, and the submission and evaluation of these licenses within (OWS) service requests" is described. In [Schäffer, 2012] a concept for the "dynamic access to and chaining of secured Geoprocessing Web Services without a priori established rights or direct trust relationships" is presented.

## Geospatial Licenses

The SLA concept is not completely new for National Spatial Data Infrastructure (NSDI) building activities, which cover not only technologies but also policies in order to support the organizational collaboration. The NSDI policies typically define requirements regarding available datasets, licenses for data sharing or minimum performance criteria. Furthermore, policies are also developed for the regulation of cost-sharing between different departments, agencies or countries. Examples for such policies are for instance the "Gesetz über den Zugang zu digitalen Geodaten (Geodatenzugangsgesetz)" [BMJ, 2009] that regulates the public access to basic geoinformation in Germany and the GeoLizenz project [Rech, 2011] that establishes a framework for providing harmonized license models for the public authorities in Germany. Beyond such an organizational and legal perspective, aspects such as the on-demand and online negotiation of SLAs in SDIs without the need of prior offline communication between service consumers and service providers have not been widely investigated yet.

This thesis is based on concepts that have appeared previously in other publications and that established the foundation for the conducted research. The work presented in [Baranski, 2009] aims at identifying and developing an abstract architecture in which different service quality levels in SDIs can be measured and managed by attaching SLA functionality to existing OWS. Based on this work, the approach described in [Baranski and Schäffer, 2010] provides a more substantiated requirements analysis for the application of the SLA concept in SDIs that are based on OGC standards. The approach provides a classification of QoS attributes that are relevant in SDIs, and proposes an architecture that covers the complete SLA negotiation process between service consumers and service providers. The OGC published the WS-Agreement Application Profile for OGC Web Services [Baranski, 2011], a domain-specific extension of the WS-Agreement specification that mainly consists of a set of XML Schema Documents (XSDs) specifying the OGC-specific content of an agreement, an Uniform Resource Name (URN) namespace for identifying OGC-specific service properties and a Domain Specific Language (DSL) for defining and evaluating domain-specific guarantee terms. The Service Level Agreements for OGC Web Services (SLA4OWS) framework [Baranski, 2012] is an Open Source implementation of the WS-Agreement Application Profile for OCG Web Services and automates typical SLM management tasks such as service level monitoring, agreement reporting and infrastructure management. The SLA4OWS framework enables SDI service providers to offer their services along with

different service levels. Furthermore, it enables service consumers to perform a cost-benefit ratio analysis of different service offerings. The SLA4OWS framework integrates the Amazon EC2[12] for realizing differentiated web services in an economic fashion.

## Cloud Computing

In the GI domain, the area of Grid Computing has been addressed very early at various levels. Many research projects investigated how to merge the SDI and the Grid Computing concept. In [Padberg and Kiehle, 2009] the conceptual differences between OWS and the Open Grid Services Architecture (OGSA) [Foster et al., 2006] are examined and an integration approach to overcome the gaps between SDIs and Grid Computing infrastructures is provided. The Globus Toolkit was integrated in the NASA Web GIS Software Suite (NWGISS) [Di et al., 2003], OGC and OGSA web services were orchestrated in workflows that are based on Business Process Execution Language (BPEL) [Hobona et al., 2007] [Fleuren and Müller, 2008], and geospatial data was processed through an OGC Web Processing Service (WPS) [Schut, 2007] in Grid Computing infrastructures [Lanig et al., 2008] [Baranski, 2008]. The GDI-Grid project [Maué and Kiehle, 2009] focused on solutions for the efficient integration and processing of geospatial data in the German national Grid Computing infrastructure D-Grid through OGC Web Services. A proof of concept was developed for "flood simulation", "noise propagation in urban areas" and "emergency routing for relief units". The Secure Access to Geospatial Services (SEE-GEO) project [Higgins, 2009] investigated the means of making geospatial data securely available in the United Kingdom (UK) National Grid Service (NGS)[13] for use by the UK academic sector. Most of conducted research focused on efficient and secure processing of huge amounts of geospatial data in Grid Computing infrastructures through the SDI standards layer as for instance exemplified in [Di et al., 2003] and [Woolf and Shaon, 2009].

First experiments have demonstrated that Cloud Computing can be used as an on-demand and cost-efficient GIS hosting platform. ESRI Inc.[14] allows users and developers to access geospatial data via web applications and services that are hosted at Amazon EC2. WeoGeo Inc.[15] provides an online marketplace for the transformation, storage and sale of geospatial data. Although scaling aspects are considered, open SDI standards and security requirements are not addressed yet. The US Federal Geographic Data Committee (FGDC)[16] is developing "The Geospatial Platform" [FGDC, 2001], a Cloud Computing infrastructure to promote the sharing of geospatial data, services, and applications to support all levels of government. These examples show that especially the commercial but also the public sector achieved considerable progress towards cost-efficient hosting of geospatial applications, services and data based on Cloud Computing infrastructures.

From a research perspective, the interoperability of OWS and the Cloud Computing concept was studied in [Ludwig and Coetzee, 2010] and [Schäffer et al., 2010b]. The

[12] http://aws.amazon.com/ec2
[13] http://www.ngs.ac.uk
[14] http://www.esri.com
[15] http://www.weogeo.com
[16] http://www.fgdc.gov

licensing concept presented in [Schäffer et al., 2010a] for realizing access control and payment models in SDIs, was further developed in [Baranski et al., 2010a] to a general framework for realizing sustainable pay-as-you-go business models for geoprocessing services hosted in Public Clouds. Furthermore, some experiments have shown that Cloud Computing is suitable for realizing high-available and high-scalable geospatial services. In [Blower, 2010] an OGC WMS implementation was deployed at Google App Engine (GAE)[17]. In [Baranski et al., 2010b] the performance of an OGC WPS implementation was evaluated for GAE and Amazon EC2. The application of Hybrid Clouds for geospatial services was first described in [Foerster et al., 2010] based on the example of an OGC WPS implementation. The approach presented in [Baranski et al., 2011] analyzed how the Hybrid Cloud concept can be used to realize the challenging INSPIRE requirements regarding availability, performance and capacity.

## 2.5 Summary

In the past, the SDI development focused on interoperable standards for the retrieval, portrayal and processing of geospatial data through web services. The development of such standards was facilitated by emerging web service technologies and mandated by legal frameworks. The most sophisticated example for such a legal framework from Europe is the INSPIRE directive which aims at building an European SDI based on the member states' national SDIs. The INSPIRE guidelines and recommendations rely on the OGC Standard Baseline, which defines several mature and widely accepted standards for geospatial data and services. The OGC Standards Baseline not only covers the interoperable retrieval, portrayal and processing of geospatial data through web services. There are also several approaches for describing different quality aspects of geospatial data or to enforce digital rights management in SDIs. The GeoDRM approach for instance supports the on-demand and online negotiation of licenses for authentication and authorization, but completely ignores SLM and SLAs aspects, which have been a common product in support of services offered by telecommunications service providers for many years.

The main reason for implementing SLM in SDIs is to deliver the accurate service quality level for a specific use case exactly when it is needed. The key element for enforcing SLM is the SLA, which is a negotiated contract that helps to identify service quality expectations, clarify responsibilities, and facilitate communication between service providers and service consumers. The basic parts of SLAs are general information about the parties that negotiated the agreement, the KPIs that define a set of quantifiable measurements regarding specific aspects of the service quality, and the SLOs that define the minimum service quality level that is considered acceptable by service consumers and that can be guaranteed by service providers. There are many standardization approaches that aim to integrate SLAs in SOAs. One of them is the WS-Agreement specification that provides not only an XML Schema document for specifying the structure of agreement templates and agreements, but also a web service interface for managing the complete life cycle of agreements. The WS-Agreement specification is focused on the submission of jobs to a batch processing system. The WS-Agreement specification neither defines domain-specific web service descriptions, expressions and

[17] https://appengine.google.com

metrics for KPIs and SLOs, nor defines how and where to measure such properties. However, the WS-Agreement specification was investigated and developed in several research projects and finally reached a certain level of maturity in terms of features, extensibility and rate of adoption.

For service providers it is important to be in a position to actually deliver promised service quality levels to specific service consumers or in an application context. The Cloud Computing paradigm partially promises to fulfill such requirements. There are various definitions for the Cloud Computing paradigm available. What all of these definitions have in common is that a) Cloud Computing resources can be acquired in nearly any quantity at any time in order to scale applications, services and storage depending on an application context, and that b) Cloud Computing technologies enable resource providers to combine their physical and virtual resources in order to serve multiple users and their dynamically changing demands in an economical fashion.

The SDI development will continue to be faced with a rapidly increasing volume of available geospatial data with a higher spatial, temporal, and thematic resolution. Furthermore, not only the available volume of geospatial data but also the potential number of users will continue to increase. Along with emerging laws and provisions such as the INSPIRE directive, these developments result in ambitious requirements regarding the reliability, performance and scalability of geospatial services and corresponding applications. Therefore, the next chapter describes the domain-specific requirements for the application of the SLA concept in SDIs.

# Chapter 3

# Requirements Analysis

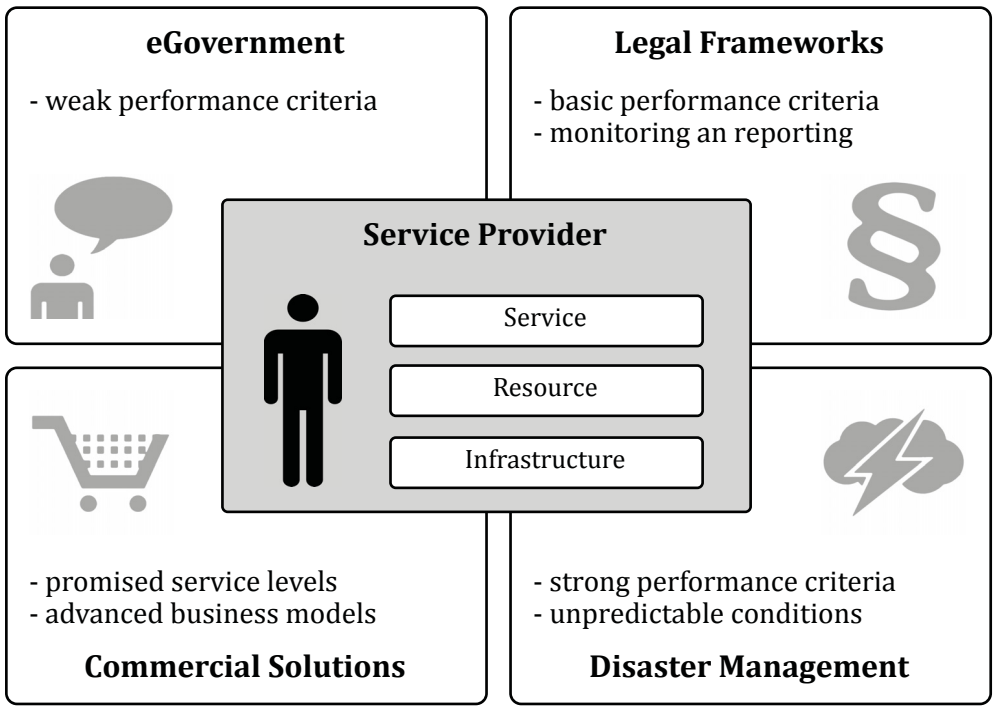
This chapter describes an abstract scenario that is composed of four application domains in which SDI services are utilized for different purposes. Based on these application domains and their particular characteristics, the conceptual requirements for the application of the SLA concept in SDIs are derived and generalized.

## 3.1 Scenario

The stakeholders in the SDI development are central or local governmental organizations, the commercial sector, non-governmental organizations, or the academic sector. The authors of [EU, 2011a] developed a typology that "distinguishes between countries that are national data producer led and those that are not" [Masser, 2005]. For example in Germany, which is not national data producer led, the governments from local, regional and national level are all generators and maintainers of public geospatial information. The authors of [EU, 2011c] provide a comprehensive overview about the nature and organization of the Geodateninfrastruktur in Deutschland (GDI-DE) initiative, which is the joint setup of the national SDI for Germany. The SDI development activities within the GDI-DE initiative are coordinated by a steering committee comprising members from the federal government, the federal states, and the communal head associations. The GDI-DE architecture describes not only the mission and the organization [GDI-DE, 2010] but also the standards and the roadmap [GDI-DE, 2010] for establishing the GDI-DE particularly with regard to the INSPIRE directive.

The GDI-DE architecture was developed to meet the requirements of several application domains as for instance eGovernment applications, legal frameworks, commercially available product solutions and disaster management. Some of the application domains may have challenging requirements regarding the service quality, while others may have lower requirements. In some cases the usage of the service is free of charge, in others the service provider charges a fee for using the service. For the purpose of this thesis, it is assumed that governmental SDI service providers offer services that are used in more than one application domain at the same time. One approach may be to deliver the highest possible service level to all service consumers and in every application context. This may however not be a reasonable solution in terms of technical and economical efficiency. Therefore, in order to realize higher levels of efficiency it is important for service providers to adjust the delivered service quality

Figure 3.1: Application Domains and Characteristics




and corresponding business conditions automatically according to the individual service consumers' requirements or the application context.

The following sections describe the GDI-DE application domains and their varying service level characteristics and corresponding business conditions (Figure 3.1).

### 3.1.1 eGovernment

The term eGovernment (also known as eParticipation) is defined as the use of Information and Communications Technology (ICT) to "move beyond the passive information-giving to active citizen involvement in the decision-making process" [Signore et al., 2005] in order to "improve the efficiency, acceptance, and legitimacy of political processes" [Sanford and Rose, 2007]. An example for such an application in the context of the GDI-DE activities is the prototypical development of an eParticipation application that "enable Wiesbaden's citizens to inform the city administration about infrastructural problems" such as pot-holes or broken street lighting [Blankenbach and Schaffert, 2009]. This web-based application is an integral part of the municipal SDI and integrates several OGC standards. Another example is TIM-online [Sandmann, 2005], an online platform that is provided by the Federal State of North-Rhine Westphalia in order to provide citizens with basic access to geographic information that is collected by the surveying and cadastre administrations.

The service delivery in eGovernment applications is focused on the overall user satisfaction, which does not only encompass weak service performance criteria for

realizing responsive applications but also other "soft" factors such as the design of the website (aesthetic), ease of use of the application (intuitiveness), and security and privacy (trust) [Alanezi et al., 2010]. Although the service quality requirements for eGovernment applications are not that much sophisticated, for service providers it is always important to provide reliable and responsive applications in order improve clients' relationships and satisfactions. The authors of [Quirchmayr et al., 2007] present a quality model of eGovernment services (e-GSQ Model) in order to verify whether government services meet citizens' needs or not.

### 3.1.2 Legal Frameworks

The governmental SDI stakeholders from state, national and regional level are usually mandated by legal frameworks to promote the SDI development. The GDI-DE activities for instance implement the INSPIRE directive, which aims at building an European SDI based on the member states' national SDIs. The INSPIRE directive defines a list of spatial data themes that must be collected and maintained by the EU member states. To ensure that the national SDIs of the member states of the EU are compatible, the INSPIRE directive also defines interoperable data encodings, standards for service interfaces and minimum service performance criteria. The INSPIRE View Service [INSPIRE, 2011a] for example is defined as an application profile of the OGC WMS specification and must be available 99% of the time (availability), the maximum initial response time of a `GetMap` request with 640x480 pixel must be 5 seconds (performance), and a service instance must be able to fulfill both of these criteria if the number of served simultaneous service requests is up to 20 per second (capacity).

Besides organizational aspects, legal frameworks such as the INSPIRE directive mostly focus on data encodings, service standards and performance criteria. The guidelines regarding the data encodings and service standards mostly refer to widely adopted OGC standards. The performance criteria guidelines mostly define acceptable values for classic web service quality attributes such as the minimum service availability (per week, month or year), the maximum service response time (for particular requests) and/or the ability to cope with a specific amount of traffic (a number of simultaneous service requests). For governmental service providers it is also of great relevance to monitor and report (to citizens or the superior governmental administration) whether their services are compliant to the regulations or not.

### 3.1.3 Commercial Solutions

The INSPIRE directive generally allows to charge fees for maintaining data and operating services, but does not define or recommend any specific pricing model. The "Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)"[1] defines guidelines in order to harmonize different pricing and accounting models for offline and online delivery of geospatial data in the federal states of Germany [AdV, 2009]. These guidelines are aligned with the INSPIRE directive and are the foundation for sales and distribution of products of the surveying and cadastre administrations within the GDI-DE initiative. Examples for such commercially

[1] Working Committee of the Surveying Authorities of the States of the Federal Republic of Germany

available product solutions are the Geobroker[2] from the "Landesvermessung und Geobasisinformation Brandenburg"[3] and the Geoserver[4] from the "Landesamt für Vermessung und Geoinformation Schleswig-Holstein"[5]. These online shops charge fees for the offline and online delivery of geospatial data, depending on the customers' affiliation, the number of users, and the area of application.

The AdV pricing models for online data delivery charge fees for service usage and downloaded content on a pay-per-request, pay-per-pixel or pay-per-feature basis. The pricing models are not intended for charging runtime-related or infrastructure-related fees. But in some scenarios, the service provider may would like to charge fees for providing specific service quality levels. In other scenarios, the service provider would like to forward the general costs for hosting a service to the service consumer. Furthermore, in the AdV pricing models it is not possible to define define penalties (or rewards) for not meeting (or fulfilling) promised service quality goals. All these aspects are important for developing innovative business models that help SDI service providers to promote their services in order to participate in the potential value chain of the future GIS marketplace [Alameh, 2003].

### 3.1.4 Disaster Management

Emergency management can be defined as "the discipline and profession of applying science, technology, planning, and management to deal with extreme events that can injure or kill large numbers of people, do extensive damage to property, and disrupt community life" [Drabek and Hoetmer, 1991]. The government agencies at all levels have primary responsibility for emergency management that encompasses a wide range of activities as for instance planning, mitigation, preparedness, response and recovery [Johnson, 2000]. In dealing with these extreme events, many of the critical problems that arise are inherently spatial [Cova, 1996] and GIS applications have become a critical tool for analysis purposes in all phases of emergency management [Su and Jin, 2009]. GIS applications can help for instance to identify the most critical areas that are affected by a hazard, or to calculate routes to location considering the impact of the hazard [Jeberson and Sasipraba, 2010]. An example for such an application in the context of the GDI-DE activities is PEGELONLINE[6], which provides data from over 500 water gauges across Germany in near real-time through an OGC Sensor Observation Service (SOS) [Bröring et al., 2012] and an intuitive website.

Providing the right information at the right time is reported to be crucial in disaster management [Meissner et al., 2002] and a couple of challenges for geographic information providers can be identified in this context [Parker, 2005]. One major challenge in disaster management is that services in spatial decision support systems must be highly-available and that requested information must be provided instantaneously (with reliable response time) [Kurzbach et al., 2009]. This is hard to realize, especially when the number of service consumers changes in disaster events

[2] http://geobroker.geobasis-bb.de
[3] The Land Survey Office of the State of Brandenburg in Germany
[4] http://www.gdi-sh.de/geoserver.html
[5] The Land Survey Office of the State of Schleswig-Holstein in Germany
[6] http://www.pegelonline.wsv.de

not only rapidly but also unpredictably. The PEGELONLINE website for example has approximately 400 users daily in times of no particular incidents, and 60000 users daily in times of flooding and heavy rainfalls [Steinmann, 2007]. An example for a productive running disaster management application is the Taiwan Debris Flow Monitoring System [Yin et al., 2010], which is operated by the Taiwan Soil and Water Conservation Bureau (SWCB)[7]. The Debris Flow Monitoring System is built upon several fixed and mobile monitoring stations equipped with many different sensors. In the case of a debris flow event in the typhoon season, not only the number of users (decision makers and citizens) that request real-time or historical information but also the amount of raw sensor data to be processed increases significantly. These changing conditions result in high requirements regarding the provision of sufficient computational resources in order to realize high-available and high-performance service delivery even in times when the service is accessed concurrently by a high number of users.

## 3.2 Requirements

Based on these application domains and their particular service characteristics, this section describes the requirements for the integration of SLAs in SDIs. An overview about all requirements can be found in Appendix A.

### 3.2.1 Roles and Relationships

The activities in SDIs are the retrieval, portrayal and processing of geospatial data through interoperable web services above organizational boundaries [Kiehle et al., 2006]. Typically SDIs implement the publish-find-bind pattern [Papazoglou, 2003], which describes the dynamic binding between service providers and service consumers in distributed and loosely-coupled environments [Massuthe et al., 2005]. The publish-find-bind scenario (Figure 3.2) identifies three different actors: the service provider, the service broker (sometimes referred to as service repository, service registry or service directory) and the service consumer. First, the service provider publishes his service along with additional metadata to a service broker. The service broker manages the service registry and helps service consumers to find service providers. Second, the service consumer performs service discovery operations on the service broker to find an adequate service provider according to functional or non-functional requirements. Finally, the service consumer uses the provided metadata to bind directly to a service.
The service consumption in the web-based SLA management architecture shall be performed only under the terms of previously created agreements. Therefore, the basic publish-find-bind pattern now must be enhanced by an additional 'agree' phase in which the service consumer and the service provider agree to certain SLA terms prior the service consumption (Figure 3.3). Such an additional agree phase is not new in the context of SDIs. The so-called 'publish-find-agree-bind' pattern presented in [INSPIRE, 2008b] and [Wagner, 2009] covers for instance rights management in SDIs with a focus on restricting the access to SDI services. Aspects such as authentication, authorization and pricing are covered, but SLM- and SLA-related aspects are missing.

[7] http://en.swcb.gov.tw

Figure 3.2: Publish-Find-Bind Pattern

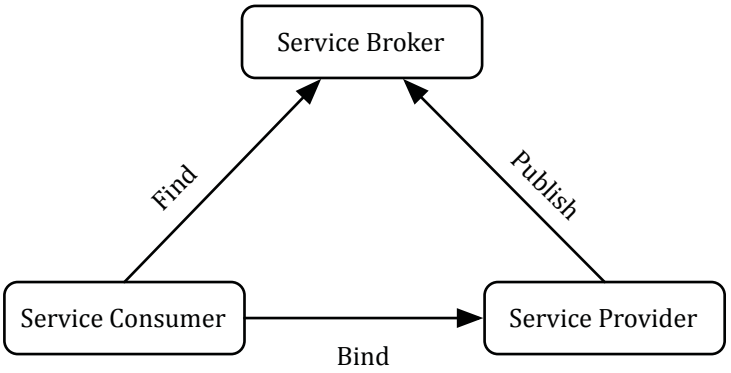
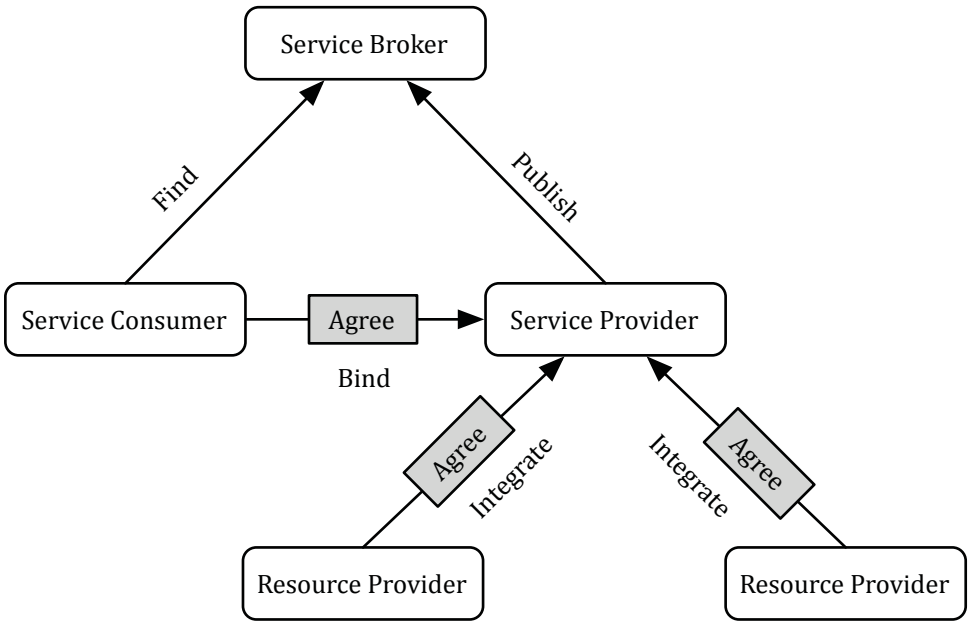



Figure 3.3: Publish-Find-Agree-Bind Pattern



The service provider hosts the offered resources (e.g. geospatial data or spatial models) in its own organizational boundaries and operates the offered SDI services in its own data center. Furthermore, the general trend in the mainstream IT hints into a future where data, services and applications are hosted on external infrastructures [Foster et al., 2008]. Therefore, data and infrastructure providers can be identified as additional roles in SDIs. This enhanced view on SDI actors could potentially be extended with a more fine-grained view on GeoDRM roles as for instance presented in [Wagner, 2006a] and [Wagner, 2009]. However, the presented differentiation into service provider, data provider and infrastructure provider is sufficient to highlight all important aspects of SLA contracting in the context of SDIs.

Each of the previously mentioned SDI actors can offer and implement different SLAs. The service provider offers for instance specific web service functionality, the data provider specific data quality and the infrastructure provider specific infrastructure reliability and capacity. Therefore, the additional agree phase encompasses that the service provider has to perform a SLA aggregation phase before being able to offer certain SLAs to the service consumer. Such a SLA aggregation phase is required even if the data and infrastructure provider are just different departments in the service providers' own organizational boundaries, since such departments are mostly separated in terms of responsibilities and accounting.

Therefore, for the integration of the SLA concept in SDIs, the publish-find-bind pattern in SDIs must be enhanced by additional SLA negotiation steps. These additional SLA negotiation steps shall not break completely the usual behavioral pattern of GI experts when searching and invoking SDI services.

**Requirement R1**

*The web-based SLA management architecture shall enable service consumers and service providers to negotiate SLA prior the service consumption.*

**Requirement R2**

*The web-based SLA management architecture shall ensure that the basic publish-find-bind pattern in SDIs remains.*

Furthermore, for a complete integration of SLAs in SDIs, each performed step (service publishing, service discovery and service consumption) must be enhanced by SLA-specific capabilities.

**Requirement R3**

*The service provider shall be able to publish his service along with additional SLA related information to the service broker.*

**Requirement R4**

*The service consumer shall be able to perform service discovery operations on the service broker to find an adequate service provider according to SLA related requirements.*

**Requirement R5**

*The web-based SLA management architecture shall ensure that service consumption is performed only under the terms of previously created agreements, in which service consumers and service providers agree to certain terms and conditions.*

### 3.2.2 Services, Resources and Quality

The most important element of an agreement is the actual description of the service. In the SDI context, such a service description mainly consists of functional and non-functional aspects of web services [Toma and Foxvog, 2008]. Furthermore, not only the encoding but also the meaning and the quality of delivered geospatial resources are an important aspect of service descriptions [Subbiah et al., 2007].

#### SERVICES

For a successful information integration in all application domains, it is always important for service consumers to find an adequate service provider according to individual functional requirements.

For the INSPIRE Network Services there are several technical guidance documents available for INSPIRE Discovery Services [INSPIRE, 2011b], INSPIRE View Services [INSPIRE, 2011a], INSPIRE Download Services [INSPIRE, 2009] and INSPIRE Transformation Services [INSPIRE, 2010a]. To ensure that the SDIs of the member states of the EU are compatible, these technical guidance documents and the "INSPIRE Network Services Architecture" document [INSPIRE, 2008b] detail mandatory service interfaces, which are aligned to corresponding OGC standards.

**Requirement R6**

*The abstract SLA model shall allow to create service offerings reflecting the service types that are standardized by the OGC and described by the INSPIRE directive.*

**Requirement R7**

*The abstract SLA model shall allow the definition of KPIs and SLOs reflecting the functional requirements as defined by the INSPIRE directive.*

**Requirement R8**

*The web-based SLA management architecture shall allow the monitoring of KPIs and the evaluation of SLOs reflecting the functional requirements as defined by the INSPIRE directive.*

#### RESOURCES

No less important for a successful information integration is the ability for service consumers to find not only appropriate service types but also applicable and accurate datasets [Paul and Ghosh, 2006].

The INSPIRE Data Specification documents[8] define 34 spatial data themes that must be

[8] http://inspire.jrc.ec.europa.eu/index.cfm/pageid/2

collected and maintained by the EU member states. The INSPIRE data specifications are separated in three different appendixes, the first of which for instance defines the spatial data themes "Administrative Units", "Cadastral Parcels", "Geographical Name", "Hydrography", "Protected Sites", "Transport Networks", "Addresses", "Coordinate Reference Systems" and "Geographical Grid Systems" [INSPIRE, 2008a].

**Requirement R9**

*The abstract SLA model shall allow the definition of spatial data themes as defined by the INSPIRE directive.*

The INSPIRE Data Specification documents also define data quality criteria for the spatial data themes. Therefore, the INSPIRE Data Specification on "Protected Sites" [INSPIRE, 2010c] for instance defines the completeness and the accuracy as relevant data quality elements that should be included in the dataset metadata. The completeness of spatial data objects can be documented by using either the number of excess (commission) or missing (omission) items in the dataset in relation to the number of items that should have been present. The accuracy of spatial objects can be documented by describing either the (absolute or external) accuracy within a dataset or the spatial resolution of a spatial object.

**Requirement R10**

*The abstract SLA model shall allow the definition of data quality elements as defined by the INSPIRE directive.*

The INSPIRE directive recognizes the need to be able "to gain access to and use spatial data and spatial data services in accordance with harmonized conditions" [EU, 2007]. The INSPIRE Data Sharing document defines that "sharing agreements can assume different forms, e.g. e-mail, license statement on a web-page, a click license, a license agreement signed by all the parties involved. Whatever form the agreement takes, it is legally binding and defines the conditions of use of the related spatial data sets and services" [INSPIRE, 2010b]. The "Interministerielle Ausschuss für Geoinformation des Bundes (IMAGI)"[9] started the pilot project GeoLizenz [Rech, 2011] that should establish a framework for providing harmonized license models for the public authorities in Germany. The framework contains 8 different license models that follow the open data philosophy [Ayers, 2007] and allow to grant usage rights independent of any pricing models.

**Requirement R11**

*The abstract SLA model shall allow the definition of license models for the provision of access to spatial datasets as recognized by the INSPIRE directive and implemented by the GeoLizenz license models.*

In some cases it is possible to check whether spatial data is inconsistent [Vallières et al., 2005] or complete [van Oort, 2005]. In other cases it is possible to attach information about the uncertainty in the data capture and mapping process [Comber and Wadsworth, 2005] or additional data provenance information

[9] http://www.imagi.de/

[Wang et al., 2008] to the data. However, the automated data validation and quality assessment processes (checking whether the delivered data is in accordance with demanded category, quality and license) is not part of this thesis.

#### QUALITY

Some of the application domains and corresponding scenarios may have challenging requirements regarding the service quality, while others may have lower requirements. However, in all application domains it is important for service consumers (service providers) to be aware of the service quality that can (shall) be delivered.

In Article 16 of the INSPIRE directive minimum performance criteria for all Network Services are required. Furthermore, the INSPIRE Network Services Performance Guidelines [INSPIRE, 2007] define such minimum performance criteria to be included in each of the Implementing Rules (IR) of all Network Services. Although the official performance guideline document is a 'non-paper' (it does not represent an official position of the European Commission), it "record(s) the consensus reached among the Network Services Drafting team, other Drafting Teams and the European Commission" [INSPIRE, 2007]. Furthermore, it helps other stakeholders to "better understand the framework used for the performance criteria included of the different Network Services Implementing Rules" [INSPIRE, 2007]. However, the performance guidelines document analyses the QoS attributes of the W3C [Lee et al., 2003] and selects all of them (except the scalability attribute) as applicable from the INSPIRE perspective. After comparing and combining the W3C and other approaches such as [Brahnmath et al., 2002], the following set of QoS attributes has been selected to guide the definition of minimum performance criteria for all Network Services: performance, reliability, capacity, availability, security, regulatory and interoperability of a web service. Based on the selected set of QoS attributes, the Network Services Drafting Team defines concrete performance criteria in the IR of all Network Services. The INSPIRE View Services for instance must implement a maximum initial response time of 5 seconds in normal situations and shall be available 99% of the time. Furthermore, all INSPIRE View Services should be able to fulfill both of these criteria if the number of served simultaneous service requests is up to 20 per second. These minimum performance criteria must be implemented by all EU member states.

**Requirement R12**

*The abstract SLA model shall allow the definition of KPIs and SLOs reflecting the minimum performance criteria as defined by the INSPIRE directive.*

**Requirement R13**

*The web-based SLA management architecture shall allow the monitoring of KPIs and the evaluation of SLOs reflecting the minimum performance criteria as defined by the INSPIRE directive.*

### 3.2.3 Pricing and Accounting

In some of the application domains service providers may would like to impose service usage fees, while in others they are obligated to do so. However, in all application

domains it is important for service consumers to be able to perform a benefit-cost ratio analysis before they bind to a service provider.

The article 14(1) of the INSPIRE directive defines that the "Member States shall ensure that the services (...) are available to the public free of charge", but the article 14(2) of the INSPIRE directive also allows "a public authority (...) to apply charges where such charges secure the maintenance of spatial datasets and corresponding data services, especially in cases involving very large volumes of frequently updated data". Furthermore, the Article 14 (4) of the INSPIRE directive defines, that "where public authorities levy charges for the services (...), Member States shall ensure that e-commerce services are available. Such services may be covered by disclaimers, click-licenses or, where necessary, licenses". In the "AdV-Gebührenrichtlinie" document [AdV, 2009] the AdV defines guidelines in order to harmonize different pricing and accounting models for offline and online delivery of geospatial data in the federal states of the Federal Republic of Germany. These guidelines are aligned with the INSPIRE directive and are the foundation for sales and distribution of products of the survey administrations within the GDI-DE initiative such as stipulated for instance in the "Gebührenordnung für das amtliche Vermessungswesen und die amtliche Grundstückswertermittlung in Nordrhein-Westfalen (VermWertGebT)" [NRW, 2010]. The VermWertGebT defines many different kinds of complex pricing models (e.g. flatrate, pay per click or sliding scale fees), which could potentially be required and implemented by all member states of the EU.

**Requirement R14**

*The abstract SLA model shall allow the definition of complex pricing models as described by the "AdV-Gebührenrichtlinie" and the "VermWertGebT".*

**Requirement R15**

*The web-based SLA management architecture shall allow the accounting of service offerings aligned with complex pricing models as described by the "AdV-Gebührenrichtlinie" and the "VermWertGebT".*

The implementation of accounting and billing procedures is out of the scope of this thesis. Nevertheless, the proposed solution for the integration of the SLA concept in SDIs should allow service providers as well as service consumers to gather and access all required information in order to realize accounting and billing without causing any legal issues.

### 3.2.4 Security and Rights Management

The on-demand and online negotiation of SLAs can result in legally binding electronic contracts [Gisler et al., 2000]. Therefore, an approach for the integration of the SLA concept in SDIs shall be developed with respect to generic security objectives as for instance confidentiality, integrity, authentication and non-repudiation [Hafner and Breu, 2008]. Furthermore, in some application domains it is important to implement rights management mechanisms in order to implement access control to geospatial resources [Herrmann, 2010].

The development of Digital Rights Management (DRM) mechanisms for geospatial data and services is not part of this thesis. Such approaches have been widely investigated as for instance in the GeoDRM reference model [Vowles, 2006], the GeoXACML specification [Matheus and Herrmann, 2011] and past OGC testbeds [Wagner, 2006b] [Gartmann and Leinenweber, 2009]. The GeoXACML specification for instance defines a geo-specific extension to the OASIS XACML specification [Moses, 2005] in order to provide support for spatial data types and spatial authorization decision functions in the encoding of access rights. The XACML model for an access control system can be used in the context of GeoXACML to enforce the access restrictions on geographic information.

**Requirement R16**

*The abstract SLA model shall allow the integration of access rights policies that are based on existing approaches such as GeoXACML.*

**Requirement R17**

*The web-based SLA management architecture shall allow the enforcement of access rights policies that are based on existing approaches such as GeoXACML.*

### 3.2.5 Infrastructure Management

In all application domains it is important for service providers and service consumers to agree on a certain service quality level that can be realized by the service provider. In some cases, service consumers may have specific requirements regarding the computing capacity of the service.

**Requirement R18**

*The abstract SLA model shall allow the definition of KPIs and SLOs reflecting infrastructure requirements and capabilities.*

**Requirement R19**

*The web-based SLA management architecture shall allow the monitoring of KPIs and the evaluation of SLOs reflecting infrastructure requirements and capabilities.*

Furthermore, in all application domains it is important for service providers to implement and maintain differentiated services in an economic fashion. In some of the application domains the service provider also may wish to pass the accruing costs for maintaining the actual service to the service consumer. That enables service providers to offer pricing models in which service consumers have to pay more or less, depending on whether they need for instance high availability or best effort service delivery.

**Requirement R20**

*The abstract SLA model shall allow the definition of complex pricing models reflecting the service infrastructure utilization.*

**Requirement R21**

*The web-based SLA management architecture shall allow the accounting of service offerings aligned with complex pricing models reflecting the service infrastructure utilization.*

For service providers it is always important to be in a position to actually deliver promised service quality to specific service consumers or in an application context.

**Requirement R22**

*The abstract SLA model shall allow the definition of infrastructure management information in order to realize differentiated services under the terms of previously created agreements.*

**Requirement R23**

*The web-based SLA management architecture shall support strategies for realizing differentiated services under the terms of previously created agreements.*

### 3.2.6 Standards and Technology

Interoperable and open standards are important key factors for the successful sharing of geospatial resources in SDIs [Wytzisk and Sliwinski, 2004]. Several potential ways for attaching SLA functionality to OWS can be identified. A conceptual redesign of all OWS interface specifications interferes with the huge number of existing OGC-compliant server and client implementations. To support SLA functionality, the modified OWS interface specifications have to be adopted by all server and client implementations. The probability of obtaining a broader acceptance of such a radical solution in the community's standardization process and for instance from government agencies appears to be very low. Therefore, the OGC Service Architecture model [Percivall, 2002] recommends that a security architecture should "leverage the existing OGC specifications by defining generic (...) licensing extensions" [Schäffer and Gartmann, 2011].

**Requirement R24**

*The abstract SLA model and the web-based SLA management architecture shall be developed with respect to the OGC Standards Baseline. They shall not replace any previous OGC specifications, but should depend and build on them.*

In the web-based SLA management architecture, service consumers should be able to perform a SLA-aware service discovery. To maintain and foster interoperability, the SLA-aware service discovery and the following SLA negotiation process should be realized independent of vendor- and implementation-specific solutions. That enables service consumers to compare service offerings and corresponding business models of different service providers.

**Requirement R25**

*The development of a standardized document encoding for the abstract SLA model is strongly recommended.*

**Requirement R26**

*The development of a standardized communication protocol for realizing SLA negotiation and service consumption under the terms of previously created SLAs is strongly recommended.*

Such a standardized document encoding should contain definitions for describing the agreement context, the service terms (including service descriptions and domain-specific KPIs) and the guarantee terms (including domain-specific SLOs and corresponding business values). Furthermore, such a standardized document encoding should contain definitions for describing meaningful metrics in order to establish a common understanding on how to monitor the KPIs and how to evaluate the SLOs. All these definitions should be designed in flexible way in order to be able to integrate new domain-specific service descriptions and KPIs.

**Requirement R27**

*The abstract SLA model and the web-based SLA management architecture should be designed in a flexible way in order to be applicable in other application domains.*

## 3.3 Summary

The stakeholders in the SDI development are central or local governmental organizations, the commercial sector, non-governmental organizations, or the academic sector. They all use the SDI building blocks in different application domains as for instance eGovernment applications, legal frameworks, commercially available product solutions and disaster management. Each of these application domains has varying service level characteristics and different business conditions. Some of the application domains may have challenging requirements regarding the service quality, while others may have lower requirements. In some cases the service usage is free of charge, in others the service provider charges a fee for using the service.

Based on these application domains and their particular service level characteristics and business conditions, this chapter describes the conceptual requirements for the integration of SLAs in SDIs. The domain-specific requirements cover aspects such as the functional and non-function description of geospatial services and datasets, the definition of dynamic pricing and accounting models for the offline and online delivery of geospatial data, the enforcement of access rights policies for geospatial data and services, the management of (virtualized) hardware infrastructure in order to fulfill challenging performance and availability criteria, and the general integration of the SLA concept in the OGC Standards Baseline.

The ORM does not contain any standards or best practices for the integration of SLAs in SDIs. Furthermore, the evaluated domain-specific requirements are not considered by existing (web service) standards for specifying the structure of agreements and for managing the complete agreement life cycle. Therefore, the next two chapters develop a concept for the integration of SLAs in SDIs. The concept consists of an abstract SLA model and a web-based SLA management architecture. The abstract SLA model describes the domain-specific structure and content of SLAs that can be applied in SDIs

that are based on standards developed by the OGC. The web-based SLA management architecture covers not only the agreement negotiation and service consumption, but also the infrastructure management under the terms of previously created agreements.

# Chapter 4

# Agreement Formalization

Based on the abstract scenario and the evaluated requirements, this chapter formalizes an abstract SLA model that is applicable in SDIs. The aim of the abstract SLA model is to provide an abstract representation of the agreement on a conceptual level, which is independent of any specific data encoding format. Although the abstract SLA model should be independent of any specific technology, most of the proposed elements and attributes are derived from common standards as for instance the OGC Web Services Common Standard [Whiteside and Greenwood, 2010] specification.

The abstract SLA model is composed of several complement parts. Section 4.1 describes the domain-specific structure and content of the abstract SLA model. Section 4.2 describes how to monitor domain-specific service properties. Section 4.3 describes how to evaluate the status of service level objectives and how to calculate certain business values.

## 4.1 Agreement Structure

The structure of the abstract SLA model is derived from the agreement structure presented in [Andrieux et al., 2005] and mainly consists of the following three parts. The *Agreement Context* part contains general information such as contact details of the contracting parties and for instance the lifetime of an agreement. The *Service Terms* part contains domain-specific information about the services to which an agreement is related (*Service Description*), a domain-specific reference to a concrete service instance (*Service Reference*) and a set of domain-specific properties associated with the service (*Service Properties*). The *Guarantee Terms* part specifies the service quality goals that the contracting parties are agreeing (*Service Level Objectives*) and general business related properties such as service usage costs and for instance penalty fees (*Business Values*).

Figure 4.1 shows the structure of the abstract SLA model as an Entity Relationship Model (ERM) [Chen, 1976] in the Crow's Foot Notation [Shelly et al., 1998]. This sort of diagram allows the creation of an information model, which is an abstract description of the entities of a system and an abstract description of how these entities relate to each other [Westerinen et al., 2001]. Figure 4.2 shows the cardinality notation of the Crow's Foot Notation. In the first example, an object A must contain exactly one object B. In the second example, an object A must contain one or many objects B. In the third example, an object A can contain zero, or one, or many objects B. In the third example, an object

Figure 4.1: Agreement Structure

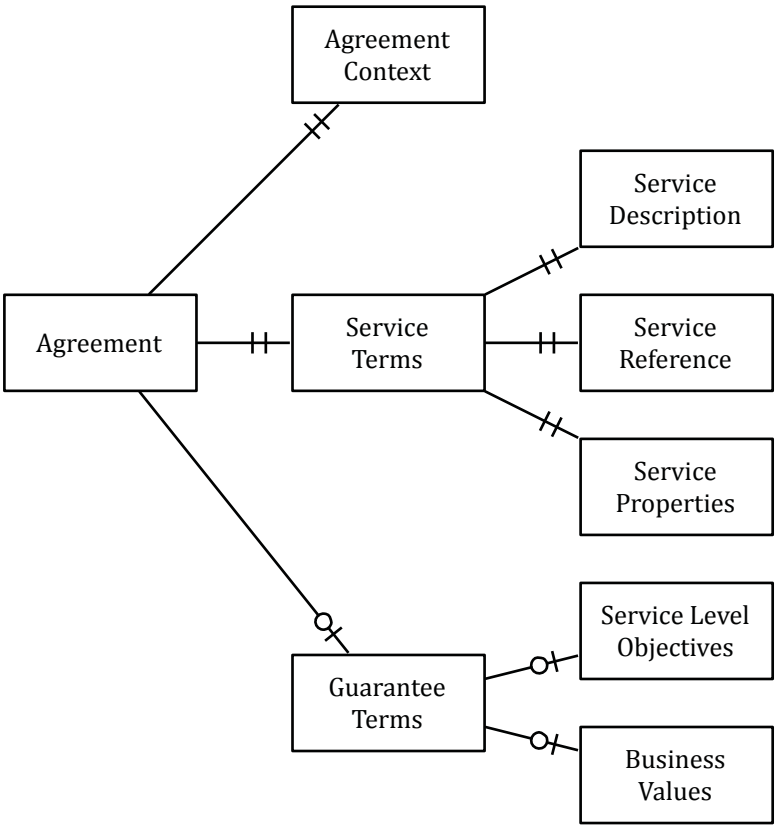


Figure 4.2: Crow's Foot Notation (adapted from [Shelly et al., 1998]).

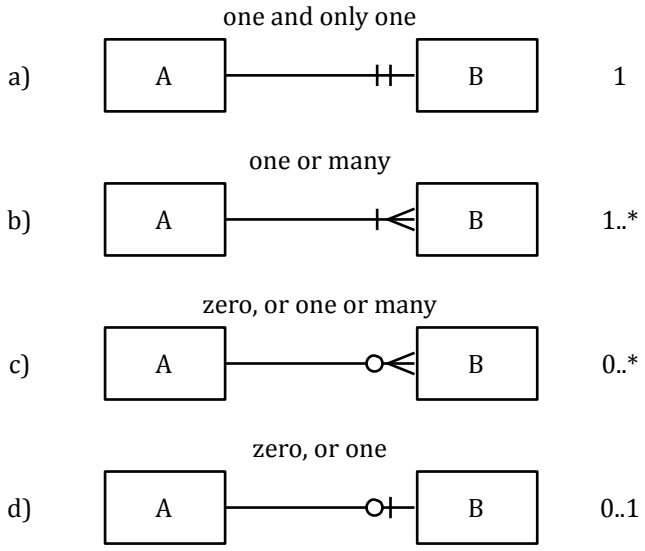
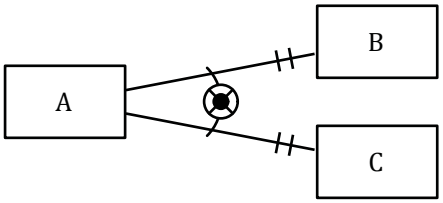



Figure 4.3: Exclusive Or Constraint (XOR)



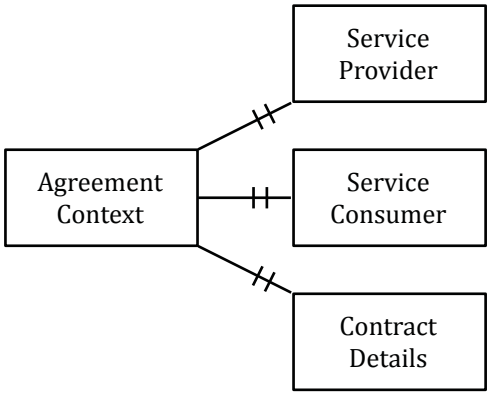

A can contain zero, or one object B. Figure 4.3 shows how to define an exclusive or (XOR) relationship between two or more entities by a dotted circle with an X through it [Halpin, 2001]. In the example, an object A either contains one object B or one object C, but not both.

The examples provided in this section are shown in JavaScript Object Notation (JSON) format [Crockford, 2006], a lightweight and human-readable open standard that is commonly used for data exchange in JavaScript [Flanagan, 1998]. Despite of the JSON format, this section provides an abstract SLA model that is independent of any specific data encoding format, and that can be implemented in several ways.

### 4.1.1 Agreement Context

An agreement is a formal contract between a service consumer and a service provider defining mutual understandings and expectations of service functionality, service quality and additional financial arrangements. Therefore, an agreement shall contain general information about the contracting parties and additional contract management information. The contracting parties and their respective roles are typically the "service

Figure 4.4: Agreement Context Structure




consumer" and the "service provider" [Jin et al., 2002]. Sometimes, the roles involved in the SLA negotiation process are referred to as the "agreement initiator" and the "agreement responder" [Andrieux et al., 2005]. In the chosen publish-find-agree-bind pattern (Section 3.2.1) the service consumer is always the initiator of the agreement negotiation process. Therefore, the abstract SLA model focus solely on the terms "service consumer" and "service provider" as the contracting parties. However, the *Agreement Context part* of the abstract SLA model shall contain the following elements (Figure 4.4):

**Service Provider**

The `Service Provider` element must contain information about the organization operating the service, such as contact details and for instance the hours of service. The `GetCapabilities` operation of an OWS returns metadata about the organization operating an OWS [Whiteside and Greenwood, 2010]. The `ProviderName` element in the `GetCapabilities` response document is an unique identifier for a service provider organization. The `ProviderSite` element in the `GetCapabilities` response document refers to the most relevant web site of the service provider and the `ServiceContact` element in the `GetCapabilities` response document specifies information for contacting a service provider. These information are derived from the corresponding classes in ISO 19115 [ISO, 2003] and contain common contact details such as the postal address, telephone or facsimile number, electronic mail address and the hours of service. All these elements are appropriate for providing information about the service provider and can be used in the *Agreement Context* part of the abstract SLA model.

**Service Consumer**

The `Service Consumer` element shall contain information about the service consumer such as contact details (maybe a different home and billing address) and for instance a banking account. The metadata elements from the `GetCapabilities` response document can also be used for providing information about the service consumer in the *Agreement Context* part of the abstract SLA model.

#### Contract Details

An agreement is naturally a temporary contract and therefore it normally has fixed start and end dates. Sometimes it is possible to end contracts ahead of schedule (e.g. in case of contract violations) or to automatically perform a contract renewal after a specific period of time under the same contract terms and conditions (e.g. after missing a contract cancellation deadline). However, such advanced opportunities for contract renegotiation are not considered by the abstract SLA model and the web-based SLA management architecture. Therefore, fixed start and end dates are entirely sufficient to support the required functionality.

An example of the *Agreement Context* part can be found in Listing 4.1.

Listing 4.1: Agreement Context Example

```
"Agreement Context":
{
  "Service Provider":
  {
    "Name": "Institute for Geoinformatics",
    "Site:" "http://www.ifgi.de",
    "Contact":
    {
      "IndividualName": "Bastian Baranski",
      "PositionName": "Research Associate",
      "ContactInfo":
      {
        "Phone":
        {
          "Voice": "+49 251 8333071",
          "Facsimile": "+49 251 8339763"
        },
        "Address":
        {
          "DeliveryPoint": "Weseler Strasse 253",
          "City": "Muenster",
          "PostalCode": "48151",
          "Country": "Germany",
          "ElectronicMailAddress": "baranski@uni-muenster.de"
        },
        "HoursOfService": "The hours of service are Monday to Friday from 8
            AM to 16 PM.",
        "ContactInstructions": "Please contact the service desk via phone or
             mail."
      }
    }
  },
  "Contract Detail":
  {
    "Contract Period":
    {
      "Start": "2010-07-04T13:00:00+02:00",
      "End": "2012-07-09T13:00:00+02:00"
    }
  }
}
```

The presented list of elements for the *Agreement Context* part can potentially be extended with a more fine-grained view on organizational processes and required contract information. Therefore, a standardized document encoding format for implementing the abstract SLA model should be designed in a flexible way to allow

the integration of additional domain-specific information.

### 4.1.2 Service Description

An agreement shall contain domain-specific information about the service to which an agreement pertains. The domain-specific service description should contain a human-readable description of the service (to make a manual service discovery more user-friendly) as well as a machine-readable description of the service (to allow the implementation of interoperable server and client implementations). Therefore, the *Service Description* section of the abstract SLA model shall contain the following elements (Figure 4.5):

**Title**

The *Service Description* section must contain a short human-readable title of the service to which an agreement pertains. The `Title` element of the `GetCapabilities` response document defines such a human-readable title of a specific service instance.

**Abstract**

The *Service Description* section should contain a brief narrative description of the service to which an agreement pertains. The `Abstract` element of the `GetCapabilities` response document defines such a human-readable narrative description of a specific service instance.

**Keywords**

The `Keywords` element of the `GetCapabilities` response document is an unordered list of one or more commonly used or formalized keywords or phrases that can be used to describe a specific service instance, the service context or the service capabilities.

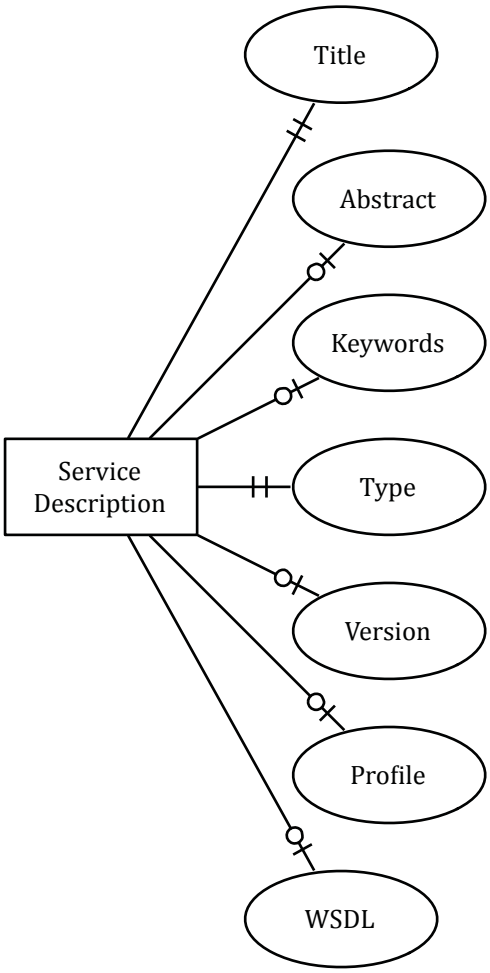
**Type**

The OGC standards are documents describing service interfaces and data encodings. To realize seamless data exchange via interoperable web services and client implementations, it is important to know the specific OGC standard that is supported by a service instance. The `ServiceType` element of the `GetCapabilities` response document specifies an URN from a registry of services and identifies an OGC service interface specification. In the `GetCapabilities` response document such an URN shall comply to the structures as specified for instance in [Cox, 2010d] and [Reed, 2004]. These URNs must be used in the *Service Description* section to provide a machine-readable type identifier of the service to which an agreement pertains.

**Version**

The `ServiceTypeVersion` element of the `GetCapabilities` response document defines the concrete version of an OGC service interface specification which is implemented by a specific service instance. This value could be used in the

Figure 4.5: Service Description Structure



*Service Description* section to provide a machine-readable version identifier of an OGC service interface specification that is supported by the service to which an agreement pertains.

**Profile**

In the context of OWS the term "Application Profile" (sometimes referred to as "Application Schema") is used whenever an extension of an OGC service interface specification or data encoding is required to match a give use case with extended functionality that could not be covered by the original OGC service interface specification or data encoding [Whiteside and Greenwood, 2010]. An application profile extends and/or restricts for instance the XML elements of an OGC standard in order to give a service provider the opportunity to gather or deliver additional information. Such an application profile can be identified by an URN from a registry of application profiles and contains for instance a reference to an XML schema describing a service interface. The `Profile` element of the `GetCapabilities` response document contains such a reference to an application profile, which can be used in the abstract SLA model to provide additional service information.

**WSDL**

The Web Services Description Language (WSDL) is an XML language for describing the functional characteristics of a web service as for instance the service interface, protocol bindings and network endpoints [Christensen et al., 2001]. An OWS typically does not have a Simple Object Access Protocol (SOAP) binding [W3C, 2007] and is therefore not described by a WSDL document. But if an OWS supports SOAP, the `WSDL` element of the `GetCapabilities` response document specifies a reference to such a WSDL document, which can be used in the abstract SLA model to provide additional machine-readable information about the service functionality.

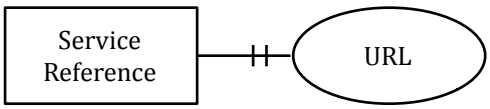
An example of the *Service Description* section can be found in Listing 4.2.

Listing 4.2: Service Description Example

```
"Service Description":
{
  "Title": "INSPIRE View Service",
  "Abstract": "This service instance is an INSPIRE View Service
      implementation.",
  "Keywords": "INSPIRE, View Service, OGC, WMS",
  "Type": "urn:ogc:doc:is:wms:1.1.1"
}
```

The presented list of potential elements for the *Service Description* section of the abstract SLA model can potentially be extended with a more fine-grained view on service descriptions. Since this thesis focuses on SDIs that are based on OGC standards, the described elements are sufficient to meet all stipulated requirements. Nevertheless, a potential standardized document encoding for implementing the abstract SLA model should be designed in a flexible way to allow the integration of additional domain-specific service descriptions.

Figure 4.6: Service Reference Structure




### 4.1.3 Service Reference

An agreement must contain a domain-specific reference to a concrete service instance to which the agreement is related. The *Service Reference* section of the abstract SLA model shall contain the following elements (Figure 4.6):

**URL**

In the context of SDIs that are based on standards developed by the OGC, services offerings are typically referenced by web service endpoints as for instance described in [Gudgin et al., 2006]. In most cases, a simple Hypertext Transfer Protocol (HTTP) Uniform Resource Locator (URL) is such an appropriate service reference.

An example of the *Service Reference* section can be found in Listing 4.3.

Listing 4.3: Service Reference Example

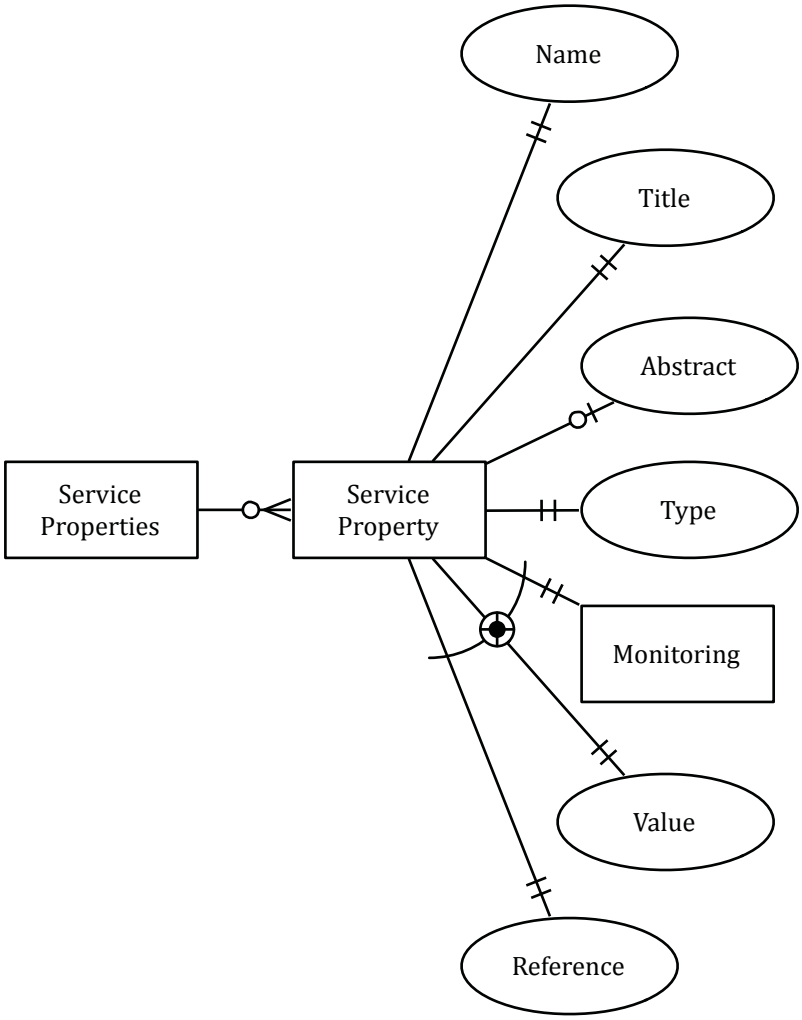
```
1 "Service Reference":
2 {
3   "URL": "http://server:port/path"
4 }
```

In some cases, the SDI service is protected with authentication mechanisms such as HTTP Authentication [Franks et al., 1999]. In that case, the service consumer has to provide additional credentials (e.g. username and password) in order to invoke a service. Therefore, a potential standardized document encoding for implementing the abstract SLA model should be designed in a flexible way to allow the integration of additional information on how users can obtain required credentials in order to invoke the service.

### 4.1.4 Service Properties

An agreement typically defines a set of domain-specific KPIs that are used to measure the service quality. Therefore, the *Service Properties* section of the abstract SLA model should contain a list of zero, or one, or many `Service Property` elements (Figure 4.7). Each of these `Service Property` elements in the *Service Properties* section specifies which service property shall be monitored by the web-based SLA management architecture during agreement runtime. Therefore, the `Service Property` element of the abstract SLA model shall contain the following elements (Figure 4.7):

Figure 4.7: Service Properties Structure



**Name**

Each `Service Property` element can be referenced in the *Service Level Objectives* and the *Business Values* section of the abstract SLA model. Therefore, each `Service Property` element must contain a short and unique name.

**Title**

Each `Service Property` element element must contain a short human-readable title of the service property that shall be monitored by the web-based SLA management architecture during agreement runtime.

**Abstract**

Each `Service Property` element should contain a brief narrative description of the service property that shall be monitored by the web-based SLA management architecture.

**Type**

Each `Service Property` element must contain an URN from a registry of service property types that identifies the concrete service property that shall be monitored by the web-based SLA management architecture during agreement runtime. Section 4.3.1 defines such an URN namespace for identifying concrete service property types.

Table 4.1 provides an overview about all service property categories that are supported by the abstract SLA model. Some of these categories reference service characteristics that can be automatically monitored during agreement runtime (e.g. the web service response time). In the case that a `Service Property` element is from one of these categories, it must contain a `Monitoring` element. This element specifies how the web-based SLA management architecture should measure the service property during agreement runtime. Other categories describe aspects of the service delivery that cannot be automatically monitored during agreement runtime (e.g. the license for sharing data). In the case that a `Service Property` element is from one of these categories, it must contain either a `Value` or a `Reference` element. These elements specify non-varying service characteristics that are determined and guaranteed by the service provider.

Table 4.1: Service Property Categories

| Category | Description |
|---|---|
| Resource | The resource-related service properties reference resources that are offered by a service as for instance maps or processes. The availability or conformance of these resources can be automatically monitored during agreement runtime. |

Table 4.1 – Continued on next page

Table 4.1 – Continued from previous page

| Category | Description |
|---|---|
| Runtime | The runtime-related service properties reference classic web service quality characteristics as for instance the service availability or the service response time. The values of such characteristics can be automatically monitored during agreement runtime. |
| Usage | The usage-related service properties reference the service or the infrastructure utilization by the service consumer. The behavior of the service consumer can be automatically recorded during agreement runtime. |
| Data | The data-related service properties reference quality aspects of the delivered data as for instance the data accuracy or the license for data sharing. These data characteristics cannot be automatically measured or validated during agreement runtime. Therefore, such service properties define guarantees that are provided by the service provider. |
| Security | The security-related service properties reference security policies that are used to enforce service access control. These policies can be used by the web-based SLA management architecture for instance to restrict the service access to identified user groups. |
| Infrastructure | The infrastructure-related service properties reference different aspects of the service hosting environment as for instance the available Central Processing Unit (CPU) clock speed. These non-varying information can be attached to an agreement in order to define infrastructure requirements of the service consumer or to define infrastructure management information that are used internally by the service provider. |

#### Monitoring

In case the `Service Property` element is from one of the categories that can be automatically monitored during agreement runtime, it must contain a description of how the web-based SLA management architecture should monitor the relevant service property. Section 4.2 describes the potential content of the `Monitoring` element and the available monitoring mechanisms of the web-based SLA management architecture.

#### Value

In case the `Service Property` element is from one of the categories that cannot be automatically monitored during agreement runtime, it must contain a description of the service property. The `Value` element provides a non-varying description directly included in the agreement.

#### Reference

In case the `Service Property` element is from one of the categories that cannot be automatically monitored during agreement runtime, it must contain a description of the service property. The `Reference` element provides a potentially varying description by an URL pointing to a public accessible resource (e.g. a file on a web server) that contains the actual description.

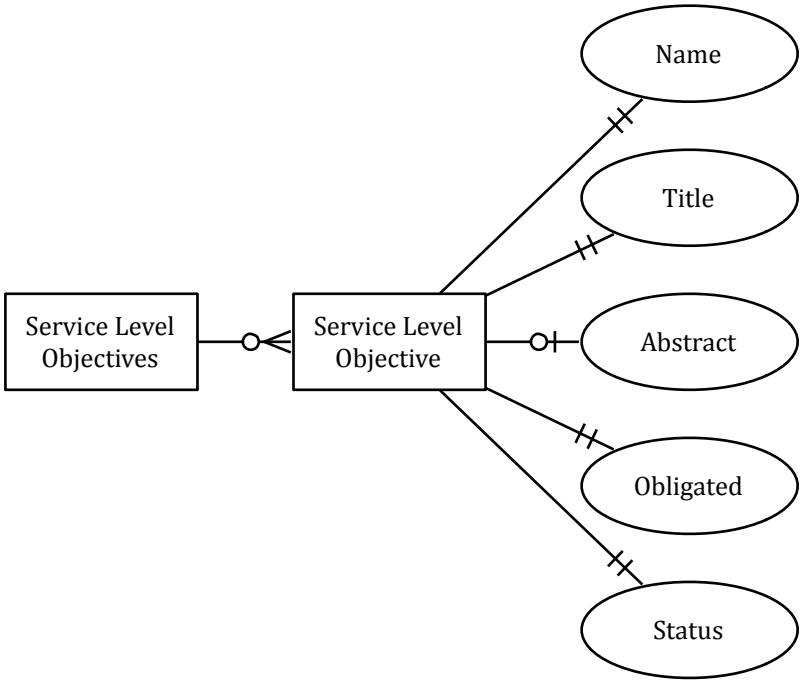
An example of the *Service Properties* section can be found in Listing 4.4.

Listing 4.4: Service Properties Example

```
"Service Properties":
{
  (...)
  "Service Property":
  {
    "Name": "availability",
    "Title": "Service Availability",
    "Abstract": "The general availability of the service.",
    "Type": "urn:ogc:def:sla:property:runtime:availability",
    "Monitoring":
    {
      (...)
    }
  },
  "Service Property":
  {
    "Name": "response",
    "Title": "Response Time",
    "Abstract": "The response time of the service.",
    "Type": "...",
    "Monitoring":
    {
      (...)
    }
  },
  "Service Property":
  {
    "Name": "capacity",
    "Title": "Service Capacity",
    "Abstract": "The response time of the service for multiple parallel
        requests.",
    "Type": "...",
    "Monitoring":
    {
      (...)
    }
  },
  "Service Property":
  {
    "Name": "pixel",
    "Title": "Sum of Pixels",
    "Abstract": "The accessed number of pixels.",
    "Type": "...",
    "Monitoring":
    {
      (...)
    }
  },
  (...)
}
```

The *Service Properties* section of the abstract SLA model allows to define more than

Figure 4.8: Service Level Objectives Structure




one `Service Property` element for the same service property type. That allows to define different monitoring setups for one service property and that helps to define more complex service level objectives.

### 4.1.5 Service Level Objectives

Based on the defined `Service Property` elements of the abstract SLA model, the *Service Level Objectives* section defines concrete service quality goals both contracting parties agreed upon in the agreement. Therefore, the *Service Level Objectives* section should contain a list of zero, or one, or many `Service Level Objective` elements (Figure 4.8). Each `Service Level Objective` element defines a concrete service quality goal and provides a description on how to evaluate whether the specific service quality goal is fulfilled or violated. Therefore, the `Service Level Objective` elements of the abstract SLA model shall contain the following elements (Figure 4.8):

**Name**

Each `Service Level Objective` element can be referenced in the *Business Values* section of the abstract SLA model. Therefore, each `Service Level Objective` element must contain a short and unique name.

**Title**

Each `Service Level Objective` element must contain a short human-readable title of the concrete service level objective that shall be evaluated by the web-based SLA management architecture.

#### Abstract

Each `Service Level Objective` element should contain a brief narrative description of the service level objective.

#### Obligated

Each `Service Level Objective` element must define which contracting party is obligated to realize the service level objective (either "Service Provider" or "Service Consumer").

#### Status

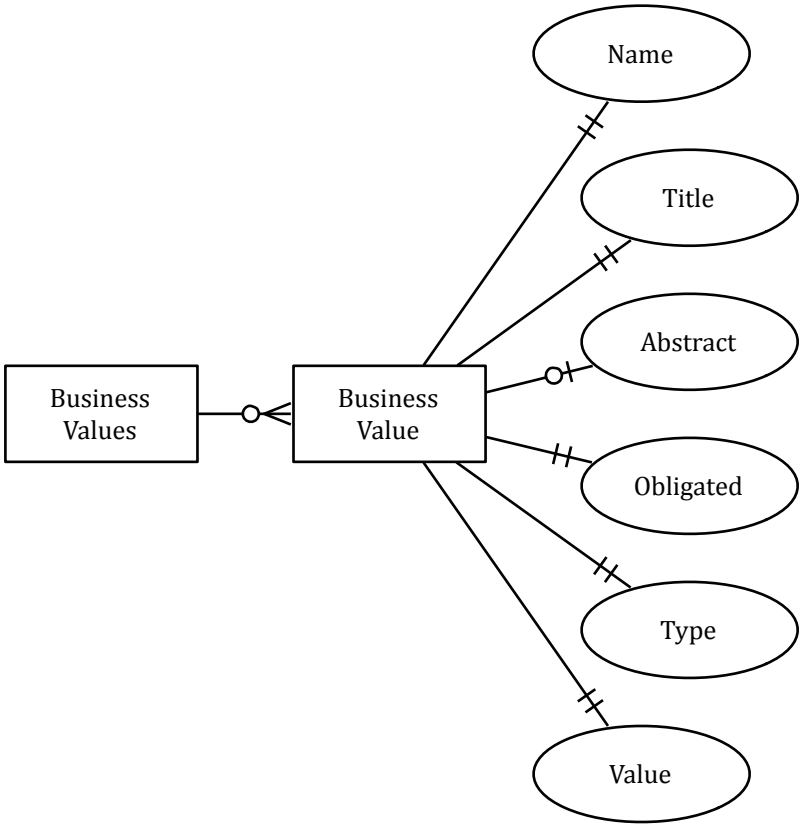
Each `Service Level Objective` element must contain a machine-readable expression defining the evaluation process of the service level objective. The result of the evaluation process is the status of the service level objective (either 'false' for a guarantee violation or 'true' for a guarantee conformance). Section 4.3 defines a DSL that can be used to define the evaluation process of the service level objective.

An example of the *Service Level Objectives* section can be found in Listing 4.5.

Listing 4.5: Service Level Objectives Example

```
"Service Level Objectives":
{
  "Service Level Objective":
  {
    "Name": "InspireAvailability"
    "Title": "INSPIRE (Availability)",
    "Abstract": "The probability of a Network Service to be available shall
        be 99% of the time.",
    "Obligated": "Service Provider",
    "Status": "..."
  },
  "Service Level Objective":
  {
    "Name": "InspirePerformance"
    "Title": "INSPIRE (Performance)",
    "Abstract": "The response time for sending the initial response to a Get
         Map Request to a view service shall be maximum 5 seconds in normal
        situation.",
    "Obligated": "Service Provider",
    "Status": "..."
  },
  "Service Level Objective":
  {
    "Name": "InspireCapacity"
    "Title": "INSPIRE (Capacity)",
    "Abstract": "The minimum number of served simultaneous service requests
        to a view service according to the performance quality of service
        shall be 20 per second.",
    "Obligated": "Service Provider",
    "Status": "..."
  }
}
```

Figure 4.9: Business Values Structure




### 4.1.6 Business Values

The *Business Values* section of the abstract SLA model defines business aspects of the agreement as for instance service usage costs. Therefore, the *Business Values* section should contain a list of zero, or one, or many `Business Value` elements (Figure 4.9). Each `Business Value` element identifies a concrete business aspect and provides a description on how to calculate the business value. Therefore, the `Business Value` element of the abstract SLA model shall contain the following elements (Figure 4.9):

**Name**

Each `Business Value` element can be referenced by another `Business Value` element of the abstract SLA model. Therefore, each `Business Value` element must contain a short and unique name.

**Title**

Each `Business Value` element must contain a short human-readable title of the business value.

**Abstract**

Each `Business Value` element should contain a brief narrative description of the business value.

**Obligated**

Each `Business Value` element must define which contracting party is charged for the business value (either "Service Provider" or "Service Consumer").

**Type**

Each `Business Value` element must contain an URN from a registry of business value types that identifies the concrete business value that shall be calculated by the web-based SLA management architecture during agreement runtime. Section 4.3.1 defines such an URN namespace for identifying concrete business value types.

**Value**

Each `Business Value` element must contain a machine-readable expression defining the calculation process of the business value. The result of the calculation process is for instance the service usage fee in Euro. Section 4.3 defines a DSL that can be used to define the calculation process of the business value.

An example of the *Business Values* section can be found in Listing 4.6.

Listing 4.6: Business Values Example

```
"Business Values"
{
  "Business Value":
  {
    "Name": "CostsPerYear",
    "Title": "Usage Costs (Year)",
    "Abstract": "The cost to be assessed for using the service on a yearly
        basis (in Euro).",
    "Obligated": "Service Consumer",
    "Type": "...",
    "Value": "..."
  },
  "Business Value":
  {
    "Name": "PenaltyPerYear",
    "Title": "Penalty (Year)",
    "Abstract": "The penalty to be assessed for not meeting service level
        objectives on a yearly basis (in Euro).",
    "Obligated": "Service Provider",
    "Type": "...",
    "Value": "..."
  }
}
```

## 4.2 Agreement Monitoring

The *Service Description* section of the abstract SLA model contains basic domain-specific information about the service to which an agreement pertains. The *Service Properties* section of the abstract SLA provides information about the resources that are offered by a service (Resource-Related Properties), information about classic web service quality (Runtime-Related Properties) and information about the service usage (Usage-Related

Properties). These service property categories can be assessed and measured during agreement runtime. The gathered information are the foundation for the evaluation of the service level objectives. Furthermore, the *Service Properties* section provides information about the service hosting environment (Infrastructure-Related Properties) or the delivered data (Data-Related Properties). These service property categories cannot be monitored during agreement runtime, but they can be use as an agreement management input for service providers and service consumers.

To obtain the concrete information about the service properties during agreement runtime, the following two main methods for monitoring web services can be identified as for instance described in [Berger, 2005] and [Dotcom-Monitor, 2012]:

- **Active Monitoring**

  The so-called "active monitoring" process simulates end-user behavior by creating real traffic at the web service (Figure 4.10a). To realize active monitoring, a monitoring software continuously sends pre-defined requests to the service and analyzes multiple metrics as for instance service availability and response time. There are a number of benefits that can be realized with the active monitoring approach. The proactive approach allows to permanently simulate peak traffic and therefore to detect potential service level problems before they actually come up at the real end-user side. Furthermore, the active monitoring approach allows to monitor the web service from nearly every location and therefore to detect issues that occur only when accessing the web service from outside of the service providers' network. The main disadvantage of the active monitoring approach is that it does not provide information about the real service usage and therefore no evidence whether a service really delivers aimed service level.

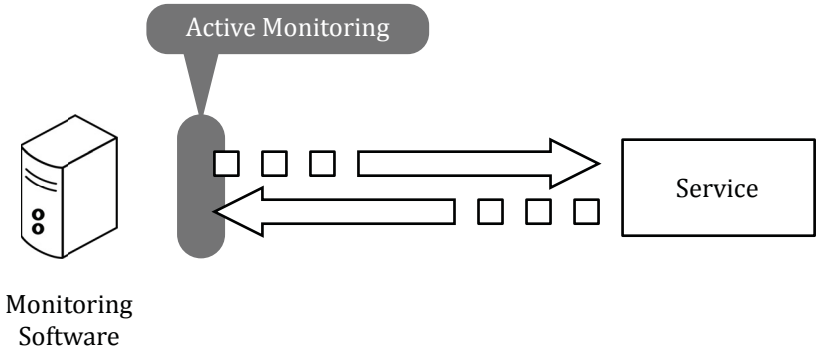
- **Passive Monitoring**

  The so-called "passive monitoring" process captures and analyzes real end-user traffic at the web service (Figure 4.10b). To capture and analyze real end-users traffic the service provider can use enterprise management tools or common software for analyzing web server logging files. The main advantage of the passive monitoring approach is to monitor the service level as seen by the real end-user and to have evidence whenever a service failed to deliver aimed service level to a specific user or in an application context. Additionally, passive monitoring allows to record real end-user behavior on the web service. That allows to check whether the user uses the service within potential constraints regarding the guarantee terms (e.g. maximum number of hourly service requests) and to realize pay-as-you-go business models (e.g. service usage costs depending on the amount of monthly service requests). The main disadvantages of the passive monitoring approach are that it requires real end-user traffic and that it offers no ability to monitor issues that occur outside of the service providers' network. Therefore, passive monitoring reports service level problems only after they happened.

To offer the ability to define whether a service property shall be monitored with active or passive monitoring mechanisms, the `Monitoring` element shall contain either an `Active Monitoring` or a `Passive Monitoring` element (Figure 4.11).

Figure 4.10: Monitoring Procedures

(a) Active Monitoring Procedure




(b) Passive Monitoring Procedure

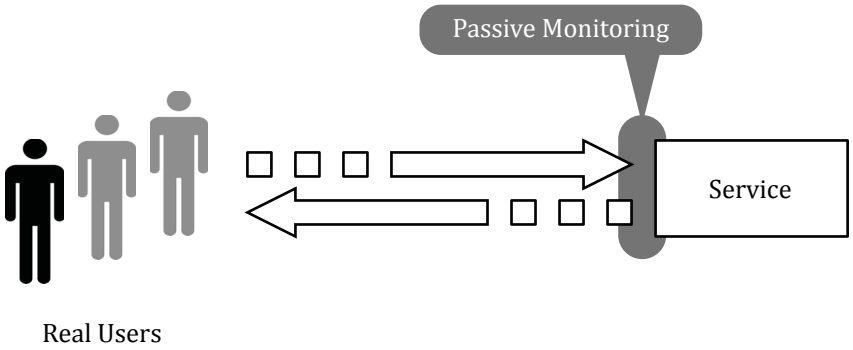



Figure 4.11: Monitoring Structure

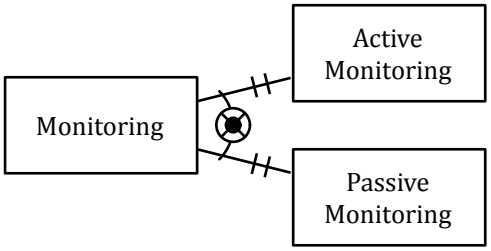
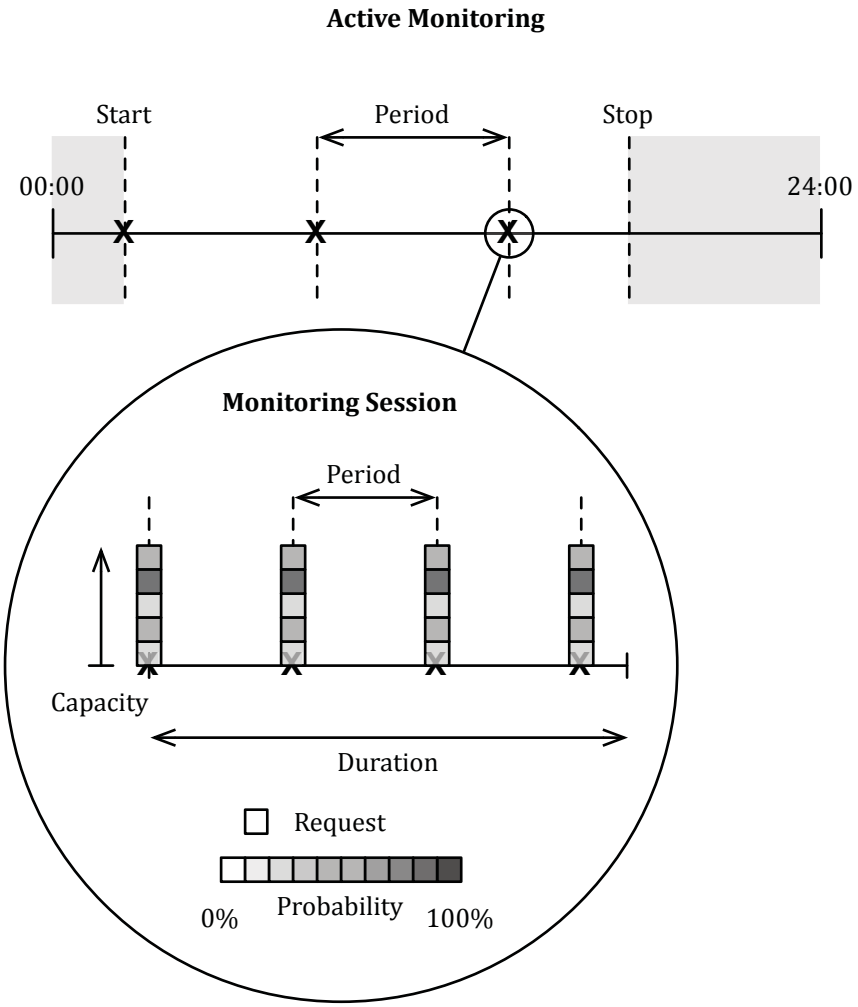


Figure 4.12: Active Monitoring Procedure




### 4.2.1 Active Monitoring

The `Active Monitoring` element of the abstract SLA model provides information about how to simulate real end-user traffic at the web service. It details the pre-defined monitoring requests and describes how often these pre-defined monitoring requests shall be send during agreement runtime to the service. Figure 4.12 shows a graphical representation of the active monitoring procedure. Therefore, the `Active Monitoring` element of the abstract SLA model shall contain the following elements (Figure 4.13):

**Start**

To avoid conflicts with real user traffic, to avoid traffic peaks at the server, or to limit the monitoring process to business hours, the `Active Monitoring` element offers the ability to limit the active monitoring process to a specific time range per day. The `Start` element defines the monitoring start time.

Figure 4.13: Active Monitoring Structure

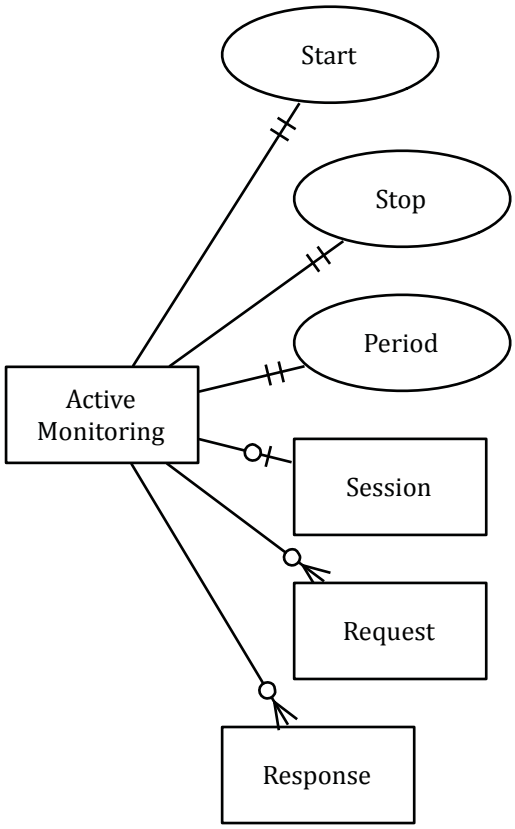
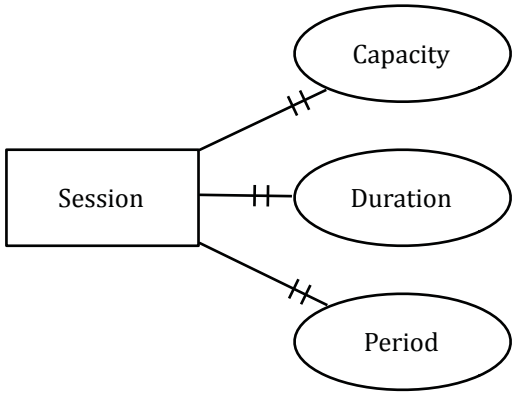


Figure 4.14: Active Monitoring Session Structure




**Stop**

The `Stop` element defines the monitoring end time. If the end time is before the start time, the defined time range covers a day changeover.

**Period**

The `Period` element defines the overall monitoring frequency. If the period value is for instance set to 360000 milliseconds, a "monitoring session" starts each 6 minutes within the defined time range.

**Session**

The "monitoring session" means the actual process of sending pre-defined requests to the web service in order to simulate end-user traffic and to measure relevant service properties. The `Session` element provides more detailed information about the monitoring session configuration.

**Request**

The active monitoring process implies the sending of one or more pre-defined requests to the web service. The `Request` element provides more detailed information about the pre-defined requests.

**Response**

In some cases it is important not only to measure runtime-related service properties but also to validate the service response. The `Response` element provides more detailed information about the validation process of the service response.

### MONITORING SESSION

The `Session` element of the abstract SLA model provides information about the monitoring session and shall contain the following elements (Figure 4.14):

Figure 4.15: Active Monitoring Request Structure

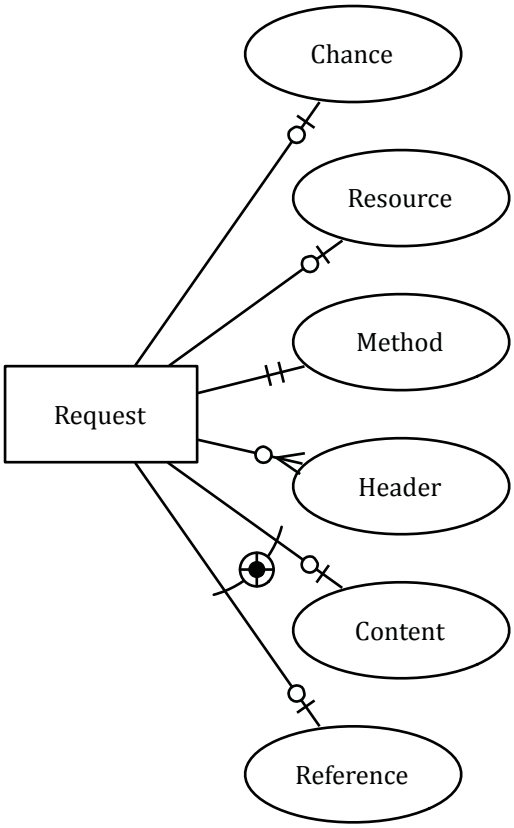
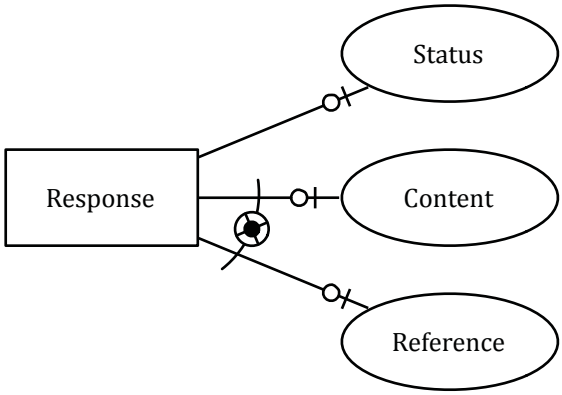



Figure 4.16: Active Monitoring Response Structure



#### Capacity

To simulate heavy traffic at the web service (e.g. many users or an large volume of input data), sometimes it is required to send more than one monitoring request at the same time to the web service. The `Capacity` element defines the number of parallel monitoring request.

#### Duration

The `Duration` element defines the overall duration of a single monitoring session. If the `Period` element of the enclosing `Active Monitoring` element is for instance set to $360000$ milliseconds and the `Duration` element is set to $60000$ milliseconds, a monitoring session that takes $1$ minute starts every $6$ minutes.

#### Period

The `Period` element defines the monitoring frequency within the monitoring session. If the `Duration` element is set to $60000$ milliseconds, the `Period` element is set to $1000$ milliseconds and the `Capacity` element is set to $20$, for $1$ minute (duration) every $1$ second (frequency) all in all $20$ requests (capacity) are send to the web service at the same time.

The `Session` element allows to simulate constant traffic at the web service for a specific period of time (e.g. $1200$ monitoring requests uniformly distributed over $1$ minute). If the `Session` element is not present, the monitoring session consists only of one single monitoring request that is send to the web service.

### MONITORING REQUEST

The `Request` elements of the abstract SLA model provide information about the predefined monitoring requests and shall contain the following elements (Figure 4.15):

#### Chance

The `Active Monitoring` element allows to define zero, or one, or many `Request` elements to define different monitoring requests. The `Chance` element defines the probability whether a specific request will be send to the service or not.

In the case of $n$ defined monitoring requests with the chance values $c_1 \dots c_n$, the probability $p_x$ (in %) that request $x \in \{1, \dots, n\}$ will be send to the service is set to

$$p_x = \frac{c_x \cdot 100\%}{\sum_{i=1}^{n} c_i} \tag{4.1}$$

#### Resource

The `Resource` element defines an additional URL path that identifies a specific resource at the server. The monitoring request will be send to the URL as specified in the *Service Reference* section of the abstract SLA model plus the URL path as defined in the `Resource` element.

#### Method

The `Method` element identifies the action to be performed on the web service. The HTTP standard [Fielding et al., 1999] defines several methods as for instance "GET" (requesting a representation of the specified resource), "POST" (submits data to specified resource) or "DELETE" (deletes the specified resource).

#### Header

The `Header` element defines HTTP header fields that shall be set when sending the request message to the server as for instance the response content type to be accepted ("Accept: text/xml").

#### Content

The `Content` element defines non-varying data that is included in the body of the HTTP request. The data to be send is directly included in the `Content` element.

#### Reference

The `Reference` element provides and URL pointing to a public accessible resource (e.g. a file on a web server) that defines the (potentially varying) data that is included in the body of the HTTP request.

The `Request` elements allow to specify zero, or one, or many HTTP requests that shall be used to measure corresponding service property. The structure of HTTP request messages and a core set of HTTP header fields is defined in [Fielding et al., 1999]. In the case that no `Request` element is present, by default a simple HTTP GET request is send to the URL as specified in the *Service Reference* section of the abstract SLA model.

### MONITORING RESPONSE

The `Response` element of the abstract SLA model provides information about the service response validation process and shall contain the following elements (Figure 4.16).

#### Status

The `Status` element defines which HTTP response status codes are accepted as valid. Typical codes are for instance 200 (standard response for successful HTTP requests) or 201 (standard response for a successful HTTP request and a new resource being created). If more than one HTTP response status code shall be accepted as valid, the different status codes must be provided by a comma-separated list.

#### Content

The `Content` element defines which HTTP response content is accepted as valid. The actual content to be compared with the service response is directly included in the `Content` element (e.g. an XML document).

**Reference**

The `Content` element provides and URL pointing to a public accessible resource (e.g. a file on a web server) that defines the (potentially varying) content to be compared with the service response.

If one of these rules is violated, the monitoring request is recorded as 'failed', which has the same effect as if the service is not available. The enclosing `Active Monitoring` element allows to define zero, or one, or many `Response` elements in order to allow different responses to be valid. If no `Response` element is present, the web service response is not evaluated. However, it is important to consider that not all services return correct HTTP response status codes and that the validation process fails even if the provided content differs from the service response just in one bit.

### MONITORING EXAMPLES

The following examples are provided to illustrate different setups for active monitoring.

#### Availability Monitoring

Listing 4.7 translates the normalized testing procedures for service availability assessment as mandated by INSPIRE to the structure of the `Active Monitoring` element. It is assumed that the assessment of service availability should be executed 24 hours a day (Line 11-12). According to INSPIRE, the service availability shall be measured consistently with minimum 10 reference requests per hour (Line 13). Such a reference request (`GetMap`) shall request images of 800x600 pixels and only one layer at a time (Line 14-18). Furthermore, the structure of the reference request is recommended to be based on varying BBOX parameters. The creation of dynamic and randomized service requests helps to reduce the impact of different caching strategies (e.g. cached tiles of a map service) on the performance measurements. To create dynamic and randomized service requests, the abstract SLA model offers a basic set of functions that can be accessed from within the `Content` element. An overview and explanation of all available functions can be found in Appendix B.1. According to the INSPIRE evaluation and assessment criteria, it is assumed that the service is available when it works properly and returns HTTP status code 200 (Line 21).

Listing 4.7: INSPIRE Availability Monitoring

```
1  "Service Property":
2  {
3    "Name": "availability",
4    "Title": "Service Availability",
5    "Abstract": "The general availability of the service.",
6    "Type": "...",
7    "Monitoring":
8    {
9      "ActiveMonitoring":
10     {
11       "Start": "00:00:00",
12       "Stop": "24:00:00",
13       "Period": 360000,
14       "Request":
15       {
16         "Method": "GET",
```

```
17          "Content": "service=WMS&version=1.3.0&request=GetMap&layers=topp:
                tasmania_state_boundaries&styles=&bbox=${__random(142.0,144.0)},
                ${__random(-46.0,-44.0)},${__random(150.0,152.0)},${__random
                (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&format=image/
                png"
18        },
19        "Response":
20        {
21          "Status": "200"
22        }
23      }
24    }
25  }
```

### Performance Monitoring

Listing 4.8 translates the normalized testing procedures for service performance assessment as mandated by INSPIRE to the proposed structure of the `Active Monitoring` element. The INSPIRE evaluation and assessment criteria regarding the performance measurement are identical to the availability assessment from the previous example. But the service property type does not reference the service availability but the service response time. Section 4.3.1 describes how to identify and reference different service property types in the `Service Property` element.

Listing 4.8: INSPIRE Performance Monitoring

```
1   "Service Property":
2   {
3     "Name": "response",
4     "Title": "Response Time",
5     "Abstract": "The response time of the service.",
6     "Type": "...",
7     "Monitoring":
8     {
9       "ActiveMonitoring":
10      {
11        "Start": "00:00:00",
12        "Stop": "24:00:00",
13        "Period": 360000,
14        "Request":
15        {
16          "Method": "GET",
17          "Content": "service=WMS&version=1.3.0&request=GetMap&layers=topp:
                tasmania_state_boundaries&styles=&bbox=${__random(142.0,144.0)},
                ${__random(-46.0,-44.0)},${__random(150.0,152.0)},${__random
                (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&format=image/
                png"
18        },
19        "Response":
20        {
21          "Status": "200"
22        }
23      }
24    }
25  }
```

### Capacity Monitoring

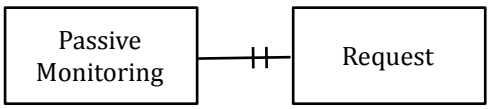
Listing 4.9 translates the normalized testing procedures for service capacity measurement as mandated by INSPIRE to the proposed structure of the

`Active Monitoring` element. The service capacity is defined as "the minimum number of served simultaneous service requests to a view service according to the performance quality (requirements)" [INSPIRE, 2011a] and therefore the service property type also references the service response time. According to INSPIRE, the service capacity shall be measured consistently based on sample reference request packages. The amount of request per package shall be 20 per second (Line 16) and shall be issued every second (Line 18) during a measurement time frame of 1 minute (Line 17). Furthermore, the sample reference request package shall be composed of 10% `GetCapabilities` requests (Line 20-25) and 90% `GetMap` requests (Line 26-31) with the same constraints as defined for the performance measurement. This active monitoring setup produces relevant traffic at the service. One monitoring session produces 1200 service requests uniformly distributed over 1 minute. According to INSPIRE, the service capacity assessment shall be performed "at regular intervals" which is not clearly specified. Therefore, to avoid conflicts with real user traffic it is assumed that the assessment of service capacity can be executed hourly after business hours from 20:00 to 04:00 every day (Line 11-12).

Listing 4.9: INSPIRE Capacity Monitoring

```
1  "Service Property":
2  {
3    "Name": "capacity",
4    "Title": "Service Capacity",
5    "Abstract": "The response time of the service for multiple parallel
         requests.",
6    "Type": "...",
7    "Monitoring":
8    {
9      "ActiveMonitoring":
10     {
11       "Start": "20:00:00",
12       "Stop": "04:00:00",
13       "Period": 3600000,
14       "Session":
15       {
16         "Capacity": 20,
17         "Duration": 60000,
18         "Period:": 1000
19       },
20       "Request":
21       {
22         "Chance": 10,
23         "Method": "GET",
24         "Content": "service=WMS&version=1.3.0&request=GetCapabilities"
25       },
26       "Request":
27       {
28         "Chance": 90,
29         "Method": "GET",
30         "Content": "service=WMS&version=1.3.0&request=GetMap&layers=topp:
               tasmania_state_boundaries&styles=&bbox=${__random(142.0,144.0)},
               ${__random(-46.0,-44.0)},${__random(150.0,152.0)},${__random
               (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&format=image/
               png"
31       },
32       "Response":
33       {
34         "Status": "200"
35       }
36     }
37   }
38 }
```

Figure 4.17: Passive Monitoring Structure




The previous examples show that it is possible to monitor one service property (e.g. the service response time) with different monitoring setups. Therefore, it is possible to define different service level objectives that are linked to the same service property (e.g. the INSPIRE performance and the INSPIRE capacity requirements). Section 4.3.2 describes how to define concrete service level objectives in the abstract SLA model. The example agreement that can be found in Appendix B.4 provides more service properties that illustrate different setups for active monitoring.

### 4.2.2 Passive Monitoring

The `Passive Monitoring` element of the abstract SLA model provides information about how to access the captured information about the real end-users traffic at the web service. But the available information and the format of these recordings always depend on implementation-specific web service hosting environments and deployed enterprise monitoring tools. Therefore, the abstract SLA model could not define a fixed and detailed way to describe the access to these type of information.

For the purpose of this thesis it is assumed that the original service is accessed by real end-users through an HTTP proxy [Fielding et al., 1999]. Such a proxy can intercept all the communication between client and server. Therefore, it is possible to protocol common resource-, runtime- and usage-related service properties for passive monitoring. Section 5.2.1 describes how the assumed HTTP proxy works and whether a “transparent proxy” or a “non-transparent proxy” is required in the web-based SLA management architecture. Therefore, the `Passive Monitoring` element of the abstract SLA model shall contain the following elements (Figure 4.17):

**Request**

> The `Request` element of the `Passive Monitoring` element has the same structure and content as the `Request` element of the `Active Monitoring` element. It provides more detailed information about how and where to receive the recorded information.

The following example is provided to illustrate the access to the recorded number of delivered pixel. Section 5.2.2 describes the passive monitoring workflow in the web-based SLA management architecture. Section 5.3.2 describes the service interface of the HTTP proxy and the encoding of requested information. The example agreement that can be found in Appendix B.4 provides more service properties that illustrate different setups for passive monitoring.

Listing 4.10: Pixel Delivery Logging

```
"Service Property":
{
  "Name": "pixel",
  "Title": "Sum of Pixels",
  "Abstract": "The accessed number of pixels.",
  "Type": "...",
  "Monitoring":
  {
    "PassiveMonitoring":
    {
      "Request":
      {
        "GET":
        {
          "Resource": "/usage/pixel"
          "Method": "GET"
        }
      }
    }
  }
}
```

## 4.3 Agreement Evaluation

This section describes how to evaluate the status of an agreement during agreement runtime. Section 4.3.1 describes an OGC URN Schema Extension that can be used to identify service property types and business value types. Section 4.3.2 defines a DSL for evaluating service level objectives and calculating business values.

### 4.3.1 OGC URN Schema Extension

An URN is a Uniform Resource Identifier (URI) that uses the URI scheme defined in [Moats, 1997]. The syntax of an URN can be described in Backus Naur Form (BNF) [Backus et al., 1963] as follows.

```
<URN> ::= "urn:" <NID> ":" <NSS>
```

The <NID> token is the namespace identifier and the <NSS> token is a namespace specific string. The OGC has an officially registered URN namespace [Reed, 2002] that is defined in [Reed, 2008]. The OGC namespace specifies "a family of identifiers for resources of interest in the context of OGC Web Services, mostly concerning resources provided or defined by OGC" [Whiteside, 2006]. The namespace identifier is "ogc" and the namespace specific string has the following form.

```
<URN> ::= "urn:ogc:" <OGCresource> ":" <ResourceSpecificString>
```

The <OGCresource> token defines one resource category out of the following top-level categories. The "specification" category identifies all published OGC specifications and related supporting schemas. The "doc" category identifies OGC documents and elements within OGC documents. The "service" category identifies access to an OWS. The "tc" category identifies work products of the various OGC working groups

and committees. The "def" category references "definitions of coordinate reference systems, coordinate (transformation) operations, and components thereof, that are specified or recognized by the OGC in a formal OGC document" [Reed, 2004]. The structure and the content of <ResourceSpecificString> token is conform to the syntax requirements of [Moats, 1997] and depends on the resource category. The rules for constructing a <ResourceSpecificString> token for the top-level categories are defined in [Cox, 2010b], [Cox, 2009] and [Cox, 2010a]. The procedures used by the OGC Naming Authority for the assignment and registration of new <OGCresource> token names and the specific registration values assigned for each resource category are described in [Cox, 2010c].

The following examples are provided to illustrate the different resource categories and corresponding resource specific strings.

**Example**

This URN identifies the Geography Markup Language (GML) Encoding Specification Version 3.00.

```
urn:ogc:specification:gml:doc-is(02-023r4):3.00
```

**Example**

This URN identifies the WMS Implementation Specification Version 1.3 document.

```
urn:ogc:doc:IS:WMS:1.3
```

**Example**

This URN specifies the service type for an OGC Catalogue service.

```
urn:ogc:service:CatalogueService:2.0:HTTP
```

**Example**

This URN identifies the minutes of the plenary session of the 50th OGC Technical Committee Meeting in Southampton, UK.

```
urn:ogc:tc:plenary:doc-minutes:20040620
```

**Example**

This URN references the definition of the Coordinate Reference System (CRS) with the code 26986 that is specified in Version 6.3 of the European Petroleum Survey Group Geodesy (EPSG) database.

```
urn:ogc:def:crs:EPSG:6.3:26986
```

The use of the "def" category "may be expanded in the future to accommodate the needs of new OGC standards" [Reed, 2004]. Therefore, the following sections define an OGC URN Schema Extension in order to identify service property types and business value types in the abstract SLA model.

### SERVICE PROPERTIES

The *Service Properties* section of the abstract SLA model contains a list of zero, or one, or many `Service Property` elements. Each `Service Property` element contains an unique identifier of the service property type that shall be monitored by the web-based SLA management architecture during agreement runtime.

The syntax of the OGC URN Schema Extension can be described in BNF as follows.

```
<URN> ::= "urn:ogc:def:sla:" <Section> ":" <Category> ":" <Indicator> [":" <
Parameter>]
```

The `<Section>` token defines whether the URN is about identifying a service property or a business value types. The "property" section identifies service property types. The structure of the `<Category>` token, the `<Indicator>` token and the optional `<Parameter>` token are conform to the syntax requirements of [Moats, 1997]. Their content depends on the specific service property category. The following paragraphs provide an overview of selected service property categories and corresponding service property types.

#### Resource-Related Properties

The web-based SLA management architecture should be able to check whether a service really delivers promised resources. Depending on the service type, such a resource can be for instance an optional service operation, an available layer, an available featureset or an offered geospatial process. To enable service providers and service consumer to agree on delivered resources, the abstract SLA model defines the following resource-related service property types.

If the `<Category>` token is "resource", the `<Indicator>` token must be "operation" or "layer". The "operation" indicator represents information about about the (optional) operations implemented by the service instance. The "layer" indicator represents information about the maps offered by the service instance. In all cases the `<Parameter>` token remains empty.

The following examples illustrate the definition of URNs for identifying resource-related service properties.

**Example**

This URN identifies the operations implemented by the service instance.

```
urn:ogc:def:sla:property:resource:operation
```

**Example**

This URN identifies the maps offered by the service instance..

```
urn:ogc:def:sla:property:resource:layer
```

### Runtime-Related Properties

To give service providers and service consumer the ability to negotiate general runtime-related service quality levels, the abstract SLA model must allow the definition of the following common QoS attributes for web services.

If the <Category> token is "runtime", the <Indicator> token must be one of "availability" or "response". The "availability" indicator represents information about the probability whether a web service is up and running over a specific period of time. The "response" indicator represents information about the time required to complete a service request. In all cases the <Parameter> token remains empty.

The following examples illustrate the definition of URNs for identifying runtime-related service properties.

**Example**

This URN identifies the probability whether a web service is up and running.

```
urn:ogc:def:sla:property:runtime:availability
```

**Example**

This URN identifies the initial response time of a service.

```
urn:ogc:def:sla:property:runtime:response
```

### Usage-Related Properties

Monitoring the service consumer behaviour is important for realizing pay-as-you-go pricing models (e.g. service usage costs depending on the monthly service requests) and for defining preconditions under which a service quality guarantee persists (e.g. a maximum response time until the number of hourly service requests does not exceed a specific threshold).

If the <Category> token is "usage", the <Indicator> token must be one of "request", "operation" or "pixel". The "request" indicator represents information about the accumulated number of service requests over a specific period of time. The "operation" indicator represents information about the accumulated number of service operation calls over a specific period of time . The "pixel" indicator represents information about the accumulated number of delivered pixels over a specific period of time. In all cases the <Parameter> token remains empty.

The following examples illustrate the definition of URNs for identifying usage-related service properties.

**Example**

This URN identifies the accumulated number of service requests.

```
urn:ogc:def:sla:property:usage:request
```

**Example**

This URN identifies the accumulated number of service operation calls.

`urn:ogc:def:sla:property:usage:operation`

**Example**

This URN identifies the accumulated number of delivered pixels.

`urn:ogc:def:sla:property:usage:pixel`

### Data-Related Properties

In some application domains it is important for service consumers to be aware of the delivered data quality and how, where, when and why content was produced (data provenance) [Groth, 2012]. In the context of GI science, the quality aspect of geospatial data has been widely examined, mostly with a focus on data accuracy and its fitness-for-use in an application context [Subbiah et al., 2007]. Further common quality aspects are for instance the resolution, the scale and the completeness of geospatial data [van Oort, 2005]. The ability to share, enrich and combine geospatial data is one important key to the success of SDIs and therefore the abstract SLA model shall enable service providers to define the limitations for the application of delivered data. Furthermore, the authors of [Van Loenen et al., 2012] compare existing licensing frameworks for geospatial data with a focus on the Creative Commons Framework[1], which had a "great influence on any licensing models for geographic data and is often used as a basis for harmonising initiatives".

If the `<Category>` token is "data", the `<Indicator>` token must be one of "accuracy", "completeness" or "license". The "accuracy" indicator represents information about the rate of excess or missing items in the delivered data. In that case, the `<Parameter>` token must be one of "commission" or "omission". The "completeness" indicator represents information about absolute or external accuracy of delivered data. In that case, the `<Parameter>` token must be one of "absolute", "external" or "resolution". The "license" indicator represents information about the conditions for sharing and modifying delivered data. In that case, the `<Parameter>` token remains empty.

The following examples illustrate the definition of URNs for identifying data-related service properties.

**Example**

This URN identifies the spatial resolution of the underlying data source.

`urn:ogc:def:sla:property:data:accuracy:resolution`

**Example**

This URN identifies the license for data sharing and reuse in another application context.

`urn:ogc:def:sla:property:data:license`

[1] http://creativecommons.org

### Security-Related Properties

For many use cases it is important to implement rights management mechanisms in order to implement access control to geospatial resources.

If the <Category> token is "security", the <Indicator> token must be "license". The "license" indicator represents access rights policies in order to enforce the access restrictions on geographic information. In that case, the <Parameter> token must be "geoxacml" or any other supported license format.

The following example illustrates the definition of URNs for identifying security-related service properties.

**Example**

This URN identifies access rights policies in GeoXACML format.

`urn:ogc:def:sla:property:security:license:geoxacml`

### Infrastructure-Related Properties

The importance of making compute resources available in a pay-as-you-go manner [Armbrust et al., 2009], and the importance of the ability to allocate compute resource on a short-term basis as needed and release them as needed [Armbrust et al., 2010] becomes evident in the Cloud Computing paradigm [Buyya et al., 2009]. To allow service providers to offer different hosting opportunities with particular infrastructure and to allow service consumers to define their compute resources requirements, the abstract SLA model defines the following infrastructure-related service property types.

If the <Category> token is "infrastructure", the <Indicator> token must be one of "provider", "vm" or "compute". The "provider" indicator represents information about the actual infrastructure provider. In that case, the <Parameter> token must be one of "name" or "region". The "name" parameter defines which infrastructure provider is responsible for hosting the service. The "region" parameter represents information about the region for the service to use. The "vm" indicator represents information about a virtualized compute resource. In that case, the <Parameter> token must be one of "name" or "persistence". The "name" parameter defines which Virtual Machine (VM) should represent the compute resource and the "persistence" parameter defines whether a VM is persistent or not. Finally, the "compute" indicator represents information about the hardware resources of a compute resource. In that case, the <Parameter> token must be one of "architecture", "cores", "speed" or "memory". The "architecture" parameter defines the CPU architecture of a compute resource. The "core" parameter defines the numbers of available CPU cores at a compute resource and the "speed" parameter defines the CPU clock speed of each available CPU core. Finally, the "memory" parameter defines the available Random Access Memory (RAM) of the compute resource.

The following examples illustrate the definition of URNs for identifying infrastructure-related service properties.

**Example**

This URN identifies an infrastructure provider.

`urn:ogc:def:sla:property:infrastructure:provider:name`

**Example**

This URN identifies a VM from a repository of registered VMs.

`urn:ogc:def:sla:property:infrastructure:vm:name`

**Example**

This URN identifies defines the available RAM of a compute resource.

`urn:ogc:def:sla:property:infrastructure:compute:memory`

The presented URN examples are an excerpt from all possible URNs. Appendix B.2.1 provides a comprehensive dictionary of all URNs that can be used in the abstract SLA model to identify most relevant service property types. Furthermore, the presented `<Category>` and `<Indicator>` tokens are not complete. Other categories of service property types such as data-, transaction-, security- and business-related service properties are described only partially or completely missing. The assessment of spatial data accuracy and spatial data completeness [Joksić Dušan, 2004] for instance includes automated and especially non-automated on-screen validation procedures [Victorian Spatial Council, 2009]. Even if there are legal requirements regarding the quality of spatial data as for instance described in the INSPIRE Data Specification on Protected Sites [INSPIRE, 2010c], these requirements neither specify test nor reference data sets [van Oort, 2005] for an automatic testing procedure. Therefore, these categories are currently not covered completely by the abstract SLA model and the web-based SLA management architecture. To accommodate the needs of specific applications and use cases, by introducing new categories (`<Category>`) and corresponding indicators (`<Indicator>`) the dictionary can be extended with a more fine-grained view on web service quality, data quality, infrastructure management.

### BUSINESS VALUES

The *Business Values* section of the abstract SLA model contains a list of zero, or one, or many `Business Value` elements. Each `Business Value` element contains an unique identifier of the business value type that should be calculated by the web-based SLA management architecture at specific time points during agreement runtime.

The syntax of the OGC URN Schema Extension can be described in BNF as follows.

```
<URN> ::= "urn:ogc:def:sla:" <Section> ":" <Category> ":" <Interval>
```

The `<Section>` token defines whether the URN is about identifying a service property or a business value type. The "business" section identifies business value types. The structure of the `<Category>` token and the `<Interval>` token are conform to the syntax requirements of [Moats, 1997]. Their content depends on the specific business value category.

For business contracts it is important to be able to define (dynamic) service usage costs, to describe penalties for not meeting a service level objective, or to offer a reward for meeting a service level objective. If the <Category> token is "cost", the <Interval> token defines the assessment interval and must be one of "day", "week", "month" or "business". If the <Category> token is "penalty", the <Interval> token defines the assessment interval and must be one of "day", "week", "month" or "business". If the <Category> token is "reward", the <Interval> token defines the assessment interval and must be one of "day", "week", "month" or "business".

The following example illustrates the definition of URNs for identifying business values.

**Example**

This URN identifies monthly service usage costs.

```
urn:ogc:def:sla:business:cost:month
```

This example illustrates the definition of URNs for identifying a business value. In contrast to the service properties, it is not allowed to define multiple business values with the same type. Appendix B.2.2 provides a comprehensive dictionary of all URNs that can be used in the abstract SLA model to identify the business value types that shall be calculated by the web-based SLA management architecture at specific time points during agreement runtime. To accommodate the needs of specific applications and use cases, by introducing new categories (<Category>) and corresponding indicators (<Indicator>) the dictionary can be extended with a more fine-grained view on business processes.

### 4.3.2 Agreement Expression Language

The authors of [van Deursen et al., 2000] propose the following DSL definition.

> *A domain-specific language (DSL) is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain.*

The reason for developing a DSL instead of using a general-purpose programming language is that a DSL can offer more appropriate or established domain-specific notations, constructs and abstractions. [Mernik et al., 2005]. Due to the fact that a DSL is tailored to a particular application domain and typically does not offer many features that can be found in general-purpose programming languages, the DSL remains small and simple. Therefore, domain experts themselves normally can "understand, validate, modify, and often even develop DSL programs" [Deursen et al., 2000] in order to express the solution to a domain-specific problem in a more natural way and from a real world point of view [Hudak, 1998].

This section develops a DSL for describing service level objectives and business values in the abstract SLA model. The development of such an *Agreement Expression Language* must consider the following requirements:

- Depending on the specific `Service Property` elements that are defined in the abstract SLA model, the DSL must provide access to all gathered information related to these service properties (e.g. the measured response time of a service gathered by active monitoring).
- The DSL shall allow the definition of service level objectives reflecting complex functional as well as non-functional requirements (e.g. guaranteed service response time for a give number of service requests per month).
- The DSL shall allow the definition of pricing models reflecting different business use cases (e.g. fixed or usage-based cost).

The proposed DSL for the abstract SLA model is based on Java EXpression Language (JEXL)[2], an expression language inspired by Apache Velocity[3] and the Unified Expression Language [Lubke et al., 2005] defined in JavaServer Pages (JSP) Version 2.1 [SUN, 2006] and the JavaServer Pages Standard Tag Library (JSTL) Version 1.2 [SUN, 2012]. The JEXL syntax supports basic language elements such as comments, variables, method calls and basic literals such as numbers, strings, arrays and access to maps. The JEXL grammar allows the definition of mathematical and boolean operators to control the program workflow via conditional statements. Whenever a JEXL script made up of one ore more statements is interpreted, the last evaluated expression is returned. When evaluating simple expressions or complex scripts, JEXL offers a context that contains a set of use case specific variables. These variables internally are representatives of concrete Java objects and JEXL allows to call any method on a Java object through these variables using the same syntax. Furthermore, JEXL allows to access methods that follow the JavaBean [Hamilton, 1997] naming convention for properties, i.e. setters and getters. However, more detailed information about the JEXL syntax and some JEXL examples can be found at the JEXL project homepage.

The following sections describe selected variables and methods that are available in the JEXL context of the `Service Level Objective` and `Business Value` elements. Appendix B.3 provides a complete overview about all variables and methods that can be accessed in the abstract SLA model. However, the variables and methods in this JEXL context define a DSL that is designed to evaluate the status of service level objectives (either 'fulfilled' or 'violated') and to calculate business related values (e.g. monthly service usage costs).

#### SERVICE LEVEL OBJECTIVES

The `Service Property` elements in the abstract SLA model either provide useful management information for the web-based SLA management architecture or define what specific service properties shall be monitored during agreement runtime. Based on acquired monitoring information, the status of all `Service Level Objective` elements in the abstract SLA model is evaluated on a regular basis during agreement runtime. The previous section defines several service property categories and corresponding service property types that cause monitoring processes during agreement runtime. Each `Service Property` element in abstract SLA model, that is from one of these categories,

[2] http://commons.apache.org/jexl
[3] http://velocity.apache.org

results in a corresponding JEXL context variable that provides access to the gathered monitoring information. The name of the JEXL context variable is identical with the name of the `Service Property` element and the methods of the JEXL context variable depend on the type of the `Service Property` element.

The previous sections exemplified for instance the following runtime-related service properties (Figure 4.11):

Listing 4.11: Runtime-Related Service Properties

```
"Service Properties":
{
  (...),
  "Service Property":
  {
    "Name": "availability",
    (...)
    "Type": "urn:ogc:def:sla:property:runtime:availability",
    (...)
  },
  "Service Property":
  {
    "Name": "response",
    (...)
    "Type": "urn:ogc:def:sla:property:runtime:response",
    (...)
    }
  },
  "Service Property":
  {
    "Name": "capacity",
    (...)
    "Type": "urn:ogc:def:sla:property:runtime:response",
    (...)
  },
  (...)
}
```

The first service property identifies the service availability. The second and the third service property identifies the service response time, but with different monitoring configurations (Listing 4.7 - 4.9).

All service properties in the *Service Properties* section of the abstract SLA model with type

`urn:ogc:def:sla:property:runtime:availability`

result in a JEXL context variable from type `AvailabilityType` that has the following variables and methods (Table 4.2).

Table 4.2: JEXL `AvailabilityType`

| `AvailabilityType` [a] | |
|---|---|
| `float` | `day`<br>Returns the measured service availability for the current day in percent (where 0.0 means 0% and 1.0 means 100%). |
| `float` | `week`<br>Returns the measured service availability for the current calendar week in percent (where 0.0 means 0% and 1.0 means 100%). |
| `float` | `month`<br>Returns the measured service availability for the current calendar month in percent (where 0.0 means 0% and 1.0 means 100%). |
| `float` | `year`<br>Returns the measured service availability for the current calendar year in percent (where 0.0 means 0% and 1.0 means 100%). |

[a] This table shows selected variables and methods of the `AvailabilityType`. A complete overview of all variables and methods can be found in Appendix B.3.

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:runtime:response
```

result in a JEXL context variable from type `ResponseType` that has the following variables and methods (Table 4.3 - 4.5).

Table 4.3: JEXL `ResponseType`

| `ResponseType` [a] | |
|---|---|
| `InitialResponseType` | `initial`<br>Returns an object of type `InitialResponseType` that provides information about the measured initial response time of the service. |
| `TotalResponseType` | `total`<br>Returns an object of type `TotalResponseType` that provides information about the measured total response time of the service. |

[a] This table shows selected variables and methods of the `ResponseType`. A complete overview of all variables and methods can be found in Appendix B.3.

Table 4.4: JEXL `InitialResponseType`

| `InitialResponseType` [a] | |
|---|---|
| `int[]` | `day`<br>Returns an array of all response time measurements for the current day. Each element in the array represents the measured initial response time of a successful request in milliseconds. |
| `int[]` | `week`<br>Returns an array of all response time measurements for the current calendar week. Each element in the array represents the measured initial response time of a successful request in milliseconds. |
| `int[]` | `month`<br>Returns an array of all response time measurements for the current calendar month. Each element in the array represents the measured initial response time of a successful request in milliseconds. |
| `int[]` | `year`<br>Returns an array of all response time measurements for the current calendar year. Each element in the array represents the measured initial response time of a successful request in milliseconds. |

[a] This table shows selected variables and methods of the `InitialResponseType`. A complete overview of all variables and methods can be found in Appendix B.3.

Table 4.5: JEXL `TotalResponseType`

| `TotalResponseType` [a] | |
|---|---|
| `int[]` | `day`<br>Returns an array of all response time measurements for the current day. Each element in the array represents the measured total response time of a successful request in milliseconds. |
| `int[]` | `week`<br>Returns an array of all response time measurements for the current calendar week. Each element in the array represents the measured total response time of a successful request in milliseconds. |

Table 4.5 – Continued on next page

Table 4.5 – Continued from previous page

| `TotalResponseType` [a] | |
|---|---|
| `int[]` | `month`<br>Returns an array of all response time measurements for the current calendar month. Each element in the array represents the measured total response time of a successful request in milliseconds. |
| `int[]` | `year`<br>Returns an array of all response time measurements for the current calendar year. Each element in the array represents the measured total response time of a successful request in milliseconds. |

[a] This table shows selected variables and methods of the `TotalResponseType`. A complete overview of all variables and methods can be found in Appendix B.3.

The following examples are provided to illustrate the utilization of JEXL context variables from type `AvailabilityType` and `ResponseType` for evaluating service level objectives that reflect INSPIRE service quality requirements.

### Availability Evaluation

Listing 4.12 translates the INSPIRE service availability requirements into the Agreement Expression Language. Based on the availability measurements, the example checks whether the probability of the service to be available is greater or equal $99\%$ per week, month and year. The measured availability can be accessed through the "availability" context variable, which was injected by the `Service Property` element with the same name. The result of the boolean function (either `true` or `false`) defines the current status of the service level objective (either 'fulfilled' or 'violated').

Listing 4.12: INSPIRE Availability Evaluation

```
1  "Service Level Objective":
2  {
3    "Name": "InspireAvailability"
4    "Title": "INSPIRE (Availability)",
5    "Abstract": "The probability of a Network Service to be available shall be
         99% of the time.",
6    "Obligated": "Service Provider",
7    "Status": "
8      (availability.week >= 0.99) and (availability.month >= 0.99) and (
           availability.year >= 0.99)
9    "
10 }
```

### Performance Evaluation

Listing 4.13 translates the INSPIRE service performance requirements into the Agreement Expression Language. Based on the actual response time measurements, the example script checks for each measurement whether the time between sending the

service request and receiving the service response is smaller or equal 5000 milliseconds. If 90% ore more measurements are below that limit, the service level objective is fulfilled (the script returns `true`). Otherwise, the service level objective is violated (the script returns `false`).

Listing 4.13: INSPIRE Performance Evaluation

```
"Service Level Objective":
{
  "Name": "InspirePerformance"
  "Title": "INSPIRE (Performance)",
  "Abstract": "The response time for sending the initial response to a Get
      Map Request to a view service shall be maximum 5 seconds in normal
      situation.",
  "Obligated": "Service Provider",
  "Status": "
    fulfilled = 0;
    for (item : response.initial.week) {
      if (item lt 5000)
      {
        fulfilled = fulfilled + 1;
      }
    }
    percent = fulfilled / (size(response.initial.week) / 100.0);
    percent gt 90.0;
  "
}
```

### Capacity Evaluation

Listing 4.14 translates the INSPIRE service capacity requirements into the Agreement Expression Language. The script is nearly identical to the previous script, but the JEXL context variable for accessing the performance measurements is different. The example still checks for each measurement whether the initial service response time is smaller or equal 5000 milliseconds and if 90% ore more measurements are below that limit, but it accesses the actual measurements through the 'capacity' variable (coming from the 'capacity' service property) instead of the 'response' variable (coming from the 'response' service property). The variable 'response' (Listing 4.8) references the service response time of a single `GetMap` request. The variable 'capacity' (Listing 4.9) references the measured service response time of a package of 20 parallel requests, composed of 10% `GetCapabilities` and 90% `GetMap` requests.

Listing 4.14: INSPIRE Capacity Evaluation

```
"Service Level Objective":
{
  "Name": "InspireCapacity"
  "Title": "INSPIRE (Capacity)",
  "Abstract": "The minimum number of served simultaneous service requests to
       a view service according to the performance quality of service shall
      be 20 per second.",
  "Obligated": "Service Provider",
  "Status": "
    fulfilled = 0;
    for (item : capacity.initial.week) {
      if (item lt 5000)
      {
        fulfilled = fulfilled + 1;
      }
```

```
14      }
15      percent = fulfilled / (size(capacity.initial.week) / 100.0);
16      percent gt 90.0;
17    "
18  }
```

The last two examples showed successfully how to define service level objectives that reference to the same service property with different monitoring setups. Therefore, it is possible to create an agreement that contains different service level objectives based one single service property.

A more detailed overview about all service property categories and corresponding JEXL context variables can be found in Appendix B.3.1. The example agreement that can be found in Appendix B.4 provides more examples that illustrate the usage of JEXL context variables for evaluating the status of service level objectives.

#### BUSINESS VALUES

The `Business Value` elements in the abstract SLA model define custom business values that shall be calculated at specific time points during agreement runtime. The `Type` element of a `Business Value` element defines what kind of business value shall be calculated (e.g. monthly usage costs) and the `Value` element defines a concrete pricing model in the proposed Agreement Expression Language.

The previous sections exemplified for instance the following usage-related service property (Figure 4.15).

Listing 4.15: Usage-Related Service Property

```
1  "Service Properties":
2  {
3    (...)
4    "Service Property":
5    {
6      "Name": "pixel",
7      (...)
8      "Type": "urn:ogc:def:sla:property:usage:pixel",
9      (...)
10   }
11   (...)
12 }
```

The service property identifies the accumulated amount of pixels delivered by a service with a specific monitoring configuration (Listing 4.10).

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:usage:pixel
```

result in a JEXL context variable from type `PixelType` that has the following variables and methods (Table 4.6).

Table 4.6: JEXL `PixelType`

| `PixelType` [a] | |
|---|---|
| `long` | `day`<br>Returns the accumulated amount of delivered pixel of all layers and for the current day. |
| `long` | `week`<br>Returns the accumulated amount of delivered pixel of all layers and for the current calendar week. |
| `long` | `month`<br>Returns the accumulated amount of delivered pixel of all layers and for the current calendar month. |
| `long` | `year`<br>Returns the accumulated amount of delivered pixel of all layers and for the current calendar year. |

[a] This table shows selected variables and methods of the `PixelType`. A complete overview of all variables and methods can be found in Appendix B.3.

The following example is provided to illustrate the utilization of JEXL context variables from type `PixelType` for realizing dynamic and usage-based pricing models as for instance mandated by the AdV. The AdV for instance defines three different pricing models for the online delivery of "Geobasisdaten" (e.g. topographical data). The pay-as-you-go pricing model defines yearly usage costs based on the delivered pixels. Each delivered 1 million pixel (MPx) costs 1 Euro. Furthermore, the pay-as-you-go pricing model defines different discount levels that depend on the total amount of delivered pixels. If the total amount of delivered pixels per year is in the range from 1.000 to 10.000 MPx, the discount factor is for instance 50%.

Listing 4.16: Yearly Usage Costs

```
1  "Business Value":
2  {
3    "Name": "CostsPerYear",
4    "Title": "Usage Costs (Year)",
5    "Abstract": "The cost to be assessed for using the service on a yearly
         basis (in Euro).",
6    "Obligated": "Service Consumer",
7    "Type": "urn:ogc:def:sla:business:cost:year",
8    "Value": "
9      factor;
10     if (pixel.year lt (1000000 * 1000))
11     {
12       factor = 1.0;
13     } else
14     if (pixel.year lt (1000000 * 10000))
15     {
16       factor = 0.5;
17     } else
18     if (pixel.year lt (1000000 * 100000))
19     {
20     factor = 0.25;
21     } else
```

```
22      if (pixel.year lt (1000000 * 1000000))
23      {
24        factor = 0.125;
25      } else
26      {
27        factor = 0.0625;
28      }
29      (factor * (pixel.year / 1000000));
30    "
31  }
```

To define penalties (rewards) for violating (fulfilling) service level objectives, each `Service Level Objective` element in the abstract SLA model results in a corresponding JEXL context variable from type `ObjectiveType` that provides access to the status of the corresponding service level objective. The name of the JEXL context variable is identical with the name of the `Service Level Objective` element and an overview about all methods of JEXL context variables from type `ObjectiveType` can be found in Table B.27.

Table 4.7: JEXL `ObjectiveType`

| `ObjectiveType` [a] | |
|---|---|
| `Boolean` | **`status`**<br>Returns `true` if the corresponding service level objective is fulfilled and `false` if the corresponding service level objective is violated. |

[a] This table shows selected variables and methods of the `ObjectiveType`. A complete overview of all variables and methods can be found in Appendix B.3.

To calculate discounts (extra charges) proportionately to actual usage costs, each `Business Value` element in the abstract SLA model results in a corresponding JEXL context variable from type `BusinessType` that provides access to a previously calculated custom business value. The name of the JEXL context variable is identical with the name of the `Business Value` element and an overview about all methods of such JEXL context variables from type `BusinessType` can be found in Table B.28.

Table 4.8: JEXL `BusinessType`

| `BusinessType` [a] | |
|---|---|
| `float` | **`value`**<br>Returns the value of the corresponding business value (normally the rate in Euro). |

[a] This table shows selected variables and methods of the `BusinessType`. A complete overview of all variables and methods can be found in Appendix B.3.

The following example is provided to illustrate the definition of penalties (discounts) in the case that one or more service level objectives are violated. The 'InspireAvailability' variable is from type `ObjectiveType` and provides access to the status of the service level

objective that reflects the INSPIRE service availability requirements. The 'CostsPerYear' variable is from type `BusinessType` and provides access to the business value that reflects the AdV pricing model. The aimed penalty reflects a 20% discount on the yearly service usage costs when the INSPIRE service availability requirement is not fulfilled.

Listing 4.17: Yearly Penalty

```
"Business Value":
{
  "Name": "PenaltyPerYear",
  "Title": "Penalty (Year)",
  "Abstract": "The penalty to be assessed for not meeting service level
      objectives on a yearly basis (in Euro).",
  "Obligated": "Service Provider",
  "Type": "urn:ogc:def:sla:business:penalty:year",
  "Value": "
    if (InspireAvailability.status == true)
    {
      factor = 0;
    }
    else
    {
      factor = 0.25;
    }
    (factor * CostsPerYear.value);
  "
}
```

The example agreement in Appendix B.4 provides more examples that illustrate the usage of JEXL context variables for calculating custom business values. The example pricing models do not cover any value added tax calculations as for instance required by local laws and regulations. They are simplified versions of real-world pricing models as for instance mandated by the AdV and the VermWertGebT.

However, the examples (Listing 4.12 - Listing 4.16) show that the JEXL context variables hide information about the actual process of measurement. The measurements provided by the 'response' variable are gathered with active monitoring, whereas the measurements provided by the 'pixel' variable are recorded service consumer behavior. That abstract representation of service level measurements prevents service level objectives to be too complex and to care about implementation-specific details.

## 4.4 Summary

This chapter formalizes an abstract SLA model that is applicable in SDIs. The aim of the abstract SLA model is to provide an abstract representation of agreements on a conceptual level that is independent of any specific data encoding format. The abstract SLA model is composed of a description of the structure and the content of the domain-specific agreements, an OGC URN Schema Extension for identifying domain-specific service property types and business value types, and a DSL for describing service level objectives and business values.

The abstract SLA model contains general information about the agreement and the contracting parties (*Agreement Context*), domain-specific information about the services to which an agreement is related (*Service Description*), a domain-specific reference to

an actual service (*Service Reference*), and a set of domain-specific service properties that are used to measure the service quality and that should be monitored during agreement runtime (*Service Properties*). Furthermore, the abstract SLA model specifies service quality goals that the contracting parties are agreeing (*Service Level Objectives*) and general business related properties such as usage costs and penalty fees (*Business Values*).

The OGC URN Schema Extension allows to identify domain-specific service property types and business value types. Furthermore, it classifies the service property types and business value types into different categories. Some of these categories reference service characteristics that can be automatically monitored during agreement runtime (e.g. the web service response time), whilst other categories specify service characteristics that are guaranteed by the service provider and that cannot be automatically monitored during agreement runtime (e.g. the license for data sharing). In the case that a service property is from one of the categories that can be automatically monitored during agreement runtime, the abstract SLA model allows to use the following two methods for monitoring web services. The active monitoring process simulates end-user behavior by creating real traffic at the web service. The passive monitoring process captures and analyzes real end-user traffic at the web service. However, the abstract SLA model allows to configure the active and passive monitoring process for each defined service property.

The DSL is tailored to the selected application domains and provides access to all gathered monitoring information. Based on the measurements, the DSL allows to define complex service level objectives reflecting functional and non-functional requirements (e.g. guaranteed service response time for a maximum number of service requests per month). Furthermore, the DSL allows to define complex pricing models reflecting different business use cases (e.g. fixed or usage-based cost).

The next chapter presents the design of a web-based SLA management architecture that allows to integrate the abstract SLA model in SDIs.

# Chapter 5

# Service Level Management Architecture

This chapter describes a web-based SLA management architecture. The purpose of this architecture is to enable the on-demand and online negotiation of SLAs in SDIs without the need of prior offline communication between service providers and service consumers.

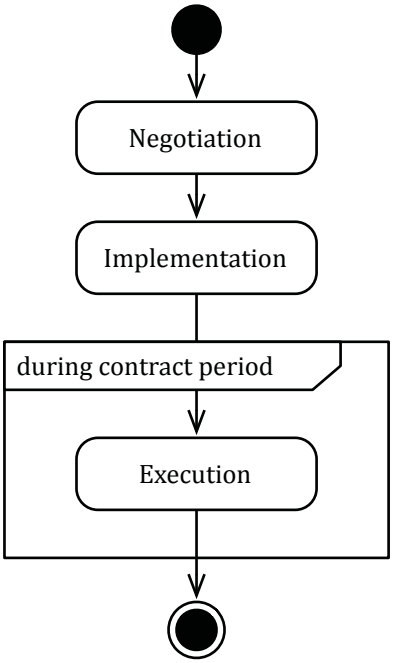
The design of the web-based SLA management architecture is inspired by policy-based management systems [Flegkas et al., 2003]. The IETF describes such a policy-based architecture for admission control [Armbrust et al., 2010] and identifies appropriate terms and components that describe such architectures [Westerinen et al., 2001]. The application of policy-based management architectures for geospatial services has already been investigated in the context of SDIs. The work presented in [Gartmann and Leinenweber, 2009], [Herrmann and Matheus, 2009], [Schäffer et al., 2010a] and [Gartmann and Schäffer, 2011] focused on role-based access control and corresponding license models as for instance XACML. This thesis relies on these previous efforts, but the proposed web-based SLA management architecture uses a more fine grained license model and covers all the SLA-related aspects that have not been addressed yet.

This chapter describes the web-based SLA management architecture in three different steps. The *process model* provides a general overview about the processes and tasks that are required for SLAs management in SDIs. The *information model* identifies all managed objects on a conceptual level and describes how these objects relate to each other independent from any specific implementation or protocol. The *data model* specifies concrete data structures and includes implementation- and protocol-specific details for realizing the design that is described by the information model. The purpose of this multi-stage approach is to design a complex system without being confronted with and distracted by technical limitations in the very beginning. Such a clear distinction allows to replace for instance one communication protocol by another one without the need to change the underlying (abstract) data model or even the actual (concrete) document encoding.

## 5.1 Process Model

The purpose of a process model is to provide "a task oriented view on a system" [Humphrey and Kellner, 1989] and to describe the "behavior of participants in a

Figure 5.1: Process Overview



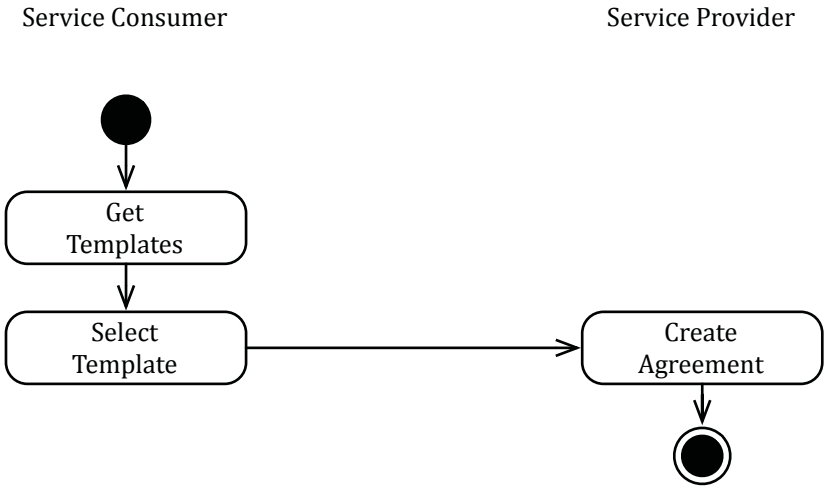

specific business interaction" [Peltz, 2003]. It provides a first overview about "the control- and data-flow among the atomic services" [Rao et al., 2004] and "describes system logic and processes that programmers use to develop necessary code modules" [Shelly et al., 1998]. This section provides process models for all phases of the agreement life cycle that are covered by the web-based SLA management architecture: agreement negotiation, agreement implementation and agreement execution (Figure 5.1).

### 5.1.1 Agreement Negotiation

The agreement negotiation phase starts with the fact that service providers offer one or more templates and that service consumers have to select one template in order to create an agreement. Each of these templates can be defined in such a way that it is a "take it or leave it proposition" [Lee and Ben-Natan, 2002] or that it can be modified and detailed by the service consumers. In the first case, the service providers do not offer the possibility for service consumers to adjust a template to their personal needs (Figure 5.2a). Therefore, the service providers clearly define the service level they are willing to deliver and indicate that service consumers can directly create legally binding contracts without intermediate steps. In the second case, service consumers are able to modify the terms of a template in order to adjust a template to their personal needs (Figure 5.2b). Based on such a configured template, a service consumer must make an agreement offer to the service provider in order to create an agreement. The service provider either can accept or reject an agreement offer. In the case of acceptance, the agreement offer directly leads to a potentially legally binding contract.

Figure 5.2: Agreement Negotiation Process

(a) Negotiation Process 1




(b) Negotiation Process 2

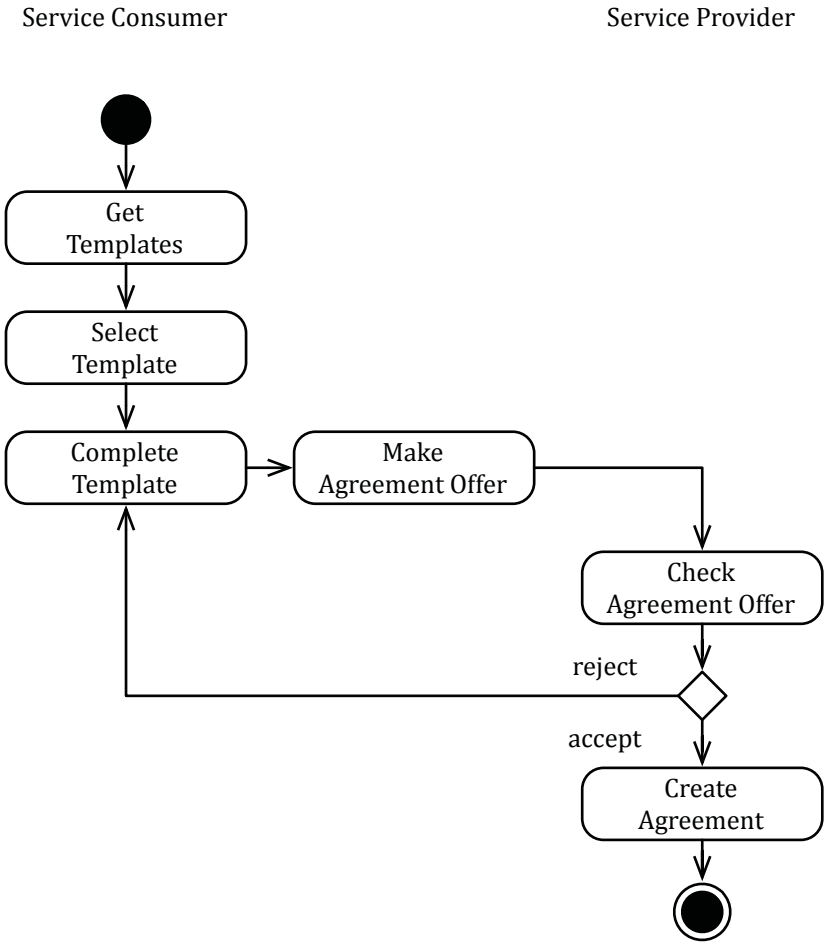
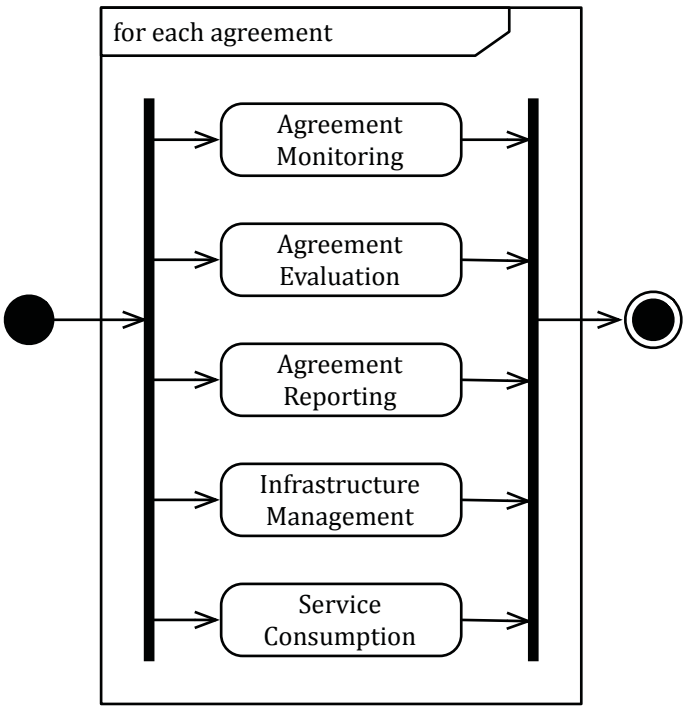


Figure 5.3: Agreement Execution Process




### 5.1.2 Agreement Implementation

The agreement implementation phase contains all activities that must be performed by the service provider in order to provide the service in compliance with the created agreement. This provisioning process covers aspects such as the initial technical setup of the service on a per instance basis, the installation of monitoring capabilities, and the configuration of reporting mechanisms [Lee and Ben-Natan, 2002].

### 5.1.3 Agreement Execution

The agreement execution phase is the "the normal day-to-day operation and associated activities related to the service being delivered" [Lee and Ben-Natan, 2002]. These activities cover agreement monitoring, agreement evaluation, agreement reporting, infrastructure management and service consumption (Figure 5.3).

The agreement monitoring phase covers the KPI measurement of all agreements on an ongoing basis. The agreement evaluation phase also covers the SLO evaluation against the KPI measurements of an agreement. Furthermore, the business values of an agreement are calculated in order to provide the input for accounting. The agreement reporting phase covers the reporting of all events that (potentially) result in SLO violations. The infrastructure management phase covers the same aspects that must be performed in the agreement implementation phase. Depending on the overall system load and the status of all active agreements, a service provider must manage the infrastructure in order to provide all services in compliance with the active agreements. The service consumption phase covers the actual service execution by the service

consumer. This phase contains all activities that must be performed in order to achieve a satisfying user experience regarding the underlying agreement. Furthermore, the service provider must enforce that only service consumers holding a valid agreement are able to execute a service. However, all these activities in the agreement execution phase can be performed in parallel on an ongoing basis during the complete agreement runtime.

## 5.2 Information Model

This section introduces the information model of the web-based SLA management architecture. The purpose of an information model is to provide an abstract representation of the objects in a managed environment [Westerinen et al., 2001]. An information model defines the relationships between managed objects at a conceptual level [Pras and Schoenwaelder, 2003] and should be "independent of any specific repository, software usage, protocol, or platform" [Westerinen et al., 2001]. The purpose of subsequent data models is to define mappings of abstract information models to concrete implementations [Moore et al., 2001].

Chapter 4 already describes an information model for templates, agreements and related documents. The following sections provide an overview about all server components of the web-based SLA management architecture.

### 5.2.1 Architecture Components

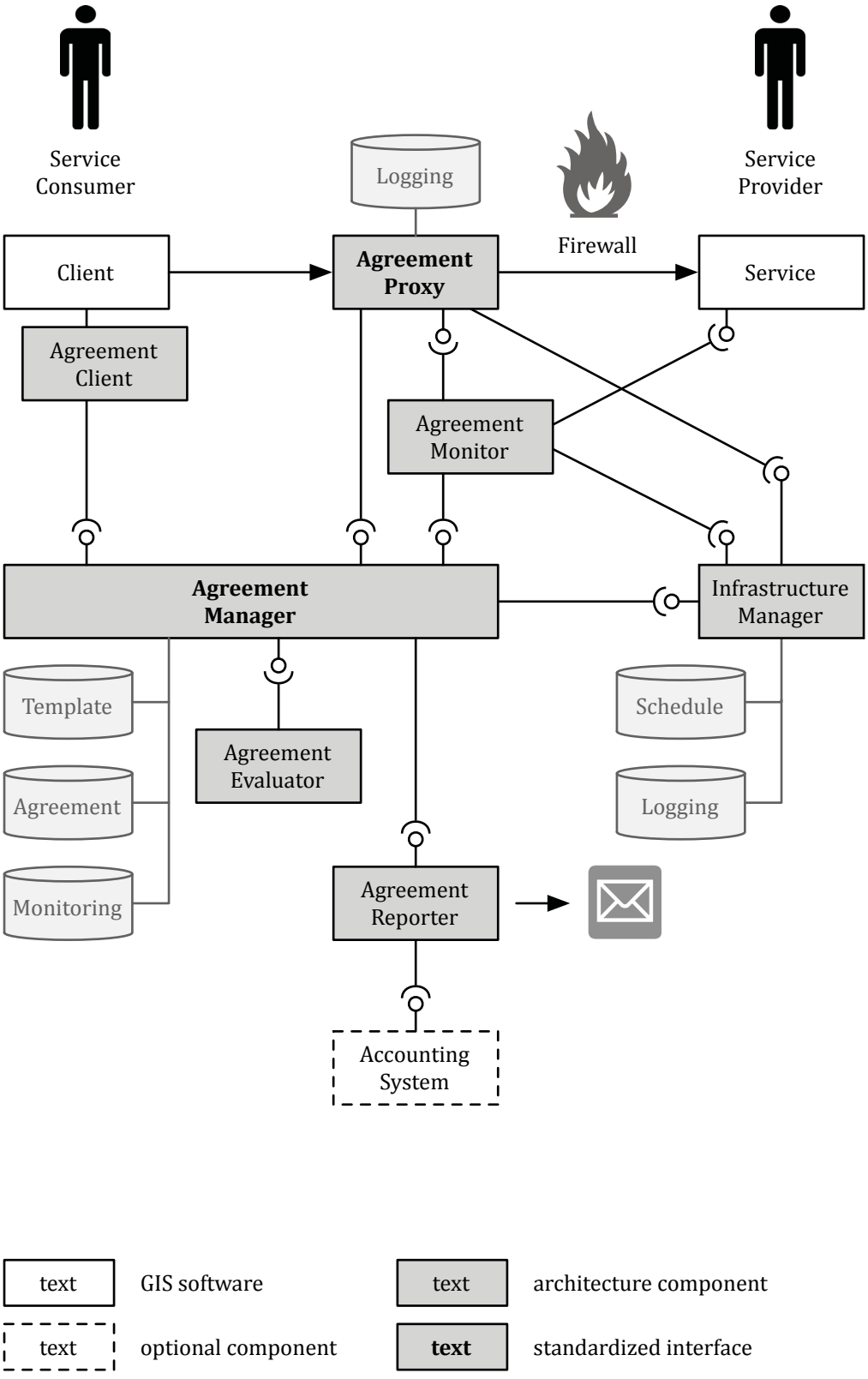
Figure 5.4 provides an overview about all components of the web-based SLA management architecture. The components and their arrangements are described as component diagrams in UML [OMG, 2011b] [OMG, 2011a]. Component diagrams are primarily used to "show the structural relationships between the components of a system" [Bell, 2004a].

At first glance, the service consumer and the service provider can be identified. The service provider maintains the original SDI service and the service consumer uses as desktop- or web-based GIS client to access the service directly. In some scenarios, the client can also be another service or software that utilizes the service for instance to complete a more complex workflow. In the diagram, such a client is represented by the 'Client' component and the original service is represented by the 'Service' component. These components are not affected directly by the web-based SLA management architecture and they don't need to be changed in order to integrate SLAs in SDIs.

To realize the publish-find-agree-bin pattern (Section 3.2.1), the following steps are carried out:

1. The original service must be protected by an internet firewall. A firewall blocks undesired incoming connections from the internet [Freed, 2000]. In combination with other efforts such as the deployment of an additional proxy component (Step 2) that helps to prevent service consumption without permission. In the context of this

Figure 5.4: Architecture Overview



thesis, a grant for service consumption is the existence of a previously negotiated and still active agreement for the service consumer and the particular service.

2. The development and deployment of a SOA for realizing agreement negotiation between service consumer and service provider. The components of the SOA must enable service providers to offer templates for their services. Furthermore, the components must enable service consumers to perform template discovery and to create agreements for particular services.

To realize the complete agreement life cycle, the following steps are carried out:

3. The development and deployment of components for realizing agreement execution. The agreement execution phase covers aspects such as agreement monitoring, agreement evaluation and agreement reporting.

4. The development and deployment of components for realizing agreement implementation. The agreement implementation phase includes all mechanisms that enable service providers to manage their infrastructure with respect to previously created agreements in order to realize promised service levels.

The following sections detail all components that are essential for the web-based SLA management architecture. Basic components such as the original SDI client and the original SDI service are not further explained. Other components such as e-commerce systems for electronic bill payment are also out of scope of this thesis.

#### AGREEMENT MANAGER

The Agreement Manager component of the web-based SLA management architecture is responsible for the management of templates, agreements and monitoring information. In the concept of policy-based management systems, the Agreement Manager realizes the 'Policy Repository'. A policy repository is "a specific data store that holds policy rules, their conditions and actions, and related policy data" [Westerinen et al., 2001]. Therefore, the Agreement Manager must provide an interface for service providers to manage templates in order to promote their service offerings, and an interface for service consumers to perform template discovery in order to find a service offering according to their personal functional or non-functional requirements. Furthermore, the Agreement Manager must provide an agreement management interface for all contracting parties and the other components of the web-based SLA management architecture. The agreement management tasks include agreement negotiation, agreement monitoring and agreement revision. Beyond basic agreement management functionality, the Agreement Manager must provide also an interface to manage related agreement data as for instance historical monitoring data that can be used for evaluating the service level objectives of an agreement.

#### AGREEMENT CLIENT

The Agreement Client component of the web-based SLA management architecture enables service consumers to obtain agreements for specific services, to get a detailed overview about the elements in an agreement and to keep track of the agreement status during the whole agreement life cycle. The Agreement Client wraps the Agreement

Manager functionality and provides an easy to use graphical interface for human users. The Agreement Client does not offer any additional features. All required functionality can directly be accessed through the Agreement Manager. Therefore, the Agreement Client component is optional but strongly recommended for the web-based SLA management architecture.

### AGREEMENT PROXY

The Agreement Proxy component of the web-based SLA management architecture acts as a proxy for the original service. A 'proxy' is a "an intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients" [Fielding et al., 1999] and a 'non-transparent proxy' is a proxy that "modifies the request or response in order to provide some added service to the user agent" [Fielding et al., 1999]. Whenever a service consumer makes a service request, the Agreement Proxy forwards the request to the target service and returns the response of the target service back to the service consumer. In some cases, the response of the target service must be modified before it is returned to the service consumer. For example, many `GetCapabilities` response documents contain URLs pointing to the original service. Service consumers and client applications normally reuse these information for accessing the service afterwards. Therefore, such URLs must be replaced with URLs pointing to the Agreement Proxy in order to enable subsequent request-response workflows. The Agreement Proxy receives requests as if it is the target server and service consumers are not aware that they are not communicating directly with the target service. Therefore, the Agreement Proxy must implement all operations and protocols of the target service. Since the Agreement Proxy acts as an intermediary of the target service, the Agreement Proxy act as a 'gateway' as for instance described in [Fielding et al., 1999].

In the concept of policy-based management systems, the Agreement Proxy is the Policy Enforcement Point (PEP) and the Policy Decision Point (PDP) of the web-based SLA management architecture combined in one single component. In policy-based management systems, the PDP is "a logical entity that makes policy decisions" that are requested by the PEP, which is a "logical entity that enforces policy decisions" [Yavatkar et al., 2000]. Whenever a service consumer makes a service request under the terms of a previously created agreement, the Agreement Proxy determines whether all agreement constraints are fulfilled or not. Obviously, the agreement runtime is such a constraint. If all agreement constraints are fulfilled, the Agreement Proxy forwards the service request to the target service. If one ore more constraints are violated, the Agreement Proxy refuses the service request.

Both aspects, the response rewriting and the agreement enforcement are following the interceptor concept [Hansen, 2007] that has already been applied successfully for dynamic rights management in geoprocessing workflows [Schäffer, 2012].

### AGREEMENT MONITOR

The Agreement Monitor component of the web-based SLA management architecture keeps track of all created agreements and calculates the values of all relevant service

properties from an agreement. To receive raw monitoring data, the Agreement Monitor relies on active and passive monitoring mechanisms (Section 4.10). In some cases, the raw monitoring data must be processed in order to calculate high-level service metrics as for instance the yearly service availability based on previous monitoring requests. Finally, the Agreement Monitor updates monitoring information for an agreement at the Agreement Manager.

#### AGREEMENT EVALUATOR

The Agreement Evaluator component of the web-based SLA management architecture keeps track of all created agreements and evaluates whether the service level objectives of an agreement are fulfilled or violated. Based on the monitoring information that are captured by the Agreement Monitor, the Agreement Evaluator evaluates the status of all service level objectives that are defined in an agreement. Based on the same monitoring information and the previously evaluated status of the service level objectives, the Agreement Evaluator also calculates the business values of an agreement. Finally, the Agreement Evaluator updates the service level objectives and the business values of an agreement at the Agreement Manager.

#### AGREEMENT REPORTER

The Agreement Reporter component of the web-based SLA management architecture keeps track of all created agreements and reports the agreement status to all contracting parties. The Agreement Reporter identifies (potential) agreement violations in order to help service providers to eliminate critical infrastructure bottlenecks (in advance). Furthermore, the Agreement Reporter can send the agreement for instance to an accounting system at certain times as defined by the business values of an agreement.

#### INFRASTRUCTURE MANAGER

The Infrastructure Manager component of the web-based SLA management architecture keeps track of all created agreements and dynamically schedules the infrastructure of the service provider according to the functional and non-functional requirements defined in all active agreements. If the infrastructure of the service provider relies on Cloud Computing, the Infrastructure Manager can, for instance, dynamically start and stop virtual machines in order to realize service offerings at exactly the right time.

Furthermore, the Infrastructure Manager records information about the infrastructure utilization and the overall system load. The information about the infrastructure utilization can be accessed by the Agreement Monitor to update the passive monitoring information for an agreement at the Agreement Manager. The information about the current and expected system load can be used to check whether the infrastructure of the service provider is capable to realize additional services offerings. The information about the current and expected system load can help the Agreement Manager to decide whether an agreement offer can be accepted or not.

In the remainder of this thesis, the Infrastructure Manager is outlined on a conceptual level as an abstract component and not detailed furthermore. Section 6.1.3 provides an example of how SDI service providers can match the basic INSPIRE service quality

requirements without investing in rarely used hardware by means of an Hybrid Cloud approach.

### 5.2.2 Component Interaction

This section describes the relationships between the components of the web-based SLA management architecture. The interactions between these components are described as sequence diagrams in UML. Sequence diagrams are primarily used to describe the interactions between managed objects in the sequential order that those interactions occur [Bell, 2004b]. They define communication patterns for a concrete implementation of a managed environment.

#### AGREEMENT NEGOTIATION

The agreement negotiation phase consists of the following steps (Figure 5.5):

1. **Template Discovery**

   First, the service consumer performs a template discovery at the Agreement Manager. Therefore, the service consumer sends a search query to the Agreement Manager, which may contain information about functional and non-functional requirements of the service consumer. The template discovery results in a list of zero, or one, or many templates that match the service consumers' requirements.

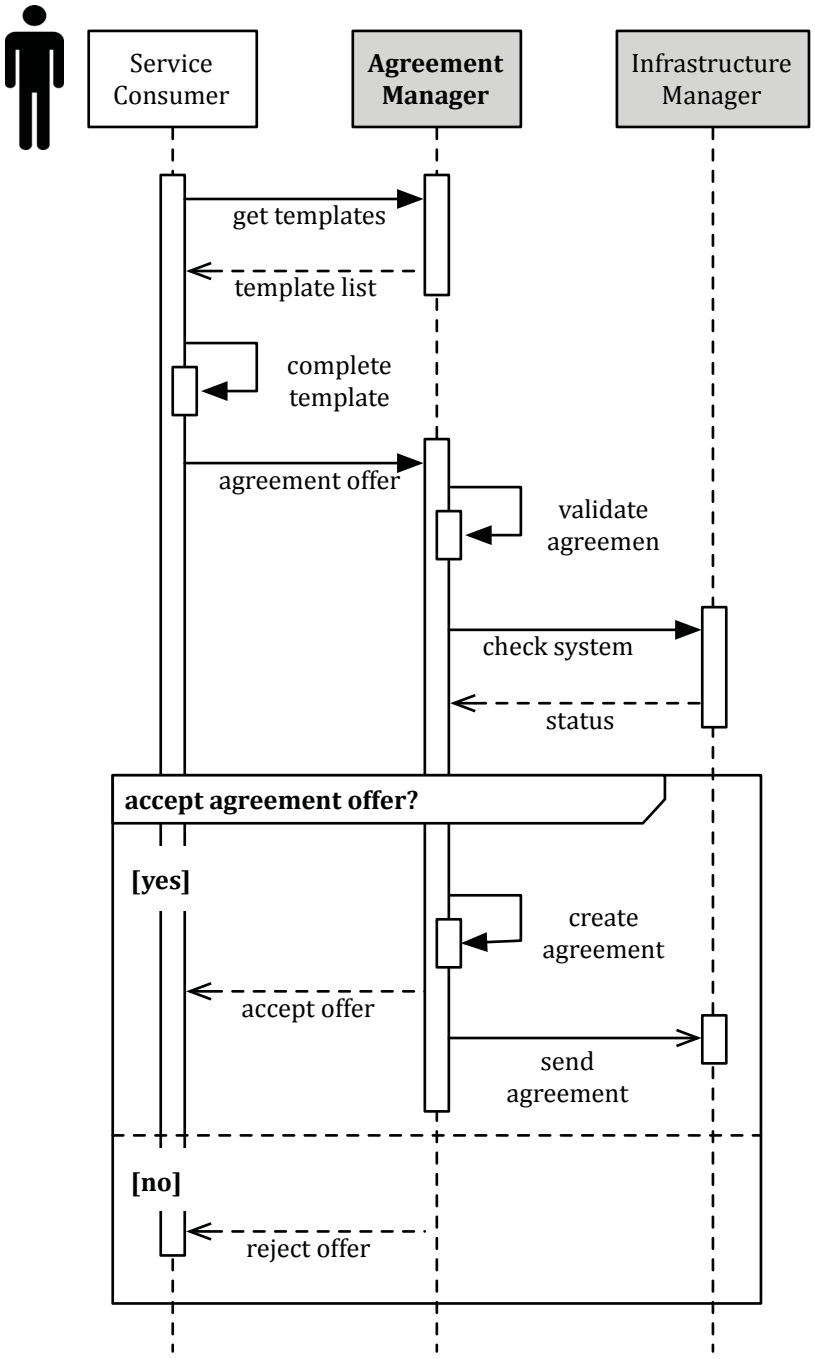
   If no template matches the specific requirements, the service consumer either can modify and resend the originally query to the Agreement Manager in order to find an adequate template, or can contact another service provider. If more than one template matches the specific requirements, the service provider must select exactly one of these templates in order to proceed. When the service consumer decides to proceed with a template, the next step is the agreement creation.

   In order to create an agreement, the service consumer must send an agreement offer to the Agreement Manager. Such an agreement offer is technically identical to the template. In some cases, the service consumer must complete a template in order to create a valid agreement offer. Maybe the service consumer must specify the contract period or accept some end user license agreements. In most cases, the service consumer has to provide additional management information such as contact and bank account details.

2. **Agreement Creation**

   When the Agreement Manager receives an agreement offer, several steps must be taken in order to validate the agreement offer, and to decide whether to accept or reject the agreement offer. First of all, the Agreement Manager checks whether the agreement offer is formally valid or not. The formal validation process checks if all required document elements are present, if the agreement offer matches the template and if the information provided by the service consumer are within the (optional) creation constraints (e.g. minimum contract period). After the formal validation process, the Agreement Manager checks whether the infrastructure of the service

Figure 5.5: Agreement Negotiation Workflow



provider is capable to deliver the demanded services in the desired service quality for the intended contract period.

If the formal validation process fails or if the infrastructure of the service provider is too busy to realize the demanded service levels, the Agreement Manager rejects the agreement offer and sends a rejection notification back to the service consumer. The rejection notification should contain detailed information about the reasons for the rejection of the agreement offer. These information can help the service consumer to modify and resend the agreement offer in order to successfully create an agreement. If the validation process and the system check succeed, the Agreement Manager creates an agreement in the local database and notifies the Infrastructure Manager about the existence of a new agreement. The agreement is enhanced by additional management information and then returned as a copy to the service consumer. The returned agreement must contain information about how to access the service and how to assess the status of the agreement during agreement runtime.

From now on, the agreement is a (potentially legally) binding contract for both contracting parties and the agreement life cycle continues with the agreement implementation and executions phases.

#### AGREEMENT IMPLEMENTATION

The Infrastructure Manager keeps track of all created agreements and initially configures the services so that they are provided in compliance with the created agreements. This initial provisioning process covers many different aspects such as the technical setup of the service, the installation of monitoring capabilities, and the configuration of reporting mechanisms.

Several approaches haven been developed to realize reliable and high-performance services, databases, networks or complete computer clusters. Implementing HTTP load balancing [Bourke, 2001] may be suited for simple application scenarios and stateless web services. Advanced failover mechanisms for databases and application states [Chouk, 2003] as well as complete computer clusters [Vogels et al., 1998] are also available for many enterprise products. However, this thesis does not develop advanced mechanisms in order to realize high-available and high-performance web services.

#### AGREEMENT EXECUTION

The agreement execution phase consists of the agreement monitoring and the agreement evaluation sub-tasks that can be executed in parallel.

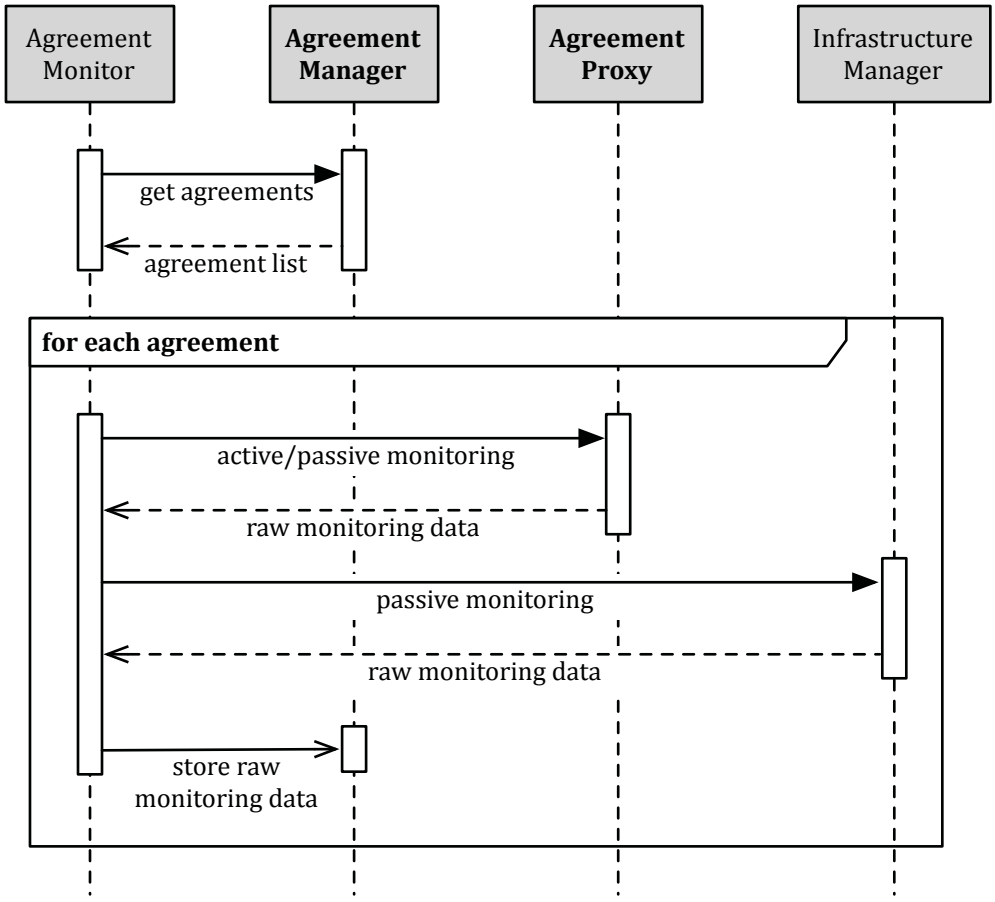
##### Agreement Monitoring

The Agreement Monitor keeps track of all created agreements and for each active agreement, the agreement monitoring task consists of the following steps (Figure 5.6):

- **Service Monitoring**

  For each resource-, runtime- and usage-related service property of an agreement, the Agreement Monitor determines the raw and atomic monitoring values for the

Figure 5.6: Agreement Monitoring Workflow



specific service property type at certain times of the day. If the agreement contains for instance a service property that represents the service availability, the Agreement Monitor determines whether the service is currently available (`true`) or not (`false`). These raw and atomic measurements can be used later by the Agreement Evaluator to calculate high-level service property values. The Agreement Evaluator can take for instance the atomic results of all availability measurements of the current calendar week (which is basically a list of boolean values) to calculate the average service availability for the current calendar week in percent (the percentage of `true` values in comparison to the total number of measurements).

Depending on the service property type and depending on the monitoring setup, the Agreement Monitor can use active or passive monitoring mechanisms to obtain the raw and atomic monitoring values (Section 4.2). If a service property defines an active monitoring setup, the Agreement Monitor sends one or more monitoring requests to the Agreement Proxy that offers the service entry point, and subsequently derives the raw and atomic monitoring values directly from the service response behavior. If a service property defines a passive monitoring setup, the Agreement Monitor requests the required information directly from the Agreement Proxy. For the agreement evaluation process it technically makes no difference whether the measurements are captured by means of active or passive monitoring mechanisms. Nevertheless, for the meaning and the interpretation of the service level objectives it is important to know whether the measurements come from real or simulated user traffic.

- **Infrastructure Monitoring**

  The raw and atomic monitoring values for the infrastructure-related service properties cannot be determined by active monitoring mechanisms. If the agreement for instance contains a service property that represents the computing resources that the service consumer actually consumed (e.g. the consumed CPU hours per accounting period), the Agreement Monitor requests the required information directly from the Infrastructure Manager.

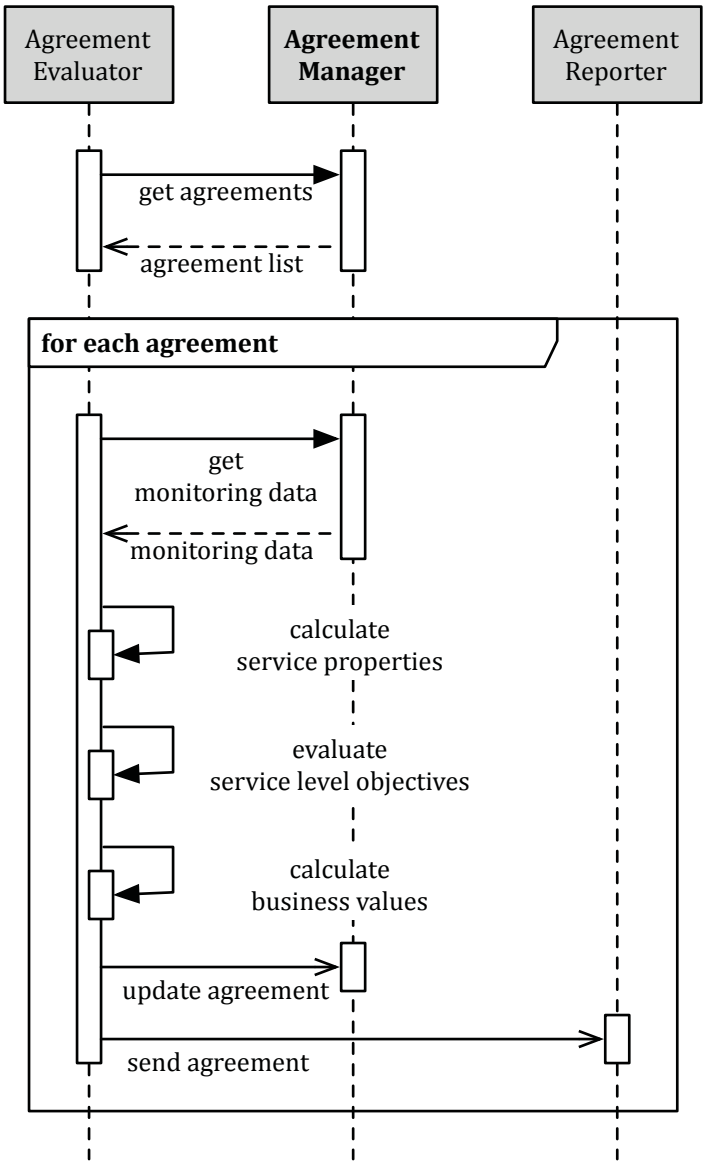
However, the monitoring setup of each service property defines the points in time when the Agreement Monitor collects the raw monitoring data. The monitoring setup also defines the content of the monitoring requests, the number of parallel requests or whether the service response should be analyzed and validated.

Finally, for each agreement the collected raw monitoring data is stored at the Agreement Manager. Therefore, the monitoring data can be reused by the Agreement Evaluator in order to evaluate service level objectives or to calculate business values. Furthermore, all contracting parties are able to review the collected raw monitoring data directly at the Agreement Manager or at the Agreement Client.

#### Agreement Evaluation

The Agreement Evaluator keeps track of all created agreements and for each active agreement, the agreement evaluation task consists of the following steps (Figure 5.7):

Figure 5.7: Agreement Evaluation Workflow



1. **Service Property Calculation**

   For each resource-, runtime-, usage- and infrastructure-related service property of an agreement, the Agreement Evaluator gathers all available measurements at the Agreement Manager and calculates the corresponding high-level service property values. Based on these values, the Agreement Evaluator creates the JEXL context variables that can be accessed from within the service level objectives and business values.

2. **Service Level Objective Evaluation**

   For each `Service Level Objective` element of an agreement, the Agreement Evaluator reads the script from the `Status` element and creates a JEXL context that contains all the previously created JEXL context variables. The Agreement Evaluator executes the script within the scope of the context in order to determine the current status of the service level objective.

3. **Business Value Calculation**

   For each `Business Value` element of an agreement, the Agreement Evaluator reads the script from the Value element and creates a JEXL context that contains all the previously created JEXL context variables. The Agreement Evaluator executes the script within the scope of the context in order to determine the current value of the business value.

Finally, for each agreement the evaluated service level objectives and the calculated business values are updated at the Agreement Manager. The high-level service property values are not stored permanently for two reasons. On the one hand, they are instantly outdated after they are calculated. On the other hand, they can be recalculated from the raw measurements at any time in the future.

### Agreement Reporting

The Agreement Reporter keeps track of all agreements and informs the contracting parties of an agreement about the overall agreement status, service property measurements, and the status of service level objectives and business values. These information are important for service consumers to understand whether the actual service is delivered with promised service quality or not. For service providers it is important to be aware of (potential) service quality bottlenecks. Without frequent reports, "the agreement is left merely as a statement of good intentions" [Sturm et al., 2000]. The reports might be delivered via mail to the service consumer and the service provider. In other use cases, the reports might be send to other (software) components in order to initiate other automated processes. The reports can be used for instance to optimize the infrastructure management process or to start an accounting process.

### Infrastructure Management

The infrastructure management task basically consists of the same steps as the agreement implementation phase, but these steps must be performed permanently during agreement runtime. The Infrastructure Manager keeps track of all active

agreements. Depending on the current and expected system load (e.g. user traffic or other workload indicators), the service provider (manually) or the Infrastructure Manager (automatically) must react on service quality fluctuations in order to avoid infrastructure bottlenecks.

#### Service Consumption

The service consumption phase consists of the following steps (Figure 5.8):

1. **Service Execution**

   First, the service consumer typically inserts the service reference, which is provided by the created agreement, into a GIS client in order to access the SDI service. This service reference points to the Agreement Proxy and not to the original service, which is protected by a firewall that only allows the Agreement Proxy to access the original SDI service. The service reference to the Agreement Proxy is unique for the created agreement, for the underlying original SDI service, and maybe unique for a specific user or a group of users. However, the GIS client is not aware that service requests are send to the service under the terms of a previously created agreement and therefore sends standards-compliant OWS service requests to the Agreement Proxy.

2. **Agreement Enforcement**

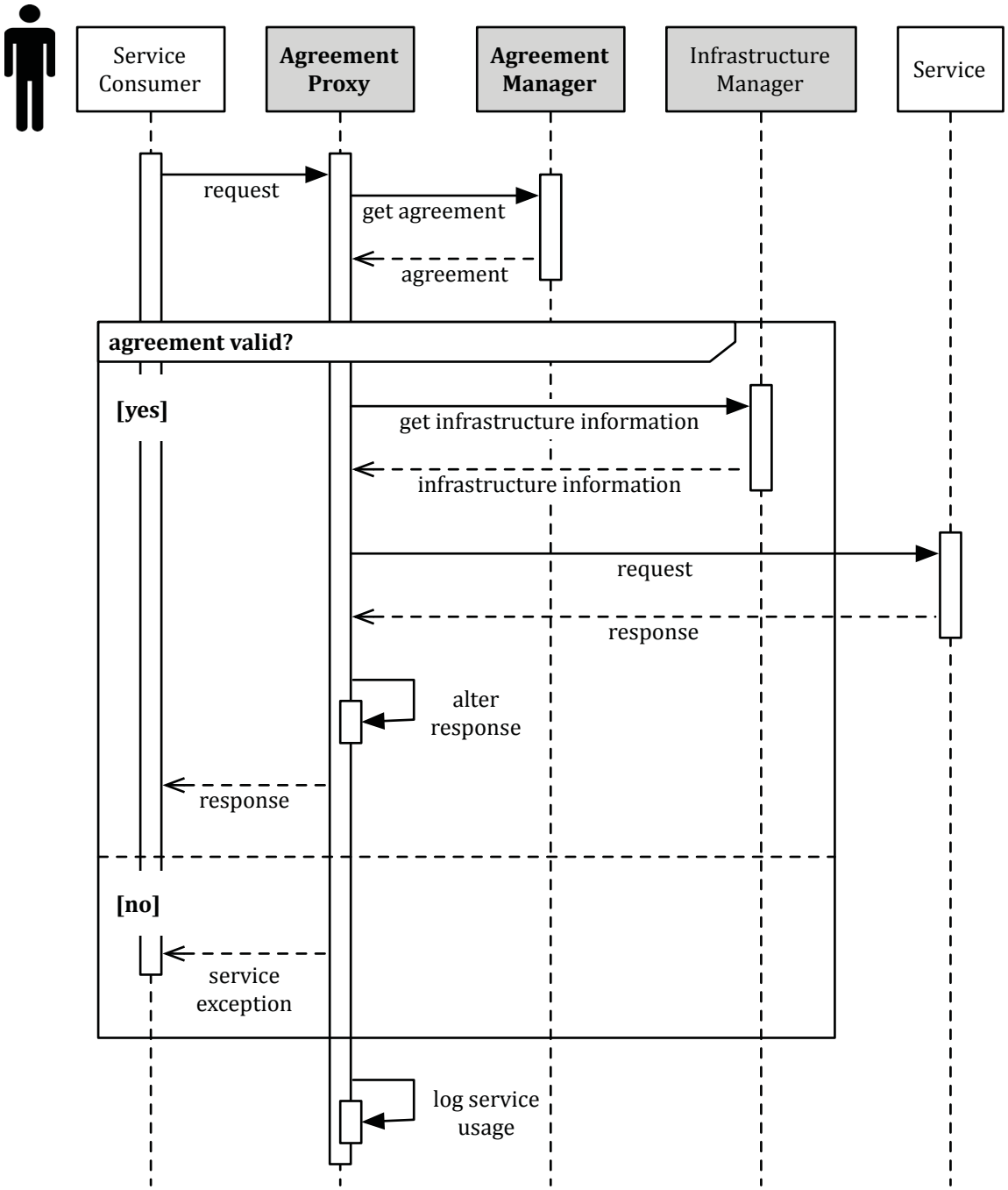
   For each service request, the Agreement Proxy checks whether the agreement, which is identified by the unique service reference, is formally valid or not. This formal validation process checks for instance whether the service execution is within the contract period or not. If the formal validation process fails, the Agreement Proxy sends an Exception Report message [Whiteside and Greenwood, 2010] back to the service consumer. The service exception should contain detailed information about the reasons for the exception. These information can help service consumers to find potential errors in their workflows or to obtain a new agreement for the targeted service. In the case the formal validation process succeeds, the Agreement Proxy checks whether the actual service request is valid.

   The service request validation process checks for instance whether the service consumer is allowed to make the specific service request or not. Limiting constraints can be for instance a maximum number of allowed service requests per month or a maximum amount of data that can be transferred to the service per month. If the service request validation process fails, the Agreement Proxy sends an Exception Report message [Whiteside and Greenwood, 2010] back to the service consumer. If the service request validation process succeeds, the Agreement Proxy continues to execute the served request at the original SDI service.

3. **Request Forwarding**

   If all preliminary checks succeed, the Agreement Proxy requests the Infrastructure Manager how to contact the original SDI service. Afterwards, the Agreement Proxy calls the original SDI service on behalf of the service consumer (proxy). In some cases, the response of the original service must be modified (non-transparent proxy) before it can be returned back to the service consumer. However, the service

Figure 5.8: Service Consumption Workflow



consumer is not aware that service requests are checked, that the Agreement Proxy is a gateway and not the original service, and that the service response may be modified.

During the service consumption phase, the Agreement Proxy measures and records all kinds of service properties as for instance the service response time or the number of delivered pixel. These measurements are stored locally at the Agreement Proxy, so that these measurements can be reused later for instance by the Agreement Monitor for passive monitoring.

## 5.3 Data Model

This section introduces the data model for the web-based SLA management architecture. A data model is a mapping of an information model into a form that is specific to a concrete implementation [Shelly et al., 1998]. It is intended for developers and includes implementation- and protocol-specific details [Pras and Schoenwaelder, 2003]. Although information models and data models normally serve different purposes, in some cases it is not possible to precisely define "what kind of details should be expressed in an information model and which ones belong in a data model" [Pras and Schoenwaelder, 2003]. One example in the context of this thesis is the utilization of the JEXL language in the abstract SLA model, which normally should contain no concrete implementation- and protocol-specific details.

This section introduces a mapping of the abstract SLA model to an extended and particular version of the WS-Agreement specification. Furthermore, this section describes the service interfaces of the Agreement Manager and the Agreement Proxy. From a standardization perspective, these components are the most relevant components in the web-based SLA management architecture. They ensure interoperable template discovery and service consumption across service provider boundaries. The Agreement Manager and the Agreement Proxy are the only components that are directly accessible for service consumers. All other components need not necessarily be standardized. They are only executed internally and can be realized by vendor specific solutions. Appendix D.2 describes example service interfaces for those internally used components.

### 5.3.1 WS-Agreement Application Profile

The WS-Agreement specification [Andrieux et al., 2005] defines an XML-based SLA template format and a SOAP-based web service protocol for establishing agreements between two parties. This section describes the WS-Agreement Application Profile for OGC Web Services, which has been presented first in [Baranski, 2011]. The application profile defines a mapping of the abstract SLA model to an extended version of the WS-Agreement specification as a concrete implementation.

#### WS-AGREEMENT SPECIFICATION

The WS-Agreement specification mainly consists of two parts. First, an XML Schema for specifying the structure of templates and agreements. Second, a web service interface

for managing the life cycle of an agreement.

### Agreement Structure

In the WS-Agreement specification an agreement is composed of several distinct parts.

#### Agreement Context

The *Agreement Context* part contains general metadata about the entire agreement as for instance domain-specific information about the participants in the agreement and the agreement's lifetime.

#### Service Terms

The *Service Terms* part contains a set of terms, each expressing the defined consensus or obligations of the participants in the agreement. It contains information to identify the service to which an agreement pertains and to which the guarantee terms can apply. The *Service Description Terms* section provides a domain-specific description of the service to which an agreement pertains. The *Service Reference* section points to the service to which an agreement pertains. The *Service Properties* section is used to define domain-specific measurable and exposed properties associated with a service. These service properties are used for expressing the domain-specific service level objectives in the *Guarantee Terms* part of an agreement.

#### Guarantee Terms

The *Guarantee Terms* part contains information about the service quality associated with the service described by the *Service Terms* part. The *Service Level Objective* section is expressed over the service properties and defines an assurance on service quality. The *Business Value List* defines an optional list of business values associated with a service level objective.

In WS-Agreement the general structure of a template is the same as that of an agreement. But a template may also contain an additional *Creation Constraints* part that contains constraints on possible values of the *Service Terms* and the *Guarantee Terms* for creating an agreement.

### Service Interface

The WS-Agreement specification defines a web service interface for creating, representing and monitoring agreements. The web service interface is based on the Web Services Resource Framework (WSRF) [Czajkowski et al., 2004], a SOAP-based web service interface that allows the modeling of stateful resources with web services. The WS-Agreement specification defines the following main port types and resource properties. The `AgreementFactory` port type offers a `CreateAgreement` operation for creating agreements. The `Template` resource property of the `AgreementFactory` port type represents a sequence of zero or more templates of agreement offers that can be accepted by the `AgreementFactory` operations in order to create an agreement. The `Agreement` port type offers different resource properties providing static information about an agreement. The `Name` resource property exposes the name of an agreement.

The `AgreementId` resource property exposes an unique identifier of an agreement. The `Context` resource property exposes the context information of the entire agreement. The `Terms` resource property exposes the service and guarantee terms of an agreement. The `AgreementState` port type offers different resource properties providing runtime information about an agreement. The `AgreementState` resource property exposes the overall agreement state. The `ServiceTermState` resource property exposes a service runtime state for each service description term of an agreement. The `GuaranteeTermState` resource property exposes a state of fulfillment for each guarantee term of an agreement.

The overall agreement compliance can be observed at agreement runtime by monitoring different types of states. The agreement states can be observed via the `AgreementState` resource property and the primary agreement states are "Pending", "Observed", "Rejected", "Completed" and "Terminated". The "Pending" state means that an agreement offer has been made, but it has been neither accepted nor rejected. The "Observed" state means that an agreement offer has been made and accepted. The "Rejected" state means that an agreement offer has been made and rejected. The "Complete" state means that an agreement offer has been received and accepted, and that all activities pertaining to the agreement are finished. The "Terminated" state means that an agreement offer has been terminated by the agreement initiator and that the obligation no longer exists. The service runtime states could be observed via the `ServiceTermState` resource property and primary service runtime states are "Not Ready", "Ready" and "Completed". The "Not Ready" state means that a service cannot be used. The "Ready" state means that a service is ready for use by a client. The "Completed" state means that a service cannot be used any more and any service provider activity is finished. The guarantee states could be observed via the `GuaranteeTermState` resource property and primary guarantee states are "Fulfilled", "Violated" and "NotDetermined". The "Fulfilled" state means that currently the specific guarantee is fulfilled. The "Violated" state means that currently the specific guarantee is violated. The "NotDetermined" state is the initial state of a guarantee term and it means that no activity regarding this guarantee has happened yet that allows evaluating whether the guarantee is met or not. However, the different state types defined by the WS-Agreement specification are independent of domain-specific service descriptions, service properties and guarantee terms. They can be applied as they are in a broad range of usage domains, but also be extended for other application scenarios.

#### APPLICATION PROFILE

The most current XML Schema of WS-Agreement is available in the schema repository of the OGF[1]. The WS-Agreement specification defines neither domain-specific expressions and metrics for service descriptions and service properties, nor how and where to measure such properties. Furthermore, the WS-Agreement specification does not define a specific condition expression language for defining and evaluating domain-specific guarantee terms. Therefore, the XSD of WS-Agreement contains various `type="xs:anyType"` attributes and `<xs:any/>` elements in order to allow the definition of any domain-specific content in XML documents that can be conform to

[1] http://schemas.ggf.org/graap/2007/03/ws-agreement

the WS-Agreement specification.

The WS-Agreement Application Profile for OGC Web Services describes a domain-specific extension of WS-Agreement and mainly consists of the following three parts:

- A set of XSD for specifying the domain-specific content of an agreement.
- An URN namespace for identifying domain-specific service properties and business values.
- A DSL for defining and evaluating domain-specific guarantee terms.

Section 4.3.1 already defines an OGC URN Schema Extension for identifying domain-specific service properties and business values in the abstract SLA model. The URN namespace can be reused without any modifications in the application profile. Section 4.3.2 already defines a DSL for evaluating the status of service level objectives and calculating business values in the abstract SLA model. The defined Agreement Expression Language can also be reused without any modifications in the application profile.

Based on the abstract SLA model, the following sections introduce a set of XSD for specifying the domain-specific content in WS-Agreement. Example XML documents implementing these XSDs can be found in Appendix C.1.2.

### Agreement Context

The `AgreementInitiator` and the `AgreementResponder` elements in the *Agreement Context* part of the WS-Agreement specification should help to identify and contact the corresponding parties, which may have obligations in an agreement. The XSD of the WS-Agreement specification defines the following XML elements for the agreement initiator and the agreement responder (Listing 5.1).

Listing 5.1: Agreement Context in WS-Agreement

```
<xs:complexType name="AgreementContextType">
  (...)
  <xs:element minOccurs="0" name="AgreementInitiator" type="xs:anyType"/>
  <xs:element minOccurs="0" name="AgreementResponder" type="xs:anyType"/>
  (...)
</xs:complexType>
```

The `type="xs:anyType"` attribute allows to define any domain-specific content in the `AgreementInitiator` and the `AgreementResponder` element. Section 4.1.1 defines the general structure and content for describing the service provider and the service consumer in the *Agreement Context* part of the abstract SLA model. These information are derived from the metadata that is delivered by the `GetCapabilities` operation of all OWS. They can be reused without any modifications to define the domain-specific content of the `AgreementInitiator` and the `AgreementResponder` element in the WS-Agreement specification. An example XML document that can be embedded for instance in the `AgreementResponder` element can be found in Listing 5.2.

Listing 5.2: Example Agreement Responder

```
<wsag:AgreementResponder>
  <wsag-ogc:Contact>
    <wsag-ogc:Name>Institute for Geoinformatics</wsag-ogc:Name>
    <wsag-ogc:Site xmlns:xlin="http://www.w3.org/1999/xlink" xlin:href="http
        ://www.ifgi.de"/>
    <wsag-ogc:Contact>
      <ows:IndividualName>Bastian Baranski</ows:IndividualName>
      <ows:PositionName>Research Associate</ows:PositionName>
      <ows:ContactInfo>
        <ows:Phone>
          <ows:Voice>+49 251 8333071</ows:Voice>
          <ows:Facsimile>+49 251 8339763</ows:Facsimile>
        </ows:Phone>
        <ows:Address>
          <ows:DeliveryPoint>Weseler Strasse 253</ows:DeliveryPoint>
          <ows:City>Muenster</ows:City>
          <ows:PostalCode>48151</ows:PostalCode>
          <ows:Country>Germany</ows:Country>
          <ows:ElectronicMailAddress>baranski@uni-muenster.de</ows:
              ElectronicMailAddress>
        </ows:Address>
        <ows:HoursOfService>The hours of service are Monday to Friday from 8
            AM to 16 PM.</ows:HoursOfService>
        <ows:ContactInstructions>Please contact the service desk via phone
            or mail.</ows:ContactInstructions>
      </ows:ContactInfo>
    </wsag-ogc:Contact>
  </wsag-ogc:Contact>
</wsag:AgreementResponder>
```

The XSD specifying the domain-specific content in the `AgreementInitiator` and the `AgreementResponder` element can be found in Appendix C.1.1.

### Service Description Terms

The *Service Description Terms* section is fundamental for agreements in WS-Agreement because the agreement is about the service(s) that are described by the list of zero, or one, or more `ServiceDescriptionTerm` elements in the *Service Description Terms* section. The XSD of the WS-Agreement specification defines the following XML elements for the *Service Description Terms* section (Listing 5.3).

Listing 5.3: Service Description in WS-Agreement

```
<xs:complexType name="TermCompositorType">
  <xs:sequence>
    <xs:choice maxOccurs="unbounded">
      (...)
      <xs:element name="ServiceDescriptionTerm" type="wsag:
          ServiceDescriptionTermType"/>
      (...)
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<xs:complexType abstract="true" name="TermType">
  <xs:attribute name="Name" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType abstract="true" name="ServiceTermType">
  <xs:complexContent>
     <xs:extension base="wsag:TermType">
       <xs:attribute name="ServiceName" type="xs:string" use="required" />
     </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

```
20 <xs:complexType name="ServiceDescriptionTermType">
21   <xs:complexContent>
22     <xs:extension base="wsag:ServiceTermType">
23       <xs:sequence>
24         <xs:any namespace="##other" processContents="strict"/>
25       </xs:sequence>
26     </xs:extension>
27   </xs:complexContent>
28 </xs:complexType>
```

The `<xs:any/>` element allows to define any domain-specific content in the `ServiceDescriptionTerm` element. To map the *Service Description* section, the *Service Properties* section, and the *Contract Period* section of the abstract SLA model to WS-Agreement, the WS-Agreement Application Profile for OGC Web Services defines three different types of `ServiceDescriptionTerm` elements.

- The *Functional Service Description* describes what kind of service (functionality) is offered by the service provider.
- The *Non-Functional Service Description* defines the service level that is guaranteed by the service provider and what service properties shall be monitored during agreement runtime.
- The *Contract Period* defines the start and the end date of the service offering.

These service description types allow the parties, which may have obligations in the agreement, to identify all relevant service aspects that must be fulfilled and monitored during the agreement life cycle.

### *Functional Service Description*

Section 4.1.2 defines the general structure and content for describing the *Service Description* section of the abstract SLA model. These information are derived from the metadata that is delivered by the `GetCapabilities` operation of all OWS. They can be reused without any modifications to provide the domain-specific functional service description in the *Service Description Terms* section in WS-Agreement.

An example XML document that can be embedded in the `ServiceDescriptionTerm` element is shown in Listing 5.4.

Listing 5.4: Example Functional Service Description

```
1 <wsag:ServiceDescriptionTerm wsag:Name="SERVICE_DESCRIPTION_SDT" wsag:
      ServiceName="INSPIRE_VIEW_SERVICE">
2   <wsag-ogc:ServiceDescription>
3     <wsag-ogc:Title>INSPIRE View Service</wsag-ogc:Title>
4     <wsag-ogc:Abstract>This service instance is an INSPIRE View Service
          implementation.</wsag-ogc:Abstract>
5     <wsag-ogc:Keywords>INSPIRE, View Service, OGC, WMS</wsag-ogc:Keywords>
6     <wsag-ogc:Type>urn:ogc:doc:is:wms:1.1.1</wsag-ogc:Type>
7   </wsag-ogc:ServiceDescription>
8 </wsag:ServiceDescriptionTerm>
```

The XSD specifying the domain-specific functional service description for the *Service Description Terms* section can be found in Appendix C.1.1.

### *Non-Functional Service Description*

Section 4.1.4 defines the general structure and content for describing the *Service Properties* section in the abstract SLA model. These information can be reused without any modifications to provide the domain-specific non-functional service description in the *Service Description Terms* section in the WS-Agreement specification.

An example XML document that can be embedded in the `ServiceDescriptionTerm` element can be found in Listing 5.5.

Listing 5.5: Example Non-Functional Service Description

```
<wsag:ServiceDescriptionTerm wsag:Name="SERVICE_PROPERTIES_SDT" wsag:
    ServiceName="INSPIRE_VIEW_SERVICE">
  <wsag-ogc:ServiceProperties>
    <!-- RESOURCE-RELATED PROPERTIES -->
    <wsag-ogc:Property>
      <wsag-ogc:Name>operations</wsag-ogc:Name>
      <wsag-ogc:Title>Supported Operations</wsag-ogc:Title>
      <wsag-ogc:Abstract>The operations that are supported by the service.</
          wsag-ogc:Abstract>
      <wsag-ogc:Type>urn:ogc:def:sla:property:resource:operation</wsag-ogc:
          Type>
      <wsag-ogc:Monitoring>
        <wsag-ogc:ActiveMonitoring>
          <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
          <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
          <wsag-ogc:Period>360000</wsag-ogc:Period>
        </wsag-ogc:ActiveMonitoring>
      </wsag-ogc:Monitoring>
    </wsag-ogc:Property>
    <!-- RUNTIME-RELATED PROPERTIES -->
    <wsag-ogc:Property>
      <wsag-ogc:Name>availability</wsag-ogc:Name>
      <wsag-ogc:Title>Service Availability</wsag-ogc:Title>
      <wsag-ogc:Abstract>The general availability of the service.</wsag-ogc:
          Abstract>
      <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:availability</wsag-ogc
          :Type>
      <wsag-ogc:Monitoring>
        <wsag-ogc:ActiveMonitoring>
          <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
          <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
          <wsag-ogc:Period>360000</wsag-ogc:Period>
          <wsag-ogc:Request>
            <wsag-ogc:Method>GET</wsag-ogc:Method>
            <wsag-ogc:Content>service=WMS&amp;version=1.3.0&amp;request=
                GetMap&amp;layers=topp:tasmania_state_boundaries&amp;styles
                =&amp;bbox=${__random(142.0,144.0)},${__random(-46.0,-44.0)
                },${__random(150.0,152.0)},${__random(-38.0,-36.0)}&amp;
                width=800&amp;height=600&amp;srs=EPSG:4326&amp;format=image/
                png</wsag-ogc:Content>
          </wsag-ogc:Request>
          <wsag-ogc:Response>
            <wsag-ogc:Status>200</wsag-ogc:Status>
          </wsag-ogc:Response>
        </wsag-ogc:ActiveMonitoring>
      </wsag-ogc:Monitoring>
    </wsag-ogc:Property>
    (...)
  </wsag-ogc:ServiceProperties>
</wsag:ServiceDescriptionTerm>
```

The OGC URN Schema Extension described in Section 4.3.1 can be used in the `Type` element to identify which specific service property shall be monitored by the web-

based SLA management architecture during agreement runtime. The `Value` element defines concrete service levels that are guaranteed by the service provider. The XSD specifying the domain-specific functional service description for the *Service Description Terms* section can be found in Appendix C.1.1.

### *Contract Period*

Section 4.1.1 defines the general structure and content for describing the *Agreement Context* section in the abstract SLA model. Beside the information about the service provider and the service consumer, the *Agreement Context* section of the abstract SLA model contains information about the contract period. These information can be reused without any modifications to provide the contract period information in the *Service Description Terms* section in WS-Agreement.

An example XML document that can be embedded in the `ServiceDescriptionTerm` element can be found in Listing 5.6.

Listing 5.6: Example Service Availability Period

```
<wsag:ServiceDescriptionTerm wsag:Name="CONTRACT_RUNTIME_SDT" wsag:
    ServiceName="INSPIRE_VIEW_SERVICE">
  <res-sla:TimeConstraint>
    <res-sla:StartTime>2010-07-04T13:00:00+02:00</res-sla:StartTime>
    <res-sla:EndTime>2012-07-09T13:00:00+02:00</res-sla:EndTime>
  </res-sla:TimeConstraint>
</wsag:ServiceDescriptionTerm>
```

The XSD specifying the domain-specific contract period for the *Service Description Terms* section can be found in Appendix C.1.1.

### Service Reference

The `ServiceReference` element of the WS-Agreement specification points to a service for instance by providing an Endpoint Reference (EPR) that identifies a web service. The XSD of the WS-Agreement specification defines the following XML elements for such a domain-specific service reference (Listing 5.7).

Listing 5.7: Service Reference in WS-Agreement

```
<xs:complexType name="TermCompositorType">
  <xs:sequence>
    <xs:choice maxOccurs="unbounded">
      (...)
      <xs:element name="ServiceReference" type="wsag:ServiceReferenceType"/>
      (...)
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceReferenceType">
  <xs:complexContent>
    <xs:extension base="wsag:ServiceTermType">
      <xs:sequence>
        <xs:any namespace="##other" processContents="strict"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The <xs:any/> element allows the definition of any domain-specific information in ServiceDescriptionTerm elements. Section 4.1.3 defines the structure and the content for the service reference in the abstract SLA model. These information can be used without any modifications to define the domain-specific content of the ServiceReference element in WS-Agreement.

An example XML document that can be embedded in the ServiceReference element can be found in Listing 5.8.

Listing 5.8: Example Service Reference

```
<wsag:ServiceReference wsag:Name="SERVICE_REFERENCE" wsag:ServiceName="
    INSPIRE_VIEW_SERVICE">
  <wsag-ogc:ServiceReference>
    <wsag-ogc:URL>http://localhost:8088/sla-proxy/DefaultWMS</wsag-ogc:URL>
  </wsag-ogc:ServiceReference>
</wsag:ServiceReference>
```

The XSD specifying the service reference for the *Service Reference* section can be found in Appendix C.1.1.

### Service Properties

The *Service Properties* section of the WS-Agreement specification does not allow to define domain-specific content. The WS-Agreement Application Profile for OGC Web Services already defines domain-specific non-functional service properties in the *Service Description Term* section and domain-specific custom service levels in the *Service Level Objective* section. Therefore, the *Service Properties* section is ignored by the WS-Agreement Application Profile for OGC Web Services.

### Service Level Objective

The ServiceLevelObjective element of the WS-Agreement specification defines a domain-specific and machine-readable condition that must be met to fulfill a service quality guarantee. The XSD of the WS-Agreement specification defines the following XML elements for the *Service Level Objective* section (Listing 5.9).

Listing 5.9: Service Level Objectives in WS-Agreement

```
<xs:complexType name="GuaranteeTermType">
  <xs:complexContent>
    <xs:extension base="wsag:TermType">
      <xs:sequence>
        (...)
        <xs:element ref="wsag:ServiceLevelObjective"/>
        (...)
      </xs:sequence>
      (...)
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="ServiceLevelObjective" type="wsag:
    ServiceLevelObjectiveType"/>
```

```
14 <xs:complexType name="ServiceLevelObjectiveType">
15   <xs:choice>
16     <xs:element name="KPITarget" type="wsag:KPITargetType"/>
17     <xs:element name="CustomServiceLevel" type="xs:anyType"/>
18   </xs:choice>
19 </xs:complexType>
20 <xs:complexType name="KPITargetType">
21   <xs:sequence>
22     <xs:element name="KPIName" type="xs:string"/>
23     <xs:element name="CustomServiceLevel" type="xs:anyType" />
24   </xs:sequence>
25 </xs:complexType>
```

The `type="xs:anyType"` attribute allows the definition of any domain-specific information in the `CustomServiceLevel` element. Section 4.1.5 defines the structure and the content for `Service Level Objective` elements in the abstract SLA model. These information can be reused without any modifications to define the domain-specific content of the `CustomServiceLevel` element in the WS-Agreement specification.

An example XML document that can be embedded in the `CustomServiceLevel` element can be found in Listing 5.10.

Listing 5.10: Example Service Level Objective

```
1  <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_AVAILABILITY" wsag:
       Obligated="ServiceProvider">
2    <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
3    <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
         QualifyingCondition>
4    <wsag:ServiceLevelObjective>
5      <wsag:CustomServiceLevel>
6        <wsag-ogc:CustomServiceLevel>
7          <wsag-ogc:Name>InspireAvailability</wsag-ogc:Name>
8          <wsag-ogc:Title>INSPIRE (Availability)</wsag-ogc:Title>
9          <wsag-ogc:Abstract>The probability of a Network Service to be
               available shall be 99% of the time.</wsag-ogc:Abstract>
10         <wsag-ogc:Status>
11           (availability.week >= 0.99) and (availability.month >= 0.99) and (
                 availability.year >= 0.99)
12         </wsag-ogc:Status>
13       </wsag-ogc:CustomServiceLevel>
14     </wsag:CustomServiceLevel>
15   </wsag:ServiceLevelObjective>
16   <wsag:BusinessValueList/>
17 </wsag:GuaranteeTerm>
```

The Agreement Expression Language described in Section 4.3.2 can be used in the `Status` element to define the process for evaluating the status of a service quality assurance. The XSD specifying domain-specific custom service levels for the *Service Level Objective* section can be found in Appendix C.1.1.

### Business Value List

The `BusinessValueList` element of the WS-Agreement specification defines a list of domain-specific and machine-readable business value aspects of an agreement. The XSD of the WS-Agreement specification defines the following XML elements for the *Business Value List* section (Listing 5.11).

Listing 5.11: Business Values in WS-Agreement

```
<xs:complexType name="GuaranteeTermType">
  <xs:complexContent>
    <xs:extension base="wsag:TermType">
      <xs:sequence>
        (...)
        <xs:element name="BusinessValueList" type="wsag:
            BusinessValueListType"/>
      </xs:sequence>
      (...)
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="BusinessValueListType">
  <xs:sequence>
    <xs:element minOccurs="0" name="Importance" type="xs:integer"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" name="Penalty" type="
        wsag:CompensationType"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" name="Reward" type="wsag
        :CompensationType"/>
    <xs:element minOccurs="0" name="Preference" type="wsag:PreferenceType"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" name="
        CustomBusinessValue" type="xs:anyType"/>
  </xs:sequence>
</xs:complexType>
```

The `type="xs:anyType"` attribute allows the definition of any domain-specific information in the `CustomBusinessValue` element. Section 4.1 defines the structure and the content for business values in the abstract SLA model. These information can be reused without any modifications to define the domain-specific content of the `CustomBusinessValue` element in the WS-Agreement specification.

An example XML document that can be embedded in the `CustomBusinessValue` element can be found in Listing 5.12.

Listing 5.12: Example Business Value

```
<wsag:GuaranteeTerm wsag:Name="COSTS_PER_YEAR" wsag:Obligated="
    ServiceProvider">
  <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
  <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
      QualifyingCondition>
  <wsag:ServiceLevelObjective/>
  <wsag:BusinessValueList>
    <wsag:CustomBusinessValue>
      <wsag-ogc:CustomBusinessValue>
        <wsag-ogc:Name>CostsPerYear</wsag-ogc:Name>
        <wsag-ogc:Title>Usage Costs (Year)</wsag-ogc:Title>
        <wsag-ogc:Abstract>The cost to be assessed for using the service on
            a yearly basis (in Euro).</wsag-ogc:Abstract>
        <wsag-ogc:Type>urn:ogc:def:sla:business:cost:year</wsag-ogc:Type>
        <wsag-ogc:Value>
          factor;
          if (pixel.year lt (1000000 * 1000))
          {
            factor = 1.0;
          } else
          if (pixel.year lt (1000000 * 10000))
          {
            factor = 0.5;
          } else
          if (pixel.year lt (1000000 * 100000))
          {
          factor = 0.25;
          } else
```

```
26            if (pixel.year lt (1000000 * 1000000))
27            {
28              factor = 0.125;
29            } else
30            {
31              factor = 0.0625;
32            }
33            (factor * (pixel.year / 1000000));
34          </wsag-ogc:Value>
35        </wsag-ogc:CustomBusinessValue>
36      </wsag:CustomBusinessValue>
37    </wsag:BusinessValueList>
38  </wsag:GuaranteeTerm>
```

The OGC URN Schema Extension described in Section 4.3.1 can be used in the `Type` element to identify which specific business value should be calculated by the web-based SLA management architecture during agreement runtime. The DSL described in Section 4.3.2 can be used in the `Status` element to define the process for calculating the specific business value. The XSD specifying domain-specific business values for the *Business Value List* section can be found in Appendix C.1.1.

### 5.3.2 Service Interfaces

This section defines the service interfaces of the Agreement Manager and the Agreement Proxy. These components are the only components in the web-based SLA management architecture that must be public accessible for potential service consumers and their respective SLA clients and GIS applications. Appendix D.2 proposes service interfaces for all other components that must not be standardized and that can be realized by implementations specific for each service provider.

The interfaces of the Agreement Manager and the Agreement Proxy are designed in Representational State Transfer (REST) architectural style [Fielding, 2000]. The fundamental principle behind the REST approach is that everything is a resource that can be identified by an URI. The REST concept makes use of the HTTP protocol in order to manipulate the content (representation) of a resource. The GET operation reads the representation of a resource. The POST operation creates a new sub-resource and returns an URI that represents the new resource. The PUT operation creates or updates the representation of a resource. The DELETE operation completely deletes a resource. However, the claim of the REST concept is that following those constraints and architectural principles [Hansen, 2007] will result in "an architecture that works well in the areas of scalability, mashup-ability, usability, and accessibility" [Wilde and Pautasso, 2011].

The WS-Agreement specification already defines a SOAP interface for managing the agreement life cycle, including creation, expiration, and monitoring of agreement states. In some cases REST offers better flexibility and control, but "when it comes to enterprise applications though, the reliability and better defined security features of WS-* make SOAP a probably more appropriate solution" [Kübert et al., 2011]. For the purpose of this thesis, the identified components are designed along a simplified version of the RESTful design of the WS-Agreement specification presented in [Kübert et al., 2011]. The major reason for this choice is that REST promises a light-weight solution for

SLA management in SDIs. The SOAP protocol seems to produce some relevant organizational and performance overhead [Mulligan and Gracanin, 2009]. Furthermore, compared to the SOAP protocol, the REST concept seems to be to a greater degree in line with the existing OGC Standards Baseline.

#### AGREEMENT MANAGER

Table 5.1 describes the service interface of the Agreement Manager component.

Table 5.1: Agreement Manager Resources

| Resource [a] | Description |
|---|---|
| `/templates` | With the `GET` method, this URI returns a list of all available templates. The output format is defined by the `TemplateList` element (Listing C.12). |
| `/template/{id}` | With the `GET` method, this URI returns a representation of the template with the unique identifier `{id}`. The output format is defined by the template format as defined in the WS-Agreement Application Profile for OGC Web Services. An example XML document can be found in Listing C.8. |
| `/agreements` | With the `GET` method, this URI returns a list of all available agreements. The output format is defined by the `AgreementList` element (Listing C.12).<br>With the `POST` method, this `URI` allows the creation of a new agreement. The input format is defined by the agreement offer format as defined in the WS-Agreement specification. An example XML document can be found in Listing C.9. |
| `/agreement/{id}` | With the `GET` method, this URI returns a representation of the agreement with the unique identifier `{id}`. The output format is defined by the agreement format as defined in the WS-Agreement Application Profile for OGC Web Services. An example XML document can be found in Listing C.10. |
| `/agreement/{id}/state` | With the `GET` method, this URI returns the state of the agreement with the unique identifier `{id}`. The output format is defined by the agreement properties format as defined in the WS-Agreement specification. An example XML document can be found in Listing C.11. |

Table 5.1 – Continued on next page

Table 5.1 – Continued from previous page

| Resource [a] | Description |
|---|---|
| | With the `POST` method, this URI allows the creation of a resource holding state for the agreement with with the unique identifier `{id}`. The input format is defined by the agreement properties format as defined in the WS-Agreement specification. An example XML document can be found in Listing C.11.<br><br>With the `PUT` method, this URI allows the update of a resource holding state for the agreement with the unique identifier `{id}`. The input format is defined by the agreement properties format as defined in the WS-Agreement specification. An example XML document can be found in Listing C.11. |

[a] In case of successful or invalid requests, all resources produce HTTP Status Codes as defined in [Fielding et al., 1999] (e.g. `201 CREATED` after a new agreement beeing created or `400 BAD REQUEST` for malformed request syntax).

The XSD in Listing C.12 defines the XML output format of the Agreement Manager component. Appendix C.3 describes an example template discovery, agreement creation and agreement monitoring workflow based on the presented Agreement Manager service interface.

### AGREEMENT PROXY

Table 5.2 describes the service interface of the Agreement Proxy component.

Table 5.2: Agreement Proxy Resources

| Resource [a] | Description |
|---|---|
| `/{service}/{agreement}` | This URI, which can be found in the *Service Reference* section of an agreement, references the service with the unique name `{service}` and the corresponding agreement with the unique identifier `{id}`.<br><br>The URI is the main entry point for service consumers and intercepts all types of requests. When a request arrives, the Agreement Proxy checks whether the service consumer is allowed to execute the service with the unique name `{service}` under the terms of the previously created agreement with the unique identifier `{id}`. If the check succeeds, the request is forwarded to the targeted service in order to execute the service consumer request. |

Table 5.2 – Continued on next page

Table 5.2 – Continued from previous page

| Resource | Description |
|---|---|

[a] In case of successful or invalid requests, all resources produce HTTP Status Codes as defined in [Fielding et al., 1999] (e.g. `201 CREATED` after a new agreement beeing created or `400 BAD REQUEST` for malformed request syntax).

The XSD in Listing C.13 defines the XML output format of the Agreement Proxy component. Appendix C.3 describes an example service consumption task based on the presented Agreement Proxy service interface.

## 5.4 Summary

This chapter presents the design of a web-based SLA management architecture. The purpose of this architecture is to enabled the on-demand and online negotiation of SLAs in SDIs without the need of prior offline communication between service providers and service consumers. The web-based SLA management architecture is developed in three different steps.

First, the process model provides a general overview about the processes and tasks that are required for a full integration of SLAs in SDIs. The web-based SLA management architecture covers three phases of the agreement life cycle: agreement negotiation, agreement implementation and agreement execution.

Second, the information model identifies all managed objects of the web-based SLA management architecture on a conceptual level and describes how these objects relate to each other independent of any specific implementation or protocol. The design of the web-based SLA management architecture is inspired by policy-based management systems. The main components of such architectures are the Policy Repository, the Policy Enforcement Point (PEP) and the Policy Decision Point (PDP). The Agreement Manager component implements the Policy Repository. The component is responsible for the management of templates, agreements and monitoring information. The Agreement Proxy component implements the PEP and the PDP. The component acts as a proxy for the original SDI service and enforces agreement terms whenever a service consumer makes a service request under the terms of a previously created agreement. The other components such as the Agreement Evaluator and the Infrastructure Manager realize automated agreement and infrastructure management.

Third, the data model specifies concrete data structures and includes implementation- and protocol-specific details for realizing the information model. The data model mainly consists of the WS-Agreement Application Profile for OGC Web Services, and the service interfaces of the Agreement Manager and the Agreement Proxy. The WS-Agreement Application Profile for OGC Web Services is a mapping of the abstract SLA model to an extended and particular version of the WS-Agreement specification. The application profile consists of a set of XSD that specifies the domain-specific content of an agreement, the OGC URN Schema Extension for identifying domain-specific service properties and business values, and the DSL for defining and evaluating

domain-specific guarantee terms. The Agreement Manager and the Agreement Proxy are the only components in the web-based SLA management architecture that must be public accessible for potential service consumers, their respective SLA clients, and GIS applications. The interfaces of the Agreement Manager and the Agreement Proxy are designed in REST architectural style, which promises a light-weight solution for the management of SLAs and is in line with the OGC Standards Baseline. The other components such as the Agreement Evaluator and the Infrastructure Manager, must not be standardized and can be realized by implementations specific for each service provider.

The next chapter presents and evaluates the abstract SLA model and the web-based SLA management architecture by means of a prototypical implementation.

# Chapter 6

# Implementation and Evaluation

This chapter presents an implementation and demonstration of the abstract SLA model and the web-based SLA management architecture. Based on the implementation, this chapter evaluates the presented concept for the integration of SLAs in SDIs with respect to the objectives and the requirements of this thesis.

## 6.1 Implementation

The implementation of the abstract SLA model and the web-based SLA management architecture mainly consist of the SLA4OWS framework, which implements the WS-Agreement Application Profile for OGC Web Services. Section 6.1.1 describes the SLA4OWS framework and other components that are used to demonstrate the concepts of this thesis. Section 6.1.2 describes how service consumers can search for templates, create agreements and observe the status of agreements. Section 6.1.3 provides an example that demonstrate how SDI service providers can manage their infrastructure by means of an Hybrid Cloud in order to match the INSPIRE service quality requirements.

### 6.1.1 Applications and Resources

The following technologies, applications and resources are used to implement and demonstrate the abstract SLA model and the web-based SLA management framework.

#### SLA4OWS

The Service Level Agreements for OGC Web Services (SLA4OWS) framework [Baranski, 2012] is an Open Source framework for the integration of SLAs in SDIs that are based on standards developed by the OGC. The SLA4OWS framework enables service providers to offer different service quality levels and pricing models for their existing services. The SLA4OWS framework utilizes Cloud Computing infrastructures to implement the offered service levels in an economical fashion. The standards and technologies that have been chosen to implement the SLA4OWS framework do not require changes to other OGC standards or compliant implementations.

The SLA4OWS framework mainly consists of two components. The server implements the WS-Agreement Application Profile for OGC Web Services. The client provides a web-based user interface for template discovery, agreement negotiation and agreement

monitoring. The SLA4OWS framework is based on common Java technologies and provides a pluggable framework for agreement negotiation, web service monitoring and service hosting in Cloud Computing environments. All INSPIRE-related web service properties and the on-demand hosting of SDI services at Amazon EC2 are supported by default. The source code and documentation of the SLA4OWS framework can be found at the project homepage[1].

The SLA4OWS framework was originally developed and evaluated in the SLA4D-Grid project. For the purpose of this thesis, the list of templates that are supported by default was extended by additional templates reflecting the requirements of the four application domains of the abstract scenario. Furthermore, the infrastructure management capabilities have been extended in order to demonstrate how service providers can deliver domain-specific service quality levels.

#### 52°NORTH WPS

The 52°North WPS [Foerster, 2006] is an Open Source implementation of the OGC WPS specification. The 52°North WPS implementation is based on common Java technologies and provides a pluggable framework for algorithms, data handling and processing frameworks. The source code and documentation of the 52°North WPS can be found at the project homepage[2].

For the purpose of this thesis, the list of processes that are supported by default was extended by additional processes such as processes for INSPIRE Coordinate Transformation. Furthermore, the processing framework was extended in order to support the distributed and parallel process execution in Grid Computing environments that are featured with the UNICORE (Uniform Interface to Computing Resources) middleware [Romberg, 1999].

#### GEOSERVER

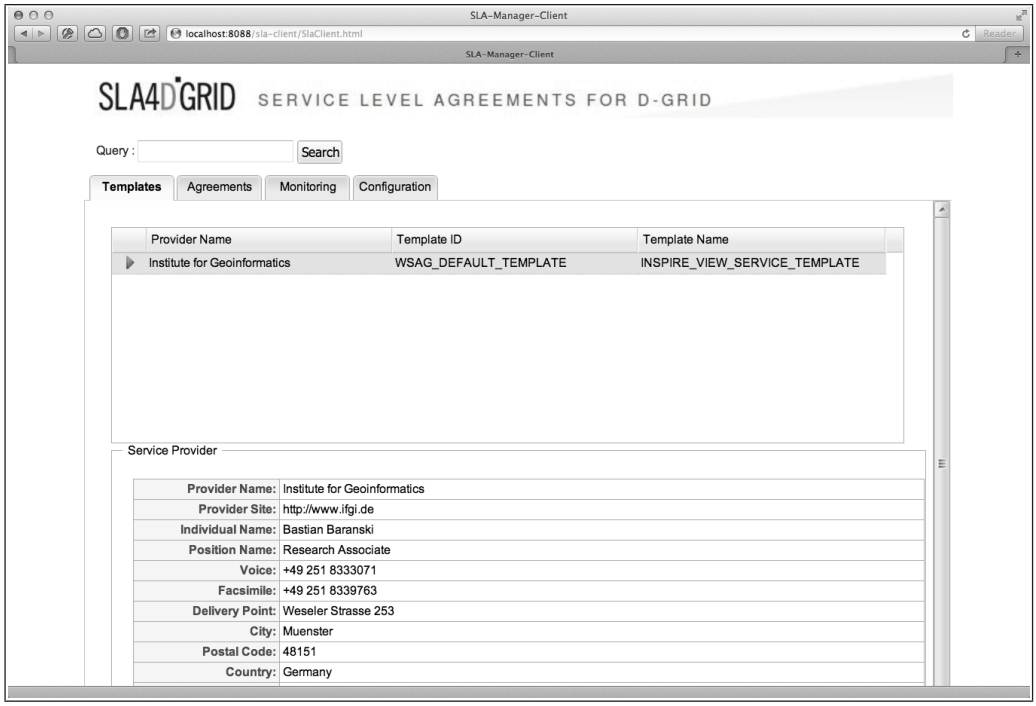
The Geoserver is an Open Source reference implementation of the OGC Web Map Service (WMS), OGC Web Feature Service (WFS) and OGC Web Coverage Service (WCS) specifications. The Geoserver implementation is based on common Java technologies and support many database engines and file formats by default. The OpenLayers JavaScript library for displaying maps in web browsers is integrated as a default preview engine. The source code and documentation of the Geoserver can be found at the project homepage[3].

#### UDIG

The uDig is an Open Source desktop GIS for data access, editing and viewing. The uDig stand-alone application is built with the Eclipse Rich Client Platform (RCP) technology and can easily be extended with RCP plugins. The source code and documentation of uDig can be found at the project homepage[4].

[1] http://www.sla4ows.org
[2] http://52north.org/wps
[3] http://geoserver.org
[4] http://udig.refractions.net

Figure 6.1: Template Discovery




### 6.1.2 Service Level Management

This section exemplifies a template discovery, agreement creation, service consumption and agreement monitoring workflow from the service consumers' perspective. The workflow description demonstrates the web-based user interface of the SLA4OWS framework. An example workflow that directly interacts with the service interface of the Agreement Manager and the Agreement Proxy can be found in Appendix C.3.

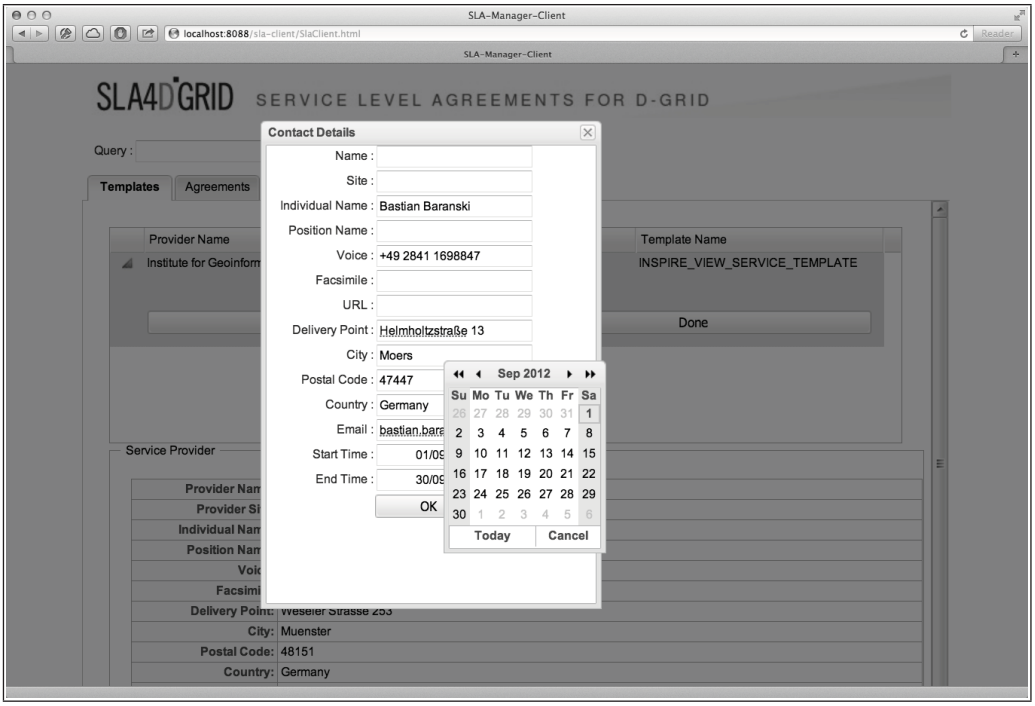
#### TEMPLATE DISCOVERY

To search for services and corresponding templates, the potential service consumer must login to the Agreement Client. The Agreement Client of the SLA4OWS framework is able to integrate multiple Agreement Manager instances at the same time. That enables service consumers to review the service offerings of different service providers at one place. After the service consumer successfully logged in, the Agreement Client presents an overview about all available templates (Figure 6.1). The service consumer is able to look at the template details (e.g. service level objectives and business values) or to search for templates that cover specific service characteristics (e.g. service types or delivered data).

#### AGREEMENT NEGOTIATION

When the service consumer decides to select a template, the next step is the agreement creation. In order to create an agreement, the service consumer must specify the contract period and accept the end use license agreements (Figure 6.2). The user details, which are extracted from the user database of the Agreement Client, can be modified and extended for instance to provide additional management information such as bank

Figure 6.2: Agreement Negotiation




account details. When the service provider finally initiates the agreement creation, the Agreement Client creates an agreement at the Agreement Manager for the particular template. Instantly, all background processes that are essential for the agreement implementation and the agreement execution are started. In case of a successful agreement creation, the service consumer receives a confirmation about the created agreement.

### SERVICE CONSUMPTION

The provided agreement creation confirmation contains an URL pointing to the Agreement Proxy component. The URL comes from the `ServiceReference` element in the WS-Agreement Application Profile for OGC Web Services and enables the service consumption under the terms of the previously created agreement. The Agreement Proxy acts as a proxy and sends all incoming requests to the actual SDI service on behalf of the requesting SDI client. Since the Agreement Proxy is implemented as a non-transparent proxy, the provided URL can be used by any OGC compliant GIS client for the retrieval, portrayal and processing of geospatial data (Figure 6.3 and 6.4).

### AGREEMENT MONITORING

The Agreement Client provides an overview about all created agreements of the service consumer (Figure 6.5). The list overview provides basic agreement information as for instance the service type and the status of service level objectives. By default, the provided URL is public accessible and can be used not only by the service consumer who created the agreement. Some use cases may require more restrictive and limiting access rules. Therefore, the Agreement Client provides basic mechanisms to protect the service

Figure 6.3: Accessing Agreement Proxy

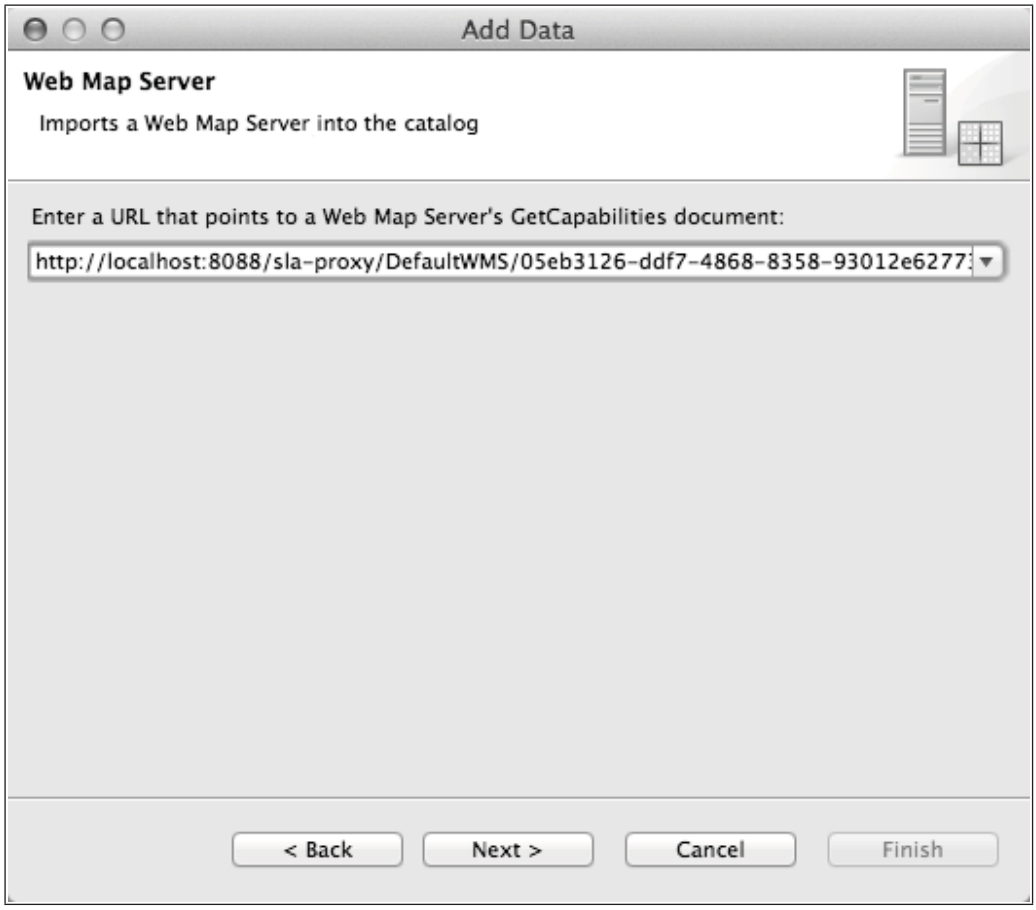



Figure 6.4: Service Consumption

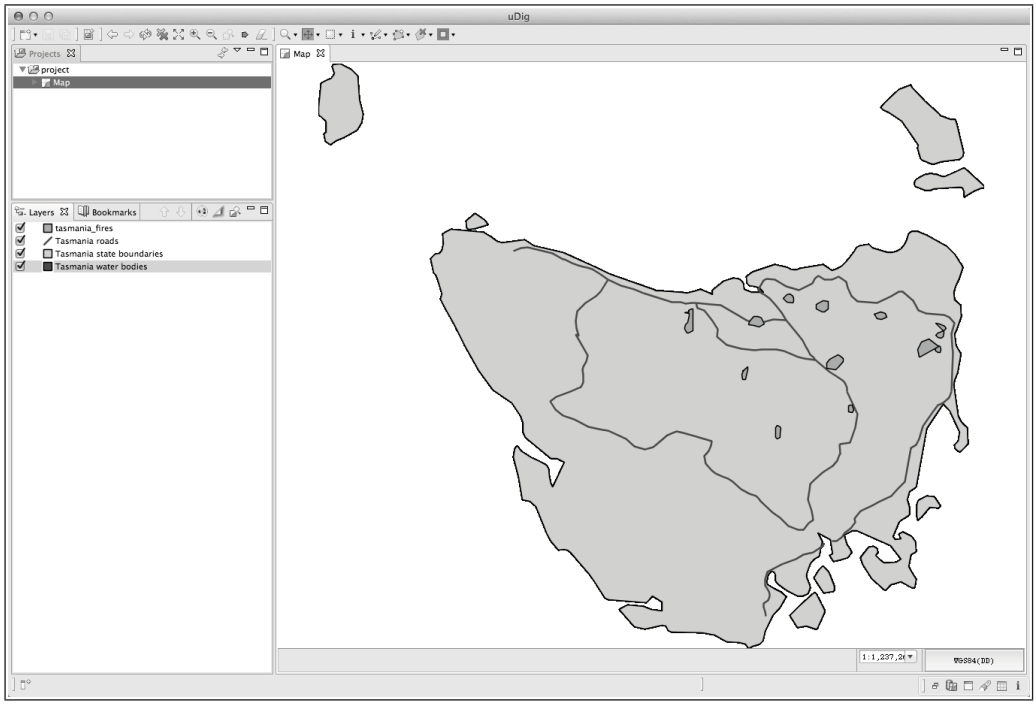


Figure 6.5: Agreement Overview

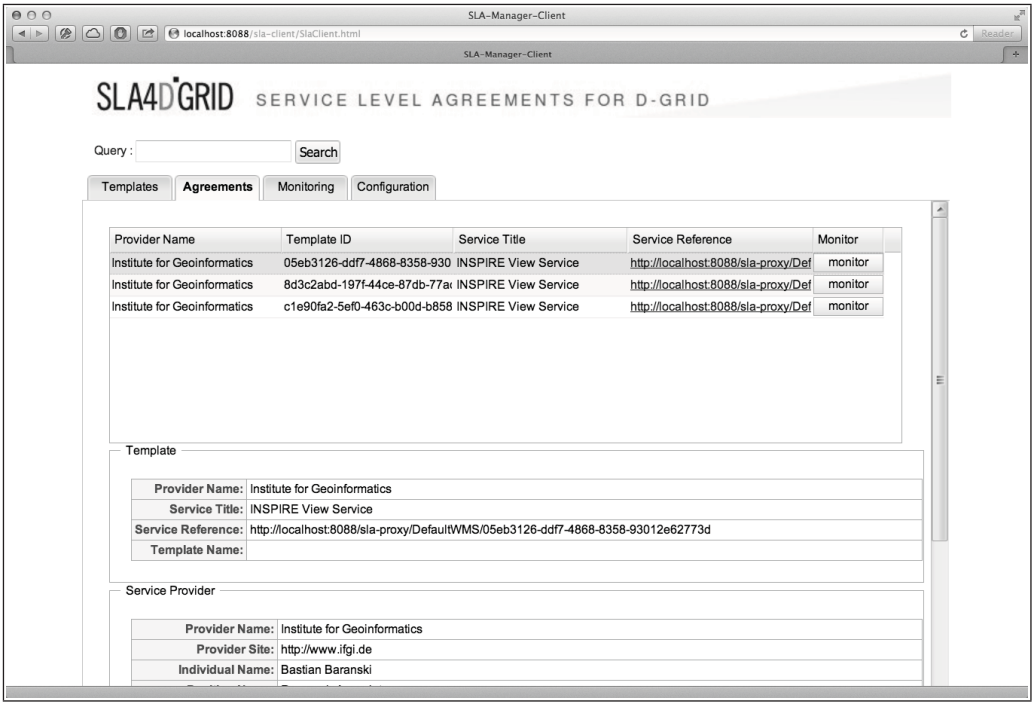
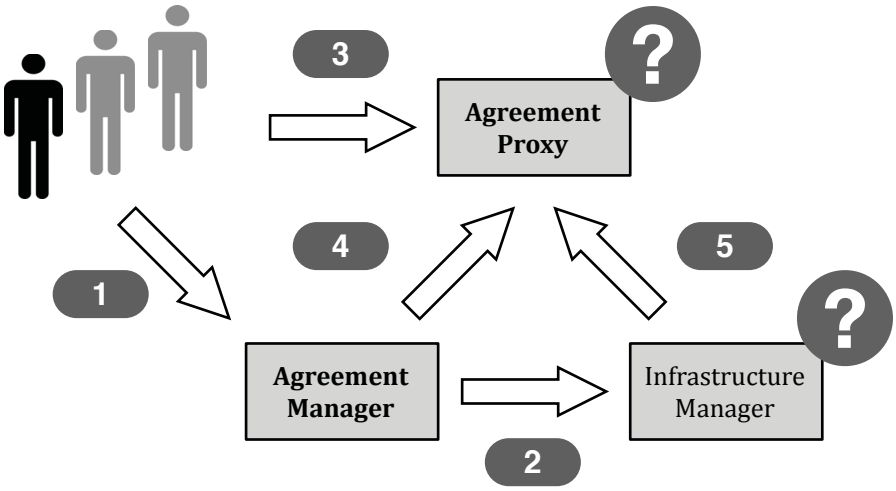



Figure 6.6: Monitoring Information

Figure 6.7: Infrastructure Manager




access against unauthorized access for instance by means of HTTP Authentication. Furthermore, service consumers are able to renew the agreement-specific service reference in order to invalidate previous URLs, which may be in circulation. The Agreement Client also provides a detailed overview about the monitoring history, which is the basis for the evaluation of the service level objectives of an agreement. Figure 6.6 shows for example the measured service response time for the service requests that are defined to evaluate the agreement against the INSPIRE Capacity requirements.

### 6.1.3 Infrastructure Management

The description of the web-based SLA management architecture (Chapter 5) covers the agreement negotiation, the agreement implementation and the agreement execution phases. The complete agreement life cycle and the service consumption have been described explicitly, but the actual infrastructure management was only touched slightly.

Figure 6.7 shows the service consumption from a simplified point of view. When the service consumer makes a successful agreement offer (1), the Agreement Manager notifies the Infrastructure Manager about the created agreement (2). After having created an agreement, the service consumer can access the service through the URL that is provided by the `ServiceReference` element of the created agreement (3). The URL points to the Agreement Proxy, which requests the particular agreement from the Agreement Manager for each incoming request (4). If the agreement and the service request are valid, the Agreement Proxy queries infrastructure-related information from the Infrastructure Manager (5). Based on these infrastructure-related information, the Agreement Proxy forwards the incoming request to the targeted service and returns the potentially altered response to the service consumer.

So the question now is what happens at the Infrastructure Manager when being notified about new agreements? How can the Infrastructure Manager schedule the

infrastructure of the SDI service provider in order to deliver specific service quality levels and realize certain revenue models? What kind of information are returned exactly from the Infrastructure Manager? How can these information be used to forward the request to the targeted service?

The following sections provide an example of how SDI service providers can match the INSPIRE service quality requirements without investing in rarely used hardware in advance by means of an Hybrid Cloud approach. The presented approach considers only stateless SDI services and neglects completely the transfer of big data in distributed infrastructures. Despite these limitations, the presented Hybrid Cloud approach can be an easy to follow and efficient "best practice" to setup scalable and reliable SDI services in order to meet the domain-specific requirements and at the same time also the technical skills of many SDI service providers.

### INSPIRE SERVICE QUALITY

The characteristics and benefits of the Hybrid Cloud approach are presented by means of an INSPIRE Coordinate Transformation Service, which is defined as an application profile of the OGC WPS specification. First implementations of such a service interface are presented in [Kubik and Kopanczyk, 2009] and [Lehto, 2009]. Following the INSPIRE service quality requirements, an INSPIRE Coordinate Transformation Service must be available 99% of the time (availability), the initial response time must be 0.5 seconds per 1 MB of input data (performance) and a service instance must be able to fulfill both of these criteria even if the number of served simultaneous requests is up to 5 per second (capacity).

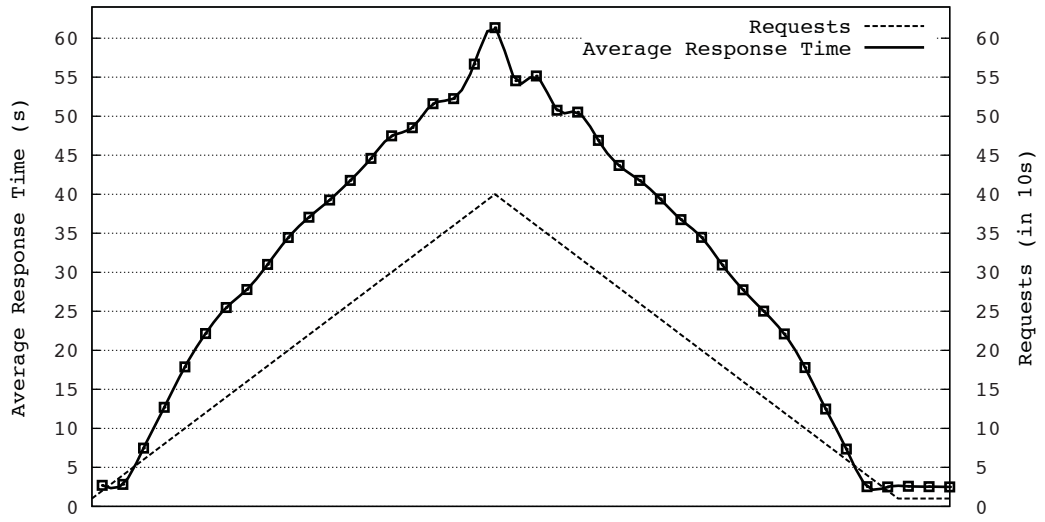
From a service provider perspective, the INSPIRE service quality requirements are very challenging in terms of their implementation. This section introduces the design and the implementation of a Hybrid Cloud architecture that enables SDI service providers to match INSPIRE service quality requirements without investing in rarely used hardware in advance. The proposed Hybrid Cloud was developed by the 52°North Geoprocessing Community[5] and presented to the public at the 2011 Esri International User Conference (UC) [Baranski et al., 2011]. The content of this section is based on ideas, fragments and figures that have appeared previously in [Foerster et al., 2010] and [Baranski et al., 2011].

### Single Server

To understand the limitations of classic service deployments, the scalability of the 52°North WPS implementation was investigated. The 52°North WPS implementation was developed and evaluated in many research projects. The implementation can be seen as a good representative of other well-engineered SDI Open Source software in terms of standard compliance, feature completeness and resource consumption [Garnett, 2011]. For the purpose of the conducted experiment, the 52°North WPS implementation was enhanced by a process that realizes INSPIRE Coordinate Transformation. The WPS was installed on a single-core machine (2.4 GHz, 4 GB RAM) and the web application container (Apache Tomcat 5.5) was configured to use

[5] http://52north.org/communities/geoprocessing

Figure 6.8: Single Server Benchmark




local memory as much as possible, accept incoming HTTP connections as many as possible and create local threads as many as possible. No clustering, load balancing or other advanced distributed computing mechanism were used to make use of multi-core processor or other servers in the network. In the conducted experiment, a randomized `Execute` request (quite simple in terms of submitted geometry) was sent to the service. Figure 6.8 visualizes the average service response time compared to the number of served simultaneous requests.

The benchmark shows that the average response time of all requests increased significantly according to the number of served simultaneous requests. The average response time increased from 2.7 seconds (2 requests in 10 seconds) up to 61.3 seconds (40 requests in 10 seconds). This behavior is not surprising. The pure processing time at the server for a single `Execute` request was approximately 2.5 seconds plus the overhead for receiving the input data and sending the response data. During this time period the average CPU load of the single-core machine was near by 100% most of the time. When 5 requests within 10 seconds were sent to the service uniformly distributed over time, the server was still able to handle all requests one after another without interfering. When more than 5 requests within 10 seconds were sent to the service, the server had to queue arriving requests, or had to pause and restart already processing requests. However, in the conducted experiment the single server was not able to handle more than 5 requests within 10 seconds without increasing the average response time significantly. The average response time increased faster than the number of requests due to a lot of computational overhead (pausing and restarting processes, memory allocation and copying). Furthermore, in nearly all conducted experiments the web services threw internal server errors during peak load (especially XML parsing errors caused by Out of Memory exceptions). Sometimes the whole server crashed under overload and further requests could not be processed anymore.

The observed performance behavior can be improved by using more powerful hardware

or by distributing the workload to more than one server. But these improvements merely shift the general barrier of scalability to another level and in most of the real-world scenarios the additional hardware will have very low utilization rates as for instance shown in [Odlyzko, 1998].

The idea of the proposed Hybrid Cloud architecture is always to provide sufficient computational resources to achieve a constant average service response time, independent of the number of users requesting a service. By incorporating external third-party resources into the local IT infrastructure, the proposed architecture offers potentially unlimited resources on-demand and nearly in real-time. The proposed Hybrid Cloud approach combines the limited hardware resources of a local IT infrastructure (local servers) and potentially unlimited resources at a Public Cloud provider (virtualized servers) in order to realize scalable SDI service in a technical and economical efficient manner.

The following sections describe the architecture and the implementation of the proposed Hybrid Cloud approach. The scalability of the Hybrid Cloud implementation is evaluated against the benchmark of the single server scenario.

## ARCHITECTURE

The Hybrid Cloud architecture is composed of the following components (Figure 6.9):

**Proxy**

The Proxy component is the main entry point for all clients (users, applications, or other services) and it controls the access to the whole (local and third-party) infrastructure. It receives all incoming requests, forwards them to the Load Balancer at the Gateway, and returns the delivered response as if it was itself the origin.

The Gateway is an organizational unit containing a Load Balancer, a Virtual Machine Repository, a Cloud Controller and a Cloud Manager.

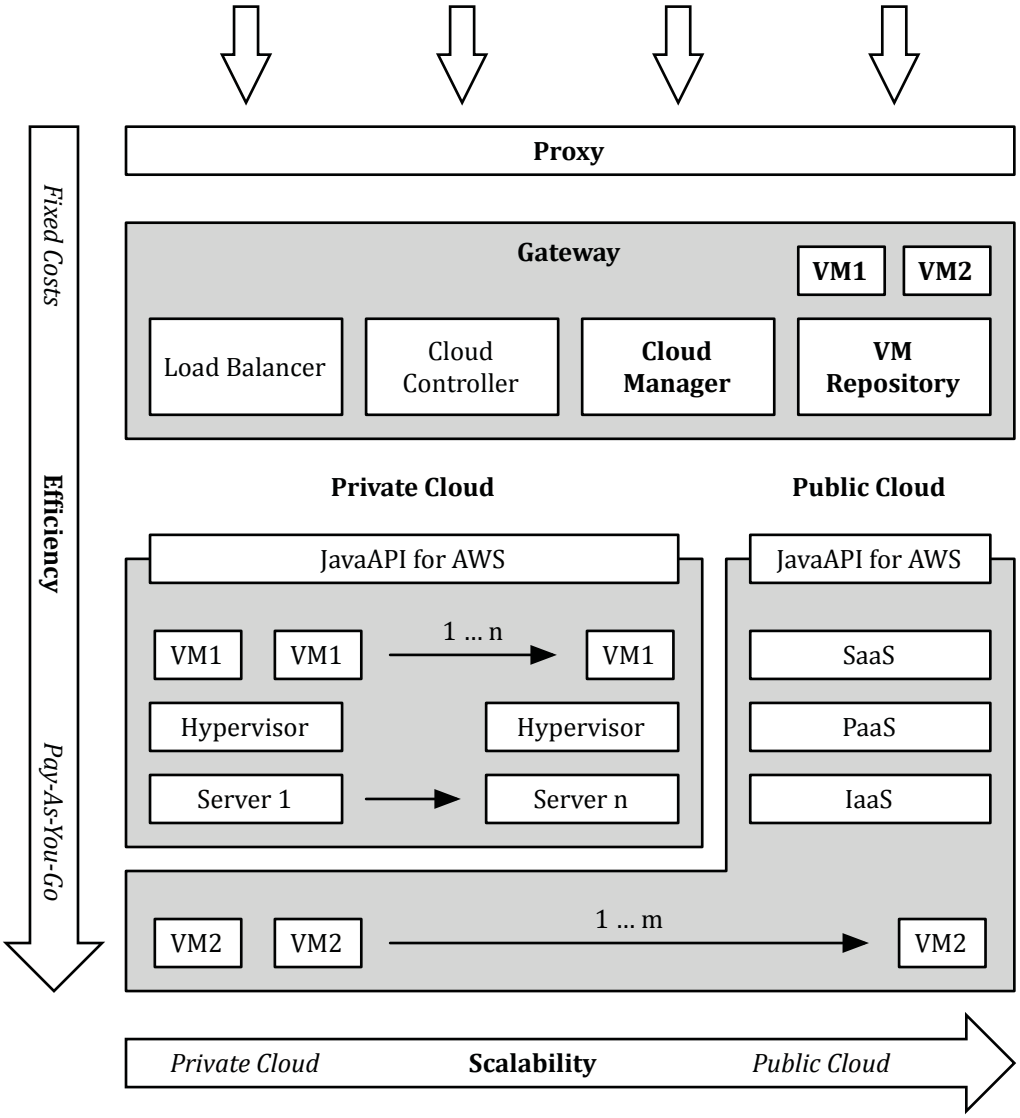
**Load Balancer**

The Load Balancer component receives all forwarded requests from the Proxy and distributes them across all available service instances independent of whether they are in the local or third-party infrastructure.

**Virtual Machine Repository**

The Virtual Machine Repository component realizes a local storage containing one or more VM images. Each VM image belongs to an offered service and contains a guest operating system, all required software components and related configurations. In the proposed Hybrid Cloud architecture two different types of VM images exists. One image type is dedicated for use in the local infrastructure. The other image type is dedicated for use at the Public Cloud. Therefore, to offer a service through the Hybrid Cloud, for each service two image types must be provided at the VM Repository.

Figure 6.9: Hybrid Cloud Architecture



#### Cloud Controller

The Cloud Controller component manages the virtualized local infrastructure by providing an interface for starting and stopping VM instance on the local servers. Therefore, on each of the local servers a host operating system together with a Hypervisor must be installed. The only task for the Hypervisor is to run the guest Operating System (OS).

#### Cloud Manager

The Cloud Manager component monitors the CPU load on each running VM in the architecture. If the overall CPU load of the system goes beyond a configured threshold, the Cloud Manager starts a new VM instance and adds the new running VM to the Internet Protocol (IP) pool of the Load Balancer. In the ideal case, the VM is started via the Cloud Controller at the local infrastructure. In the case that all local servers are busy, the VM is started at the Public Cloud. If the overall CPU load of the system goes below a configured threshold, the Cloud Manager stops the running VM instance with the lowest CPU load (with a priority for running VM instances at the Public Cloud in order to reduce external costs). Before the Cloud Manager stops a running VM, the VM is removed from the IP pool of the Load Balancer.

Each time a new VM instance is added/removed from the IP pool of the Load Balancer, the Load Balancer must be restarted for technical reasons to notice the new resources. To avoid connection interruptions between the Proxy and the requesting clients (in the case the Load Balancer is not available for a short period of time), the Proxy component re-sends the forwarded requests to the Load Balancer until they could be processed successfully.

### IMPLEMENTATION

The Hybrid Cloud architecture was implemented as a proof of concept. Most of the components have been realized by means of common Open Source software packages. The Proxy component was realized through an Apache HTTP Server[6] combined with the mod_proxy[7] module. The Apache HTTP Server was configured to act as a reverse proxy and therefore appeared to requesting clients just like an ordinary web server. The Load Balancer was realized through the nginx[8] web server that was also configured to act as a reverse proxy. The Kernel-based Virtual Machine (KVM)[9] hypervisor was used to realize virtualization in the local infrastructure and Amazon EC2 was selected to be the Public Cloud provider. Therefore, the Virtual Machine Repository must contain a KVM image and an Amazon Machine Image (AMI) image for each of the offered services. Eucalyptus is a software package for implementing Hybrid Cloud infrastructures using the Amazon Web Services (AWS) API. The Cloud Controller component used the Eucalyptus Cloud Controller (CLC), the Eucalyptus Storage Controller (SC) and the

[6] http://httpd.apache.org
[7] http://httpd.apache.org/docs/2.4/mod/mod_proxy.html
[8] http://wiki.nginx.org/Main
[9] http://www.linux-kvm.org

Eucalyptus Cluster Controller (CC) to manage the virtualized local infrastructure. The local servers run an Ubuntu 9.10 operating system, a KVM hypervisor and the Eucalyptus Node Controller (NC) to offer a cloud abstraction layer at each local server.

The Cloud Manager component was developed from scratch to suit the INSPIRE service quality requirements. It was implemented in Java and is publicly available as Open Source in the 52°North source repository[10]. The Cloud Manager is the core component of the proposed Hybrid Cloud solution and provides the scalability of the system either by starting/stopping VM instances in the local infrastructure or at the Public Cloud. The Cloud Manager configuration affects the dynamic behavior and therefore the scalability of the overall system (e.g. how fast new VM instances are available in case of high user demands). Since the resources at the Public Cloud are allocated based on a pay-as-you-go manner, the configuration also affects the financial efficiency of the proposed Hybrid Cloud (e.g. how much the service provider has to pay for a more dynamic scalability).

Table 6.1 shows the default configuration parameters of the Cloud Manager.

Table 6.1: Cloud Manager Configuration

| Parameter | Value | Description |
|---|---|---|
| `Breach Duration` | `15s` | This parameter describes the period after which VM instances are stopped when the lower threshold is reached. |
| `Period` | `5s` | This parameter describes the interval for monitoring the CPU load of each running VM. |
| `Upper Threshold` | `20%` | This parameter describes the upper CPU load threshold that is relevant for starting new VM instances. |
| `Lower Threshold` | `10%` | This parameter describes the lower CPU load threshold that is relevant for stopping running VM instances. |
| `Statistics` | `"average"` | This parameter describes how the monitored CPU load history influences the calculation whether the upper/lower threshold is reached. Possible values are “minimum”, “maximum” and “average”. |
| `Maximum Cloud Instances` | `6` | This parameter describes how many Amazon EC2 instances can be started by the Cloud Manager. |

[10] https://svn.52north.org/svn/studentscorner/CloudFramework

### BENCHMARK

There are several Open Source tools available for performing web server benchmarks or load testing as for instance Apache JMeter[11] or ApacheBench[12]. To exactly control the amount of workload that was sent to a web service in a specific period of time (the number of parallel requests) and to control the logged benchmark indicators, a simple benchmark tool tailored to the particular requirements of the intended INSPIRE scenario was developed. The tailored benchmark tool was implemented in Java and it is also publicly available as Open Source in the 52°North source repository[13]. The benchmark tool sends a specific number of requests per sequence to a web service and slightly increases/decreases the number of requests per sequence over time.

In the performed benchmarks the Private Cloud consisted of up to 4 virtual machine instances, based on 2 servers with multi-core processors containing 2 cores each. The underlying virtualized hardware acquired at Amazon EC2 is unknown and only the family of instance types is known. In all benchmarks where the Public Cloud was activated, the Hybrid Cloud was configured to start up to 6 instances at Amazon EC2 from the "Small Instance" family. To get representative results, all of the experiments were repeated several times with both the same and slightly different benchmark configurations. Relevant deviations from normal behavior are described explicitly.

#### Private Cloud

The benchmark for the Private Cloud was performed without allowing the Cloud Manager to acquire additional resources at Amazon EC2. Therefore, the scalability of a Cloud-managed internal data center is reviewed. The visualized average response time of a request sent to the Private Cloud in comparison to the local hardware utilization rate is depicted in Figure 6.10.

The benchmark shows that the average response time of all requests that were sent to the INSPIRE Coordinate Transformation Service that was deployed in the Private Cloud still increased according to the number of served simultaneous requests. The average response time increased from 1.6 seconds (4 requests in 10 seconds) up to 5.9 seconds (40 requests in 10 seconds). Compared to the benchmark results of the Single Server deployment, this is a significant improvement. Even in times of peak load, the average response time is at maximum by a factor 3.6 higher than in idle times, compared to a maximum of 22.7 times higher than in idle times for the Single Server deployment. Furthermore, in none of the conducted experiments the web services threw internal server errors or crashed.

#### Hybrid Cloud

The benchmark for the complete Hybrid Cloud was performed with allowing the Cloud Manager to use all 4 local instances (Private Cloud) and to acquire up to 6 additional VM instances at Amazon EC2 (Public Cloud). Therefore, the scalability of the complete Hybrid Cloud architecture is reviewed. The average response time of a request sent to

[11] http://jmeter.apache.org

[12] http://httpd.apache.org/docs/2.0/programs/ab.html

[13] https://svn.52north.org/svn/studentscorner/Cloud/monitoring-app

Figure 6.10: Private Cloud Benchmark

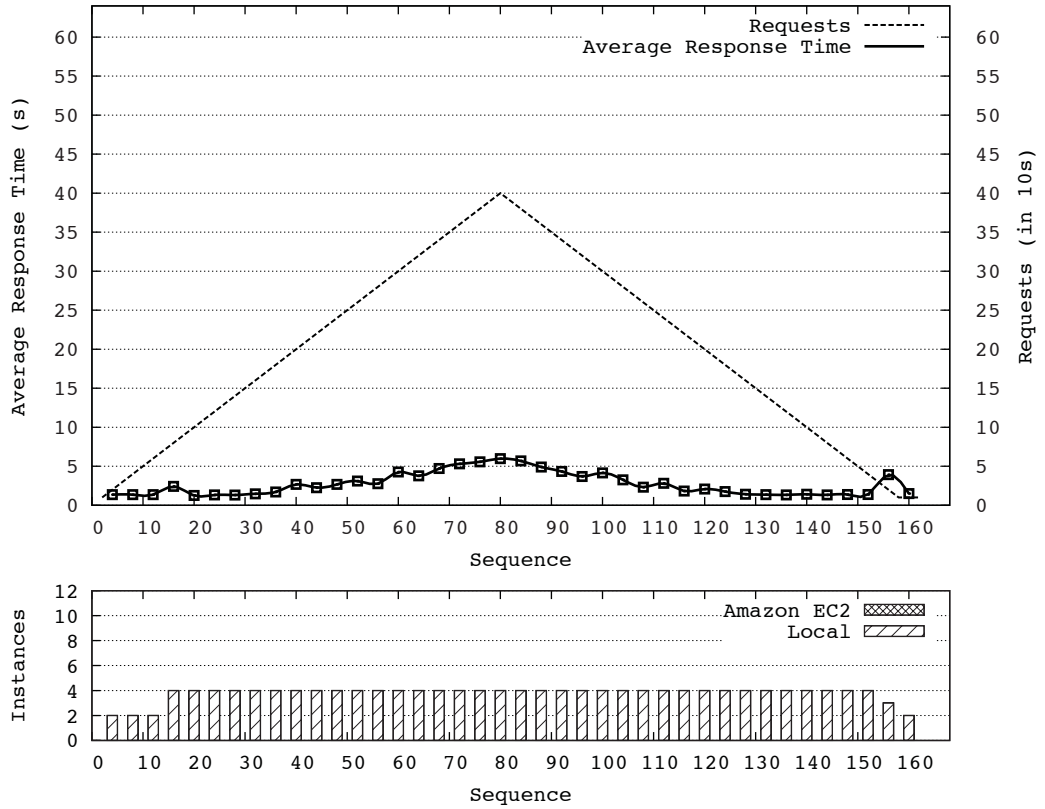



Figure 6.11: Hybrid Cloud Benchmark

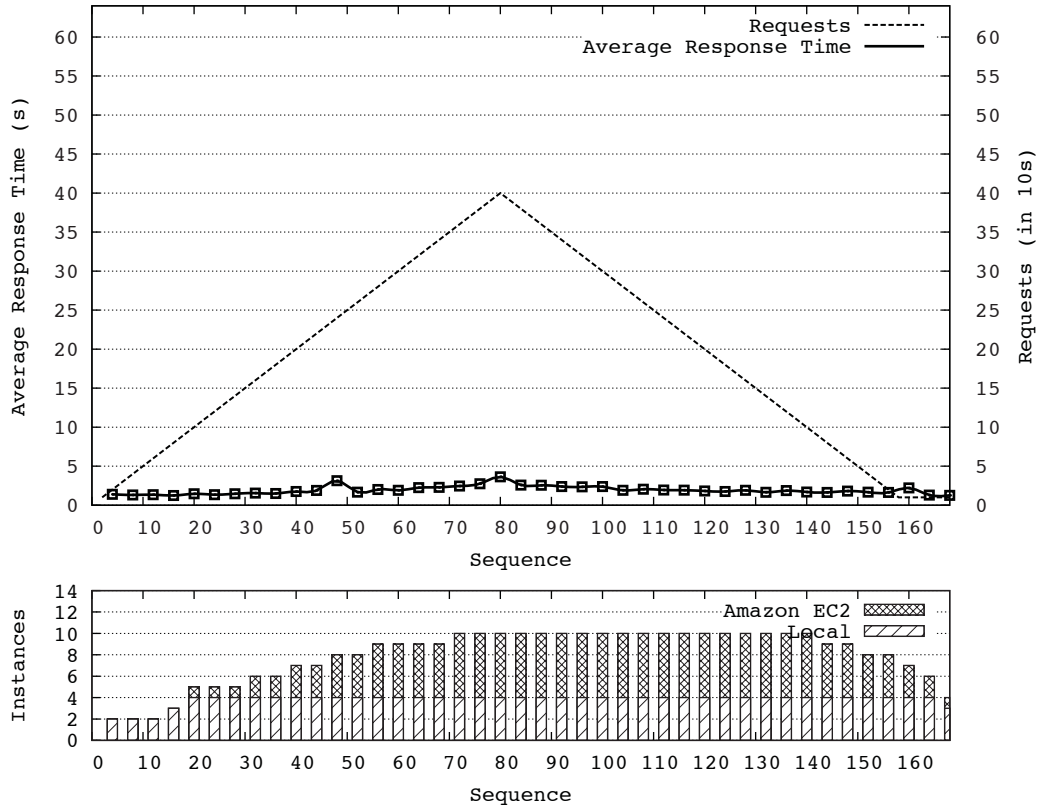
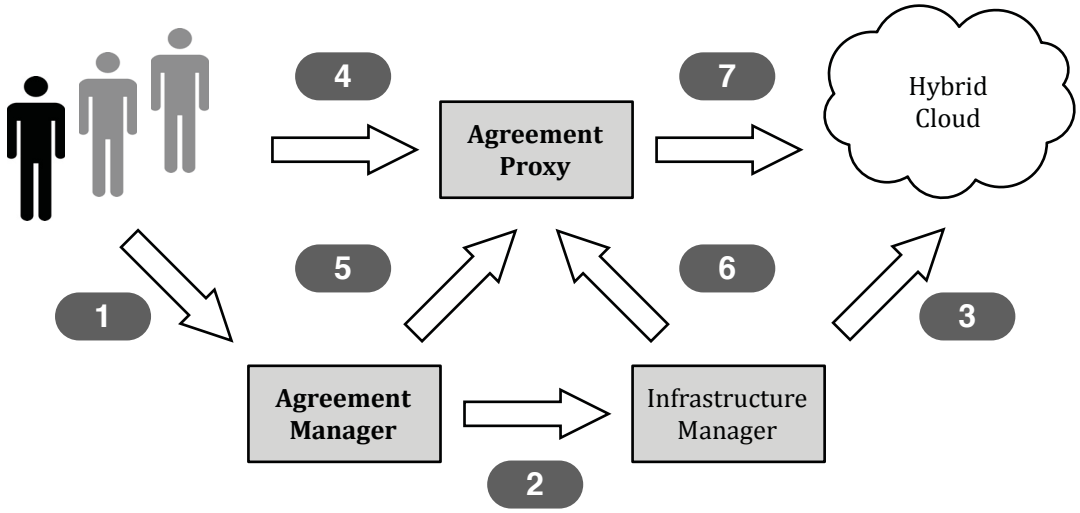


Figure 6.12: Hybrid Cloud Integration




the Hybrid Cloud in comparison to the local and third-party hardware utilization rate is shown in Figure 6.11.

The benchmark shows that the average response time of all requests sent to the INSPIRE Coordinate Transformation Service deployed in the Hybrid Cloud still slightly increases but stays nearly constant independent of the number of served simultaneous requests. The average response time increased from 1.7 seconds (4 requests in 10 seconds) up to 3.3 seconds (40 requests in 10 seconds). Compared to the benchmark results of the Single Server deployment and also compared to the benchmark results of the Private Cloud deployment, this is again a significant improvement. Even in times of peak load, the average response time is maximum 1.9 times higher than in idle times (compared to 22.7 times for the Single Server deployment and 3.6 times for the Private Cloud deployment). Furthermore, in none of the conducted experiments the web services threw internal server errors or crashed.

### INTEGRATION

The presented Hybrid Cloud architecture and implementation can be used as a standalone solution for realizing reliable and scalable web services. Figure 6.12 shows how the Hybrid Cloud can be integrated in the web-based SLA management architecture in order to provide particular service levels under the terms of previously created agreements.

When the service consumer makes a successful agreement offer (1), the Agreement Manager notifies the Infrastructure Manager about the created agreement (2). The Infrastructure Manager immediately sends all infrastructure-related information of the agreement to the Hybrid Cloud (3). The abstract SLA model and the derived WS-Agreement Application Profile for OGC Web Services enable service providers to define such non-varying infrastructure management information in the *Service Properties* section. Listing 6.1 shows how to define infrastructure management information such

as the (internal) name of the infrastructure provider and the (internal) name of the VM image template that provides the designated service. The Cloud Manager component of the Hybrid Cloud transforms these information to the internal Cloud Manager configuration parameters (Table 6.1) and schedules all activities that are required to deliver the service in the intended service quality.

Listing 6.1: Infrastructure Management Information

```
"Service Properties":
{
  (...)
  "Service Property":
  {
    "Name": "provider",
    "Title": "Infrastructure Provider",
    "Abstract": "The name of the infrastructure provider.",
    "Type": "urn:ogc:def:sla:property:infrastructure:provider:name",
    "Value": "MyHybridCloud"
  },
  "Service Property":
  {
    "Name": "image",
    "Title": "template",
    "Abstract": "The name of the VM template.",
    "Type": "urn:ogc:def:sla:property:infrastructure:vm:name",
    "Value": "MyImage"
  }
}
```

After having created an agreement, the service consumer can access the requested service through the URL that is provided by the `Service Reference` element of the created agreement (4). The URL points to the Agreement Proxy, which requests the created agreement from the Agreement Manager for each incoming service request (5). If the agreement and the service request are valid, the Agreement Proxy requests an IP address from the Infrastructure Manager in order to forward the incoming request to the actual service and to return the (altered) response to the service consumer (7). In the meantime, the Cloud Manager started one or more VM instances in order to fulfill the service quality guarantees from the previously created agreement. Therefore, the Infrastructure Manager receives the requested IP address from the IP pool that is managed by the Cloud Manager for the specific service instance (6).

## 6.2 Evaluation

The main goal of this thesis was to develop a concept for the integration of SLAs in SDIs with a multi-step approach. First, an abstract SLA model was developed in order to document the service quality expectations and the conditions of service delivery. Second, a web-based SLA management architecture was developed in order to enable the on-demand and online negotiation of SLAs in SDIs without the need of prior offline communication between service providers and service consumers. Third, an Hybrid Cloud architecture was developed as a best practice for service providers to match the INSPIRE service quality requirements.

The general applicability of the presented concepts has been validated by the prototypical implementation of the SLA4OWS framework (Section 6.1.1) and the Hybrid

Cloud (Section 6.1.3). The following sections evaluate the abstract SLA model and the web-based SLA management architecture against the objectives (Section 1.2) and the requirements (Section 3.2) of this thesis. Furthermore, the general advantages and the limitations of the concepts are described in order to assess the general fitness for use of the presented concepts in other real world applications.

### 6.2.1 Agreement Model

The abstract SLA model (Chapter 4) is composed of a description of the domain-specific structure and content of agreements (Section 4.1), an OGC URN Schema Extension to identify particular service properties and business values (Section 4.3.1), and an Agreement Expression Language for evaluating the status of service level objectives and for calculating business values (Section 4.3.2).

The *Service Description* section of the abstract SLA model provides domain-specific information about the services to which an agreement is related. The content and structure of the *Service Description* section reuse many elements from the OGC Standards Baseline and therefore allow to describe service offerings reflecting common SDI service types as for instance standardized by the OGC and described by the INSPIRE directive (R6). The *Service Properties* section of the abstract SLA model provides a set of domain-specific and exposed properties associated with the service. The OGC URN Schema Extension allows to define service properties that can be automatically monitored during agreement runtime and that reflect available service operations (R7) and spatial data themes (R9), measured web service performance (R12), and logged service usage (R14) and infrastructure utilization (R20). Furthermore, the OGC URN Schema Extension allows to define other import aspects of the service delivery such as data quality (R10) and data licenses (R11), access rights policies (R16), and infrastructure management information (R18, R22). The context variables in the Agreement Expression Language correspond to these service property categories. They provide access either to the raw monitoring data or to pre-processed high-level service metrics. That allows to define concrete service level objectives in the *Service Level Objectives* section of the abstract SLA model reflecting the functional and non-functional service aspects. Furthermore, that allows to define complex pricing models in the *Business Values* section of the abstract SLA model reflecting the service usage and the infrastructure utilization.

In order to facilitate interoperability and create an incentive for adopting the proposed concepts, a standardized document encoding format for the abstract SLA model should be developed (R25) with respect to existing OGC standards (R24). Furthermore, the standardized document encoding format for the abstract SLA model should be developed in a flexible way in order to be applicable in other application domains and use cases (R27). The developed WS-Agreement Application Profile for OGC Web Services (Section 5.3.1) provides such a standardized document encoding format that incorporates many elements from the OGC Standards Baseline. Furthermore, the approach for identifying service properties and business values by means of the OGC URN Schema Extension and the Agreement Expression Language can easily be extended in order to match the requirements of future application domains and use cases.

### 6.2.2 Management Architecture

The design of the web-based SLA management architecture (Chapter 5) follows the model of policy-based management systems. The Agreement Manager implements the Policy Repository, which allows service providers to advertise their services along with SLA metadata (R3), and service consumers to perform service discovery operations in order to find an adequate service according to individual functional and non-functional requirements (R4). Furthermore, the Agreement Manager provides additional functionality for the on-demand and online negotiation of SLAs without the need of prior offline communication between service providers and service consumers (R1). The selected policy-based approach and the design of the Agreement Proxy ensure that the general publish-find-bind pattern in SDIs remains (R2) and that existing SDI client and server implementations don't need to be modified (R24). The Agreement Proxy acts as a (non-transparent) proxy for the original SDI service and implements the PEP and the PDP in one single component. The Agreement Proxy makes sure that service consumption is performed only under the terms of previously created agreements (R5). Furthermore, the Agreement Proxy is responsible for the enforcement of access rights policies once they are defined in an agreement (R17).

In addition to these components that are elementary for policy-based systems, the web-based SLA management architecture identifies several other components for realizing the agreement life cycle. The Agreement Monitor keeps track of all created agreements and calculates the values of all relevant service properties from an agreement. Depending on the service property type, which is defined through the OGC URN Schema Extension, the Agreement Monitor uses active or passive monitoring mechanisms to receive the raw monitoring data and calculate derived high-level service metrics. The Agreement Evaluator keeps track of all created agreements and evaluates the service level objectives and calculates the business values of an agreement. Based on the monitoring information of the Agreement Monitor, the Agreement Evaluator creates a set of context variables corresponding to the service properties of an agreement, and executes the particular Agreement Expression Language script within this context in order to evaluate service level objectives or to calculate business values. This mechanism realizes the monitoring of KPIs and the evaluation of SLOs reflecting the functional (R8) and non-functional (R13) requirements as for instance defined by the INSPIRE directive. Furthermore, this mechanism realizes the monitoring of KPIs and the evaluation of SLOs reflecting infrastructure requirements of service consumers and infrastructure capabilities of service providers (R19). This mechanism also allows the accounting of complex pricing models reflecting the service usage (R15) and the infrastructure utilization (R21). Finally, the Infrastructure Manager is able to forward infrastructure management information for instance to the Hybrid Cloud in order to offer differentiated services under the terms of previously created agreements (R23).

The WS-Agreement Application Profile for OGC Web Services and the standardized interfaces of the Agreement Manager and the Agreement Proxy (Section 5.3.2) ensure the interoperable agreement negotiation and service consumption above service providers' boundaries (R26). The interfaces of internal components for agreement monitoring, agreement evaluation, agreement reporting and infrastructure management don't need to be standardized since they are not accessed directly by

service consumers. Any service provider is free to develop proprietary solutions for internal agreement management tasks. The REST communication protocol developed for the web-based SLA model is quite simple in terms of methods and resources implemented. It can easily be extended in order to match the requirements of future application domains and use cases (R27).

### 6.2.3 Advantages and Limitations

The following paragraphs provide an overview of the main advantages and limitations of the presented concepts and implementations.

#### ADVANTAGES OF THE APPROACH

The abstract SLA model and the WS-Agreement Application Profile for OGC Web Services are straightforward approaches for defining SLA templates. The SLA4OWS framework implements the web-based SLA management architecture and can easily be deployed in the service providers' infrastructure in order to attach SLA templates to existing and already productive running SDI services. The SLA4OWS framework instantly enables service providers to monitor and report the status of their services. The SLA4OWS framework also enables service providers to promote sustainable business models for their services, and service consumers to search for services that match their individual functional and non-functional requirements. These advantages are outstanding in particular with regard to the fact that service providers and service consumers don't have to make any modifications to their server and client implementations.

The WS-Agreement Application Profile for OGC Web Services and the SLA4OWS framework can easily be extended with new features. The OGC URN Schema Extension can be extended in order to identify and define service characteristics that are currently not covered by this thesis. The pluggable SLA4OWS framework also can easily be extended in order to use these service characteristics for active and passive monitoring, evaluation of service level objectives, and calculation of business values. The functionality that is provided by the agreement execution language is quite simple but sufficient to realize even complex service level objectives and business values.

The Hybrid Cloud is realized with common Open Source software and can easily be deployed and maintained. The combination of local infrastructure (Private Cloud) and resources of third-party providers (Public Cloud) enables SDI service providers to realize particular service quality levels for individual users or in an application context under the terms of previously created agreements in an economical fashion. The availability of information about the infrastructure utilization in the Agreement Expression Language allows to realize sustainable business models by means of revenue models that forward infrastructure costs to service consumers.

#### LIMITATIONS OF THE APPROACH

The abstract SLA model and the WS-Agreement Application Profile for OGC Web Services are designed to match the domain-specific requirements of the abstract scenario in which SDI services are utilized for different purposes at the same time. On a

conceptual level, they are both highly specialized for their specific purpose and partially ignore aspects that might be important in other concrete use cases. With the current model it is not possible, for instance, to define human-related service characteristics or more complex monitoring functions. In some cases an agreement shall define for instance the time period for free phone support, in others it may be important to define an increasing/decreasing number of monitoring requests over time in order to simulate changing user behavior, or to check whether a monitoring response is an empty (white) image or not. These limitations can be removed by enhancing the abstract SLA model and the web-based SLA management architecture. As mentioned earlier, the OGC URN Schema Extension and the XML Schema of the WS-Agreement Application Profile for OGC Web Services can easily be extended with such new definitions. Furthermore, the open and pluggable architecture of the SLA4OWS framework allows to easily adopt these and other use case specific developments.

On a conceptual and implementation level, the Agreement Proxy component is a performance bottleneck that limits the capacity and scalability of the overall system. The Agreement Proxy must handle the complete user traffic, which can be very large in terms of number of requests and delivered geospatial data. Furthermore, the Agreement Proxy must implement the PEP and PDP mechanisms, which can also be time-consuming and which are not to be neglected. Advanced methods and technologies such as multithreading programming [Prasad, 1996] and native programming languages [Prechelt, 2000] can be used to improve the general performance not only of the Agreement Proxy but also of all other components of the SLA4OWS framework. The general scalability of the Agreement Proxy component can be increased by means of classic load balancing mechanisms over many parallel Agreement Proxy instances.

## 6.3 Summary

This chapter presents an implementation of the abstract SLA model and the web-based SLA management architecture. Based on the implementation, this chapter evaluates the advantages and limitations of the concept for the integration of SLAs in SDIs with respect to the objectives and the requirements of this thesis.

The abstract SLA model allows to describe any type of OGC service and to define common SDI service quality requirements. Furthermore, it is possible to define complex pricing models in order to charge the service consumer for service usage or infrastructure utilization. The web-based SLA management architecture enhance the general publish-find-bind pattern in SDIs with an additional 'agree' phase. The 'agree' phase allows to publish services along with additional SLA related information, to create SLAs for specific services and to perform service consumption under the terms of previously created agreements. Furthermore, the web-based SLA management architecture permanently monitors the service health and evaluates the overall status of all available SLAs. The data encodings and service interfaces chosen for the abstract SLA model and the web-based SLA management architecture can easily be extended in order to match the requirements in future application scenarios.

The SLA4OWS framework implements the WS-Agreement Application Profile for OGC

Web Services and enables SDI service providers to offer different service quality levels and pricing models for their existing SDI services. The chosen standards and technologies do not require changes to other OGC standards or compliant implementations. The SLA4OWS framework utilizes Cloud Computing infrastructures to implement offered service levels in an economical fashion. The 52°North Hybrid Cloud helps SDI service providers to match basic INSPIRE service quality requirements without investing in rarely used hardware in advance by means of an Hybrid Cloud approach. The idea of the Hybrid Cloud approach is always to provide sufficient computational resources to achieve a constant average service response time independent of the number of users requesting a service. By incorporating resources from Amazon EC2 into the local IT infrastructure, the architecture offers potentially unlimited resources on-demand and nearly in real-time. The Hybrid Cloud combines the limited hardware resources of a local IT infrastructure (local servers) and potentially unlimited resources of, a Public Cloud provider (virtualized servers) in order to realize scalable SDI services in a technical and economical efficient manner. Both implementations are based on common Open Source software and can easily be deployed and maintained. The implementations can easily be extended with new features, such as the monitoring of unregarded service characteristics and the connection to existing accounting systems.

The next chapter summarizes the conducted research, highlights the contribution of this thesis and provides directions for future research.

# Chapter 7

# Conclusion and Outlook

This chapter provides the answers to the research questions and describes the major contribution of this thesis. Furthermore, directions for future research are provided.

## 7.1 Research Questions

The main research question of this thesis has been divided into several sub-questions, which are answered in the following.

*a) What are the domain-specific requirements for the integration of SLAs in SDIs?*

This research question aims at the evaluation of the domain-specific requirements of different SDI stakeholders for the integration of SLAs in SDIs.

The major stakeholders in the SDI development are central or local governmental organizations, the commercial sector, non-governmental organizations, or the academic sector. For the purpose of this thesis, it is assumed that governmental SDI service providers offer SDI services that are used in more than one application domain at the same time. These application domains can be for instance eGovernment applications, legal frameworks, commercially available product solutions and disaster management. Some of the application domains may have challenging requirements regarding the service quality, while others may have lower requirements. In all application domains it is important for service providers to be aware of the minimum service quality that shall be delivered to individual customers or in an application context. At the same time it is also important for service providers to actually be in a position to adjust the delivered service quality automatically according to the individual service consumers' requirements or the application context. All these aspects can be summarized under the heading of SLM and SLA, which is an integral part of SLM.

Section 3.1 describes four different application domains and their particular service level characteristics and business conditions. Section 3.2 derives the requirements for the integration of SLAs in SDIs from these application domains. The domain-specific requirements cover aspects such as the functional and non-function description of geospatial services and datasets, the definition of dynamic pricing and accounting models for the offline and online delivery of geospatial data,

the enforcement of access rights policies for geospatial data and services, the management of (virtualized) hardware infrastructure, and the general integration of the SLA concept in the OGC Standard Baseline.

The transformation of these domain-specific requirements into an abstract SLA model is part of the next research question.

b) *What is the domain-specific content of SLAs that can be applied in SDIs?*

This research question aims at the development of the abstract SLA model and the WS-Agreement Application Profile for OGC Web Services, which is an implementation of the abstract SLA model.

Chapter 4 formalizes an abstract SLA model that is applicable in SDIs. The aim of the abstract SLA model is to provide an abstract representation of the agreement on a conceptual level, which is independent of any specific data encoding format. Section 4.1 describes general information about the agreement and the contracting parties, domain-specific information about the services to which an agreement is related, a domain-specific reference to an actual service, and a set of domain-specific service properties that are used to measure the service quality and that should be monitored during agreement runtime. Furthermore, the agreement specifies domain-specific service quality goals that the contracting parties are agreeing and general business related properties such as usage costs and penalty fees. Section 4.2 describes how to monitor domain-specific service properties. Section 4.3 describes how to evaluate the status of service level objectives and how to calculate certain business values. All these agreement details are structured in accordance with the evaluated requirements.

Section 5.3.1 describes the WS-Agreement Application Profile for OGC Web Services, a mapping of the abstract SLA model into an extended and particular version of the WS-Agreement specification. The WS-Agreement Application Profile for OGC Web Services mainly consists of a set of XSD that specifies the domain-specific content of an agreement, an URN namespace for identifying domain-specific service properties and business values, and a DSL for defining and evaluating domain-specific guarantee terms.

The conceptual and technical development of an abstract SLA model is one important aspect. The actual integration of SLAs in the SDI workflow is part of the next research question.

c) *How can service consumers and service providers negotiate SLAs for SDI services?*

This research question aims at the development of the web-based SLA management architecture and the SLA4OWS framework, which is an implementation of the web-based SLA management architecture.

Chapter 5 presents the design of a web-based SLA management architecture that is inspired by policy-based management systems. The main components

of the architecture are the Agreement Manager and the Agreement Proxy. The Agreement Manager is responsible for the management of templates, agreements and monitoring information. The optional Agreement Client enables service consumers to easily obtain agreements for specific services, to get a detailed overview about the elements in an agreement and to keep track of the agreement status during the whole agreement life cycle. The Agreement Proxy acts as a proxy for the original service and determines whether all agreement constraints are fulfilled or not. If all agreement constraints are fulfilled, the Agreement Proxy forwards service requests to the target service and returns the service response back to the service consumer. The Agreement Manager enhances the classic publish-find-bind pattern by an additional 'agree' phase in which the service consumer and the service provider agree to certain SLA terms prior the service consumption. The Agreement Proxy enforces that service consumption is performed only under the terms of previously created agreements. The Agreement Manager and the Agreement Proxy are the only components in the web-based SLA management architecture that are public accessible for potential service consumers and their respective SLA clients and GIS applications. Therefore, standard service interfaces for the Agreement Manager and the Agreement Proxy are proposed in order to maintain and foster interoperability.

Section 6.1.1 presents the SLA4OWS framework. The SLA4OWS framework is an Open Source implementation of the WS-Agreement Application Profile for OGC Web Services, including standardized document encodings and service interfaces. The chosen standards and technologies do not require changes to other OGC standards or compliant implementations. The provided solution can easily be integrated in any SDI environment in order to provide agreement negotiation for existing SDI services.

The web-based SLA management architecture must also be able to handle the agreement implementation and agreement execution phases, which are part of the next research question.

d) *How can the complete SLA life cycle be integrated into SDIs?*

This research question is also related to the development of the web-based SLA management architecture.

The web-based SLA management architecture also contains components for agreement implementation and agreement execution. The Agreement Monitor keeps track of all created agreements, monitors the service to which an agreement is related, and calculates all relevant service property values of an agreement. The Agreement Evaluator keeps track of all created agreements and evaluates whether the service level objectives of an agreement are fulfilled or violated. The Agreement Reporter keeps track of all created agreements and reports the agreement status to all contracting parties. These components are not public accessible for potential service consumers. They must not be standardized and they can be realized by implementations specific for each service provider.

The SLA4OWS framework also provides components that are required for agreement implementation and agreement execution. The standard service property types and monitoring mechanism, which are described in the abstract SLA model and the web-based SLA management architecture, are supported by default. The provided software solution can easily be integrated in any SDI environment in order to provide agreement implementation and agreement execution for previously created agreements.

For service providers it is important to be in a position to actually deliver particular service quality levels to specific service consumers or in an application context, which is part of the next research question.

e) *How can SDI service providers deliver promised service levels to individual customers or in an application context?*

This research question aims at the development of guidelines for SDI service providers to manage their computing infrastructure under the terms of previously created SLAs in order to deliver promised service levels to individual service consumers or in an application context.

The web-based SLA management architecture contains components that help service provider to actually adjust the delivered service quality automatically according to the agreement terms under which a service request is performed. The Infrastructure Manager keeps track of all created agreements and can dynamically schedule the infrastructure of the service provider according to the functional and non-functional requirements defined in all agreements.

Section 6.1.3 presents a Hybrid Cloud architecture that offers potentially unlimited resources on-demand and nearly in real-time by incorporating external third-party resources into the local IT infrastructure. The idea of the proposed Hybrid Cloud architecture is to keep track of all created agreements and always to provide sufficient computational resources to achieve a constant average service response time, which should be independent of the number of users requesting a service.

The Hybrid Cloud was implemented as a proof of concept by means of common Open Source software packages. The main controller of the Hybrid Cloud, which receives all created agreements from the Infrastructure Manager and adjusts the elasticity of the Cloud solution accordingly, was developed from scratch to suit the INSPIRE service quality requirements.

## 7.2 Contribution

This thesis contributes mainly to research on SDI governance. In particular, the ideas and concepts presented in this thesis can be seen as best-practices and recommendations for SDI service providers to offer SLAs for existing SDI services.

The requirements evaluation (Chapter 3) fosters the understanding what can and should be part of SLAs that can be applied in SDIs, which is a common question in many SDI building activities. The requirements analysis enables both service consumers and service providers to manage the expectations what service quality should and actually can be delivered to specific users or in an application context.

The abstract SLA model (Chapter 4) and the WS-Agreement Application Profile for OGC Web Services (Section 5.3.1) provide an SLA formalization not only on a conceptual but also on a technical level. Based on the requirements analysis, the proposed document encoding format and the proposed mandatory public service interfaces provide an interoperable and extendable foundation for integrating SLAs in SDIs.

The web-based SLA management architecture (Section 5) helps SDI service providers to identify what kind of processes and software components are required to actually offer the online and on-demand SLA negotiation for their services, and to ensure an appropriate service delivery under the terms of previously negotiated SLAs. The chosen policy-based approach shows that it is possible to extend SDI services with SLA capabilities without the need to change other OGC standards or compliant implementations.

The SLA4OWS framework (Section 6.1.1) can help SDI service providers to setup an environment that instantaneously extends already deployed and productive running SDI services with SLAs capabilities. The Hybrid Cloud (Section 6.1.3) provides a solution to always provide sufficient computational resources to realize the challenging INSPIRE performance and capacity requirements, which is a challenging task for many SDI service providers. The Hybrid Cloud allows to run services in the local IT infrastructure or to transfer the service hosting step-by-step to third-party facilities. The developments established in this thesis are validated in the SLA4D-Grid project and are public available through the 52°North Geoprocessing Community as Open Source software.

The general discussion about the content and the availability of SLAs in SDIs is not completely new, but the presented concepts provide a reliable solution for the online and on-demand negotiation and enforcement of SLAs SDIs for the first time. Therefore, all these aspects are a substantial step towards an infrastructure that is prepared for future SDI application domains including potential highly competitive GIS markets, not only on a technological but also on an economical level.

## 7.3 Future Work

Based on the results of this thesis, the following items are identified as future work:

- *Improve Data Models and Service Interfaces*

  The abstract SLA model and the corresponding WS-Agreement Application Profile for OGC Web Services are derived from the requirements analysis that is based on four selected applications domains and their particular characteristics regarding service level objectives and business models. However, there are several other important application domains in which SDI services are utilized for different

purposes and which may have different requirements that are not met yet. The developed mechanisms for identifying service properties through the OGC URN Schema Extension and for defining service level objectives by means of the Agreement Expression Language are designed in a way that they can easily be extended in order to match the requirements of other application scenarios. Therefore, one of the next steps should be to extend the URN dictionary for identifying service property types, and to revise the Agreement Expression Language for defining service level objectives and business values. Especially the integration of advanced concepts for ensuring data quality and providing data provenance information seems to be very promising and valuable for the application of the developed SLA concept in all application domains where the data applicability is from a very fundamental importance.

Furthermore, the abstract SLA model and the web-based SLA management architecture are developed with a focus on, but are not limited to SDI services that are based on standards developed by the OGC. The development of templates and agreements for other vendor-specific services and applications such as the ArcGIS for Server[1] platform seems to be very promising in order to promote the acceptance and to foster the rate of adoption of the developed concepts.

- *Harmonize Concepts for Service Level Agreements and Rights Management*

  There have been several approaches developed in the past to implement digital rights mechanisms in SDIs. Some of them take place on a standardization level as for instance the GeoDRM architecture for the management of digital rights in the area of geospatial data and services, the GeoXACML specification for deploying access control for protecting access to distributed geographic information, and the interoperability experiments that have been accomplished in past OGC testbeds. Other approaches present advanced and beyond state of art concepts for allowing the dynamic access to and chaining of secured geoprocessing services without a-priori established rights or direct trust relationships in contradiction to classic role-based access control schemes, as for instance presented in [Schäffer, 2012].

  The security-related service properties of the abstract SLA model already allow to integrate policies such as those in the GeoXACML format. These policies can be used by the web-based SLA management architecture for instance to restrict the service access to specific service operations or geospatial resources. Beside authentication and authorization, both the abstract SLA model and the web-based SLA management architecture do not define how to realize important security-related aspects such as intrusion detection, message integrity, privacy and non-repudiation. Some of the existing and previously mentioned approaches target these issues by using the advanced security mechanisms provided for instance by WS-Security. These solutions mostly rely on SOAP, which cannot easily be integrated in the JSON-based and RESTful WS-Agreement Application Profile for OGC Web Services. As mentioned earlier in this thesis, the abstract SLA model and the web-based SLA management architecture are independent of any specific data encoding format and can be implemented in several ways. Therefore, one of the next steps should be to merge the aforementioned concepts with the proposed solution for the integration of SLAs in SDIs, and to adopt mechanisms and technologies from mainstream IT in

[1] http://www.esri.com/software/arcgis/arcgisserver

order to extend the proposed solution for the integration of SLAs in SDIs with the missing security-related features.

- *Workflow Composition of Service Level Agreements*

  This thesis provides a solution for the negotiation, implementation and execution of agreements regarding single SDI services. However, many SDI application domains and use cases often require the functionality of multiple SDI services composed in cross-enterprise workflows. In this respect, the BPEL specification is the de-facto standard in the mainstream IT for describing workflows based on web services using an XML encoding. It also has been adopted successfully in the SDI domain as for instance shown in [Schäffer, 2009]. Consequently, in addition to just composing SDI services that fulfill functional requirements, the actual end-users will need to assure that services are compatible also with regards to offered service levels and business values. In the mainstream IT, the workflow composition of SLAs has already been addressed for instance in [Dyachuk and Deters, 2007] and [Blake and Cummings, 2007]. Therefore, to improve the proposed solution for the integration of SLAs in SDIs, consequently the implications of multilateral agreement creation in the SDI domain need to be investigated, and the presented concepts for the integration of SLAs in SDIs need to be adjusted in order to implement the agreement negotiation, the agreement implementation and the agreement execution in cross-enterprise workflows.

To promote the acceptance and to foster the rate of adoption of the WS-Agreement Application Profile for OGC Web Services, which already has been published as an OGC Discussion Paper in the OGC Geoprocessing Domain Working Group (DWG), it is recommended to proceed with the standardization efforts and to pursue for instance a publication as an OGC Best Practice or OGC Implementation Specification. The opportunities regarding a successful standardization process heavily depend on the overall acceptance of the proposed concepts in the standardization community, which can be evaluated for instance in future OGC interoperability experiments. Furthermore, it is recommended to harmonize the outcomes of this thesis with other OGC activities that may benefit from it and that are also related to service and data quality, service metadata, geoprocessing and geospatial rights management. Examples for such harmonization activities are for instance the integration of SLA metadata directly in the OGC Catalogue Service (CSW) and to consider infrastructure management information in the WPS service interface for process execution.

# Bibliography

[Aagedal and Ecklund, 2002] Aagedal, J. and Ecklund, E. (2002). Modelling QoS: Towards a UML Profile. In *Proceedings of the 5th International Conference on The Unified Modeling Language*, UML '02, pages 275–289, London, UK, UK. Springer-Verlag.

[AdV, 2009] AdV (2009). Richtlinie über Gebühren für die Bereitstellung und Nutzung von Geobasisdaten der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV-Gebührenrichtlinie). Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV).

[Alameh, 2003] Alameh, N. (2003). Service Chaining of Interoperable Geographic Information Web Service. *IEEE Internet Computing*, 7(5):22–29.

[Alanezi et al., 2010] Alanezi, M., Kamil, A., and Basri, S. (2010). A proposed instrument dimensions for measuring e-government service quality. *International Journal of u- and e- Service, Science and Technology*, Vol. 3, No. 4:1–18.

[Anastassova et al., 2010] Anastassova, M., Magnusson, C., Pielot, M., Randall, G., and Claassen, G. B., editors (2010). *Proceedings of the Workshop on Using Audio and Haptics for Delivering Spatial Information via Mobile Devices at MobileHCI 2010, Lisbon, Portugal*, MobileHCI '10, New York, NY, USA. ACM. 12th International Conference on Human Computer Interaction with Mobile Devices and Services.

[Andrieux et al., 2005] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M. (2005). Web Services Agreement Specification (WS-Agreement). Open Grid Forum (OGF), GFD-R-P.107.

[Armbrust et al., 2010] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.

[Armbrust et al., 2009] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., and Zaharia, M. (2009). Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, Electrical Engineering and Computer Sciences University of California at Berkeley.

[Attiya and Welch, 2004] Attiya, H. and Welch, J. (2004). *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Wiley Series on Parallel and Distributed Computing. Wiley.

[Ayers, 2007] Ayers, D. (2007). Evolving the Link. *IEEE Internet Computing*, 11(3):94–95.

[Backus et al., 1963] Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijngaarden, A., and Woodger, M. (1963). Revised report on the algorithm language ALGOL 60. *Commun. ACM*, 6:1–17.

[Bajaj et al., 2006] Bajaj, S., Box, D., Chappell, D., Curbera, F., Daniels, G., Hallam-Baker, P., Hondo, M., Kaler, C., Langworthy, D., Nadalin, A., Nagaratnam, N., Prafullchandra, H., von Riegen, C., Roth, D., Schlimmer, J., Sharp, C., Shewchuk, J., Vedamuthu, A., Yalçinalp, U., and Orchard, D. (2006). Web Services Policy 1.2 - Framework (WS-Policy). Technical report, World Wide Web Consortium (W3C).

[Bank, 2004] Bank, E. (2004). Importance of Open Spatial Data Infrastructure for Data Sharing. In Altan, O., editor, *Proceedings of the ISPRS Congress Istanbul 2004*, volume XXXV, Part B4 of *Commission IV Papers*, pages 271–276.

[Baranski, 2008] Baranski, B. (2008). Grid Computing Enabled Web Processing Service. In Bishr, M., Pebesma, E., and Bartoschek, T., editors, *Proceedings of the 6th Geographic Information Days. Ifgi Prints.*, volume 32, pages 243–256.

[Baranski, 2009] Baranski, B. (2009). Guaranteeing Quality of Service in a Spatial Data Infrastructure by using Service Level Agreements. Presented at GSDI 11 World Conference, Rotterdam, The Netherlands.

[Baranski, 2011] Baranski, B. (2011). WS-Agreement Application Profile for OGC Web Services. Open Geospatial Consortium (OGC), OGC 11-094 (Discussion Paper).

[Baranski, 2012] Baranski, B. (2012). The Service Level Agreements for OGC Web Services (SLA4OWS) Framework. In Löwner, M., Hillen, F., and Wohlfahrt, R., editors, *Geoinformatik 2012 - Mobilität und Umwelt*, pages 383–388. Shaker Verlag.

[Baranski et al., 2010a] Baranski, B., Deelmann, T., and Schäffer, B. (2010a). Pay-per-Use Revenue Models for Geoprocessing Services in the Cloud. 1st International Workshop on Pervasive Web Mapping, Geoprocessing and Services (WebMGS 2010). Como, Italy.

[Baranski et al., 2011] Baranski, B., Foerster, T., Schäffer, B., and Lange, K. (2011). Matching INSPIRE Quality of Service Requirements with Hybrid Clouds. In Wilson, J. P., Stewart Fotheringham, A., and O'Sullivan, D., editors, *Transactions in GIS*, volume 15, pages 125–142. Wiley Online Library.

[Baranski and Schäffer, 2010] Baranski, B. and Schäffer, B. (2010). Towards Service Level Agreements in Spatial Data Infrastructures. In Rajabifard, A., Crompvoets, J., Kalantari, M., and B., K., editors, *Spatially Enabling Society: Research, Emerging Trends, and Critical Assessment*, pages 149–162. Leuven University Press.

[Baranski et al., 2010b] Baranski, B., Schäffer, B., and Redweik, R. (2010b). Geoprocessing in the Clouds. In *OSGeo Journal*, volume 8, pages 17–22. Open Source Geospatial Foundation (OSGeo).

[Baranski et al., 2009] Baranski, B., Woolf, A., Shaon, A., and Kurzbach, S. (2009). OWS-6 WPS Grid Processing Profile Engineering Report. Open Geospatial Consortium (OGC), OGC 09-041 (Engineering Report).

[Battré, 2009] Battré, D. (2009). Time Constraint Profile, Version 1.0. Technical report, Open Grid Forum (OGF).

[Bell, 2004a] Bell, D. (2004a). IBM Developer Works. UML basics: The component diagram. Online. Visited 2012-03-29, http://www.ibm.com/developerworks/rational/library/dec04/bell/.

[Bell, 2004b] Bell, D. (2004b). IBM Developer Works. UML basics: The sequence diagram. Online. Visited 2012-01-06, http://www.ibm.com/developerworks/rational/library/3101.html.

[Berger, 2005] Berger, T. (2005). *Konzeption und Management von Service-Level-Agreements für IT-Dienstleistungen.* PhD thesis, TU Darmstadt, Darmstadt.

[Bishr et al., 1999] Bishr, Y., Pundt, H., Kuhn, W., and Radwan, M. (1999). Probing the Concept of Information Communities - A First Step Toward Semantic Interoperability. *Interoperating Geographic Information Systems*, 495:203–215.

[Blake and Cummings, 2007] Blake, M. and Cummings, D. (2007). Workflow Composition of Service Level Agreements. In *Services Computing, 2007. SCC 2007. IEEE International Conference on*, pages 138 –145.

[Blake et al., 1997] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. (1997). An Architecture for Differentiated Services. Internet Engineering Task Force (IETF), RFC 2475 (Informational).

[Blankenbach and Schaffert, 2009] Blankenbach, J. and Schaffert, M. (2009). A SDI and Web 2.0 based Approach to Support E-Participation in Municipal Administration and Planning Strategies. In *Facing the Challenges. Proceedings of the FIG Congress 2010. Sydney, Australia.*

[Blower, 2010] Blower, J. D. (2010). GIS in the Cloud: Implementing a Web Map Service on Google App Engine. In *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application*, COM.Geo '10, pages 34:1–34:4, New York, NY, USA. ACM.

[BMJ, 2009] BMJ (2009). Gesetz über den Zugang zu digitalen Geodaten (Geodatenzugangsgesetz - GeoZG). Bundesministerium der Justiz (BMJ). 10. Februar 2009 (BGBl. I S. 278).

[BMWI, 2010] BMWI (2010). Sichere Internet-Dienste - Sicheres Cloud Computing für Mittelstand und öffentlichen Sektor (Trusted Cloud). Bundesministerium für Wirtschaft und Technologie (BMWi). Ein Technologiewettbewerb des Bundesministeriums für Wirtschaft und Technologie.

[BMWI, 2012] BMWI (2012). Das Normungs- und Standardisierungsumfeld von Cloud Computing. Bundesministerium für Wirtschaft und Technologie (BMWi). Ein Technologiewettbewerb des Bundesministeriums für Wirtschaft und Technologie.

[Bourke, 2001] Bourke, T. (2001). *Server Load Balancing*. O'Reilly & Associates, Inc., Sebastopol, CA, USA.

[Bradner, 1997] Bradner, S. (1997). Key words for use in RFCs to Indicate Requirement Levels. Internet Engineering Task Force (IETF), RFC 2119 (Best Current Practice).

[Brahnmath et al., 2002] Brahnmath, G. J., Rajeev, R. R., Olson, A. M., Auguston, M., Bryant, B. R., and Burt, C. C. (2002). A Quality of Service Catalog for Software Components. In *Proceedings of the Southeastern Software Engineering Conference*.

[Brauner et al., 2009] Brauner, J., Foerster, T., Schäffer, B., and Baranski, B. (2009). Towards a Research Agenda for Geoprocessing Services. In J. Haunert, B. K. and Milde, J., editors, *Proceedings of 12th AGILE International Conference on Geographic Information Science*. Hanover, Germany: IKG, Leibniz University of Hanover.

[Brunette and Mogull, 2009] Brunette, G. and Mogull, R. (2009). Security Guidance for Critical Areas of Focus in Cloud Computing V2. 1. *CSA (Cloud Security Alliance), USA.*, 1.

[Bröring et al., 2012] Bröring, A., Stasch, C., and Echterhoff, J. (2012). OGC Sensor Observation Service Interface Standard. Open Geospatial Consortium (OGC), OGC 12-006 (Implementation Specification).

[BSI, 2012] BSI (2012). Sicherheitsempfehlungen für Cloud Computing Anbieter - Mindestanforderungen in der Informationssicherheit. Bundesamt für Sicherheit in der Informationstechnik (BSI). Eckpunktepapier.

[Budhathoki and Nedovi-Budi, 2006] Budhathoki, N. R. and Nedovi-Budi, Z. (2006). Expanding the Spatial Data Infrastructure Knowledge Base. In Onsrud, H., editor, *Research and Theory in Advancing Spatial Data Infrastructure Concepts*, pages 7–31. ESRI Press.

[Buyya et al., 2009] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616.

[Cannon, 2007] Cannon, D. L. (2007). *ITIL Service Operation*. TSO The Stationery Office.

[Carver, 2003] Carver, S. (2003). The Future of Participatory Approaches Using Geographic Information: developing a research agenda for the 21st Century. *Urban & Regional Information Systems Association*, 15(APA 1):61–71.

[CCUCG, 2009] CCUCG (2009). Cloud Computing Use Cases - A White Paper. *October*, 2. Cloud Computing Use Case Discussion Group.

[CGDI Architecture Working Group, 2001] CGDI Architecture Working Group (2001). *Canadian Geospatial Data Infrastructure, architecture description*. Natural Resources Canada.

[Chen, 1976] Chen, P. P.-S. (1976). The Entity-Relationship Model - Toward a Unified View of Data. *ACM Trans. Database Syst.*, 1:9–36.

[Chouk, 2003] Chouk, M. (2003). *Master-slave replication, failover and distributed recovery in PostgreSQL database*. McGill University.

[Christensen et al., 2001] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001). Web Service Definition Language (WSDL). Technical report, World Wide Web Consortium (W3C).

[Chung et al., 2009] Chung, L., Fang, Y., Chang, Y., Chou, T., Lee, B., Yin, H., and Baranski, B. (2009). A SOA based debris flow monitoring system. In *Proceedings of the 17th International Conference on Geoinformatics, 2009*, pages 1–6. IEEE.

[Comber and Wadsworth, 2005] Comber, A. and Wadsworth, R. Fisher, P. (2005). Reasoning Methods for Handling Uncertain Information in Land Cover Mapping. In Devillers, R. and Jeansoulin, R., editors, *Fundamentals of Spatial Data Quality*, chapter 7, pages 123–139. Hermes Science/Lavoisier.

[Cova, 1996] Cova, T. (1996). GIS in emergency management. In Longley, P., Goodchild, M., Maguire, D., and Rhind, D., editors, *Geographical Information Systems Principles Techniques Applications*, volume 2, chapter 60, pages 845–858. Wiley.

[Cox, 2009] Cox, S. (2009). Name type specification - documents. Open Geospatial Consortium (OGC), OGC 09-047r3 (OpenGIS Policy).

[Cox, 2010a] Cox, S. (2010a). Name type specification - definitions - part 1 - basic name. Open Geospatial Consortium (OGC), OGC 09-048r3 (OpenGIS Policy).

[Cox, 2010b] Cox, S. (2010b). Name type specification - specification elements. Open Geospatial Consortium (OGC), OGC 10-103 (OpenGIS Policy).

[Cox, 2010c] Cox, S. (2010c). OGC Naming Authority - Policies and Procedures. Open Geospatial Consortium (OGC), OGC 09-046r2 (OpenGIS Policy).

[Cox, 2010d] Cox, S. (2010d). OGC Naming Authority: Procedures. Open Geospatial Consortium (OGC), OGC 09-046r2 (OpenGIS Policy).

[Craglia et al., 2008] Craglia, M., Goodchild, M. F., Annoni, A., Camara, G., Gould, M., Kuhn, W., Mark, D., Masser, I., and Maguire, D. (2008). Next-Generation Digital Earth. *International Journal of Spatial Data Infrastructure Research*, 3(3):146–167.

[Crockford, 2006] Crockford, D. (2006). The application/json Media Type for JavaScript Object Notation (JSON). Internet Engineering Task Force (IETF), RFC 4627 (Informational).

[Czajkowski et al., 2004] Czajkowski, K., Ferguson, D. F., Foster, I., Frey, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S., and Vambenepe, W. (2004). The WS-Resource Framework Version 1.0. OASIS Standard, Organization for the Advancement of Structured Information Standards (OASIS).

[de la Beaujardiere, 2006] de la Beaujardiere, J. (2006). OpenGIS Web Map Server Implementation Specification. Open Geospatial Consortium (OGC), OGC 06-042 (Implementation Specification).

[De Smith et al., 2007] De Smith, M., J, G., M F, L., and P, A. (2007). *Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software Tools*, volume 1. Troubador.

[DeCapua and Bhaduri, 2007] DeCapua, C. and Bhaduri, B. (2007). Applications of Geospatial Technology in International Disasters and During Hurricane Katrina. Technical report, Oak Ridge National Laboratory.

[Deelman et al., 2004] Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M.-H., Vahi, K., and Livny, M. (2004). Pegasus: Mapping Scientific Workflows onto the Grid. In Dikaiakos, M. D., editor, *European Across Grids Conference*, volume 3165 of *Lecture Notes in Computer Science*, pages 11–20. Springer.

[Densham, 1991] Densham, P. (1991). Spatial Decision Support Systems. *Geographical Information Systems*, 1:403–412.

[Deursen et al., 2000] Deursen, A., Klint, P., and Visser, J. M. (2000). Domain-Specific Languages. Technical report, Amsterdam, The Netherlands, The Netherlands.

[Di et al., 2003] Di, L., Chen, A., Yang, W., and Zhao, P. (2003). The Integration of Grid Technology with OGC Web Services (OWS) in NWGISS for NASA EOS Data. In *in NWGISS for NASA EOS Data, in GGF8 & HPDC12 2003*, pages 24–27. Science Press.

[DMTF, 1999] DMTF (1999). Common Information Model (CIM) Specification. Distributed Management Task Force (DMTF).

[DMTF, 2009] DMTF (2009). Open Virtualization Format Specification. Distributed Management Task Force (DMTF).

[Donker, 2009] Donker, F. (2009). Public Sector Geo Web Services: Which Business Model Will Pay for a Free Lunch? In van Loenen, B., Besemer, J., and Zevenbergen, J., editors, *SDI Convergence. Research, Emerging Trends, and Critical Assessment*. Nederlandse Commissie voor Geodesie.

[Dotcom-Monitor, 2012] Dotcom-Monitor (2012). Active vs. Passive Web Performance Monitoring. Online. Visited 2012-02-28, http://www.dotcom-monitor.com/release-active-vs-passive-web-performance-monitoring.asp.

[Drabek and Hoetmer, 1991] Drabek, T. and Hoetmer, G. (1991). *Emergency management: principles and practice for local government*. Municipal management series. International City Management Association.

[Driver, 2008] Driver, M. (2008). Gartner Says Cloud Application Infrastructure Technologies Need Seven Years to Mature. Gartner Press Release.

[Dyachuk and Deters, 2007] Dyachuk, D. and Deters, R. (2007). Service Level Agreement Aware Workflow Scheduling. In *IEEE SCC*, pages 715–716. IEEE Computer Society.

[Díaz et al., 2012] Díaz, L., Remke, A., Kauppinen, T., Degbelo, A., Foerster, T., Stasch, C., Rieke, M., Schaeffer, B., Baranski, B., Broering, A., and Wytzisk, A. (2012). Future SDI – Impulses from Geoinformatics Research and IT Trends. *International Journal of Spatial Data Infrastructures Research (IJSDIR)*, Volume 7.

[EC, 2004] EC (2004). Multi-channel delivery of eGovernment services. Technical report, European Commission. Interchange of Data between Administrations Programme.

[Eggert and Heidemann, 1999] Eggert, L. and Heidemann, J. (1999). Application-Level Differentiated Services for Web Servers. *World Wide Web*, 2(3):133–142.

[EU, 1995] EU (1995). Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Union*, pages 0031–0050.

[EU, 2007] EU (2007). DIRECTIVE 2007/2/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL: establishing an Infrastructure for Spatial Information in the European Community (INSPIRE). *Official Journal of the European Union.*

[EU, 2011a] EU (2011a). INSPIRE & NSDI SoP. D4.2 - Summary report regarding the results of the European Assessment of 34 NSDI (first year). State of Play: Report from the Europen Union, Spatial Applications Division K.U. Leuven.

[EU, 2011b] EU (2011b). INSPIRE Roadmap. Online. Visited 2012-06-09, http://inspire.jrc.ec.europa.eu/index.cfm/pageid/44.

[EU, 2011c] EU (2011c). Spatial Data Infrastructures in Germany: State of Play 2011. State of Play: Report from the Europen Union, Spatial Applications Division K.U. Leuven.

[FGDC, 2001] FGDC (2001). *Building the Business Case for the Geospatial Platform - The Value Proposition.* Federal Geographic Data Committee (FGDC).

[FGDC, 2005] FGDC (2005). *The National Spatial Data Infrastructure.* The Federal Geographic Data Committee (FGDC). Online Available: http://www.fgdc.gov/nsdi/library/factsheets/documents/nsdi.pdf [Accessed 3 June 2012].

[Fielding et al., 1999] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1. Internet Engineering Task Force (IETF), RFC 2616 (Standards Track).

[Fielding, 2000] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, California.

[Flanagan, 1998] Flanagan, D. (1998). *JavaScript: The Definitive Guide.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 3rd edition.

[Flegkas et al., 2003] Flegkas, P., Flegkas, P., Trimintzios, P., Trimintzios, P., Pavlou, G., Pavlou, G., Liotta, A., and Liotta, A. (2003). Design and Implementation of a Policy-Based Resource Management Architecture. In *IEEE/IFIP Integrated Management Symposium (IM 2003*, pages 215–229. Kluwer.

[Fleuren and Müller, 2008] Fleuren, T. and Müller, P. (2008). BPEL Workflows Combining Standard OGC Web Services and Grid-enabled OGC Web Services. In *Proceedings of the 2008 34th Euromicro Conference Software Engineering and Advanced Applications*, SEAA '08, pages 337–344, Washington, DC, USA. IEEE Computer Society.

[Foerster, 2006] Foerster, T. (2006). An open software framework for Web Service-based geo-processes. In *FOSS4G2006 - Free And Open Source Software for Geoinformatics*, volume 8, pages 17–22. Open Source Geospatial Foundation (OSGeo). Presented at Free and Open Source Software for Geospatial (FOSS4G) Conference, Sydney, Australia.

[Foerster et al., 2010] Foerster, T., Baranski, B., Schäffer, B., and Lange, K. (2010). Geoprocessing in Hybrid Clouds. In Zipf, A., Behncke, K., Hillen, F., and Schaefermeyer, J., editors, *Die Welt im Netz*, pages 13–19. Akademische Verlagsgesellschaft.

[Foerster et al., 2011] Foerster, T., Schäffer, B., Brauner, J., and Baranski, B. (2011). Geospatial Web Services for Distributed Processing - Applications and Scenarios. In Zhao, P. and Di, P., editors, *Geospatial Web Services: Advances in Information Interoperability*, pages 245–286. Hershey.

[Fornefeld et al., 2008] Fornefeld, M., Boele-Keimer, G., Recher, S., and Fanning, M. (2008). Assessment of the Re-use of Public Sector Information (PSI) in the Geographical information, Meteorological Information and Legal Information Sectors. Final Report. Technical report, MICUS Management Consulting GmbH, Düsseldorf, Germany.

[Fornefeld et al., 2003] Fornefeld, M., Oefinger, P., and Rausch, U. (2003). The Market for Geospatial Information: Potentials for Employment, Innovation and Value Added. Technical report, MICUS Management Consulting GmbH, Düsseldorf, Germany.

[Foster, 2003] Foster, I. (2003). *The Grid: A New Infrastructure for 21st Century Science*, pages 51–63. John Wiley & Sons, Ltd.

[Foster et al., 2006] Foster, I., Kishimoto, H., Savva, A., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J., and Von Reich, J. (2006). The Open Grid Services Architecture, Version 1.5. Open Grid Forum (OGF), GFD-I.080.

[Foster et al., 2008] Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud Computing and Grid Computing 360-Degree Compared.

[Franks et al., 1999] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and Stewart, L. (1999). HTTP Authentication: Basic and Digest Access Authentication. Internet Engineering Task Force (IETF), RFC 2617 (Standards Track).

[Freed, 2000] Freed, N. (2000). Behavior of and Requirements for Internet Firewalls. Internet Engineering Task Force (IETF), RFC 2979 (Informational).

[Frick et al., 2002] Frick, R., Keller, M., Vettori, A., Meier, J., and Spahni, D. (2002). Analyse Geodatenmarkt Schweiz. Technical report, Institut für Wirtschaft und Verwaltung (IWV). Bern, 31. Oktober 2002.

[Frolund and Koistinen, 1998] Frolund, S. and Koistinen, J. (1998). Quality of Service Specification in Distributed Object Systems Design. In *Proceedings of the 4th USENIX Conference on Object-Oriented Technology and Systems (COOTS)*, Santa Fe, New Mexico.

[Garnett, 2011] Garnett, J. (2011). Web Processing Service Shootout - Execute Process Posse Panel Discussion. In *FOSS4G 2011 - Free And Open Source Software for Geoinformatics*, volume 8, pages 17–22. Open Source Geospatial Foundation (OSGeo). Presented at Free and Open Source Software for Geospatial (FOSS4G) Conference, Sydney, Australia.

[Gartmann and Leinenweber, 2009] Gartmann, R. and Leinenweber, L. (2009). OWS-6 Security Engineering Report. Open Geospatial Consortium (OGC), OGC 09-035 (Engineering Report).

[Gartmann and Schäffer, 2011] Gartmann, R. and Schäffer, B. (2011). License-Based Access Control. Open Geospatial Consortium (OGC), OGC 11-018 (Discussion Paper).

[GDI-DE, 2010] GDI-DE (2010). *Architektur der Geodateninfrastruktur Deutschland, Version 2.0*. Lenkungsgremium GDI-DE. Arbeitskreis Architektur der GDI-DE und Koordinierungsstelle GDI-DE.

[Gisler et al., 2000] Gisler, M., Stanoevska-Slabeva, K., and Greunz, M. (2000). Legal Aspects of Electronic Contracts. CAiSE 2000 Workshop: Infrastructure for Dynamic Business-to-Business Service Outsourcing (ISDO 2000).

[Goodchild, 1991] Goodchild, M. F. (1991). Geographic Information Systems. *Progress in Human Geography*, 15(2):192–200.

[Goodchild, 2010] Goodchild, M. F. (2010). Twenty years of progress: GIScience in 2010. *Journal of Spatial Information Science*, 1(1):3–20.

[Goodchild and Glennon, 2010] Goodchild, M. F. and Glennon, J. A. (2010). Crowdsourcing geographic information for disaster response: a research frontier. *International Journal of Digital Earth*, 3(3):231–241.

[Groot and McLaughlin, 2000] Groot, R. and McLaughlin, J. D. (2000). *Geospatial Data Infrastructure: Concepts, Cases, and Good Practice*. Oxford ; New York : Oxford University Press. Includes bibliographical references and index.

[Groth, 2012] Groth, P. (2012). Feedback Welcome: An Overview of the Provenance (PROV) family of specs. Online. Visised 2012-09-17, http://www.w3.org/blog/SW/2012/01/11/feedback-welcome-an-overview-of-the-provenance-prov-family-of-specs/.

[Gudgin et al., 2006] Gudgin, M., Hadley, M., and Rogers, T. (2006). Web Services Addressing - Core. Technical report, World Wide Web Consortium (W3C).

[Hafner and Breu, 2008] Hafner, M. and Breu, R. (2008). *Security Engineering for Service-Oriented Architectures*. Springer Publishing Company, Incorporated.

[Halpin, 2001] Halpin, T. (2001). *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann.

[Hamilton, 1997] Hamilton, G. (1997). JavaBeans Specification, Version 1.01. Online. Visited 2012-02-08, http://www.oracle.com/technetwork/java/javase/documentation/spec-136004.html.

[Hansen, 2007] Hansen, M. D. (2007). *SOA Using Java Web Services*. Prentice Hall.

[Hartig, 2009] Hartig, K. (2009). Cloud Computing Journal. What is Cloud Computing? Online. Visited 2012-05-10, http://cloudcomputing.sys-con.com/node/579826.

[Herrmann, 2010] Herrmann, J. (2010). Access Control Systems for Spatial Data Infrastructures and their Administration. In *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research and Application*, COM.Geo '10, pages 47:1–47:2, New York, NY, USA. ACM.

[Herrmann and Matheus, 2009] Herrmann, J. and Matheus, A. (2009). OWS-6 GeoXACML Engineering Report. Open Geospatial Consortium (OGC), OGC 09-036r2 (Engineering Report).

[Higgins, 2009] Higgins, C.;Sinnott, R. M. J. A. A. (2009). Spatial Data e-Infrastructure. In *Proceedings of International Conference on e-Social Science, Cologne, Germany, May 2009*. N/A.

[Hiles, 2002] Hiles, A. (2002). *The Complete Guide to IT Service Level Agreements: Aligning IT Services to Business Needs*. Service Level Management Series. Rothstein Associates Incorporated.

[Hillmann-Köster, 2011] Hillmann-Köster, B. (2011). Arbeiten in der Cloud // Working in the Cloud. *gis.TRENDS+MARKETS: The Geomatics News Magazine*, 1:42–49.

[Hobona et al., 2007] Hobona, G., Fairbairn, D., and P., J. (2007). Workflow Enactment of Grid-Enabled Geospatial Web Services. In *Proceedings of the UK e-Science All Hands Meeting*. National e-Science Centre.

[Hudak, 1998] Hudak, P. (1998). Domain Specific Languages. In *Handbook of Programming Languages, Vol. III: Little Languages and Tools*, chapter 3, pages 39–60. MacMillan, Indianapolis.

[Humphrey and Kellner, 1989] Humphrey, W. S. and Kellner, M. I. (1989). Software process modeling: principles of entity process models. In *Proceedings of the 11th international conference on Software engineering*, ICSE '89, pages 331–342, New York, NY, USA. ACM.

[INSPIRE, 2007] INSPIRE (2007). INSPIRE Network Services Performance Guidelines. Technical report, INSPIRE Infrastructure for Spatial Information in Europe, INSPIRE Consolidation Team.

[INSPIRE, 2008a] INSPIRE (2008a). Definition of Annex Themes and Scope, Version 3.0. Technical report, INSPIRE Infrastructure for Spatial Information in Europe, Drafting Team "Data Specifications".

[INSPIRE, 2008b] INSPIRE (2008b). Network Services Architecture, Version 3.0. Technical report, INSPIRE Infrastructure for Spatial Information in Europe, Network Services Drafting Team.

[INSPIRE, 2009] INSPIRE (2009). Draft Technical Guidance for INSPIRE Download Services, Version 2.0. Technical report, INSPIRE Infrastructure for Spatial Information in Europe, Drafting Team "Network Services".

[INSPIRE, 2010a] INSPIRE (2010a). Draft Technical Guidance for INSPIRE Coordinate Transformation Services, Version 2.1. Technical report, INSPIRE Infrastructure for Spatial Information in Europe, Drafting Team "Network Services".

[INSPIRE, 2010b] INSPIRE (2010b). Guidance on the "Regulation on access to spatial data sets and services of the Member States by Community institutions and bodies under harmonised conditions". Technical report, INSPIRE Infrastructure for Spatial Information in Europe, DT Data and Service Sharing.

[INSPIRE, 2010c] INSPIRE (2010c). INSPIRE Data Specification on Protected Sites - Guidelines. Technical report, INSPIRE Infrastructure for Spatial Information in Europe, INSPIRE Thematic Working Group Protected Sites.

[INSPIRE, 2011a] INSPIRE (2011a). Technical Guidance for the implementation of INSPIRE View Services, Version 3.1. Technical report, INSPIRE Infrastructure for Spatial Information in Europe, IOC Task Force for Network Services.

[INSPIRE, 2011b] INSPIRE (2011b). Technical Guidance to implement INSPIRE Discovery Services, Version 3.1. Technical report, INSPIRE Infrastructure for Spatial Information in Europe, IOC Task Force "Network Services".

[ISO, 2003] ISO (2003). ISO 19115:2003 Geographic information – Metadata. Technical report, International Organization for Standardization (TC 211).

[ISO, 2005] ISO (2005). ISO ISO 19128:2005 Geographic information – Web map server interface. Technical report, International Organization for Standardization (TC 211).

[ITU, 1994] ITU (1994). *ITU-T Recommendation E.800: Telephone Network and ISDN Quality of Service, Network Management and Traffic Engineering : Terms and Definitions of Traffic Engineering*. International Telecommunication Union (ITU).

[Jeberson and Sasipraba, 2010] Jeberson, R. and Sasipraba, T. (2010). *Disaster Management System based on GIS Web Services*, page 252–261. IEEE.

[Jin et al., 2002] Jin, L., Machiraju, V., and Sahai, A. (2002). Analysis on Service Level Agreement of Web Services. *HP LABORATORIES*.

[Johnson, 2000] Johnson, R. (2000). *GIS Technology for Disasters and Emergency Management*. ESRI White Paper.

[Joksić Dušan, 2004] Joksić Dušan, B. B. (2004). Elements of spatial data quality as information technology support for sustainable development planning. *Spatium*, Issue 11:pp. 77–83.

[Kalbfleisch et al., 1999] Kalbfleisch, C., Krupczak, C., Presuhn, R., and Saperia, J. (1999). Application Management MIB. Internet Engineering Task Force (IETF), RFC 2564 (Proposed Standard).

[Karten, 1998] Karten, N. (1998). *How to establish service level agreements*. Naomi Karten Associates.

[Kearney et al., 2010] Kearney, K. T., Torelli, F., and Kotsokalis, C. (2010). SLA: An abstract syntax for Service Level Agreements. In *GRID*, pages 217–224. IEEE.

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

[Kiehle et al., 2006] Kiehle, C., Greve, K., and Heier, C. (2006). Standardized Geoprocessing - Taking Spatial Data Infrastructures One Step Further. In *9th AGILE International Conference on Geographic Information Science*, Visegrad, Hungary.

[Kim and Lee, 2005] Kim, E. and Lee, Y. (2005). OASIS Web Services Quality Model TC. Technical report, Organization for the Advancement of Structured Information Standards (OASIS).

[Kübert et al., 2011] Kübert, R., Katsaros, G., and Wang, T. (2011). A RESTful implementation of the WS-agreement specification. In *Proceedings of the Second International Workshop on RESTful Design*, WS-REST '11, pages 67–72, New York, NY, USA. ACM.

[Kubik and Kopanczyk, 2009] Kubik, T. and Kopanczyk, B. (2009). Implementing WPS as an Coordinate Transformation Service. *GSDI 11 World Conference*.

[Kuebert et al., 2010] Kuebert, R., Tenschert, A., Wä andldrich, O., Ziegler, W., and Battre, D. (2010). A service level agreement layer for the D-Grid infrastructure. In *eChallenges, 2010*, pages 1 –9.

[Kurzbach et al., 2009] Kurzbach, S., Braune, S., Pasche, E., and Smith, M. (2009). *Angewandte Geoinformatik 2010. Beiträge zum 22. AGIT-Symposium Salzburg*, chapter Operative Hochwasservorhersage-Dienste im Geodateninfrastruktur-Grid, page 881–886. Wichmann.

[Lamanna et al., 2003] Lamanna, D. D., Skene, J., and Emmerich, W. (2003). SLAng: A Language for Defining Service Level Agreements. In *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, FTDCS '03, pages 100–, Washington, DC, USA. IEEE Computer Society.

[Lanig et al., 2008] Lanig, S., Schilling, A., Stollberg, B., and Zipf, A. (2008). Towards Standards-Based Processing of Digital Elevation Models for Grid Computing through Web Processing Service (WPS). *Computational Science and Its Applications. Lecture Notes in Computer Science.*, 5073/2008:191–203.

[Lee and Ben-Natan, 2002] Lee, J. and Ben-Natan, R. (2002). *Integrating Service Level Agreements: Optimizing Your OSS for SLA Delivery*. John Wiley & Sons, Inc., New York, NY, USA.

[Lee et al., 2003] Lee, K., Jeon, J., Lee, W., Jeong, S.-H., and Park, S.-W. (2003). QoS for Web Services: Requirements and Possible Approaches. Technical report, World Wide Web Consortium (W3C).

[Lehto, 2009] Lehto, L. (2009). Real-Time content transformations in the European Spatial Data Infrastructure. *GSDI 11 World Conference*.

[Leiba, 2012] Leiba, B. (2012). OAuth Web Authorization Protocol. *IEEE Internet Computing*, 16:74–77.

[Leimeister et al., 2010] Leimeister, S., Riedl, C., Böhm, M., and Krcmar, H. (2010). The Business Perspective of Cloud Computing: Actors, Roles, and Value Networks. In *Proceedings of 18th European Conference on Information Systems (ECIS 2010)*, Pretoria, South Africa.

[Lloyd, 2008] Lloyd, V. (2008). *ITIL Service Design*. TSO The Stationery Office, Norwich.

[Lubke et al., 2005] Lubke, R., Ball, J., and Delisle, P. (2005). Unified Expression Language. Online. Visised 2012-02-23, http://java.sun.com/products/jsp/reference/techart/unifiedEL.html.

[Ludwig and Coetzee, 2010] Ludwig, B. and Coetzee, S. (2010). A Comparison of Platforms as a Service (PaaS) Clouds with a detailed Reference to Security and Geoprocessing Services. 1st International Workshop on Pervasive Web Mapping, Geoprocessing and Services (WebMGS 2010). Como, Italy.

[Ludwig et al., 2002] Ludwig, H., Keller, A., Dan, A., and King, R. (2002). A Service Level Agreement Language for Dynamic Electronic Services. In *Proceedings of the Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS'02)*, WECWIS '02, pages 25–, Washington, DC, USA. IEEE Computer Society.

[Marston et al., 2011] Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., and Ghalsasi, A. (2011). Cloud computing - The business perspective. *Decis. Support Syst.*, 51(1):176–189.

[Masser, 2005] Masser, I. (2005). *GIS Worlds: Creating Spatial Data Infrastructures*. ESRI Press, Redlands, California.

[Massuthe et al., 2005] Massuthe, P., Reisig, W., and Schmidt, K. (2005). An Operating Guideline Approach to the SOA. *Annals of Mathematics, Computing & Teleinformatics*, 1(3):35–43.

[Matheus and Herrmann, 2011] Matheus, A. and Herrmann, J. (2011). Geospatial eXtensible Access Control Markup Language (GeoXACML). Open Geospatial Consortium (OGC), OGC 11-017 (Abstract Specification).

[Maué and Kiehle, 2009] Maué, P. and Kiehle, C. (2009). Grid Technologies for Geospatial Applications – An Overview. *GIS.Science*, 3:65–67.

[McConnell and Siegel, 2004] McConnell, J. and Siegel, E. (2004). *Practical Service level Management: Delivering High-Quality Web-Based Services*. Cisco Press.

[McElhearn, 2004] McElhearn, K. (2004). *The Mac OS X Command Line: Unix Under the Hood*. SYBEX Inc., Alameda, CA, USA.

[Meissner et al., 2002] Meissner, A., Luckenbach, T., Risse, T., Kirste, T., and Kirchner, H. (2002). Design Challenges for an Integrated Disaster Management Communication and Information System. In *The First IEEE Workshop on Disaster Recovery Networks*.

[Mell and Grance, 2009] Mell, P. and Grance, T. (2009). The NIST Definition of Cloud Computing. National Institute of Standards and Technology. *Information Technology Laboratory*, 15(6):10–7.

[Mernik et al., 2005] Mernik, M., Heering, J., and Sloane, A. M. (2005). When and How to develop domain-specific languages. *ACM Comput. Surv.*, 37:316–344.

[Metsch and Edmonds, 2011a] Metsch, T. and Edmonds, A. (2011a). Open Cloud Computing Interface - Infrastructure. Open Grid Forum (OGF), GFD-P-R.184.

[Metsch and Edmonds, 2011b] Metsch, T. and Edmonds, A. (2011b). Open Cloud Computing Interface - RESTful HTTP Rendering. Open Grid Forum (OGF), GFD-P-R.185.

[Moats, 1997] Moats, R. (1997). URN Syntax. Internet Engineering Task Force (IETF), RFC 2141 (Standards Track).

[Moore et al., 2001] Moore, B., Ellesson, E., Strassner, J., and Westerinen, A. (2001). Policy Core Information Model – Version 1 Specification. RFC 3060 (Proposed Standard). Updated by RFC 3460.

[Moses, 2005] Moses, T. (2005). eXtensible Access Control Markup Language TC v2.0 (XACML). Technical report, Organization for the Advancement of Structured Information Standards (OASIS).

[Mulligan and Gracanin, 2009] Mulligan, G. and Gracanin, D. (2009). A comparison of SOAP and REST implementations of a service based interaction independence middleware framework. *Architecture*, pages 1423–1432.

[Myerson, 2009] Myerson, J. (2009). IBM Developer Works. Cloud Computing versus Grid Computing. Online. Visited 2012-05-10, http://www.ibm.com/developerworks/web/library/wa-cloudgrid/.

[Nadalin et al., 2006] Nadalin, A., Kaler, C., Monzillo, R., and Hallam-Baker, P. (2006). Web Services Security: SOAP Message Security 1.1. OASIS Standard, Organization for the Advancement of Structured Information Standards (OASIS).

[Nebert, 2004] Nebert, D. D. (2004). *Developing Spatial Data Infrastructures: The SDI Cookbook*. GSDI.

[Nogueras-Iso et al., 2004] Nogueras-Iso, J., Latre-Abadia, M. A., Muro-Medrano, P. R., and Zarazaga-Soria, F. J. (2004). Building e-Government Services over Spatial Data Infrastructures. In Traunmüller, R., editor, *Electronic Government and the Information Systems Perspective*, volume 3183 of *Lecture Notes in Computer Science*, pages 387–391. Springer.

[NRW, 2010] NRW (2010). Gebührenordnung für das amtliche Vermessungswesen und die amtliche Grundstückswertermittlung in Nordrhein-Westfalen (VermWertGebT). Gesetz- und Verordnungsblatt (GV. NRW.) Ausgabe 2010 Nr. 23 vom 16.7.2010 Seite 389 bis 406, Ministerium für Inneres und Kommunales des Landes Nordrhein-Westfalen.

[Nyrén et al., 2011] Nyrén, R., Edmonds, A., Papaspyrou, A., and Metsch, T. (2011). Open Cloud Computing Interface - Core. Open Grid Forum (OGF), GFD-P-R.183.

[Odlyzko, 1998] Odlyzko, A. (1998). The Economics of the Internet: Utility, Utilization, Pricing, and Quality of Service. Technical report, AT&T Labs - Research.

[OGC, 2003] OGC (2003). OGC Reference Model (ORM). Open Geospatial Consortium (OGC), OGC 08-062r4.

[OMG, 2011a] OMG (2011a). Unified Modeling Language (OMG UML), Infrastructure, V2.4.1. Technical report, Object Management Group.

[OMG, 2011b] OMG (2011b). Unified Modeling Language (OMG UML), Superstructure, V2.4.1. Technical report, Object Management Group.

[Onchaga, 2005] Onchaga, R. (2005). On Quality-Aware Composition of Geographic Information Services for Disaster Management. In Van Oosterom, P., Zlatanova, S., and Fendel, E., editors, *Geo-Information for Disaster Management*, pages 751–766. Springer.

[Ortmann et al., 2011] Ortmann, J., Limbu, M., Wang, D., and Kauppinen, T. (2011). Crowdsourcing Linked Open Data for Disaster Management. In *Proceedings of Terra Cognita 2011, The 10th International Semantic Web Conference (ISWC2011)*, Bonn, Germany.

[Padberg and Kiehle, 2009] Padberg, A. and Kiehle, C. (2009). Towards a grid-enabled SDI: Matching the paradigms of OGC Web Services and Grid Computing. *GSDI 11 World Conference*, (Special Issue GSDI-11).

[Papaioannou et al., 2006] Papaioannou, I. V., Tsesmetzis, D. T., Roussaki, I. G., and Anagnostou, M. E. (2006). A QoS Ontology Language for Web-Services. In *Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 01*, AINA '06, pages 101–106, Washington, DC, USA. IEEE Computer Society.

[Papazoglou, 2003] Papazoglou, M. P. (2003). Service -Oriented Computing: Concepts, Characteristics and Directions. *International Conference on Web Information Systems Engineering*, 0:3–12.

[Parker, 2005] Parker, C. (2005). Disaster Management: The Challenges for a National Geographic Information Provider. In Van Oosterom, P., Zlatanova, S., and Fendel, E., editors, *Geo-Information for Disaster Management*, pages 191–214. Springer.

[Parkin et al., 2008] Parkin, M., Badia, R. M., and Martrat, J. (2008). A Comparison of SLA Use in Six of the European Commissions FP6 Projects. *CoreGRID Technical Report*, TR-0129.

[Patricio et al., 2009] Patricio, L., Falcao, C., and Fisk, R. (2009). Requirements engineering for multi-channel services: the SEB method and its application to a multi-channel bank. *Requirements Engineering*, 14(3):209–227.

[Paul and Ghosh, 2006] Paul, M. and Ghosh, S. K. (2006). An Approach for Service Oriented Discovery and Retrieval of Spatial Data. In *Proceedings of the 2006 International Workshop on Service-Oriented Software Engineering*, SOSE '06, pages 88–94, New York, NY, USA. ACM.

[Peltz, 2003] Peltz, C. (2003). Web Services Orchestration and Choreography. *Computer*, 36:46–52.

[Percivall, 2002] Percivall, G. (2002). ISO 19119 and OGC Service Architecture.

[Pras and Schoenwaelder, 2003] Pras, A. and Schoenwaelder, J. (2003). On the Difference between Information Models and Data Models. Internet Engineering Task Force (IETF), RFC 3444 (Informational).

[Prasad, 1996] Prasad, S. (1996). *Multithreading Programming Techniques*. McGraw-Hill, Inc., New York, NY, USA.

[Prechelt, 2000] Prechelt, L. (2000). An Empirical Comparison of Seven Programming Languages. *Computer*, 33(10):23–29.

[Quirchmayr et al., 2007] Quirchmayr, G., Funilkul, S., and Chutimaskul, W. (2007). A Quality Model of e-Government Services Based on the ISO/IEC 9126 Standard. In *Proceedings of International Legal Informatics Symposium (IRIS), Salzburg, Austria*.

[Rajabifard, 1999] Rajabifard, A. (1999). The Nature of Regional Spatial Data Infrastructures. In *The 27th Annual Conference of AURISA Fairmont Resort, Blue Mountains, NSW*, pages 22–26.

[Rajabifard et al., 2004] Rajabifard, A., Mansourian, A., Williamson, I., Valadan, Z., and Mohammad, J. (2004). Developing Spatial Data Infrastructure to Facilitate Disaster Management. *Proceedings GEOMATICS 83 Conference*.

[Rajabifard and Williamson, 2001] Rajabifard, A. and Williamson, I. P. (2001). Spatial Data Infrastructures: Concept, SDI Hierarchy and Future Directions. *Culture*, 80(80):1–10.

[Rajabifard et al., 2000] Rajabifard, A., Williamson, I. P., Holland, P., and Johnstone, G. (2000). From Local to Global SDI initiatives : a pyramid of building blocks. *Proceedings of the 4th GSDI Conference*, page 13–15.

[Ramirez, 2001] Ramirez, A. (2001). LINUX Journal. Three-Tier Architecture. Online. Visited 2012-05-10,
http://www.linuxjournal.com/article/3508.

[Ran, 2003] Ran, S. (2003). A model for web services discovery with QoS. *SIGecom Exch.*, 4(1):1–10.

[Rao et al., 2004] Rao, J., Kungas, P., and Matskin, M. (2004). Logic-based web services composition: From service description to process model. In *In Intl. Conference on Web Services (ICWS*, pages 446–453. IEEE.

[Rech, 2011] Rech, M. (2011). Mit der Lizenz zum Wirken. *gis.BUSINESS*, 07:14–15.

[Reed, 2002] Reed, C. (2002). Uniform Resource Names (URN) Namespace Definition Mechanisms. Internet Engineering Task Force (IETF), RFC 3406 (Best Current Practice).

[Reed, 2004] Reed, C. (2004). A URN namespace for the Open Geospatial Consortium (OGC). Open Geospatial Consortium (OGC), OGC 07-107r3 (Best Practices).

[Reed, 2008] Reed, C. (2008). A Uniform Resource Name (URN) Namespace for the Open Geospatial Consortium (OGC). Internet Engineering Task Force (IETF), RFC 5165 (Informational).

[Rodosek and Lewis, 2001] Rodosek, G. D. and Lewis, L. (2001). Dynamic Service Provisioning: A User-Centric Approach. In *DSOM*, pages 37–48.

[Romberg, 1999] Romberg, M. (1999). The UNICORE Architecture: Seamless Access to Distributed Resources. In *Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing*, HPDC '99, pages 44–, Washington, DC, USA. IEEE Computer Society.

[Sahai et al., 2002] Sahai, A., Durante, A., and Machiraju, V. (2002). Towards Automated SLA Management for Web Services. Technical Report HPL-2001-310R1, HP Laboratories, Palo Alto, California.

[Sandmann, 2005] Sandmann, S. (2005). TIM-online - A part of the eGovernment strategy by the Federal State North-Rhine Westphalia. *Proceedings of the 8th AGILE International Conference on Geographic Information Science*.

[Sanford and Rose, 2007] Sanford, C. and Rose, J. (2007). Characterizing eParticipation. *International Journal of Information Management*, 27:406–421.

[Schut, 2007] Schut, P. (2007). OpenGIS Web Processing Service. Open Geospatial Consortium (OGC), OGC 05-007r7 (OpenGIS Standard).

[Schäffer, 2009] Schäffer, B. (2009). OWS-6 Geoprocessing Workflow Architecture Engineering Report. Open Geospatial Consortium (OGC), OGC 09-053r5 (Engineering Report).

[Schäffer, 2012] Schäffer, B. (2012). *Dynamic Rights Management in Cross-Domain Geoprocessing Workflows*. Dissertations in Geographic Information Science. Ios Pr Inc.

[Schäffer et al., 2010a] Schäffer, B., Baranski, B., and Foerster, T. (2010a). Licensing OGC Geoprocessing Services as a Foundation for Commercial Use in SDIs. In *Second International Conference on Advanced Geographic Information Systems, Applications and Services*, pages 111–116. IEEE Computer Society.

[Schäffer et al., 2010b] Schäffer, B., Baranski, B., and Foerster, T. (2010b). Towards Spatial Data Infrastructures in the Clouds. In Painho, M., Santos, M. Y., and Pundt, H., editors, *Geospatial Thinking. Lecture Notes in Geoinformation and Cartography*, pages 399–418. Springer.

[Schäffer et al., 2012] Schäffer, B., Baranski, B., Foerster, T., and Brauner, J. (2012). A Service-Oriented Framework for Real-time and Distributed Geoprocessing. *Geospatial Free and Open Source Software in the 21st Century. Lecture Notes in Geoinformation and Cartography*, pages 3–20.

[Schäffer and Gartmann, 2011] Schäffer, B. and Gartmann, R. (2011). Security and Licensing for Geospatial Web Services. In Zhao, P. and Di, P., editors, *Geospatial Web Services: Advances in Information Interoperability*, pages 64–95. Hershey.

[Shelly et al., 1998] Shelly, G. B., Cashman, T. J., and Rosenblatt, H. J. (1998). *Systems Analysis and Design*. International Thomson Publishing, 3rd edition.

[Shiers, 2007] Shiers, J. (2007). The Worldwide LHC Computing Grid (worldwide LCG). *Computer Physics Communications*, 177(1-2):219–223.

[Signore et al., 2005] Signore, O., Chesi, F., and Pallotti, M. (2005). E-Government: Challenges and Opportunities. In *Proceedings of the CMG Italy - XIX Annual Conference*.

[Steinmann, 2007] Steinmann, F. (2007). Pegel-Online und BS-Elbe. eGovernmentlösungen für Umweltdaten. Presentation at eGovernment Symposium, Bern, Switzerland.

[Sturm et al., 2000] Sturm, R., Morris, W., and Jander, M. (2000). *Foundations of Service Level Management*. SAMS Publishing.

[Su and Jin, 2009] Su, Y. and Jin, Z. (2009). Building Service Oriented Applications for Disaster Management - An Earthquake Assessment Example. In *Proceedings of the 2009 Fourth International Conference on Cooperation and Promotion of Information Resources in Science and Technology*, COINFO '09, pages 3–8, Washington, DC, USA. IEEE Computer Society.

[Subbiah et al., 2007] Subbiah, G., Alam, A., Khan, L., and Thuraisingham, B. (2007). Geospatial Data Qualities as Web Services Performance Metrics. In *Proceedings of the 15th International Symposium on Advances in Geographic Information Systems (ACM GIS 200)*, pages 1–4, New York, NY, USA. ACM.

[SUN, 2006] SUN (2006). JavaServer Pages Specification, Version 2.1. Online. Visited 2012-02-08, http://jcp.org/aboutJava/communityprocess/final/jsr245/index.html.

[SUN, 2009] SUN (2009). Take your business to a Higher Level - Sun cloud computing technology scales your infrastructure to take advantage of new business opportunities. Online. Visited 2012-05-12, http://www.progression.com/casestudies/studies/Sun_Cloud_Computing.pdf.

[SUN, 2012] SUN (2012). JavaServer Pages Standard Tag Library (JSTL), Version 1.2. Online. Visited 2012-02-06, http://www.oracle.com/technetwork/java/index-jsp-135995.html.

[Sun et al., 2005] Sun, W., Xu, Y., and Liu, F. (2005). The role of XML in service level agreements management. *Services Systems and Services Management, 2005. Proceedings of ICSSSM '05. 2005 International Conference on*, 2:1118–1120 Vol. 2.

[Tian et al., 2003] Tian, M., Gramm, A., Naumowicz, T., Ritter, H., and Schiller, J. (2003). A Concept for QoS Integration in Web Services. In *Proceedings of the Fourth international conference on Web information systems engineering workshops*, WISEW'03, pages 149–155, Washington, DC, USA. IEEE Computer Society.

[Toma and Foxvog, 2008] Toma, I. and Foxvog, D. (2008). Non-Functional Properties in Web services. *Web Service Modeling Ontology (WSMO) Working Group (WG) Deliverable*, D28.4 V0.1.

[Tu et al., 2004] Tu, S., Flanagin, M., Wu, Y., Abdelguerfi, M., Normand, E., Mahadevan, V., Ratcliff, J., and Shaw, K. (2004). Design Strategies to Improve Performance of GIS Web Services. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2*, ITCC '04, pages 444–, Washington, DC, USA. IEEE Computer Society.

[Vallières et al., 2005] Vallières, S., Brodeur, J., and Pilon, D. (2005). Spatial Integrity Constraints: A Tool for Improving the Internal Quality of Spatial Data. In Devillers, R. and Jeansoulin, R., editors, *Fundamentals of Spatial Data Quality*, chapter 9, pages 161–178. Hermes Science/Lavoisier.

[van Deursen et al., 2000] van Deursen, A., Klint, P., and Visser, J. (2000). Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN Not.*, 35:26–36.

[Van Loenen et al., 2012] Van Loenen, B., Janssen, K., and Welle Donker, F. (2012). Quest for a Global Standard for Geo-data Licenses. In *Spatially Enabling Government, Industry and Citizens: Research and Development Perspectives*. GSDI Association Press.

[van Oort, 2005] van Oort, P. (2005). *Spatial data quality: from description to application.* Publications on Geodesy. NCG, Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission.

[Victorian Spatial Council, 2009] Victorian Spatial Council (2009). Spatial Information Data Quality Guidelines - Part of Victoria's Spatial Information Management Framework. Technical report, Victorian Spatial Council, Department of Sustainability and Environment.

[Vogels et al., 1998] Vogels, W., Dumitriu, D., Birman, K. P., Gamache, R., Massa, M., Short, R., Vert, J., Barrera, J., and Gray, J. (1998). The Design and Architecture of the Microsoft Cluster Service - A Practical Approach to High-Availability and Scalability. In *FTCS*, pages 422–431. IEEE Computer Society.

[Vowles, 2006] Vowles, G. (2006). Geospatial Digital Rights Management Reference Model (GeoDRM RM). Open Geospatial Consortium (OGC), OGC 06-004r3 (Abstract Specification).

[Vretanos, 2010] Vretanos, P. (2010). OpenGIS Web Feature Service 2.0 Interface Standard. Open Geospatial Consortium (OGC), OGC 09-025r1 and ISO/DIS 19142 (Implementation Specification).

[W3C, 2007] W3C (2007). SOAP Version 1.2 Part 0: Primer (Second Edition). online. World Wide Web Consortium (W3C). W3C Recommendation.

[Wagner, 2006a] Wagner, R. (2006a). A Roaming-enabled SDI (rSDI): Balancing Interests, Opportunities, Investments and Risks. *GSDI 9 World Conference*.

[Wagner, 2006b] Wagner, R. (2006b). OWS-3 GeoDRM Thread Activity and Interoperability Program Report: Access Control & Terms of Use (ToU) "Click-through" IPR Management. Open Geospatial Consortium (OGC), OGC 05-111r2 (Discussion Paper).

[Wagner, 2009] Wagner, R. (2009). OpenGIS GeoRM Role Model. Open Geospatial Consortium (OGC), OGC 09-123 (Discussion Paper).

[Wang et al., 2008] Wang, S., Padmanabhan, A., Myers, J. D., Tang, W., and Liu, Y. (2008). Towards Provenance-aware Geographic Information Systems. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, GIS '08, pages 70:1–70:4, New York, NY, USA. ACM.

[Westerinen et al., 2001] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., and Waldbusser, S. (2001). Terminology for Policy-Based Management. Internet Engineering Task Force (IETF), RFC 3198 (Informational).

[Whiteside, 2005] Whiteside, A. (2005). OpenGIS Web Services Architecture Description. Open Geospatial Consortium (OGC), OGC 05-042r2 (Best Practices Paper).

[Whiteside, 2006] Whiteside, A. (2006). Definition identifier URNs in OGC namespace. Open Geospatial Consortium (OGC), OGC 06-023r1 (Best Practices).

[Whiteside and Greenwood, 2010] Whiteside, A. and Greenwood, J. (2010). OGC Web Services Common Standard. Open Geospatial Consortium (OGC), OGC 06-121r9 (Implementation Standard).

[Wieder et al., 2011] Wieder, P., Butler, J., Theilmann, W., and Yahyapour, R. (2011). *Service Level Agreements for Cloud Computing*. Springer.

[Wilde and Pautasso, 2011] Wilde, E. and Pautasso, C. (2011). *Rest: From Research to Practice*. Springer.

[Woolf and Shaon, 2009] Woolf, A. and Shaon, A. (2009). An approach to encapsulation of Grid processing within an OGC Web Processing Service. Workshop on Grid Technologies for Geospatial Applications. Presented at The 12th AGILE International Conference on Geographic Information Science, Hannover, Germany.

[Wytzisk and Sliwinski, 2004] Wytzisk, A. and Sliwinski, A. (2004). Quo Vadis SDI? In *Proceedings of the 7th AGILE Conference on Geographic Information Science*, pages 43–49.

[Wäldrich, 2011] Wäldrich, O. (2011). *Orchestration of Resources in Distributed, Heterogeneous Grid Environments Using Dynamic Service Level Agreements*. PhD thesis, Fraunhofer Institute for Scientific Computing and Algorithms, Irvine, California.

[Yavatkar et al., 2000] Yavatkar, R., Pendarakis, D., and Guerin, R. (2000). A Framework for Policy-based Admission Control. Internet Engineering Task Force (IETF), RFC 2753 (Informational).

[Yin et al., 2010] Yin, H.-Y., Huang, C.-J., Fang, Y.-M., Lee, B.-J., and Chou, T.-Y. (2010). The Present Development of Debris Flow Monitoring Technology in Taiwan. *Proceedings of the International Symposium in Pacific Rim (INTERPRAEVENT 2010), Taipei, Taiwan*, pages 992–1000.

[Yu-jie et al., 2005] Yu-jie, M., Jian, C., Shen-sheng, Z., and Jian-hong, Z. (2005). Interactive Web Service Choice-Making Based on Extended QoS Model. In *Proceedings of the The Fifth International Conference on Computer and Information Technology*, CIT '05, pages 1130–1134, Washington, DC, USA. IEEE Computer Society.

[Zhang et al., 2010] Zhang, S., Zhang, S., Chen, X., and Huo, X. (2010). Cloud Computing Research and Development Trend. *2010 Second International Conference on Future Networks*, pages 93–97.

[Zhou et al., 2005] Zhou, C., Chia, L.-T., and Lee, B.-S. (2005). QoS Measurement Issues with DAML-QoS Ontology. In *Proceedings of the IEEE International Conference on e-Business Engineering*, ICEBE '05, pages 395–403, Washington, DC, USA. IEEE Computer Society.

# Appendix A

# Requirements Analysis

Table A.1 provides an overview about all evaluated requirements.

Table A.1: Requirements Overview

| Requirement | Description |
|---|---|
| **Roles and Relationships** | |
| R1 | The web-based SLA management architecture shall enable service consumers and service providers to negotiate SLA prior the service consumption. |
| R2 | The web-based SLA management architecture shall ensure that the basic publish-find-bind pattern in SDIs remains. |
| R3 | The service provider shall be able to publish his service along with additional SLA related information to the service broker. |
| R4 | The service consumer shall be able to perform service discovery operations on the service broker to find an adequate service provider according to SLA related requirements. |
| R5 | The web-based SLA management architecture shall ensure that service consumption is performed only under the terms of previously created agreements, in which service consumers and service providers agree to certain terms and conditions. |
| **Services, Resources and Quality** | |
| R6 | The abstract SLA model shall allow to create service offerings reflecting the service types that are standardized by the OGC and described by the INSPIRE directive. |
| R7 | The abstract SLA model shall allow the definition of KPIs and SLOs reflecting the functional requirements as defined by the INSPIRE directive. |

Table A.1 – Continued on next page

Table A.1 – Continued from previous page

| Requirement | Description |
|---|---|
| R8 | The web-based SLA management architecture shall allow the monitoring of KPIs and the evaluation of SLOs reflecting the functional requirements as defined by the INSPIRE directive. |
| R9 | The abstract SLA model shall allow the definition of spatial data themes as defined by the INSPIRE directive. |
| R10 | The abstract SLA model shall allow the definition of data quality elements as defined by the INSPIRE directive. |
| R11 | The abstract SLA model shall allow the definition of license models for the provision of access to spatial datasets as recognized by the INSPIRE directive and implemented by the GeoLizenz license models. |
| R12 | The abstract SLA model shall allow the definition of KPIs and SLOs reflecting the minimum performance criteria as defined by the INSPIRE directive. |
| R13 | The web-based SLA management architecture shall allow the monitoring of KPIs and the evaluation of SLOs reflecting the minimum performance criteria as defined by the INSPIRE directive. |
| **Pricing and Accounting** | |
| R14 | The abstract SLA model shall allow the definition of complex pricing models as described by the "AdV-Gebührenrichtlinie" and the "VermWertGebT". |
| R15 | The web-based SLA management architecture shall allow the accounting of service offerings aligned with complex pricing models as described by the "AdV-Gebührenrichtlinie" and the "VermWertGebT". |
| **Security and Rights Management** | |
| R16 | The abstract SLA model shall allow the integration of access rights policies that are based on existing approaches such as GeoXACML. |
| R17 | The web-based SLA management architecture shall allow the enforcement of access rights policies that are based on existing approaches such as GeoXACML. |
| **Infrastructure Management** | |
| R18 | The abstract SLA model shall allow the definition of KPIs and SLOs reflecting infrastructure requirements and capabilities. |

Table A.1 – Continued on next page

Table A.1 – Continued from previous page

| Requirement | Description |
|---|---|
| R19 | The web-based SLA management architecture shall allow the monitoring of KPIs and the evaluation of SLOs reflecting infrastructure requirements and capabilities. |
| R20 | The abstract SLA model shall allow the definition of complex pricing models reflecting the service infrastructure utilization. |
| R21 | The web-based SLA management architecture shall allow the accounting of service offerings aligned with complex pricing models reflecting the service infrastructure utilization. |
| R22 | The abstract SLA model shall allow the definition of infrastructure management information in order to realize differentiated services under the terms of previously created agreements. |
| R23 | The web-based SLA management architecture shall support strategies for realizing differentiated services under the terms of previously created agreements. |
| **Standards and Technology** | |
| R24 | The abstract SLA model and the web-based SLA management architecture shall be developed with respect to the OGC Standards Baseline. They shall not replace any previous OGC specifications, but should depend and build on them. |
| R25 | The development of a standardized document encoding for the abstract SLA model is strongly recommended. |
| R26 | The development of a standardized communication protocol for realizing SLA negotiation and service consumption under the terms of previously created SLAs is strongly recommended. |
| R27 | The abstract SLA model and the web-based SLA management architecture should be designed in a flexible way in order to be applicable in other application domains. |

# Appendix B

# Service Level Agreement Formalization

This chapter provides additional information about the abstract SLA model.

## B.1 Monitoring Functions

The abstract SLA model offers a basic set of functions that can be accessed from within the Agreement Expression Language scripts in order to create varying monitoring requests. The basic set of functions is aligned to the functions that are available in Apache JMeter[1]. The structure of the functions looks like

```
${__functionName(var1, var2, varN)}
```

where `__functionName` matches the name of a function and `var1` to `varN` reference the input arguments that are passed to the function. Table B.1 provides an overview about the functions that are supported in the abstract SLA model by default.

Table B.1: Active Monitoring Functions

| |
|---|
| `${__random(min, max)}` |
| This function returns a random floating point number that is between the `min` and `max` floating point parameters.<br>Example<br>The following function can be used to create varying BBOX parameters for a `GetMap` request.<br>`${__random(142.0, 144.0)}` |
| `${__random(str1, str2, strN)}` |
| This function returns a random string from the set of one or more string parameters. |

Table B.1 – Continued on next page

[1] http://jmeter.apache.org

Table B.1 – Continued from previous page

| |
|---|
| Example<br>The following function can be used to query varying layers in a `GetMap` request.<br>`${__random("topp:tasmania_cities", "topp:tasmania_roads", "topp:tasmania_state_boundaries")}` |

The basic set of functions can be extended in order to accommodate the needs of further applications and use cases.

## B.2 OGC URN Schema Extension

The abstract SLA model defines an OGC URN Schema Extension in order to identify service property types and business value types. The following sections provide a comprehensive dictionary of all URNs that can be used in the abstract SLA model.

### B.2.1 Service Property Types

Table B.2 - Table B.7 provide an overview and explanation of all URNs that can be used to identify domain-specific service properties in the *Service Properties* section of the abstract SLA model. All URNs are structured in accordance with the OGC URN Schema described in [Reed, 2008] and have the form

```
urn:ogc:def:sla:property:{URN}
```

where the `URN` token must be one of the following URNs.

Table B.2: Resource-Related Service Property Types

| URN | Description |
|---|---|
| `resource:operation` | The standardized operations implemented by the service instance. |
| `resource:feature` | The objects provided by the service instance. |
| `resource:layer` | The layers provided by the service instance. |
| `resource:process` | The processes provided by the service instance. |

Table B.3: Runtime-Related Service Property Types

| URN | Description |
|---|---|
| `runtime:availability` | The probability whether the service is up and running. |
| `runtime:response` | The response time of the service (the time between sending the request and receiving the response). |

Table B.4: Usage-Related Service Property Types

| URN | Description |
|---|---|
| `usage:request` | The total number of times the service consumer accessed the service. |
| `usage:operation` | The total number of times the service consumer calls a standardized operation at the web service. |
| `usage:object` | The total number of geometric objects delivered to the service consumer. |
| `usage:pixel` | The total number of pixels delivered to the service consumer. |
| `usage:area` | The total covered area delivered to the service consumer. |
| `usage:process` | The total number of times the service consumer executes a process at the web service. |
| `usage:transfer` | The total amount of transferred data from and to the web service. |
| `usage:cpu` | The total amount of CPU utilization caused by service consumption. |

Table B.5: Data-Related Service Property Types

| URN | Description |
|---|---|
| `data:completeness:commission` | The number of excess items in the delivered dataset in relation to the number of items that should have been present. |

Table B.5 – Continued on next page

Table B.5 – Continued from previous page

| URN | Description |
|---|---|
| `data:completeness:omission` | The number of missing items in the delivered dataset in relation to the number of items that should have been present. |
| `data:accuracy:absolute` | A number that indicates the absolute accuracy of the data set. |
| `data:accuracy:external` | A number that indicates the external accuracy of the data set. |
| `data:accuracy:resolution` | A number that indicates the spatial resolution of delivered geometry. |
| `data:license` | An identifier of the license for data sharing and reuse in another application context. |

Table B.6: Security-Related Service Property Types

| URN | Description |
|---|---|
| `security:license:geoxacml` | The access rights policies defined in the GeoXACML format. |

Table B.7: Infrastructure-Related Service Property Types

| URN | Description |
|---|---|
| `infrastructure:provider:name` | The unique name of the infrastructures provider. |
| `infrastructure:provider:region` | The unique name of the region where the service provider is located. |
| `infrastructure:vm:name` | The unique name of the VM template that provides the service deployment. |
| `infrastructure:compute:architecture` | The CPU architecture of the compute resource which hosts the service. |
| `infrastructure:compute:cores` | The number of available CPU cores at the compute resource which hosts the service. |

Table B.7 – Continued on next page

Table B.7 – Continued from previous page

| URN | Description |
|---|---|
| `infrastructure:compute:speed` | The CPU clock speed for each available CPU core. |
| `infrastructure:compute:memory` | The available RAM of the compute resource which hosts the service. |

This comprehensive dictionary of URNs for identifying domain-specific service properties can be extended in order to accommodate the needs of further application domains and use cases. For the purpose of this thesis the provided list of URNs is quite sufficient to meet all requirements.

### B.2.2 Business Value Types

Table B.8 provides an overview and explanation of all URNs that can be used to identify domain-specific business values in the *Business Values* section of the abstract SLA model. All URNs are structured in accordance with the OGC URN Schema described in [Reed, 2008] and have the form

```
urn:ogc:def:sla:business:{URN}
```

where the `URN` token must be one of the following URNs.

Table B.8: Business Value Types

| URN | Description |
|---|---|
| `cost:day` | The cost to be assessed for using the service on a daily basis. |
| `cost:week` | The cost to be assessed for using the service on a weekly basis. |
| `cost:month` | The cost to be assessed for using the service on a monthly basis. |
| `cost:year` | The cost to be assessed for using the service on a yearly basis. |
| `penalty:day` | The penalty to be assessed for not meeting service level objectives on a daily basis. |
| `penalty:week` | The penalty to be assessed for not meeting service level objectives on a weekly basis. |
| `penalty:month` | The penalty to be assessed for not meeting service level objectives on a monthly basis. |

Table B.8 – Continued on next page

Table B.8 – Continued from previous page

| URN | Description |
|---|---|
| `penalty:year` | The penalty to be assessed for not meeting service level objectives on a yearly basis. |
| `reward:day` | The reward to be assessed for meeting service level objectives on a daily basis. |
| `reward:week` | The reward to be assessed for meeting service level objectives on a weekly basis. |
| `reward:month` | The reward to be assessed for meeting service level objectives on a monthly basis. |
| `reward:year` | The reward to be assessed for meeting service level objectives on a yearly basis. |

This comprehensive dictionary of URNs for identifying domain-specific business values can be extended in order to accommodate the needs of further application domains and use cases. For the purpose of this thesis the provided list of URNs is quite sufficient to meet all requirements.

## B.3 Agreement Expression Language

The abstract SLA model defines a DSL in order to define service level objectives and business values. The following sections provide a comprehensive dictionary of all context variables and functions that can be used in the abstract SLA model.

### B.3.1 Variables

All resource-, runtime- and infrastructure-related `Service Property` elements lead to custom JEXL context variables that can be used to access monitoring information that are collected during agreement runtime. Furthermore, all `Service Level Objective` elements and all `Business Value` elements lead also to JEXL context variables that can be used to access agreement status information and to create nested pricing models.

#### SERVICE PROPERTIES

The following sections describe the JEXL context variable types that result from resource-, runtime- and infrastructure-related service properties in the *Service Properties* section of the abstract SLA model.

##### Resource-Related Variables

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:resource:operation
```

result in a JEXL context variable from type `ResourceOperationType` that has the following variables and methods (Table B.9).

Table B.9: JEXL `ResourceOperationType`

| `ResourceOperationType` | |
|---|---|
| `String[]` | **`name`**<br>Returns an array of all available OWS operations. |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:resource:feature
```

result in a JEXL context variable from type `ResourceFeatureType` that has the following variables and methods (Table B.10).

Table B.10: JEXL `ResourceFeatureType`

| `ResourceFeatureType` | |
|---|---|
| `String[]` | **`name`**<br>Returns an array of all available objects (names of feature type instances). |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:resource:layer
```

result in a JEXL context variable from type `ResourceLayerType` that has the following variables and methods (Table B.11).

Table B.11: JEXL `ResourceLayerType`

| `ResourceLayerType` | |
|---|---|
| `String[]` | **`name`**<br>Returns an array of all available layers (names of layers). |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:resource:process
```

result in a JEXL context variable from type `ResourceProcessType` that has the following variables and methods (Table B.12).

Table B.12: JEXL `ResourceProcessType`

| `ResourceProcessType` | |
|---|---|
| `String[]` | **`name`**<br>Returns an array of all available processes (names of processes). |

### Runtime-Related Variables

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:runtime:availability
```

result in a JEXL context variable from type `AvailabilityType` that has the following variables and methods (Table B.13).

Table B.13: JEXL `AvailabilityType`

| `AvailabilityType` | |
|---|---|
| `float` | **`day`**<br>Returns the measured service availability for the current day in percent (where 0.0 means 0% and 1.0 means 100%). |
| `float` | **`week`**<br>Returns the measured service availability for the current calendar week in percent (where 0.0 means 0% and 1.0 means 100%). |
| `float` | **`month`**<br>Returns the measured service availability for the current calendar month in percent (where 0.0 means 0% and 1.0 means 100%). |
| `float` | **`year`**<br>Returns the measured service availability for the current calendar year in percent (where 0.0 means 0% and 1.0 means 100%). |
| `float` | **`total`**<br>Returns the measured service availability for the complete agreement runtime in percent (where 0.0 means 0% and 1.0 means 100%). |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:runtime:response
```

result in a JEXL context variable from type `ResponseType` that has the following variables and methods (Table B.14).

Table B.14: JEXL `ResponseType`

| `ResponseType` | |
|---|---|
| `InitialResponseType` | `initial`<br>Returns an object of type `InitialResponseType` that provides information about the measured initial response time of the service. |
| `TotalResponseType` | `total`<br>Returns an object of type `TotalResponseType` that provides information about the measured total response time of the service. |

Table B.15: JEXL `InitialResponseType`

| `InitialResponseType` | |
|---|---|
| `int[]` | `day`<br>Returns an array of all response time measurements for the current day. Each element in the array represents the measured initial response time of a successful request in milliseconds. |
| `int[]` | `week`<br>Returns an array of all response time measurements for the current calendar week. Each element in the array represents the measured initial response time of a successful request in milliseconds. |
| `int[]` | `month`<br>Returns an array of all response time measurements for the current calendar month. Each element in the array represents the measured initial response time of a successful request in milliseconds. |

Table B.15 – Continued on next page

Table B.15 – Continued from previous page

| InitialResponseType | |
|---|---|
| int[] | year<br>Returns an array of all response time measurements for the current calendar year. Each element in the array represents the measured initial response time of a successful request in milliseconds. |
| int[] | total<br>Returns an array of all response time measurements for the complete agreement runtime. Each element in the array represents the measured initial response time of a successful request in milliseconds. |

Table B.16: JEXL TotalResponseType

| TotalResponseType | |
|---|---|
| int[] | day<br>Returns an array of all response time measurements for the current day. Each element in the array represents the measured total response time of a successful request in milliseconds. |
| int[] | week<br>Returns an array of all response time measurements for the current calendar week. Each element in the array represents the measured total response time of a successful request in milliseconds. |
| int[] | month<br>Returns an array of all response time measurements for the current calendar month. Each element in the array represents the measured total response time of a successful request in milliseconds. |
| int[] | year<br>Returns an array of all response time measurements for the current calendar year. Each element in the array represents the measured total response time of a successful request in milliseconds. |
| int[] | total<br>Returns an array of all response time measurements for the complete agreement runtime. Each element in the array represents the measured total response time of a successful request in milliseconds. |

### Usage-Related Variables

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:usage:request
```

result in a JEXL context variable from type `RequestType` that has the following variables and methods (Table B.17).

Table B.17: JEXL `RequestType`

| `RequestType` | |
|---|---|
| `long` | `day`<br>Returns the logged number of service requests for the current day. |
| `long` | `week`<br>Returns the logged number of service requests for the current calendar week. |
| `long` | `month`<br>Returns the logged number of service requests for the current calendar month. |
| `long` | `year`<br>Returns the logged number of service requests for the current calendar year. |
| `long` | `total`<br>Returns the logged number of service requests for the complete agreement runtime. |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:usage:operation
```

result in a JEXL context variable from type `OperationType` that has the following variables and methods (Table B.18).

Table B.18: JEXL `OperationType`

| `OperationType` | |
|---|---|
| `long` | `day`[a]<br>Returns the logged number of OWS operation calls for the current day. |

Table B.18 – Continued on next page

Table B.18 – Continued from previous page

| `OperationType` | |
|---|---|
| `long` | **`week`**[a]<br>Returns the logged number of OWS operation calls for the current calendar week. |
| `long` | **`month`**[a]<br>Returns the logged number of OWS operation calls for the current calendar month. |
| `long` | **`year`**[a]<br>Returns the logged number of OWS operation calls for the current calendar year. |
| `long` | **`total`**[a]<br>Returns the logged number of OWS operation calls for the complete agreement runtime. |
| `Map<String, OperationType>` | **`operation`**<br>Returns a map containing objects of type `OperationType` that provide information about the logged number of operation calls for a specific OWS operation. The string key of the map is the name of the OWS operation. If there are no information available for a designated operation the map returns `null`. |
| [a] In cases of vendor-specific and non-standard operation calls, this number may differ from the total number of service requests. | |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:usage:object
```

result in a JEXL context variable from type `ObjectType` that has the following variables and methods (Table B.19).

Table B.19: JEXL `ObjectType`

| `ObjectType` | |
|---|---|
| `long` | **`day`**<br>Returns the accumulated amount of delivered objects for the current day. |
| `long` | **`week`**<br>Returns the accumulated amount of delivered objects for the current calendar week. |

Table B.19 – Continued on next page

Table B.19 – Continued from previous page

| `ObjectType` | |
|---|---|
| `long` | `month`<br>Returns the accumulated amount of delivered objects for the current calendar month. |
| `long` | `year`<br>Returns the accumulated amount of delivered objects for the current calendar year. |
| `long` | `total`<br>Returns the accumulated amount of delivered objects for the complete agreement runtime. |
| `Map<String, ObjectType>` | `feature`<br>Returns a map containing objects of type `ObjectType` that provide information about the accumulated amount of delivered objects for a specific feature type that was queried. The string key of the map is the name of a feature type instance. If there are no information available for a designated feature type the map returns `null`. |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:usage:pixel
```

result in a JEXL context variable from type `PixelType` that has the following variables and methods (Table B.20).

Table B.20: JEXL `PixelType`

| `PixelType` | |
|---|---|
| `long` | `day`<br>Returns the accumulated amount of delivered pixel of all layers and for the current day. |
| `long` | `week`<br>Returns the accumulated amount of delivered pixel of all layers and for the current calendar week. |
| `long` | `month`<br>Returns the accumulated amount of delivered pixel of all layers and for the current calendar month. |

Table B.20 – Continued on next page

Table B.20 – Continued from previous page

| `PixelType` | |
|---|---|
| `long` | `year`<br>Returns the accumulated amount of delivered pixel of all layers and for the current calendar year. |
| `long` | `total`<br>Returns the accumulated amount of delivered pixel of all layers and for the complete agreement runtime. |
| `Map<String, PixelType>` | `layer`<br>Returns a map containing objects of type `PixelType` that provide information about the accumulated amount of delivered pixel for a specific layer. The string key of the map is the name of a layer. If there are no information available for a designated layer the map returns `null`. |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:usage:area
```

result in a JEXL context variable from type `AreaType` that has the following variables and methods (Table B.21).

Table B.21: JEXL `AreaType`

| `AreaType` | |
|---|---|
| `float` | `day`<br>Returns the accumulated area that is requested for the current day in square kilometer ($km^2$). |
| `float` | `week`<br>Returns the accumulated area that is requested for the current calendar week in square kilometer ($km^2$). |
| `float` | `month`<br>Returns the accumulated area that is requested for the current calendar month in square kilometer ($km^2$). |
| `float` | `year`<br>Returns the accumulated area that is requested for the current calendar year in square kilometer ($km^2$). |

Table B.21 – Continued on next page

Table B.21 – Continued from previous page

| `AreaType` | |
|---|---|
| `float` | `total`<br>Returns the accumulated area that is requested for the complete agreement runtime in square kilometer ($km^2$). |
| `Map<String, AreaType>` | `layer`<br>Returns a map containing objects of type `AreaType` that provide information about the accumulated area that is requested for a specific layer. The string key of the map is the name of a layer. If the service does not support layers, this method returns `null`. |
| `Map<String, AreaType>` | `feature`<br>Returns a map containing objects of type `AreaType` that provide information about the accumulated area that is requested for a specific feature type that was queried. The string key of the map is the name of a feature type instance. If the service does not support vector data, this method returns `null`. |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:usage:process
```

result in a JEXL context variable from type `ProcessType` that has the following variables and methods (Table B.22).

Table B.22: JEXL `ProcessType`

| `ProcessType` | |
|---|---|
| `long` | `day`<br>Returns the logged number of process executions for the current day. |
| `long` | `week`<br>Returns the logged number of process executions for the current calendar week. |
| `long` | `month`<br>Returns the logged number of process executions for the current calendar month. |

Table B.22 – Continued on next page

Table B.22 – Continued from previous page

| ProcessType | |
|---|---|
| long | year<br>Returns the logged number of process executions for the current calendar year. |
| long | total<br>Returns the logged number of process executions for the complete agreement runtime. |
| Map<String, ProcessType> | process<br>Returns a map containing objects of type ProcessType that provide information about the logged number of process executions for a specific process. The string key of the map is the name of the process. If there are no information available for a designated process the map returns null. |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:usage:transfer
```

result in a JEXL context variable from type TransferType that has the following variables and methods (Table B.23).

Table B.23: JEXL TransferType

| TransferType | |
|---|---|
| TransferInType | in<br>Returns an object of type TransferInType that provides information about the logged amount of data that was transferred to the service. |
| TransferOutType | out<br>Returns an object of type TransferOutType that provides information about the logged amount of data that was delivered by the service. |

Table B.24: JEXL `TransferInType`

| `TransferInType` | |
|---|---|
| `long` | `day`<br>Returns the accumulated amount of data that was transferred to the service for the current day in megabyte (MB). |
| `long` | `week`<br>Returns the accumulated amount of data that was transferred to the service for the current calendar week in megabyte (MB). |
| `long` | `month`<br>Returns the accumulated amount of data that was transferred to the service for the current calendar month in megabyte (MB). |
| `long` | `year`<br>Returns the accumulated amount of data that was transferred to the service for the current calendar year in megabyte (MB). |
| `long` | `total`<br>Returns the accumulated amount of data that was transferred to the service for the complete agreement runtime in megabyte (MB). |

Table B.25: JEXL `TransferOutType`

| `TransferOutType` | |
|---|---|
| `long` | `day`<br>Returns the accumulated amount of data that was delivered by the service for the current day in megabyte (MB). |
| `long` | `week`<br>Returns the accumulated amount of data that was delivered by the service for the current calendar week in megabyte (MB). |
| `long` | `month`<br>Returns the accumulated amount of data that was delivered by the service for the current calendar month in megabyte (MB). |

Table B.25 – Continued on next page

Table B.25 – Continued from previous page

| TransferOutType | |
|---|---|
| long | year<br>Returns the accumulated amount of data that was delivered by the service for the current calendar year in megabyte (MB). |
| long | total<br>Returns the accumulated amount of data that was delivered by the service for the complete agreement runtime in megabyte (MB). |

All service properties in the *Service Properties* section of the abstract SLA model with type

```
urn:ogc:def:sla:property:usage:cpu
```

result in a JEXL context variable from type `CpuType` that has the following variables and methods (Table B.26).

Table B.26: JEXL `CpuType`

| CpuType | |
|---|---|
| long | day<br>Returns the CPU utilization for the current day (e.g. in clock ticks or seconds). |
| long | week<br>Returns the CPU utilization for the current calendar week (e.g. in clock ticks or seconds). |
| long | month<br>Returns the CPU utilization for the current calendar month (e.g. in clock ticks or seconds). |
| long | year<br>Returns the CPU utilization for the current calendar year (e.g. in clock ticks or seconds). |
| long | total<br>Returns the CPU utilization for the complete agreement runtime (e.g. in clock ticks or seconds). |

#### BUSINESS VALUES

All service level objectives in the abstract SLA model result in a JEXL context variable from type `ObjectiveType` that has the following variables and methods (Table B.27).

Table B.27: JEXL `ObjectiveType`

| `ObjectiveType` | |
|---|---|
| `Boolean` | `status`<br>Returns `true` if the corresponding service level objective is fulfilled and `false` if the corresponding service level objective is violated. |

All business values in the abstract SLA model result in a JEXL context variable from type `BusinessType` that has the following variables and methods (Table B.28).

Table B.28: JEXL `BusinessType`

| `BusinessType` | |
|---|---|
| `float` | `value`<br>Returns the value of the corresponding business value (normally the rate in Euro). |

The list of variable types can be extended in order to accommodate the needs of further application domains and use cases. For the purpose of this thesis the provided list of variable types is quite sufficient to meet all requirements.

### B.3.2 Functions

Beside the custom context variables, the JEXL language provides some basic functions. Table B.29 provides an overview about the functions that are supported by default. A more detailed overview can be found at the JEXL homepage.

Table B.29: DSL Functions

| `empty` |
|---|
| This function returns `true` if the expression following is either `null`, an empty string, an array of length zero, a collection of size zero, or an empty map. |

| `size` |
|---|
| This function returns the length of an Array, the size of a List, the size of a Map, the size of a Set, or the length of a string. |

The list of functions can be extended in order to accommodate the needs of further application domains and use cases. For the purpose of this thesis the provided list of functions is quite sufficient to meet all requirements.

## B.4 Agreement Example

Listing B.1 shows an example abstract SLA model in JSON format.

Listing B.1: Example Abstract SLA Model

```
{
  /* ######################################## */
  /* AGREEMENT CONTEXT                        */
  /* ######################################## */
  "Agreement Context":
  {
    "Service Provider":
    {
      "Name": "Institute for Geoinformatics",
      "Site:" "http://www.ifgi.de",
      "Contact":
      {
        "IndividualName": "Bastian Baranski",
        "PositionName": "Research Associate",
        "ContactInfo":
        {
          "Phone":
          {
            "Voice": "+49 251 8333071",
            "Facsimile": "+49 251 8339763"
          },
          "Address":
          {
            "DeliveryPoint": "Weseler Strasse 253",
            "City": "Muenster",
            "PostalCode": "48151",
            "Country": "Germany",
            "ElectronicMailAddress": "baranski@uni-muenster.de"
          },
          "HoursOfService": "The hours of service are Monday to Friday from
              8 AM to 16 PM.",
          "ContactInstructions": "Please contact the service desk via phone
              or mail."
        }
      }
    },
    "Service Consumer":
    {
      "Name": null,
      "Site:" null,
      "Contact":
      {
        "IndividualName": "Bastian Baranski",
        "PositionName": null,
        "ContactInfo":
        {
          "Phone":
          {
            "Voice": "+49 251 8333071",
            "Facsimile": null
          },
          "Address":
          {
            "DeliveryPoint": "Weseler Strasse 253",
            "City": "Muenster",
            "PostalCode": "48151",
            "Country": "Germany",
            "ElectronicMailAddress": "bastian.baranski@uni-muenster.de"
          },
          "HoursOfService": null,
          "ContactInstructions": null
        }
      }
    },
```

```
63      "Contract Detail":
64      {
65        "Contract Period":
66        {
67          "Start": "2010-07-04T13:00:00+02:00",
68          "End": "2012-07-09T13:00:00+02:00"
69        }
70      }
71    },
72    "Service Terms":
73    {
74      /* ######################################## */
75      /* SERVICE DESCRIPTION                      */
76      /* ######################################## */
77      "Service Description":
78      {
79        "Title": "INSPIRE View Service",
80        "Abstract": "This service instance is an INSPIRE View Service
              implementation.",
81        "Keywords": "INSPIRE, View Service, OGC, WMS"
82        "Type": "urn:ogc:doc:is:wms:1.1.1",
83      },
84      /* ######################################## */
85      /* SERVICE REFERENCE                        */
86      /* ######################################## */
87      "Servie Reference":
88      {
89        "URL": "http://server:port/path"
90      },
91      /* ######################################## */
92      /* SERVICE PROPERTIES                       */
93      /* ######################################## */
94      "Service Properties":
95      {
96        /* RESOURCE-RELATED PROPERTIES */
97        "Service Property":
98        {
99          "Name": "operations",
100         "Title": "Supported Operations",
101         "Abstract": "The operations that are supported by the service.",
102         "Type": "urn:ogc:def:sla:property:resource:operation",
103         "Monitoring":
104         {
105           "ActiveMonitoring":
106           {
107             "Start": "00:00:00",
108             "Stop": "24:00:00",
109             "Period": 360000,
110           }
111         }
112       },
113       /* RUNTIME-RELATED PROPERTIES */
114       "Service Property":
115       {
116         "Name": "availability",
117         "Title": "Service Availability",
118         "Abstract": "The general availability of the service.",
119         "Type": "urn:ogc:def:sla:property:runtime:availability",
120         "Monitoring":
121         {
122           "ActiveMonitoring":
123           {
124             "Start": "00:00:00",
125             "Stop": "24:00:00",
126             "Period": 360000,
127             "Request":
128             {
129               "Method": "GET",
130               "Content": "service=WMS&version=1.3.0&request=GetMap&layers=
                      topp:tasmania_state_boundaries&styles=&bbox=${__random
                      (142.0,144.0)},${__random(-46.0,-44.0)},${__random
                      (150.0,152.0)},${__random(-38.0,-36.0)}&width=800&height
                      =600&srs=EPSG:4326&format=image/png"
```

```
            },
            "Response":
            {
              "Status": "200",
            }
          }
        }
      },
      "Service Property":
      {
        "Name": "response",
        "Title": "Response Time",
        "Abstract": "The response time of the service.",
        "Type": "urn:ogc:def:sla:property:runtime:response",
        "Monitoring":
        {
          "ActiveMonitoring":
          {
            "Start": "00:00:00",
            "Stop": "24:00:00",
            "Period": 360000,
            "Request":
            {
              "Method": "GET",
              "Content": "service=WMS&version=1.3.0&request=GetMap&layers=
                  topp:tasmania_state_boundaries&styles=&bbox=${__random
                  (142.0,144.0)},${__random(-46.0,-44.0)},${__random
                  (150.0,152.0)},${__random(-38.0,-36.0)}&width=800&height
                  =600&srs=EPSG:4326&format=image/png"
            },
            "Response":
            {
              "Status": "200",
            }
          }
        }
      },
      "Service Property":
      {
        "Name": "capacity",
        "Title": "Service Capacity",
        "Abstract": "The response time of the service for multiple parallel
            requests.",
        "Type": "urn:ogc:def:sla:property:runtime:response",
        "Monitoring":
        {
          "ActiveMonitoring":
          {
            "Start": "20:00:00",
            "Stop": "04:00:00",
            "Period": 3600000,
            "Session":
            {
              "Capacity": 20,
              "Duration": 60000,
              "Period:": 1000
            },
            "Request":
            {
              "Chance": 10,
              "Method": "GET",
              "Content": "service=WMS&version=1.3.0&request=GetCapabilities"
            },
            "Request":
            {
              "Chance": 90,
              "Method": "GET",
              "Content": "service=WMS&version=1.3.0&request=GetMap&layers=
                  topp:tasmania_state_boundaries&styles=&bbox=${__random
                  (142.0,144.0)},${__random(-46.0,-44.0)},${__random
                  (150.0,152.0)},${__random(-38.0,-36.0)}&width=800&height
                  =600&srs=EPSG:4326&format=image/png"
            },
```

```
            "Response":
            {
              "Status": "200",
            }
          }
        }
      },
      /* USAGE-RELATED PROPERTIES */
      "Service Property":
      {
        "Name": "pixel",
        "Title": "Sum of Pixels",
        "Abstract": "The accessed number of pixels.",
        "Type": "urn:ogc:def:sla:property:usage:pixel",
        "Monitoring":
        {
          "PassiveMonitoring":
          {
            "Request":
            {
              "GET":
              {
                "Resource": "/state/urn:ogc:def:sla:property:usage:pixel"
                "Method": "GET"
              }
            }
          }
        }
      },
      /* DATA-RELATED PROPERTIES */
      "Service Property":
      {
        "Name": "license",
        "Title": "Data License",
        "Abstract": "...",
        "Type": "urn:ogc:def:sla:property:data:license:",
        "Value": "http://www.geolizenz.org/modules/geolizenz/docs/1.1/
            GeoLizenz_V1-1
            _Ia_kommerziell_Weiterverarbeitung_oeffentliche_Netzwerke_110831
            .pdf"
      },
      /* SECURITY-RELATED PROPERTIES */
      "Service Property":
      {
        "Name": "license",
        "Title": "Data License",
        "Abstract": "...",
        "Type": "urn:ogc:def:sla:property:data:license:",
        "Value":
        "
        "
      },
      /* INFRASTRUCTURE-RELATED PROPERTIES */
      "Service Property":
      {
        "Name": "provider",
        "Title": "Infrastructure Provider",
        "Abstract": "The name of the infrastructures provider.",
        "Type": "urn:ogc:def:sla:property:infrastructure:provider:name",
        "Value": "default"
      },
      "Service Property":
      {
        "Name": "image",
        "Title": "Virtual Machine",
        "Abstract": "The name of the Virtual Machine (VM) template.",
        "Type": "urn:ogc:def:sla:property:infrastructure:vm:name",
        "Value": "ami-59f9c62d"
      }
    }
  },
  "Guarantee Terms":
  {
```

```
265       /* ######################################## */
266       /* THE SERVICE LEVEL OBJECTIVES SECTION     */
267       /* ######################################## */
268       "Service Level Objectives":
269       {
270         /* INSPIRE OPERATIONS REQUIREMENTS */
271         "Service Level Objective":
272         {
273           "Name": "InspireOperations"
274           "Title": "INSPIRE (Operations)",
275           "Abstract": "The following operations shall be implemented for an
                  INSPIRE View service: GetCapabilities, GetMap.",
276           "Obligated": "Service Provider",
277           "Status": "
278             isGetCapabilities = false;
279             isGetMap = false;
280             for (item : operations.name)
281             {
282               if (item.equalsIgnoreCase('GetCapabilities'))
283               {
284                 isGetCapabilities = true;
285               }
286               if (item.equalsIgnoreCase('GetMap'))
287               {
288                 isGetMap = true;
289               }
290             }
291             (isGetCapabilities and isGetMap);
292           "
293         },
294         /* INSPIRE QUALITY OF SERVICE REQUIREMENTS */
295         "Service Level Objective":
296         {
297           "Name": "InspireAvailability"
298           "Title": "INSPIRE (Availability)",
299           "Abstract": "The probability of a Network Service to be available
                  shall be 99% of the time.",
300           "Obligated": "Service Provider",
301           "Status": "
302             (availability.week >= 0.99) and (availability.month >= 0.99) and (
                    availability.year >= 0.99)
303           "
304         },
305         "Service Level Objective":
306         {
307           "Name": "InspirePerformance"
308           "Title": "INSPIRE (Performance)",
309           "Abstract": "The response time for sending the initial response to a
                   Get Map Request to a view service shall be maximum 5 seconds in
                   normal situation.",
310           "Obligated": "Service Provider",
311           "Status": "
312             fulfilled = 0;
313             for (item : response.initial.week) {
314               if (item lt 5000)
315               {
316                 fulfilled = fulfilled + 1;
317               }
318             }
319             percent = fulfilled / (size(response.initial.week) / 100.0);
320             percent gt 90.0;
321           "
322         },
323         "Service Level Objective":
324         {
325           "Name": "InspireCapacity"
326           "Title": "INSPIRE (Capacity)",
327           "Abstract": "The minimum number of served simultaneous service
                  requests to a view service according to the performance quality
                  of service shall be 20 per second.",
328           "Obligated": "Service Provider",
329           "Status": "
330             fulfilled = 0;
```

```
331             for (item : capacity.initial.week) {
332               if (item lt 5000)
333               {
334                 fulfilled = fulfilled + 1;
335               }
336             }
337             percent = fulfilled / (size(capacity.initial.week) / 100.0);
338             percent gt 90.0;
339           "
340         }
341       },
342       /* ######################################## */
343       /* THE BUSINESS VALUES SECTION              */
344       /* ######################################## */
345       "Business Values"
346       {
347         /* AdV PRICING MODEL FOR ONLINE DATA DELIVERY */
348         "Business Value":
349         {
350           "Name": "CostsPerYear",
351           "Title": "Usage Costs (Year)",
352           "Abstract": "The cost to be assessed for using the service on a
                  yearly basis (in Euro).",
353           "Obligated": "Service Consumer",
354           "Type": "urn:ogc:def:sla:business:cost:year",
355           "Value": "
356             factor;
357             if (pixel.year lt (1000000 * 1000))
358             {
359               factor = 1.0;
360             } else
361             if (pixel.year lt (1000000 * 10000))
362             {
363               factor = 0.5;
364             } else
365             if (pixel.year lt (1000000 * 100000))
366             {
367             factor = 0.25;
368             } else
369             if (pixel.year lt (1000000 * 1000000))
370             {
371               factor = 0.125;
372             } else
373             {
374               factor = 0.0625;
375             }
376             (factor * (pixel.year / 1000000));
377           "
378         },
379         /* EXAMPLE DISCOUNT FOR NOT MEETING A SERVICE LEVEL OBJECTIVE */
380         "Business Value":
381         {
382           "Name": "PenaltyPerYear",
383           "Title": "Penalty (Year)",
384           "Abstract": "The penalty to be assessed for not meeting service
                  level objectives on a yearly basis (in Euro).",
385           "Obligated": "Service Provider",
386           "Type": "urn:ogc:def:sla:business:penalty:year",
387           "Value": "
388             if (InspireAvailability.status == true)
389             {
390               factor = 0;
391             }
392             else
393             {
394               factor = 0.25;
395             }
396             (factor * CostsPerYear.value);
397           "
398         }
399       }
400     }
401   }
```

# Appendix C

# Service Level Management Architecture

This chapter provides additional information about the web-based SLA management architecture.

## C.1 WS-Agreement Application Profile

The WS-Agreement Application Profile for OGC Web Services describes a domain-specific extension of the WS-Agreement specification.

### C.1.1 XML Schema

This section describes the XSDs for defining the domain-specific content in the WS-Agreement Application Profile for OGC Web Services.

#### AGREEMENT CONTEXT

The XSD in Listing C.1 defines the domain-specific structure and content for the `AgreementInitiator` and `AgreementResponder` elements in WS-Agreement.

Listing C.1: XSD for Agreement Context

```
1  <?xml version="1.0" encoding="utf-8" standalone="yes"?>
2  <xs:schema xmlns:wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:
       ows="http://www.opengis.net/ows/2.0" xmlns:xs="http://www.w3.org/2001/
       XMLSchema" targetNamespace="http://www.ifgi.org/namespaces/wsag/ogc"
       elementFormDefault="qualified">
3    <!-- ######################################## -->
4    <!-- IMPORT SCHEMA                            -->
5    <!-- ######################################## -->
6    <xs:import schemaLocation="ows/ows19115subset.xsd" namespace="http://www.
         opengis.net/ows/2.0"/>
7    <!-- ######################################## -->
8    <!-- ELEMENT DEFINITIONS                      -->
9    <!-- ######################################## -->
10   <xs:element name="Contact" type="wsag-ogc:ContactType"/>
11   <!-- ######################################## -->
12   <!-- CONTACT TYPE                             -->
13   <!-- ######################################## -->
14   <xs:complexType name="ContactType">
15     <xs:sequence>
16       <xs:element minOccurs="1" maxOccurs="1" name="Name" type="xs:string"/>
17       <xs:element minOccurs="0" maxOccurs="1" name="Site" type="ows:
             OnlineResourceType"/>
```

```
18        <xs:element minOccurs="1" maxOccurs="1" name="Contact" type="ows:
              ResponsiblePartySubsetType"/>
19      </xs:sequence>
20    </xs:complexType>
21  </xs:schema>
```

The imported XSD encodes the parts of ISO 19115 [ISO, 2003] that are used in the `ServiceIdentification` and `ServiceProvider` elements of the `GetCapabilities` operation response document of all OWS. The most current OGC schemas are available in the official schema repository of the OGC [1].

### SERVICE DESCRIPTION TERMS

The XSD in Listing C.2 defines the domain-specific structure and content of functional service descriptions for the `ServiceDescriptionTerm` element in WS-Agreement.

Listing C.2: XSD for Functional Service Description

```
1  <?xml version="1.0" encoding="utf-8" standalone="yes"?>
2  <xs:schema xmlns:wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:
       res-sla="http://schemas.wsag4j.org/2009/07/wsag4j-scheduling-extensions"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www
       .ifgi.org/namespaces/wsag/ogc" elementFormDefault="qualified">
3    <!-- ######################################## -->
4    <!-- ELEMENT DEFINITIONS                      -->
5    <!-- ######################################## -->
6    <xs:element name="ServiceDescription" type="wsag-ogc:
         ServiceDescriptionType"/>
7    <!-- ######################################## -->
8    <!-- SERVICE DESCRIPTION TYPE                 -->
9    <!-- ######################################## -->
10   <xs:complexType name="ServiceDescriptionType">
11     <xs:sequence>
12       <xs:element minOccurs="1" maxOccurs="1" name="Title" type="xs:string
             "/>
13       <xs:element minOccurs="0" maxOccurs="1" name="Abstract" type="xs:
             string"/>
14       <xs:element minOccurs="0" maxOccurs="1" name="Keywords" type="xs:
             string"/>
15       <xs:element minOccurs="1" maxOccurs="1" name="Type" type="xs:anyURI"/>
16       <xs:element minOccurs="0" maxOccurs="1" name="Version" type="xs:string
             "/>
17       <xs:element minOccurs="0" maxOccurs="1" name="Profile" type="xs:anyURI
             "/>
18       <xs:element minOccurs="0" maxOccurs="1" name="WSDL" type="xs:string"/>
19     </xs:sequence>
20   </xs:complexType>
21 </xs:schema>
```

The XSD in Listing C.3 defines the domain-specific structure and content of non-functional service descriptions for the `ServiceDescriptionTerm` element in WS-Agreement.

Listing C.3: XSD for Non-Functional Service Description

```
1  <?xml version="1.0" encoding="utf-8" standalone="yes"?>
2  <xs:schema xmlns:wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:
       res-sla="http://schemas.wsag4j.org/2009/07/wsag4j-scheduling-extensions"
```

[1] http://schemas.opengis.net

```
    xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www
    .ifgi.org/namespaces/wsag/ogc" elementFormDefault="qualified">
  <!-- ######################################## -->
  <!-- ELEMENT DEFINITIONS                      -->
  <!-- ######################################## -->
  <xs:element name="ServiceProperties" type="wsag-ogc:ServicePropertiesType
      "/>
  <xs:element name="Property" type="wsag-ogc:PropertyType"/>
  <xs:element name="Monitoring" type="wsag-ogc:MonitoringType"/>
  <xs:element name="ActiveMonitoring" type="wsag-ogc:ActiveMonitoringType"/>
  <xs:element name="PassiveMonitoring" type="wsag-ogc:PassiveMonitoringType
      "/>
  <xs:element name="Session" type="wsag-ogc:ActiveMonitoringSessionType"/>
  <xs:element name="Request" type="wsag-ogc:ActiveMonitoringRequestType"/>
  <xs:element name="Response" type="wsag-ogc:ActiveMonitoringResponseType"/>
  <!-- ######################################## -->
  <!-- SERVICE PROPERTIES TYPE                  -->
  <!-- ######################################## -->
  <xs:complexType name="ServicePropertiesType">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="wsag-ogc:Property
          "/>
    </xs:sequence>
  </xs:complexType>
  <!-- ######################################## -->
  <!-- PROPERTY TYPE                            -->
  <!-- ######################################## -->
  <xs:complexType name="PropertyType">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="Name" type="xs:string"/>
      <xs:element minOccurs="1" maxOccurs="1" name="Title" type="xs:string
          "/>
      <xs:element minOccurs="0" maxOccurs="1" name="Abstract" type="xs:
          string"/>
      <xs:element minOccurs="1" maxOccurs="1" name="Type" type="xs:string"/>
      <xs:element minOccurs="0" maxOccurs="1" name="Value" type="xs:string
          "/>
      <xs:element minOccurs="0" maxOccurs="1" name="Reference" type="xs:
          string"/>
      <xs:element minOccurs="0" maxOccurs="1" ref="wsag-ogc:Monitoring"/>
    </xs:sequence>
  </xs:complexType>
  <!-- ######################################## -->
  <!-- MONITORING TYPE                          -->
  <!-- ######################################## -->
  <xs:complexType name="MonitoringType">
    <xs:choice minOccurs="1" maxOccurs="1">
      <xs:element ref="wsag-ogc:ActiveMonitoring"/>
      <xs:element ref="wsag-ogc:PassiveMonitoring"/>
    </xs:choice>
  </xs:complexType>
  <!-- ######################################## -->
  <!-- ACTIVE MONITORING TYPE                   -->
  <!-- ######################################## -->
  <xs:complexType name="ActiveMonitoringType">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="Start" type="xs:time"/>
      <xs:element minOccurs="1" maxOccurs="1" name="Stop" type="xs:time"/>
      <xs:element minOccurs="1" maxOccurs="1" name="Period" type="xs:int"/>
      <xs:element minOccurs="0" maxOccurs="1" ref="wsag-ogc:Session"/>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="wsag-ogc:Request
          "/>
      <xs:element minOccurs="0" maxOccurs="1" ref="wsag-ogc:Response"/>
    </xs:sequence>
  </xs:complexType>
  <!-- ######################################## -->
  <!-- ACTIVE MONITORING TYPE                   -->
  <!-- ######################################## -->
  <xs:complexType name="PassiveMonitoringType">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" ref="wsag-ogc:Request"/>
    </xs:sequence>
  </xs:complexType>
```

```
66    <!-- ######################################## -->
67    <!-- SESSION TYPE                             -->
68    <!-- ######################################## -->
69    <xs:complexType name="ActiveMonitoringSessionType">
70      <xs:sequence>
71        <xs:element minOccurs="1" maxOccurs="1" name="Capacity" type="xs:int
              "/>
72        <xs:element minOccurs="1" maxOccurs="1" name="Duration" type="xs:int
              "/>
73        <xs:element minOccurs="1" maxOccurs="1" name="Period" type="xs:int"/>
74      </xs:sequence>
75    </xs:complexType>
76    <!-- ######################################## -->
77    <!-- REQUEST TYPE                             -->
78    <!-- ######################################## -->
79    <xs:complexType name="ActiveMonitoringRequestType">
80      <xs:sequence>
81        <xs:element minOccurs="0" maxOccurs="1" name="Chance" type="xs:int"/>
82        <xs:element minOccurs="0" maxOccurs="1" name="Resource" type="xs:
              string"/>
83        <xs:element minOccurs="1" maxOccurs="1" name="Method" type="xs:string
              "/>
84        <xs:element minOccurs="0" maxOccurs="unbounded" name="Header" type="xs
              :string"/>
85        <xs:element minOccurs="0" maxOccurs="1" name="Content" type="xs:string
              "/>
86      </xs:sequence>
87    </xs:complexType>
88    <!-- ######################################## -->
89    <!-- RESPONSE TYPE                            -->
90    <!-- ######################################## -->
91    <xs:complexType name="ActiveMonitoringResponseType">
92      <xs:sequence>
93        <xs:element minOccurs="0" maxOccurs="1" name="Status" type="xs:string
              "/>
94        <xs:element minOccurs="0" maxOccurs="1" name="Content" type="xs:string
              "/>
95      </xs:sequence>
96    </xs:complexType>
97  </xs:schema>
```

The XSD in Listing C.4 defines the domain-specific structure and content of the contract period for the `ServiceDescriptionTerm` element in WS-Agreement.

Listing C.4: XSD for Service Availability Period

```
1   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2   <xs:schema xmlns:res-sla="http://schemas.wsag4j.org/2009/07/wsag4j-
        scheduling-extensions" xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://schemas.wsag4j.org/2009/07/wsag4j-scheduling-
        extensions" elementFormDefault="qualified" attributeFormDefault="
        qualified">
3     <!-- ######################################## -->
4     <!-- ELEMENT DEFINITIONS                      -->
5     <!-- ######################################## -->
6     <xs:element name="TimeConstraint" type="res-sla:TimeConstraintType"/>
7     <!-- ######################################## -->
8     <!-- TIME CONSTRAINT TYPE                     -->
9     <!-- ######################################## -->
10    <xs:complexType name="TimeConstraintType">
11      <xs:sequence>
12        <xs:element name="StartTime" type="xs:dateTime" minOccurs="0"
              maxOccurs="1"/>
13        <xs:element name="EndTime" type="xs:dateTime" minOccurs="0" maxOccurs
              ="1"/>
14        <xs:element name="Duration" type="xs:int" minOccurs="0" maxOccurs
              ="1"/>
15      </xs:sequence>
16    </xs:complexType>
```

```
17 </xs:schema>
```

This XSD is originally published in [Battré, 2009], a profile that describes "a minimal set of terms that allow expressing constraints about the time during which a service must be delivered".

### SERVICE REFERENCE

The XSD in Listing C.5 defines the domain-specific structure and content of service references for the `ServiceReference` element in WS-Agreement.

Listing C.5: XSD for Service Reference

```
1  <?xml version="1.0" encoding="utf-8" standalone="yes"?>
2  <xs:schema xmlns:wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:xs
       ="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.ifgi.org
       /namespaces/wsag/ogc" elementFormDefault="qualified">
3    <!-- ######################################## -->
4    <!-- ELEMENT DEFINITIONS                      -->
5    <!-- ######################################## -->
6    <xs:element name="ServiceReference" type="wsag-ogc:ServiceReferenceType"/>
7    <!-- ######################################## -->
8    <!-- SERVICE REFERENCE TYPE                   -->
9    <!-- ######################################## -->
10   <xs:complexType name="ServiceReferenceType">
11     <xs:sequence>
12       <xs:element minOccurs="1" maxOccurs="1" name="URL" type="xs:anyURI"/>
13     </xs:sequence>
14   </xs:complexType>
15 </xs:schema>
```

### SERVICE LEVEL OBJECTIVES

The XSD in Listing C.6 defines the domain-specific structure and content of custom service levels for the `CustomServiceLevel` element in WS-Agreement.

Listing C.6: XSD for Service Level Objectives

```
1  <?xml version="1.0" encoding="utf-8" standalone="yes"?>
2  <xs:schema xmlns:wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:xs
       ="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.ifgi.org
       /namespaces/wsag/ogc" elementFormDefault="qualified">
3    <!-- ######################################## -->
4    <!-- ELEMENT DEFINITIONS                      -->
5    <!-- ######################################## -->
6    <xs:element name="CustomServiceLevel" type="wsag-ogc:
         CustomServiceLevelType"/>
7    <!-- ######################################## -->
8    <!-- CUSTOM SERVICE LEVEL TYPE                -->
9    <!-- ######################################## -->
10   <xs:complexType name="CustomServiceLevelType">
11     <xs:sequence>
12       <xs:element minOccurs="1" maxOccurs="1" name="Name" type="xs:string"/>
13       <xs:element minOccurs="1" maxOccurs="1" name="Title" type="xs:string
             "/>
14       <xs:element minOccurs="0" maxOccurs="1" name="Abstract" type="xs:
             string"/>
15       <xs:element minOccurs="1" maxOccurs="1" name="Obligated" type="xs:
             string"/>
```

```
16        <xs:element minOccurs="1" maxOccurs="1" name="Status" type="xs:string
              "/>
17      </xs:sequence>
18    </xs:complexType>
19  </xs:schema>
```

#### BUSINESS VALUES

The XSD in Listing C.7 defines the domain-specific structure and content of custom business values for the `CustomBusinessValue` element in WS-Agreement.

Listing C.7: XSD for Business Values

```
1   <?xml version="1.0" encoding="utf-8" standalone="yes"?>
2   <xs:schema xmlns:wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:xs
        ="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.ifgi.org
        /namespaces/wsag/ogc" elementFormDefault="qualified">
3     <!-- ######################################## -->
4     <!-- ELEMENT DEFINITIONS                      -->
5     <!-- ######################################## -->
6     <xs:element name="CustomBusinessValue" type="wsag-ogc:
          CustomBusinessValueType"/>
7     <!-- ######################################## -->
8     <!-- CUSTOM BUSINESS VALUE TYPE               -->
9     <!-- ######################################## -->
10    <xs:complexType name="CustomBusinessValueType">
11      <xs:sequence>
12        <xs:element minOccurs="1" maxOccurs="1" name="Name" type="xs:string"/>
13        <xs:element minOccurs="1" maxOccurs="1" name="Title" type="xs:string
              "/>
14        <xs:element minOccurs="1" maxOccurs="1" name="Abstract" type="xs:
              string"/>
15        <xs:element minOccurs="1" maxOccurs="1" name="Obligated" type="xs:
              string"/>
16        <xs:element minOccurs="1" maxOccurs="1" name="Type" type="xs:string"/>
17        <xs:element minOccurs="1" maxOccurs="1" name="Value" type="xs:string
              "/>
18      </xs:sequence>
19    </xs:complexType>
20  </xs:schema>
```

### C.1.2 XML Example

This section provides example XML documents implementing the XSDs of the WS-Agreement Application Profile for OGC Web Services.

#### AGREEMENT TEMPLATE

The XML in Listing C.8 shows an example template document.

Listing C.8: Example Agreement Template

```
1   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2   <wsag:Template xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement
        " xmlns:wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:ows="
        http://www.opengis.net/ows/2.0" xmlns:res-sla="http://schemas.wsag4j.org
        /2009/07/wsag4j-scheduling-extensions" xmlns:xs="http://www.w3.org/2001/
        XMLSchema" xmlns:wsrf="http://docs.oasis-open.org/wsrf/bf-2" xmlns:
```

```
    addressing="http://www.w3.org/2005/08/addressing" wsag:TemplateId="
    WSAG_DEFAULT_TEMPLATE">
  <wsag:Name>INSPIRE_VIEW_SERVICE_TEMPLATE</wsag:Name>
  <!-- ######################################## -->
  <!-- AGREEMENT CONTEXT                        -->
  <!-- ######################################## -->
  <wsag:Context>
    <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
    <wsag:AgreementResponder>
      <wsag-ogc:Contact>
        <wsag-ogc:Name>Institute for Geoinformatics</wsag-ogc:Name>
        <wsag-ogc:Site xmlns:xlin="http://www.w3.org/1999/xlink" xlin:href="
            http://www.ifgi.de"/>
        <wsag-ogc:Contact>
          <ows:IndividualName>Bastian Baranski</ows:IndividualName>
          <ows:PositionName>Research Associate</ows:PositionName>
          <ows:ContactInfo>
            <ows:Phone>
              <ows:Voice>+49 251 8333071</ows:Voice>
              <ows:Facsimile>+49 251 8339763</ows:Facsimile>
            </ows:Phone>
            <ows:Address>
              <ows:DeliveryPoint>Weseler Strasse 253</ows:DeliveryPoint>
              <ows:City>Muenster</ows:City>
              <ows:PostalCode>48151</ows:PostalCode>
              <ows:Country>Germany</ows:Country>
              <ows:ElectronicMailAddress>baranski@uni-muenster.de</ows:
                  ElectronicMailAddress>
            </ows:Address>
            <ows:HoursOfService>The hours of service are Monday to Friday
                from 8 AM to 16 PM.</ows:HoursOfService>
            <ows:ContactInstructions>Please contact the service desk via
                phone or mail.</ows:ContactInstructions>
          </ows:ContactInfo>
        </wsag-ogc:Contact>
      </wsag-ogc:Contact>
    </wsag:AgreementResponder>
    <wsag:TemplateId>WSAG_DEFAULT_TEMPLATE_8</wsag:TemplateId>
    <wsag:TemplateName>INSPIRE_VIEW_SERVICE_TEMPLATE</wsag:TemplateName>
  </wsag:Context>
  <wsag:Terms>
    <wsag:All>
      <!-- ######################################## -->
      <!-- FUNCTIONAL SERVICE DESCRIPTION           -->
      <!-- ######################################## -->
      <wsag:ServiceDescriptionTerm wsag:Name="SERVICE_DESCRIPTION_SDT" wsag:
          ServiceName="INSPIRE_VIEW_SERVICE">
        <wsag-ogc:ServiceDescription>
          <wsag-ogc:Title>INSPIRE View Service</wsag-ogc:Title>
          <wsag-ogc:Abstract>This service instance is an INSPIRE View
              Service implementation.</wsag-ogc:Abstract>
          <wsag-ogc:Keywords>INSPIRE, View Service, OGC, WMS</wsag-ogc:
              Keywords>
          <wsag-ogc:Type>urn:ogc:doc:is:wms:1.1.1</wsag-ogc:Type>
        </wsag-ogc:ServiceDescription>
      </wsag:ServiceDescriptionTerm>
      <!-- ######################################## -->
      <!-- NON-FUNCTIONAL SERVICE DESCRIPTION       -->
      <!-- ######################################## -->
      <wsag:ServiceDescriptionTerm wsag:Name="SERVICE_PROPERTIES_SDT" wsag:
          ServiceName="INSPIRE_VIEW_SERVICE">
        <wsag-ogc:ServiceProperties>
          <!-- RESOURCE-RELATED PROPERTIES -->
          <wsag-ogc:Property>
            <wsag-ogc:Name>operations</wsag-ogc:Name>
            <wsag-ogc:Title>Supported Operations</wsag-ogc:Title>
            <wsag-ogc:Abstract>The operations that are supported by the
                service.</wsag-ogc:Abstract>
            <wsag-ogc:Type>urn:ogc:def:sla:property:resource:operation</wsag
                -ogc:Type>
            <wsag-ogc:Monitoring>
              <wsag-ogc:ActiveMonitoring>
                <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
```

```
64                  <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
65                  <wsag-ogc:Period>360000</wsag-ogc:Period>
66                </wsag-ogc:ActiveMonitoring>
67              </wsag-ogc:Monitoring>
68            </wsag-ogc:Property>
69            <!-- RUNTIME-RELATED PROPERTIES -->
70            <wsag-ogc:Property>
71              <wsag-ogc:Name>availability</wsag-ogc:Name>
72              <wsag-ogc:Title>Service Availability</wsag-ogc:Title>
73              <wsag-ogc:Abstract>The general availability of the service.</
                    wsag-ogc:Abstract>
74              <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:availability</
                    wsag-ogc:Type>
75              <wsag-ogc:Monitoring>
76                <wsag-ogc:ActiveMonitoring>
77                  <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
78                  <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
79                  <wsag-ogc:Period>360000</wsag-ogc:Period>
80                  <wsag-ogc:Request>
81                    <wsag-ogc:Method>GET</wsag-ogc:Method>
82                    <wsag-ogc:Content>service=WMS&amp;version=1.3.0&amp;
                          request=GetMap&amp;layers=topp:
                          tasmania_state_boundaries&amp;styles=&amp;bbox=${
                          __random(142.0,144.0)},${__random(-46.0,-44.0)},${
                          __random(150.0,152.0)},${__random(-38.0,-36.0)}&amp;
                          width=800&amp;height=600&amp;srs=EPSG:4326&amp;format=
                          image/png</wsag-ogc:Content>
83                  </wsag-ogc:Request>
84                  <wsag-ogc:Response>
85                    <wsag-ogc:Status>200</wsag-ogc:Status>
86                  </wsag-ogc:Response>
87                </wsag-ogc:ActiveMonitoring>
88              </wsag-ogc:Monitoring>
89            </wsag-ogc:Property>
90            <wsag-ogc:Property>
91              <wsag-ogc:Name>response</wsag-ogc:Name>
92              <wsag-ogc:Title>Response Time</wsag-ogc:Title>
93              <wsag-ogc:Abstract>The response time of the service.</wsag-ogc:
                    Abstract>
94              <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:response</wsag-
                    ogc:Type>
95              <wsag-ogc:Monitoring>
96                <wsag-ogc:ActiveMonitoring>
97                  <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
98                  <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
99                  <wsag-ogc:Period>360000</wsag-ogc:Period>
100                 <wsag-ogc:Request>
101                   <wsag-ogc:Method>GET</wsag-ogc:Method>
102                   <wsag-ogc:Content>service=WMS&amp;version=1.3.0&amp;
                          request=GetMap&amp;layers=topp:
                          tasmania_state_boundaries&amp;styles=&amp;bbox=${
                          __random(142.0,144.0)},${__random(-46.0,-44.0)},${
                          __random(150.0,152.0)},${__random(-38.0,-36.0)}&amp;
                          width=800&amp;height=600&amp;srs=EPSG:4326&amp;format=
                          image/png</wsag-ogc:Content>
103                 </wsag-ogc:Request>
104                 <wsag-ogc:Response>
105                   <wsag-ogc:Status>200</wsag-ogc:Status>
106                 </wsag-ogc:Response>
107               </wsag-ogc:ActiveMonitoring>
108             </wsag-ogc:Monitoring>
109           </wsag-ogc:Property>
110           <wsag-ogc:Property>
111             <wsag-ogc:Name>capacity</wsag-ogc:Name>
112             <wsag-ogc:Title>Service Capacity</wsag-ogc:Title>
113             <wsag-ogc:Abstract>The response time of the service for multiple
                    parallel requests.</wsag-ogc:Abstract>
114             <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:response</wsag-
                    ogc:Type>
115             <wsag-ogc:Monitoring>
116               <wsag-ogc:ActiveMonitoring>
117                 <wsag-ogc:Start>20:00:00</wsag-ogc:Start>
118                 <wsag-ogc:Stop>04:00:00</wsag-ogc:Stop>
```

```
119                 <wsag-ogc:Period>3600000</wsag-ogc:Period>
120                 <wsag-ogc:Session>
121                   <wsag-ogc:Capacity>20</wsag-ogc:Capacity>
122                   <wsag-ogc:Duration>60000</wsag-ogc:Duration>
123                   <wsag-ogc:Period>1000</wsag-ogc:Period>
124                 </wsag-ogc:Session>
125                 <wsag-ogc:Request>
126                   <wsag-ogc:Chance>10</wsag-ogc:Chance>
127                   <wsag-ogc:Method>GET</wsag-ogc:Method>
128                   <wsag-ogc:Content>service=WMS&amp;version=1.3.0&amp;
                          request=GetCapabilities</wsag-ogc:Content>
129                 </wsag-ogc:Request>
130                 <wsag-ogc:Request>
131                   <wsag-ogc:Chance>90</wsag-ogc:Chance>
132                   <wsag-ogc:Method>GET</wsag-ogc:Method>
133                   <wsag-ogc:Content>service=WMS&amp;version=1.3.0&amp;
                          request=GetMap&amp;layers=topp:
                          tasmania_state_boundaries&amp;styles=&amp;bbox=${
                          __random(142.0,144.0)},${__random(-46.0,-44.0)},${
                          __random(150.0,152.0)},${__random(-38.0,-36.0)}&amp;
                          width=800&amp;height=600&amp;srs=EPSG:4326&amp;format=
                          image/png</wsag-ogc:Content>
134                 </wsag-ogc:Request>
135                 <wsag-ogc:Response>
136                   <wsag-ogc:Status>200</wsag-ogc:Status>
137                 </wsag-ogc:Response>
138               </wsag-ogc:ActiveMonitoring>
139             </wsag-ogc:Monitoring>
140           </wsag-ogc:Property>
141           <!-- USAGE-RELATED PROPERTIES -->
142           <wsag-ogc:Property>
143             <wsag-ogc:Name>pixel</wsag-ogc:Name>
144             <wsag-ogc:Title>Sum of Pixels</wsag-ogc:Title>
145             <wsag-ogc:Abstract>The accessed number of pixels.</wsag-ogc:
                    Abstract>
146             <wsag-ogc:Type>urn:ogc:def:sla:property:usage:pixel</wsag-ogc:
                    Type>
147             <wsag-ogc:Monitoring>
148               <wsag-ogc:PassiveMonitoring>
149                 <wsag-ogc:Request>
150                   <wsag-ogc:Resource>/state/urn:ogc:def:sla:property:usage:
                          pixel</wsag-ogc:Resource>
151                   <wsag-ogc:Method>GET</wsag-ogc:Method>
152                 </wsag-ogc:Request>
153               </wsag-ogc:PassiveMonitoring>
154             </wsag-ogc:Monitoring>
155           </wsag-ogc:Property>
156           <!-- INFRASTRUCTURE-RELATED PROPERTIES -->
157           <wsag-ogc:Property>
158             <wsag-ogc:Name>provider</wsag-ogc:Name>
159             <wsag-ogc:Title>Infrastructure Provider</wsag-ogc:Title>
160             <wsag-ogc:Abstract>The name of the infrastructure provider.</
                    wsag-ogc:Abstract>
161             <wsag-ogc:Type>urn:ogc:def:sla:property:infrastructure:provider:
                    name</wsag-ogc:Type>
162             <wsag-ogc:Value>default</wsag-ogc:Value>
163           </wsag-ogc:Property>
164           <wsag-ogc:Property>
165             <wsag-ogc:Name>image</wsag-ogc:Name>
166             <wsag-ogc:Title>Virtual Machine</wsag-ogc:Title>
167             <wsag-ogc:Abstract>The name of the Virtual Machine (VM) template
                    .</wsag-ogc:Abstract>
168             <wsag-ogc:Type>urn:ogc:def:sla:property:infrastructure:vm:name</
                    wsag-ogc:Type>
169             <wsag-ogc:Value>ami-59f9c62d</wsag-ogc:Value>
170           </wsag-ogc:Property>
171         </wsag-ogc:ServiceProperties>
172       </wsag:ServiceDescriptionTerm>
173       <!-- ######################################## -->
174       <!-- CONTRACT PERIOD                          -->
175       <!-- ######################################## -->
176       <wsag:ServiceDescriptionTerm wsag:Name="CONTRACT_PERIOD_SDT" wsag:
              ServiceName="INSPIRE_VIEW_SERVICE">
```

```
177           <res-sla:TimeConstraint>
178             <res-sla:StartTime>2010-07-04T13:00:00+02:00</res-sla:StartTime>
179             <res-sla:EndTime>2012-07-09T13:00:00+02:00</res-sla:EndTime>
180           </res-sla:TimeConstraint>
181         </wsag:ServiceDescriptionTerm>
182         <!-- ######################################## -->
183         <!-- SERVICE REFERENCE                        -->
184         <!-- ######################################## -->
185         <wsag:ServiceReference wsag:Name="SERVICE_REFERENCE" wsag:ServiceName
                ="INSPIRE_VIEW_SERVICE">
186           <wsag-ogc:ServiceReference>
187             <wsag-ogc:URL>http://localhost:8088/sla-proxy/DefaultWMS</wsag-ogc
                    :URL>
188           </wsag-ogc:ServiceReference>
189         </wsag:ServiceReference>
190         <!-- ######################################## -->
191         <!-- SERVICE PROPERTIES                       -->
192         <!-- ######################################## -->
193         <wsag:ServiceProperties wsag:Name="SERVICE_PROPERTIES" wsag:
                ServiceName="INSPIRE_VIEW_SERVICE">
194           <wsag:VariableSet>
195             <wsag:Variable wsag:Name="SERVICE_PROPERTIES_STATE" wsag:Metric="
                    xs:string">
196               <wsag:Location>declare namespace ws='http://schemas.ggf.org/
                        graap/2007/03/ws-agreement';declare namespace wsag-ogc='http
                        ://www.ifgi.org/namespaces/wsag/ogc';declare namespace wsag
                        ='http://schemas.ggf.org/graap/2007/03/ws-agreement';/ws:
                        AgreementProperties/ws:ServiceTermState[@termName='
                        SERVICE_PROPERTIES_SDT']/ws:State/text()</wsag:Location>
197             </wsag:Variable>
198           </wsag:VariableSet>
199         </wsag:ServiceProperties>
200         <!-- ######################################## -->
201         <!-- GUARANTEE TERMS                          -->
202         <!-- ######################################## -->
203         <!-- INSPIRE OPERATIONS REQUIREMENTS -->
204         <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RESOURCE_OPERATIONS" wsag:
                Obligated="ServiceProvider">
205           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
206           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
207           <wsag:ServiceLevelObjective>
208             <wsag:CustomServiceLevel>
209               <wsag-ogc:CustomServiceLevel>
210                 <wsag-ogc:Name>InspireOperations</wsag-ogc:Name>
211                 <wsag-ogc:Title>INSPIRE (Operations)</wsag-ogc:Title>
212                 <wsag-ogc:Abstract>The following operations shall be
                        implemented for an INSPIRE View service: GetCapabilities,
                        GetMap.</wsag-ogc:Abstract>
213                 <wsag-ogc:Status>
214                   isGetCapabilities = false;
215                   isGetMap = false;
216                   for (item : operations.name)
217                   {
218                     if (item.equalsIgnoreCase('GetCapabilities'))
219                     {
220                       isGetCapabilities = true;
221                     }
222                     if (item.equalsIgnoreCase('GetMap'))
223                     {
224                       isGetMap = true;
225                     }
226                   }
227                   (isGetCapabilities and isGetMap);
228                 </wsag-ogc:Status>
229               </wsag-ogc:CustomServiceLevel>
230             </wsag:CustomServiceLevel>
231           </wsag:ServiceLevelObjective>
232           <wsag:BusinessValueList/>
233         </wsag:GuaranteeTerm>
234         <!-- INSPIRE QUALITY OF SERVICE REQUIREMENTS -->
235         <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_AVAILABILITY" wsag:
                Obligated="ServiceProvider">
```

```
236           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
237           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
238           <wsag:ServiceLevelObjective>
239             <wsag:CustomServiceLevel>
240               <wsag-ogc:CustomServiceLevel>
241                 <wsag-ogc:Name>InspireAvailability</wsag-ogc:Name>
242                 <wsag-ogc:Title>INSPIRE (Availability)</wsag-ogc:Title>
243                 <wsag-ogc:Abstract>The probability of a Network Service to be
                        available shall be 99% of the time.</wsag-ogc:Abstract>
244                 <wsag-ogc:Status>
245                   (availability.week >= 0.99) and (availability.month >= 0.99)
                          and (availability.year >= 0.99)
246                 </wsag-ogc:Status>
247               </wsag-ogc:CustomServiceLevel>
248             </wsag:CustomServiceLevel>
249           </wsag:ServiceLevelObjective>
250           <wsag:BusinessValueList/>
251         </wsag:GuaranteeTerm>
252         <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_RESPONSE" wsag:
              Obligated="ServiceProvider">
253           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
254           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
255           <wsag:ServiceLevelObjective>
256             <wsag:CustomServiceLevel>
257               <wsag-ogc:CustomServiceLevel>
258                 <wsag-ogc:Name>InspirePerformance</wsag-ogc:Name>
259                 <wsag-ogc:Title>INSPIRE (Performance)</wsag-ogc:Title>
260                 <wsag-ogc:Abstract>The response time for sending the initial
                        response to a Get Map Request to a view service shall be
                        maximum 5 seconds in normal situation.</wsag-ogc:Abstract>
261                 <wsag-ogc:Status>
262                   fulfilled = 0;
263                   for (item : response.initial.week) {
264                     if (item lt 5000)
265                     {
266                       fulfilled = fulfilled + 1;
267                     }
268                   }
269                   percent = fulfilled / (size(response.initial.week) / 100.0);
270                   percent gt 90.0;
271                 </wsag-ogc:Status>
272               </wsag-ogc:CustomServiceLevel>
273             </wsag:CustomServiceLevel>
274           </wsag:ServiceLevelObjective>
275           <wsag:BusinessValueList/>
276         </wsag:GuaranteeTerm>
277         <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_CAPACITY" wsag:
              Obligated="ServiceProvider">
278           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
279           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
280           <wsag:ServiceLevelObjective>
281             <wsag:CustomServiceLevel>
282               <wsag-ogc:CustomServiceLevel>
283                 <wsag-ogc:Name>InspireCapacity</wsag-ogc:Name>
284                 <wsag-ogc:Title>INSPIRE (Capacity)</wsag-ogc:Title>
285                 <wsag-ogc:Abstract>The minimum number of served simultaneous
                        service requests to a view service according to the
                        performance quality of service shall be 20 per second.</
                        wsag-ogc:Abstract>
286                 <wsag-ogc:Status>
287                   fulfilled = 0;
288                   for (item : capacity.initial.week) {
289                     if (item lt 5000)
290                     {
291                       fulfilled = fulfilled + 1;
292                     }
293                   }
294                   percent = fulfilled / (size(capacity.initial.week) / 100.0);
295                   percent gt 90.0;
296                 </wsag-ogc:Status>
```

```
297             </wsag-ogc:CustomServiceLevel>
298           </wsag:CustomServiceLevel>
299         </wsag:ServiceLevelObjective>
300         <wsag:BusinessValueList/>
301       </wsag:GuaranteeTerm>
302       <!-- AdV PRICING MODEL FOR ONLINE DATA DELIVERY -->
303       <wsag:GuaranteeTerm wsag:Name="COSTS_PER_YEAR" wsag:Obligated="
              ServiceProvider">
304         <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
305         <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
              QualifyingCondition>
306         <wsag:ServiceLevelObjective/>
307         <wsag:BusinessValueList>
308           <wsag:CustomBusinessValue>
309             <wsag-ogc:CustomBusinessValue>
310               <wsag-ogc:Name>CostsPerYear</wsag-ogc:Name>
311               <wsag-ogc:Title>Usage Costs (Year)</wsag-ogc:Title>
312               <wsag-ogc:Abstract>The cost to be assessed for using the
                      service on a yearly basis (in Euro).</wsag-ogc:Abstract>
313               <wsag-ogc:Type>urn:ogc:def:sla:business:cost:year</wsag-ogc:
                      Type>
314               <wsag-ogc:Value>
315                 factor;
316                 if (pixel.year lt (1000000 * 1000))
317                 {
318                   factor = 1.0;
319                 } else
320                 if (pixel.year lt (1000000 * 10000))
321                 {
322                   factor = 0.5;
323                 } else
324                 if (pixel.year lt (1000000 * 100000))
325                 {
326                 factor = 0.25;
327                 } else
328                 if (pixel.year lt (1000000 * 1000000))
329                 {
330                   factor = 0.125;
331                 } else
332                 {
333                   factor = 0.0625;
334                 }
335                 (factor * (pixel.year / 1000000.0));
336                               </wsag-ogc:Value>
337             </wsag-ogc:CustomBusinessValue>
338           </wsag:CustomBusinessValue>
339         </wsag:BusinessValueList>
340       </wsag:GuaranteeTerm>
341       <!--  EXAMPLE DISCOUNT FOR NOT MEETING A SERVICE LEVEL OBJECTIVE -->
342       <wsag:GuaranteeTerm wsag:Name="PENALTY_PER_YEAR" wsag:Obligated="
              ServiceProvider">
343         <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
344         <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
              QualifyingCondition>
345         <wsag:ServiceLevelObjective/>
346         <wsag:BusinessValueList>
347           <wsag:CustomBusinessValue>
348             <wsag-ogc:CustomBusinessValue>
349               <wsag-ogc:Name>PenaltyPerYear</wsag-ogc:Name>
350               <wsag-ogc:Title>Penalty (Year)</wsag-ogc:Title>
351               <wsag-ogc:Abstract>The penalty to be assessed for not meeting
                      service level objectives on a yearly basis (in Euro).</
                      wsag-ogc:Abstract>
352               <wsag-ogc:Type>urn:ogc:def:sla:business:penalty:year</wsag-ogc
                      :Type>
353               <wsag-ogc:Value>
354                 factor;
355                 if (InspireAvailability.status == true)
356                 {
357                   factor = 0;
358                 }
359                 else
360                 {
```

```
                    factor = 0.25;
                  }
                  (factor * CostsPerYear.value);
                </wsag-ogc:Value>
              </wsag-ogc:CustomBusinessValue>
            </wsag:CustomBusinessValue>
          </wsag:BusinessValueList>
        </wsag:GuaranteeTerm>
      </wsag:All>
    </wsag:Terms>
</wsag:Template>
```

### AGREEMENT OFFER

The XML in Listing C.9 shows an example agreement offer document.

Listing C.9: Example Agreement Offer

```
<ws:AgreementOffer ws:AgreementId="b7f8fc14-17e3-499b-9845-975e7f41542e"
    xmlns:ws="http://schemas.ggf.org/graap/2007/03/ws-agreement">
  <wsag:Context xmlns:wsrf="http://docs.oasis-open.org/wsrf/bf-2" xmlns:wsag
      ="http://schemas.ggf.org/graap/2007/03/ws-agreement" xmlns:ows="http
      ://www.opengis.net/ows/2.0" xmlns:xs="http://www.w3.org/2001/XMLSchema
      " xmlns:wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:
      addressing="http://www.w3.org/2005/08/addressing" xmlns:res-sla="http
      ://schemas.wsag4j.org/2009/07/wsag4j-scheduling-extensions">
    <wsag:AgreementInitiator>
      <wsag-ogc:Contact>
        <wsag-ogc:Name>Instititute for Geoinformatics</wsag-ogc:Name>
        <wsag-ogc:Site xlin:href="http://www.ifgi.de" xmlns:xlin="http://www
            .w3.org/1999/xlink"/>
        <wsag-ogc:Contact>
          <ows:IndividualName>Kristof Lange</ows:IndividualName>
          <ows:PositionName>Student Assistance</ows:PositionName>
          <ows:ContactInfo>
            <ows:Phone>
              <ows:Voice>+49 251 833307</ows:Voice>
              <ows:Facsimile>+49 251 8339763</ows:Facsimile>
            </ows:Phone>
            <ows:Address>
              <ows:DeliveryPoint>Weseler Strasse 253</ows:DeliveryPoint>
              <ows:City>Muenster</ows:City>
              <ows:PostalCode>48151</ows:PostalCode>
              <ows:Country>Germany</ows:Country>
              <ows:ElectronicMailAddress>kristof.lange@uni-muenster.de</ows:
                  ElectronicMailAddress>
            </ows:Address>
            <ows:OnlineResource xlin:href="http://www.ifgi.de" xmlns:xlin="
                http://www.w3.org/1999/xlink"/>
          </ows:ContactInfo>
        </wsag-ogc:Contact>
      </wsag-ogc:Contact>
    </wsag:AgreementInitiator>
    <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
    <wsag:AgreementResponder>
      <wsag-ogc:Contact>
        <wsag-ogc:Name>Institute for Geoinformatics</wsag-ogc:Name>
        <wsag-ogc:Site xlin:href="http://www.ifgi.de" xmlns:xlin="http://www
            .w3.org/1999/xlink"/>
        <wsag-ogc:Contact>
          <ows:IndividualName>Bastian Baranski</ows:IndividualName>
          <ows:PositionName>Research Associate</ows:PositionName>
          <ows:ContactInfo>
            <ows:Phone>
              <ows:Voice>+49 251 8333071</ows:Voice>
              <ows:Facsimile>+49 251 8339763</ows:Facsimile>
            </ows:Phone>
```

```
40              <ows:Address>
41                <ows:DeliveryPoint>Weseler Strasse 253</ows:DeliveryPoint>
42                <ows:City>Muenster</ows:City>
43                <ows:PostalCode>48151</ows:PostalCode>
44                <ows:Country>Germany</ows:Country>
45                <ows:ElectronicMailAddress>baranski@uni-muenster.de</ows:
                      ElectronicMailAddress>
46              </ows:Address>
47              <ows:HoursOfService>The hours of service are Monday to Friday
                      from 8 AM to 16 PM.</ows:HoursOfService>
48              <ows:ContactInstructions>Please contact the service desk via
                      phone or mail.</ows:ContactInstructions>
49            </ows:ContactInfo>
50          </wsag-ogc:Contact>
51        </wsag-ogc:Contact>
52      </wsag:AgreementResponder>
53      <wsag:TemplateId>WSAG_DEFAULT_TEMPLATE_8</wsag:TemplateId>
54      <wsag:TemplateName>INSPIRE_VIEW_SERVICE_TEMPLATE</wsag:TemplateName>
55    </wsag:Context>
56    <ws:Terms>
57      <ws:All>
58        <wsag:ServiceDescriptionTerm wsag:Name="SERVICE_DESCRIPTION_SDT" wsag:
              ServiceName="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-
              open.org/wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap
              /2007/03/ws-agreement" xmlns:ows="http://www.opengis.net/ows/2.0"
              xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http
              ://www.ifgi.org/namespaces/wsag/ogc" xmlns:addressing="http://www.
              w3.org/2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.
              org/2009/07/wsag4j-scheduling-extensions">
59          <wsag-ogc:ServiceDescription>
60            <wsag-ogc:Title>INSPIRE View Service</wsag-ogc:Title>
61            <wsag-ogc:Abstract>This service instance is an INSPIRE View
                  Service implementation.</wsag-ogc:Abstract>
62            <wsag-ogc:Keywords>INSPIRE, View Service, OGC, WMS</wsag-ogc:
                  Keywords>
63            <wsag-ogc:Type>urn:ogc:doc:is:wms:1.1.1</wsag-ogc:Type>
64          </wsag-ogc:ServiceDescription>
65        </wsag:ServiceDescriptionTerm>
66        <wsag:ServiceDescriptionTerm wsag:Name="SERVICE_PROPERTIES_SDT" wsag:
              ServiceName="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-
              open.org/wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap
              /2007/03/ws-agreement" xmlns:ows="http://www.opengis.net/ows/2.0"
              xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http
              ://www.ifgi.org/namespaces/wsag/ogc" xmlns:addressing="http://www.
              w3.org/2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.
              org/2009/07/wsag4j-scheduling-extensions">
67          <wsag-ogc:ServiceProperties>
68            <!--RESOURCE-RELATED PROPERTIES-->
69            <wsag-ogc:Property>
70              <wsag-ogc:Name>operations</wsag-ogc:Name>
71              <wsag-ogc:Title>Supported Operations</wsag-ogc:Title>
72              <wsag-ogc:Abstract>The operations that are supported by the
                    service.</wsag-ogc:Abstract>
73              <wsag-ogc:Type>urn:ogc:def:sla:property:resource:operation</wsag
                    -ogc:Type>
74              <wsag-ogc:Monitoring>
75                <wsag-ogc:ActiveMonitoring>
76                  <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
77                  <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
78                  <wsag-ogc:Period>360000</wsag-ogc:Period>
79                </wsag-ogc:ActiveMonitoring>
80              </wsag-ogc:Monitoring>
81            </wsag-ogc:Property>
82            <!--RUNTIME-RELATED PROPERTIES-->
83            <wsag-ogc:Property>
84              <wsag-ogc:Name>availability</wsag-ogc:Name>
85              <wsag-ogc:Title>Service Availability</wsag-ogc:Title>
86              <wsag-ogc:Abstract>The general availability of the service.</
                    wsag-ogc:Abstract>
87              <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:availability</
                    wsag-ogc:Type>
88              <wsag-ogc:Monitoring>
89                <wsag-ogc:ActiveMonitoring>
```

```
90                  <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
91                  <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
92                  <wsag-ogc:Period>360000</wsag-ogc:Period>
93                  <wsag-ogc:Request>
94                    <wsag-ogc:Method>GET</wsag-ogc:Method>
95                    <wsag-ogc:Content><![CDATA[service=WMS&version=1.3.0&
                          request=GetMap&layers=topp:tasmania_state_boundaries&
                          styles=&bbox=${__random(142.0,144.0)},${__random
                          (-46.0,-44.0)},${__random(150.0,152.0)},${__random
                          (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&
                          format=image/png]]></wsag-ogc:Content>
96                  </wsag-ogc:Request>
97                  <wsag-ogc:Response>
98                    <wsag-ogc:Status>200</wsag-ogc:Status>
99                  </wsag-ogc:Response>
100               </wsag-ogc:ActiveMonitoring>
101             </wsag-ogc:Monitoring>
102           </wsag-ogc:Property>
103           <wsag-ogc:Property>
104             <wsag-ogc:Name>response</wsag-ogc:Name>
105             <wsag-ogc:Title>Response Time</wsag-ogc:Title>
106             <wsag-ogc:Abstract>The response time of the service.</wsag-ogc:
                    Abstract>
107             <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:response</wsag-
                    ogc:Type>
108             <wsag-ogc:Monitoring>
109               <wsag-ogc:ActiveMonitoring>
110                 <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
111                 <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
112                 <wsag-ogc:Period>360000</wsag-ogc:Period>
113                 <wsag-ogc:Request>
114                   <wsag-ogc:Method>GET</wsag-ogc:Method>
115                   <wsag-ogc:Content><![CDATA[service=WMS&version=1.3.0&
                          request=GetMap&layers=topp:tasmania_state_boundaries&
                          styles=&bbox=${__random(142.0,144.0)},${__random
                          (-46.0,-44.0)},${__random(150.0,152.0)},${__random
                          (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&
                          format=image/png]]></wsag-ogc:Content>
116                 </wsag-ogc:Request>
117                 <wsag-ogc:Response>
118                   <wsag-ogc:Status>200</wsag-ogc:Status>
119                 </wsag-ogc:Response>
120               </wsag-ogc:ActiveMonitoring>
121             </wsag-ogc:Monitoring>
122           </wsag-ogc:Property>
123           <wsag-ogc:Property>
124             <wsag-ogc:Name>capacity</wsag-ogc:Name>
125             <wsag-ogc:Title>Service Capacity</wsag-ogc:Title>
126             <wsag-ogc:Abstract>The response time of the service for multiple
                     parallel requests.</wsag-ogc:Abstract>
127             <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:response</wsag-
                    ogc:Type>
128             <wsag-ogc:Monitoring>
129               <wsag-ogc:ActiveMonitoring>
130                 <wsag-ogc:Start>20:00:00</wsag-ogc:Start>
131                 <wsag-ogc:Stop>04:00:00</wsag-ogc:Stop>
132                 <wsag-ogc:Period>3600000</wsag-ogc:Period>
133                 <wsag-ogc:Session>
134                   <wsag-ogc:Capacity>20</wsag-ogc:Capacity>
135                   <wsag-ogc:Duration>60000</wsag-ogc:Duration>
136                   <wsag-ogc:Period>1000</wsag-ogc:Period>
137                 </wsag-ogc:Session>
138                 <wsag-ogc:Request>
139                   <wsag-ogc:Chance>10</wsag-ogc:Chance>
140                   <wsag-ogc:Method>GET</wsag-ogc:Method>
141                   <wsag-ogc:Content>service=WMS&amp;version=1.3.0&amp;
                          request=GetCapabilities</wsag-ogc:Content>
142                 </wsag-ogc:Request>
143                 <wsag-ogc:Request>
144                   <wsag-ogc:Chance>90</wsag-ogc:Chance>
145                   <wsag-ogc:Method>GET</wsag-ogc:Method>
146                   <wsag-ogc:Content><![CDATA[service=WMS&version=1.3.0&
                          request=GetMap&layers=topp:tasmania_state_boundaries&
```

```
                    styles=&bbox=${__random(142.0,144.0)},${__random
                    (-46.0,-44.0)},${__random(150.0,152.0)},${__random
                    (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&
                    format=image/png]]></wsag-ogc:Content>
147               </wsag-ogc:Request>
148               <wsag-ogc:Response>
149                 <wsag-ogc:Status>200</wsag-ogc:Status>
150               </wsag-ogc:Response>
151             </wsag-ogc:ActiveMonitoring>
152           </wsag-ogc:Monitoring>
153         </wsag-ogc:Property>
154         <!--USAGE-RELATED PROPERTIES-->
155         <wsag-ogc:Property>
156           <wsag-ogc:Name>pixel</wsag-ogc:Name>
157           <wsag-ogc:Title>Sum of Pixels</wsag-ogc:Title>
158           <wsag-ogc:Abstract>The accessed number of pixels.</wsag-ogc:
                Abstract>
159           <wsag-ogc:Type>urn:ogc:def:sla:property:usage:pixel</wsag-ogc:
                Type>
160           <wsag-ogc:Monitoring>
161             <wsag-ogc:PassiveMonitoring>
162               <wsag-ogc:Request>
163                 <wsag-ogc:Resource>/state/urn:ogc:def:sla:property:usage:
                      pixel</wsag-ogc:Resource>
164                 <wsag-ogc:Method>GET</wsag-ogc:Method>
165               </wsag-ogc:Request>
166             </wsag-ogc:PassiveMonitoring>
167           </wsag-ogc:Monitoring>
168         </wsag-ogc:Property>
169         <!--INFRASTRUCTURE-RELATED PROPERTIES-->
170         <wsag-ogc:Property>
171           <wsag-ogc:Name>provider</wsag-ogc:Name>
172           <wsag-ogc:Title>Infrastructure Provider</wsag-ogc:Title>
173           <wsag-ogc:Abstract>The name of the infrastructure provider.</
                wsag-ogc:Abstract>
174           <wsag-ogc:Type>urn:ogc:def:sla:property:infrastructure:provider:
                name</wsag-ogc:Type>
175           <wsag-ogc:Value>default</wsag-ogc:Value>
176         </wsag-ogc:Property>
177         <wsag-ogc:Property>
178           <wsag-ogc:Name>image</wsag-ogc:Name>
179           <wsag-ogc:Title>Virtual Machine</wsag-ogc:Title>
180           <wsag-ogc:Abstract>The name of the Virtual Machine (VM) template
                .</wsag-ogc:Abstract>
181           <wsag-ogc:Type>urn:ogc:def:sla:property:infrastructure:vm:name</
                wsag-ogc:Type>
182           <wsag-ogc:Value>ami-59f9c62d</wsag-ogc:Value>
183         </wsag-ogc:Property>
184       </wsag-ogc:ServiceProperties>
185     </wsag:ServiceDescriptionTerm>
186     <ws:ServiceDescriptionTerm ws:Name="TIME_CONSTRAINT_SDT" ws:
            ServiceName="INSPIRE_VIEW_SERVICE">
187       <wsag:TimeConstraint xmlns:wsag="http://schemas.wsag4j.org/2009/07/
              wsag4j-scheduling-extensions">
188         <wsag:StartTime>2011-05-27T11:00:00</wsag:StartTime>
189         <wsag:EndTime>2013-05-27T11:00:00</wsag:EndTime>
190       </wsag:TimeConstraint>
191     </ws:ServiceDescriptionTerm>
192     <wsag:ServiceProperties wsag:Name="SERVICE_PROPERTIES" wsag:
            ServiceName="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-
            open.org/wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap
            /2007/03/ws-agreement" xmlns:ows="http://www.opengis.net/ows/2.0"
            xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http
            ://www.ifgi.org/namespaces/wsag/ogc" xmlns:addressing="http://www.
            w3.org/2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.
            org/2009/07/wsag4j-scheduling-extensions">
193       <wsag:VariableSet>
194         <wsag:Variable wsag:Name="SERVICE_PROPERTIES_STATE" wsag:Metric="
                xs:string">
195           <wsag:Location>declare namespace ws='http://schemas.ggf.org/
                  graap/2007/03/ws-agreement';declare namespace wsag-ogc='http
                  ://www.ifgi.org/namespaces/wsag/ogc';declare namespace wsag
                  ='http://schemas.ggf.org/graap/2007/03/ws-agreement';/ws:
```

```
                AgreementProperties/ws:ServiceTermState[@termName='
                SERVICE_PROPERTIES_SDT']/ws:State/text()</wsag:Location>
196           </wsag:Variable>
197         </wsag:VariableSet>
198       </wsag:ServiceProperties>
199       <wsag:ServiceReference wsag:Name="SERVICE_REFERENCE" wsag:ServiceName
              ="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-open.org/
              wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
200         <wsag-ogc:ServiceReference>
201           <wsag-ogc:URL>http://localhost:8088/sla-proxy/DefaultWMS</wsag-ogc
                  :URL>
202         </wsag-ogc:ServiceReference>
203       </wsag:ServiceReference>
204       <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RESOURCE_OPERATIONS" wsag:
              Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
              /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
205         <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
206         <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
207         <wsag:ServiceLevelObjective>
208           <wsag:CustomServiceLevel>
209             <wsag-ogc:CustomServiceLevel>
210               <wsag-ogc:Name>InspireOperations</wsag-ogc:Name>
211               <wsag-ogc:Title>INSPIRE (Operations)</wsag-ogc:Title>
212               <wsag-ogc:Abstract>The following operations shall be
                      implemented for an INSPIRE View service: GetCapabilities,
                      GetMap.</wsag-ogc:Abstract>
213               <wsag-ogc:Status>isGetCapabilities = false;
214                 isGetMap = false;
215                 for (item : operations.name)
216                 {
217                   if (item.equalsIgnoreCase('GetCapabilities'))
218                   {
219                     isGetCapabilities = true;
220                   }
221                   if (item.equalsIgnoreCase('GetMap'))
222                   {
223                     isGetMap = true;
224                   }
225                 }
226                 (isGetCapabilities and isGetMap);</wsag-ogc:Status>
227             </wsag-ogc:CustomServiceLevel>
228           </wsag:CustomServiceLevel>
229         </wsag:ServiceLevelObjective>
230         <wsag:BusinessValueList/>
231       </wsag:GuaranteeTerm>
232       <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_AVAILABILITY" wsag:
              Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
              /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
233         <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
234         <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
235         <wsag:ServiceLevelObjective>
236           <wsag:CustomServiceLevel>
237             <wsag-ogc:CustomServiceLevel>
238               <wsag-ogc:Name>InspireAvailability</wsag-ogc:Name>
239               <wsag-ogc:Title>INSPIRE (Availability)</wsag-ogc:Title>
240               <wsag-ogc:Abstract>The probability of a Network Service to be
```

```
                    available shall be 99% of the time.</wsag-ogc:Abstract>
241                 <wsag-ogc:Status>(availability.week >= 0.99) and (availability
                        .month >= 0.99) and (availability.year >= 0.99)</wsag-ogc:
                        Status>
242               </wsag-ogc:CustomServiceLevel>
243             </wsag:CustomServiceLevel>
244           </wsag:ServiceLevelObjective>
245           <wsag:BusinessValueList/>
246         </wsag:GuaranteeTerm>
247         <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_RESPONSE" wsag:
                Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
                /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
                agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
                http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
                org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
                /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
                /2009/07/wsag4j-scheduling-extensions">
248           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
249           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
250           <wsag:ServiceLevelObjective>
251             <wsag:CustomServiceLevel>
252               <wsag-ogc:CustomServiceLevel>
253                 <wsag-ogc:Name>InspirePerformance</wsag-ogc:Name>
254                 <wsag-ogc:Title>INSPIRE (Performance)</wsag-ogc:Title>
255                 <wsag-ogc:Abstract>The response time for sending the initial
                        response to a Get Map Request to a view service shall be
                        maximum 5 seconds in normal situation.</wsag-ogc:Abstract>
256                 <wsag-ogc:Status>fulfilled = 0;
257                   for (item : response.initial.week) {
258                     if (item lt 5000)
259                     {
260                       fulfilled = fulfilled + 1;
261                     }
262                   }
263                   percent = fulfilled / (size(response.initial.week) / 100.0);
264                   percent gt 90.0;</wsag-ogc:Status>
265               </wsag-ogc:CustomServiceLevel>
266             </wsag:CustomServiceLevel>
267           </wsag:ServiceLevelObjective>
268           <wsag:BusinessValueList/>
269         </wsag:GuaranteeTerm>
270         <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_CAPACITY" wsag:
                Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
                /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
                agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
                http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
                org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
                /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
                /2009/07/wsag4j-scheduling-extensions">
271           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
272           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
273           <wsag:ServiceLevelObjective>
274             <wsag:CustomServiceLevel>
275               <wsag-ogc:CustomServiceLevel>
276                 <wsag-ogc:Name>InspireCapacity</wsag-ogc:Name>
277                 <wsag-ogc:Title>INSPIRE (Capacity)</wsag-ogc:Title>
278                 <wsag-ogc:Abstract>The minimum number of served simultaneous
                        service requests to a view service according to the
                        performance quality of service shall be 20 per second.</
                        wsag-ogc:Abstract>
279                 <wsag-ogc:Status>fulfilled = 0;
280                   for (item : capacity.initial.week) {
281                     if (item lt 5000)
282                     {
283                       fulfilled = fulfilled + 1;
284                     }
285                   }
286                   percent = fulfilled / (size(capacity.initial.week) / 100.0);
287                   percent gt 90.0;</wsag-ogc:Status>
288               </wsag-ogc:CustomServiceLevel>
289             </wsag:CustomServiceLevel>
```

```
290           </wsag:ServiceLevelObjective>
291           <wsag:BusinessValueList/>
292         </wsag:GuaranteeTerm>
293         <wsag:GuaranteeTerm wsag:Name="COSTS_PER_YEAR" wsag:Obligated="
                ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org/wsrf/bf-2"
                 xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement"
                xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="http://www.w3
                .org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.org/
                namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org/2005/08/
                addressing" xmlns:res-sla="http://schemas.wsag4j.org/2009/07/
                wsag4j-scheduling-extensions">
294           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
295           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
296           <wsag:ServiceLevelObjective/>
297           <wsag:BusinessValueList>
298             <wsag:CustomBusinessValue>
299               <wsag-ogc:CustomBusinessValue>
300                 <wsag-ogc:Name>CostsPerYear</wsag-ogc:Name>
301                 <wsag-ogc:Title>Usage Costs (Year)</wsag-ogc:Title>
302                 <wsag-ogc:Abstract>The cost to be assessed for using the
                        service on a yearly basis (in Euro).</wsag-ogc:Abstract>
303                 <wsag-ogc:Type>urn:ogc:def:sla:business:cost:year</wsag-ogc:
                        Type>
304                 <wsag-ogc:Value>factor;
305                   if (pixel.year lt (1000000 * 1000))
306                   {
307                     factor = 1.0;
308                   } else
309                   if (pixel.year lt (1000000 * 10000))
310                   {
311                     factor = 0.5;
312                   } else
313                   if (pixel.year lt (1000000 * 100000))
314                   {
315                   factor = 0.25;
316                   } else
317                   if (pixel.year lt (1000000 * 1000000))
318                   {
319                     factor = 0.125;
320                   } else
321                   {
322                     factor = 0.0625;
323                   }
324                   (factor * (pixel.year / 1000000.0));</wsag-ogc:Value>
325               </wsag-ogc:CustomBusinessValue>
326             </wsag:CustomBusinessValue>
327           </wsag:BusinessValueList>
328         </wsag:GuaranteeTerm>
329         <wsag:GuaranteeTerm wsag:Name="PENALTY_PER_YEAR" wsag:Obligated="
                ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org/wsrf/bf-2"
                 xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement"
                xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="http://www.w3
                .org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.org/
                namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org/2005/08/
                addressing" xmlns:res-sla="http://schemas.wsag4j.org/2009/07/
                wsag4j-scheduling-extensions">
330           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
331           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
332           <wsag:ServiceLevelObjective/>
333           <wsag:BusinessValueList>
334             <wsag:CustomBusinessValue>
335               <wsag-ogc:CustomBusinessValue>
336                 <wsag-ogc:Name>PenaltyPerYear</wsag-ogc:Name>
337                 <wsag-ogc:Title>Penalty (Year)</wsag-ogc:Title>
338                 <wsag-ogc:Abstract>The penalty to be assessed for not meeting
                        service level objectives on a yearly basis (in Euro).</
                        wsag-ogc:Abstract>
339                 <wsag-ogc:Type>urn:ogc:def:sla:business:penalty:year</wsag-ogc
                        :Type>
340                 <wsag-ogc:Value>factor;
341                   if (InspireAvailability.status == true)
```

```
342                 {
343                   factor = 0;
344                 }
345                 else
346                 {
347                   factor = 0.25;
348                 }
349                 (factor * CostsPerYear.value);</wsag-ogc:Value>
350               </wsag-ogc:CustomBusinessValue>
351             </wsag:CustomBusinessValue>
352           </wsag:BusinessValueList>
353         </wsag:GuaranteeTerm>
354       </ws:All>
355     </ws:Terms>
356   </ws:AgreementOffer>
```

### AGREEMENT

The XML in Listing C.10 shows an example agreement document.

Listing C.10: Example Agreement

```
1   <ws:AgreementOffer ws:AgreementId="b7f8fc14-17e3-499b-9845-975e7f41542e"
        xmlns:ws="http://schemas.ggf.org/graap/2007/03/ws-agreement">
2     <wsag:Context xmlns:wsrf="http://docs.oasis-open.org/wsrf/bf-2" xmlns:wsag
          ="http://schemas.ggf.org/graap/2007/03/ws-agreement" xmlns:ows="http
          ://www.opengis.net/ows/2.0" xmlns:xs="http://www.w3.org/2001/XMLSchema
          " xmlns:wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:
          addressing="http://www.w3.org/2005/08/addressing" xmlns:res-sla="http
          ://schemas.wsag4j.org/2009/07/wsag4j-scheduling-extensions">
3       <wsag:AgreementInitiator>
4         <wsag-ogc:Contact>
5           <wsag-ogc:Name>Instititute for Geoinformatics</wsag-ogc:Name>
6           <wsag-ogc:Site xlin:href="http://www.ifgi.de" xmlns:xlin="http://www
                .w3.org/1999/xlink"/>
7           <wsag-ogc:Contact>
8             <ows:IndividualName>Kristof Lange</ows:IndividualName>
9             <ows:PositionName>Student Assistance</ows:PositionName>
10            <ows:ContactInfo>
11              <ows:Phone>
12                <ows:Voice>+49 251 833307</ows:Voice>
13                <ows:Facsimile>+49 251 8339763</ows:Facsimile>
14              </ows:Phone>
15              <ows:Address>
16                <ows:DeliveryPoint>Weseler Strasse 253</ows:DeliveryPoint>
17                <ows:City>Muenster</ows:City>
18                <ows:PostalCode>48151</ows:PostalCode>
19                <ows:Country>Germany</ows:Country>
20                <ows:ElectronicMailAddress>kristof.lange@uni-muenster.de</ows:
                      ElectronicMailAddress>
21              </ows:Address>
22              <ows:OnlineResource xlin:href="http://www.ifgi.de" xmlns:xlin="
                    http://www.w3.org/1999/xlink"/>
23            </ows:ContactInfo>
24          </wsag-ogc:Contact>
25        </wsag-ogc:Contact>
26      </wsag:AgreementInitiator>
27      <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
28      <wsag:AgreementResponder>
29        <wsag-ogc:Contact>
30          <wsag-ogc:Name>Institute for Geoinformatics</wsag-ogc:Name>
31          <wsag-ogc:Site xlin:href="http://www.ifgi.de" xmlns:xlin="http://www
                .w3.org/1999/xlink"/>
32          <wsag-ogc:Contact>
33            <ows:IndividualName>Bastian Baranski</ows:IndividualName>
34            <ows:PositionName>Research Associate</ows:PositionName>
35            <ows:ContactInfo>
```

```
36              <ows:Phone>
37                <ows:Voice>+49 251 8333071</ows:Voice>
38                <ows:Facsimile>+49 251 8339763</ows:Facsimile>
39              </ows:Phone>
40              <ows:Address>
41                <ows:DeliveryPoint>Weseler Strasse 253</ows:DeliveryPoint>
42                <ows:City>Muenster</ows:City>
43                <ows:PostalCode>48151</ows:PostalCode>
44                <ows:Country>Germany</ows:Country>
45                <ows:ElectronicMailAddress>baranski@uni-muenster.de</ows:
                      ElectronicMailAddress>
46              </ows:Address>
47              <ows:HoursOfService>The hours of service are Monday to Friday
                    from 8 AM to 16 PM.</ows:HoursOfService>
48              <ows:ContactInstructions>Please contact the service desk via
                    phone or mail.</ows:ContactInstructions>
49            </ows:ContactInfo>
50          </wsag-ogc:Contact>
51        </wsag-ogc:Contact>
52      </wsag:AgreementResponder>
53      <wsag:TemplateId>WSAG_DEFAULT_TEMPLATE_8</wsag:TemplateId>
54      <wsag:TemplateName>INSPIRE_VIEW_SERVICE_TEMPLATE</wsag:TemplateName>
55    </wsag:Context>
56    <ws:Terms>
57      <ws:All>
58        <wsag:ServiceDescriptionTerm wsag:Name="SERVICE_DESCRIPTION_SDT" wsag:
              ServiceName="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-
              open.org/wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap
              /2007/03/ws-agreement" xmlns:ows="http://www.opengis.net/ows/2.0"
              xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http
              ://www.ifgi.org/namespaces/wsag/ogc" xmlns:addressing="http://www.
              w3.org/2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.
              org/2009/07/wsag4j-scheduling-extensions">
59          <wsag-ogc:ServiceDescription>
60            <wsag-ogc:Title>INSPIRE View Service</wsag-ogc:Title>
61            <wsag-ogc:Abstract>This service instance is an INSPIRE View
                  Service implementation.</wsag-ogc:Abstract>
62            <wsag-ogc:Keywords>INSPIRE, View Service, OGC, WMS</wsag-ogc:
                  Keywords>
63            <wsag-ogc:Type>urn:ogc:doc:is:wms:1.1.1</wsag-ogc:Type>
64          </wsag-ogc:ServiceDescription>
65        </wsag:ServiceDescriptionTerm>
66        <wsag:ServiceDescriptionTerm wsag:Name="SERVICE_PROPERTIES_SDT" wsag:
              ServiceName="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-
              open.org/wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap
              /2007/03/ws-agreement" xmlns:ows="http://www.opengis.net/ows/2.0"
              xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http
              ://www.ifgi.org/namespaces/wsag/ogc" xmlns:addressing="http://www.
              w3.org/2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.
              org/2009/07/wsag4j-scheduling-extensions">
67          <wsag-ogc:ServiceProperties>
68            <!--RESOURCE-RELATED PROPERTIES-->
69            <wsag-ogc:Property>
70              <wsag-ogc:Name>operations</wsag-ogc:Name>
71              <wsag-ogc:Title>Supported Operations</wsag-ogc:Title>
72              <wsag-ogc:Abstract>The operations that are supported by the
                    service.</wsag-ogc:Abstract>
73              <wsag-ogc:Type>urn:ogc:def:sla:property:resource:operation</wsag
                    -ogc:Type>
74              <wsag-ogc:Monitoring>
75                <wsag-ogc:ActiveMonitoring>
76                  <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
77                  <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
78                  <wsag-ogc:Period>360000</wsag-ogc:Period>
79                </wsag-ogc:ActiveMonitoring>
80              </wsag-ogc:Monitoring>
81            </wsag-ogc:Property>
82            <!--RUNTIME-RELATED PROPERTIES-->
83            <wsag-ogc:Property>
84              <wsag-ogc:Name>availability</wsag-ogc:Name>
85              <wsag-ogc:Title>Service Availability</wsag-ogc:Title>
86              <wsag-ogc:Abstract>The general availability of the service.</
                    wsag-ogc:Abstract>
```

```
87              <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:availability</
                    wsag-ogc:Type>
88              <wsag-ogc:Monitoring>
89                <wsag-ogc:ActiveMonitoring>
90                  <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
91                  <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
92                  <wsag-ogc:Period>360000</wsag-ogc:Period>
93                  <wsag-ogc:Request>
94                    <wsag-ogc:Method>GET</wsag-ogc:Method>
95                    <wsag-ogc:Content><![CDATA[service=WMS&version=1.3.0&
                          request=GetMap&layers=topp:tasmania_state_boundaries&
                          styles=&bbox=${__random(142.0,144.0)},${__random
                          (-46.0,-44.0)},${__random(150.0,152.0)},${__random
                          (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&
                          format=image/png]]></wsag-ogc:Content>
96                  </wsag-ogc:Request>
97                  <wsag-ogc:Response>
98                    <wsag-ogc:Status>200</wsag-ogc:Status>
99                  </wsag-ogc:Response>
100               </wsag-ogc:ActiveMonitoring>
101             </wsag-ogc:Monitoring>
102           </wsag-ogc:Property>
103           <wsag-ogc:Property>
104             <wsag-ogc:Name>response</wsag-ogc:Name>
105             <wsag-ogc:Title>Response Time</wsag-ogc:Title>
106             <wsag-ogc:Abstract>The response time of the service.</wsag-ogc:
                    Abstract>
107             <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:response</wsag-
                    ogc:Type>
108             <wsag-ogc:Monitoring>
109               <wsag-ogc:ActiveMonitoring>
110                 <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
111                 <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
112                 <wsag-ogc:Period>360000</wsag-ogc:Period>
113                 <wsag-ogc:Request>
114                   <wsag-ogc:Method>GET</wsag-ogc:Method>
115                   <wsag-ogc:Content><![CDATA[service=WMS&version=1.3.0&
                          request=GetMap&layers=topp:tasmania_state_boundaries&
                          styles=&bbox=${__random(142.0,144.0)},${__random
                          (-46.0,-44.0)},${__random(150.0,152.0)},${__random
                          (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&
                          format=image/png]]></wsag-ogc:Content>
116                 </wsag-ogc:Request>
117                 <wsag-ogc:Response>
118                   <wsag-ogc:Status>200</wsag-ogc:Status>
119                 </wsag-ogc:Response>
120               </wsag-ogc:ActiveMonitoring>
121             </wsag-ogc:Monitoring>
122           </wsag-ogc:Property>
123           <wsag-ogc:Property>
124             <wsag-ogc:Name>capacity</wsag-ogc:Name>
125             <wsag-ogc:Title>Service Capacity</wsag-ogc:Title>
126             <wsag-ogc:Abstract>The response time of the service for multiple
                     parallel requests.</wsag-ogc:Abstract>
127             <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:response</wsag-
                    ogc:Type>
128             <wsag-ogc:Monitoring>
129               <wsag-ogc:ActiveMonitoring>
130                 <wsag-ogc:Start>20:00:00</wsag-ogc:Start>
131                 <wsag-ogc:Stop>04:00:00</wsag-ogc:Stop>
132                 <wsag-ogc:Period>3600000</wsag-ogc:Period>
133                 <wsag-ogc:Session>
134                   <wsag-ogc:Capacity>20</wsag-ogc:Capacity>
135                   <wsag-ogc:Duration>60000</wsag-ogc:Duration>
136                   <wsag-ogc:Period>1000</wsag-ogc:Period>
137                 </wsag-ogc:Session>
138                 <wsag-ogc:Request>
139                   <wsag-ogc:Chance>10</wsag-ogc:Chance>
140                   <wsag-ogc:Method>GET</wsag-ogc:Method>
141                   <wsag-ogc:Content>service=WMS&amp;version=1.3.0&amp;
                          request=GetCapabilities</wsag-ogc:Content>
142                 </wsag-ogc:Request>
143                 <wsag-ogc:Request>
```

```
144                     <wsag-ogc:Chance>90</wsag-ogc:Chance>
145                     <wsag-ogc:Method>GET</wsag-ogc:Method>
146                     <wsag-ogc:Content><![CDATA[service=WMS&version=1.3.0&
                            request=GetMap&layers=topp:tasmania_state_boundaries&
                            styles=&bbox=${__random(142.0,144.0)},${__random
                            (-46.0,-44.0)},${__random(150.0,152.0)},${__random
                            (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&
                            format=image/png]]></wsag-ogc:Content>
147                   </wsag-ogc:Request>
148                   <wsag-ogc:Response>
149                     <wsag-ogc:Status>200</wsag-ogc:Status>
150                   </wsag-ogc:Response>
151                 </wsag-ogc:ActiveMonitoring>
152               </wsag-ogc:Monitoring>
153             </wsag-ogc:Property>
154             <!--USAGE-RELATED PROPERTIES-->
155             <wsag-ogc:Property>
156               <wsag-ogc:Name>pixel</wsag-ogc:Name>
157               <wsag-ogc:Title>Sum of Pixels</wsag-ogc:Title>
158               <wsag-ogc:Abstract>The accessed number of pixels.</wsag-ogc:
                      Abstract>
159               <wsag-ogc:Type>urn:ogc:def:sla:property:usage:pixel</wsag-ogc:
                      Type>
160               <wsag-ogc:Monitoring>
161                 <wsag-ogc:PassiveMonitoring>
162                   <wsag-ogc:Request>
163                     <wsag-ogc:Resource>/state/urn:ogc:def:sla:property:usage:
                            pixel</wsag-ogc:Resource>
164                     <wsag-ogc:Method>GET</wsag-ogc:Method>
165                   </wsag-ogc:Request>
166                 </wsag-ogc:PassiveMonitoring>
167               </wsag-ogc:Monitoring>
168             </wsag-ogc:Property>
169             <!--INFRASTRUCTURE-RELATED PROPERTIES-->
170             <wsag-ogc:Property>
171               <wsag-ogc:Name>provider</wsag-ogc:Name>
172               <wsag-ogc:Title>Infrastructure Provider</wsag-ogc:Title>
173               <wsag-ogc:Abstract>The name of the infrastructure provider.</
                      wsag-ogc:Abstract>
174               <wsag-ogc:Type>urn:ogc:def:sla:property:infrastructure:provider:
                      name</wsag-ogc:Type>
175               <wsag-ogc:Value>default</wsag-ogc:Value>
176             </wsag-ogc:Property>
177             <wsag-ogc:Property>
178               <wsag-ogc:Name>image</wsag-ogc:Name>
179               <wsag-ogc:Title>Virtual Machine</wsag-ogc:Title>
180               <wsag-ogc:Abstract>The name of the Virtual Machine (VM) template
                      .</wsag-ogc:Abstract>
181               <wsag-ogc:Type>urn:ogc:def:sla:property:infrastructure:vm:name</
                      wsag-ogc:Type>
182               <wsag-ogc:Value>ami-59f9c62d</wsag-ogc:Value>
183             </wsag-ogc:Property>
184           </wsag-ogc:ServiceProperties>
185         </wsag:ServiceDescriptionTerm>
186         <ws:ServiceDescriptionTerm ws:Name="TIME_CONSTRAINT_SDT" ws:
                ServiceName="INSPIRE_VIEW_SERVICE">
187           <wsag:TimeConstraint xmlns:wsag="http://schemas.wsag4j.org/2009/07/
                  wsag4j-scheduling-extensions">
188             <wsag:StartTime>2011-05-27T11:00:00</wsag:StartTime>
189             <wsag:EndTime>2013-05-27T11:00:00</wsag:EndTime>
190           </wsag:TimeConstraint>
191         </ws:ServiceDescriptionTerm>
192         <wsag:ServiceProperties wsag:Name="SERVICE_PROPERTIES" wsag:
                ServiceName="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-
                open.org/wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap
                /2007/03/ws-agreement" xmlns:ows="http://www.opengis.net/ows/2.0"
                xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http
                ://www.ifgi.org/namespaces/wsag/ogc" xmlns:addressing="http://www.
                w3.org/2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.
                org/2009/07/wsag4j-scheduling-extensions">
193           <wsag:VariableSet>
194             <wsag:Variable wsag:Name="SERVICE_PROPERTIES_STATE" wsag:Metric="
                    xs:string">
```

```
195             <wsag:Location>declare namespace ws='http://schemas.ggf.org/
                    graap/2007/03/ws-agreement';declare namespace wsag-ogc='http
                    ://www.ifgi.org/namespaces/wsag/ogc';declare namespace wsag
                    ='http://schemas.ggf.org/graap/2007/03/ws-agreement';/ws:
                    AgreementProperties/ws:ServiceTermState[@termName='
                    SERVICE_PROPERTIES_SDT']/ws:State/text()</wsag:Location>
196           </wsag:Variable>
197         </wsag:VariableSet>
198       </wsag:ServiceProperties>
199       <wsag:ServiceReference wsag:Name="SERVICE_REFERENCE" wsag:ServiceName
              ="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-open.org/
              wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
200         <wsag-ogc:ServiceReference>
201           <wsag-ogc:URL>http://localhost:8088/sla-proxy/DefaultWMS</wsag-ogc
                  :URL>
202         </wsag-ogc:ServiceReference>
203       </wsag:ServiceReference>
204       <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RESOURCE_OPERATIONS" wsag:
              Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
              /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
205         <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
206         <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
207         <wsag:ServiceLevelObjective>
208           <wsag:CustomServiceLevel>
209             <wsag-ogc:CustomServiceLevel>
210               <wsag-ogc:Name>InspireOperations</wsag-ogc:Name>
211               <wsag-ogc:Title>INSPIRE (Operations)</wsag-ogc:Title>
212               <wsag-ogc:Abstract>The following operations shall be
                      implemented for an INSPIRE View service: GetCapabilities,
                      GetMap.</wsag-ogc:Abstract>
213               <wsag-ogc:Status>isGetCapabilities = false;
214                 isGetMap = false;
215                 for (item : operations.name)
216                 {
217                   if (item.equalsIgnoreCase('GetCapabilities'))
218                   {
219                     isGetCapabilities = true;
220                   }
221                   if (item.equalsIgnoreCase('GetMap'))
222                   {
223                     isGetMap = true;
224                   }
225                 }
226                 (isGetCapabilities and isGetMap);</wsag-ogc:Status>
227             </wsag-ogc:CustomServiceLevel>
228           </wsag:CustomServiceLevel>
229         </wsag:ServiceLevelObjective>
230         <wsag:BusinessValueList/>
231       </wsag:GuaranteeTerm>
232       <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_AVAILABILITY" wsag:
              Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
              /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
233         <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
234         <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
235         <wsag:ServiceLevelObjective>
236           <wsag:CustomServiceLevel>
```

```
237             <wsag-ogc:CustomServiceLevel>
238               <wsag-ogc:Name>InspireAvailability</wsag-ogc:Name>
239               <wsag-ogc:Title>INSPIRE (Availability)</wsag-ogc:Title>
240               <wsag-ogc:Abstract>The probability of a Network Service to be
                      available shall be 99% of the time.</wsag-ogc:Abstract>
241               <wsag-ogc:Status>(availability.week >= 0.99) and (availability
                      .month >= 0.99) and (availability.year >= 0.99)</wsag-ogc:
                      Status>
242             </wsag-ogc:CustomServiceLevel>
243           </wsag:CustomServiceLevel>
244         </wsag:ServiceLevelObjective>
245         <wsag:BusinessValueList/>
246       </wsag:GuaranteeTerm>
247       <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_RESPONSE" wsag:
              Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
              /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
248         <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
249         <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
250         <wsag:ServiceLevelObjective>
251           <wsag:CustomServiceLevel>
252             <wsag-ogc:CustomServiceLevel>
253               <wsag-ogc:Name>InspirePerformance</wsag-ogc:Name>
254               <wsag-ogc:Title>INSPIRE (Performance)</wsag-ogc:Title>
255               <wsag-ogc:Abstract>The response time for sending the initial
                      response to a Get Map Request to a view service shall be
                      maximum 5 seconds in normal situation.</wsag-ogc:Abstract>
256               <wsag-ogc:Status>fulfilled = 0;
257                 for (item : response.initial.week) {
258                   if (item lt 5000)
259                   {
260                     fulfilled = fulfilled + 1;
261                   }
262                 }
263                 percent = fulfilled / (size(response.initial.week) / 100.0);
264                 percent gt 90.0;</wsag-ogc:Status>
265             </wsag-ogc:CustomServiceLevel>
266           </wsag:CustomServiceLevel>
267         </wsag:ServiceLevelObjective>
268         <wsag:BusinessValueList/>
269       </wsag:GuaranteeTerm>
270       <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_CAPACITY" wsag:
              Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
              /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
271         <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
272         <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
273         <wsag:ServiceLevelObjective>
274           <wsag:CustomServiceLevel>
275             <wsag-ogc:CustomServiceLevel>
276               <wsag-ogc:Name>InspireCapacity</wsag-ogc:Name>
277               <wsag-ogc:Title>INSPIRE (Capacity)</wsag-ogc:Title>
278               <wsag-ogc:Abstract>The minimum number of served simultaneous
                      service requests to a view service according to the
                      performance quality of service shall be 20 per second.</
                      wsag-ogc:Abstract>
279               <wsag-ogc:Status>fulfilled = 0;
280                 for (item : capacity.initial.week) {
281                   if (item lt 5000)
282                   {
283                     fulfilled = fulfilled + 1;
284                   }
285                 }
```

```
286                 percent = fulfilled / (size(capacity.initial.week) / 100.0);
287                 percent gt 90.0;</wsag-ogc:Status>
288             </wsag-ogc:CustomServiceLevel>
289           </wsag:CustomServiceLevel>
290         </wsag:ServiceLevelObjective>
291         <wsag:BusinessValueList/>
292       </wsag:GuaranteeTerm>
293       <wsag:GuaranteeTerm wsag:Name="COSTS_PER_YEAR" wsag:Obligated="
              ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org/wsrf/bf-2"
               xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement"
              xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="http://www.w3
              .org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.org/
              namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org/2005/08/
              addressing" xmlns:res-sla="http://schemas.wsag4j.org/2009/07/
              wsag4j-scheduling-extensions">
294         <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
295         <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
296         <wsag:ServiceLevelObjective/>
297         <wsag:BusinessValueList>
298           <wsag:CustomBusinessValue>
299             <wsag-ogc:CustomBusinessValue>
300               <wsag-ogc:Name>CostsPerYear</wsag-ogc:Name>
301               <wsag-ogc:Title>Usage Costs (Year)</wsag-ogc:Title>
302               <wsag-ogc:Abstract>The cost to be assessed for using the
                      service on a yearly basis (in Euro).</wsag-ogc:Abstract>
303               <wsag-ogc:Type>urn:ogc:def:sla:business:cost:year</wsag-ogc:
                      Type>
304               <wsag-ogc:Value>factor;
305                 if (pixel.year lt (1000000 * 1000))
306                 {
307                   factor = 1.0;
308                 } else
309                 if (pixel.year lt (1000000 * 10000))
310                 {
311                   factor = 0.5;
312                 } else
313                 if (pixel.year lt (1000000 * 100000))
314                 {
315                 factor = 0.25;
316                 } else
317                 if (pixel.year lt (1000000 * 1000000))
318                 {
319                   factor = 0.125;
320                 } else
321                 {
322                   factor = 0.0625;
323                 }
324                 (factor * (pixel.year / 1000000.0));</wsag-ogc:Value>
325             </wsag-ogc:CustomBusinessValue>
326           </wsag:CustomBusinessValue>
327         </wsag:BusinessValueList>
328       </wsag:GuaranteeTerm>
329       <wsag:GuaranteeTerm wsag:Name="PENALTY_PER_YEAR" wsag:Obligated="
              ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org/wsrf/bf-2"
               xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement"
              xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="http://www.w3
              .org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.org/
              namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org/2005/08/
              addressing" xmlns:res-sla="http://schemas.wsag4j.org/2009/07/
              wsag4j-scheduling-extensions">
330         <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
331         <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
332         <wsag:ServiceLevelObjective/>
333         <wsag:BusinessValueList>
334           <wsag:CustomBusinessValue>
335             <wsag-ogc:CustomBusinessValue>
336               <wsag-ogc:Name>PenaltyPerYear</wsag-ogc:Name>
337               <wsag-ogc:Title>Penalty (Year)</wsag-ogc:Title>
338               <wsag-ogc:Abstract>The penalty to be assessed for not meeting
                      service level objectives on a yearly basis (in Euro).</
                      wsag-ogc:Abstract>
```

```
339                <wsag-ogc:Type>urn:ogc:def:sla:business:penalty:year</wsag-ogc
                       :Type>
340                <wsag-ogc:Value>factor;
341                  if (InspireAvailability.status == true)
342                  {
343                    factor = 0;
344                  }
345                  else
346                  {
347                    factor = 0.25;
348                  }
349                  (factor * CostsPerYear.value);</wsag-ogc:Value>
350              </wsag-ogc:CustomBusinessValue>
351            </wsag:CustomBusinessValue>
352          </wsag:BusinessValueList>
353        </wsag:GuaranteeTerm>
354      </ws:All>
355    </ws:Terms>
356  </ws:AgreementOffer>
```

### AGREEMENT PROPERTIES

The XML in Listing C.11 shows an example agreement properties document.

Listing C.11: Example Agreement Properties

```
1   <ws:AgreementProperties xmlns:ws="http://schemas.ggf.org/graap/2007/03/ws-
        agreement">
2     <ws:Name xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
          instance"/>
3     <ws:AgreementId>246fc372-5aeb-4f76-a718-775e944d1698</ws:AgreementId>
4     <wsag:Context xmlns:wsrf="http://docs.oasis-open.org/wsrf/bf-2" xmlns:wsag
          ="http://schemas.ggf.org/graap/2007/03/ws-agreement" xmlns:ows="http
          ://www.opengis.net/ows/2.0" xmlns:xs="http://www.w3.org/2001/XMLSchema
          " xmlns:wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:
          addressing="http://www.w3.org/2005/08/addressing" xmlns:res-sla="http
          ://schemas.wsag4j.org/2009/07/wsag4j-scheduling-extensions">
5       <wsag:AgreementInitiator>
6         <wsag-ogc:Contact>
7           <wsag-ogc:Name>Instititute for Geoinformatics</wsag-ogc:Name>
8           <wsag-ogc:Site xlin:href="http://www.ifgi.de" xmlns:xlin="http://www
                .w3.org/1999/xlink"/>
9           <wsag-ogc:Contact>
10            <ows:IndividualName>Kristof Lange</ows:IndividualName>
11            <ows:PositionName>Student Assistance</ows:PositionName>
12            <ows:ContactInfo>
13              <ows:Phone>
14                <ows:Voice>+49 251 833307</ows:Voice>
15                <ows:Facsimile>+49 251 8339763</ows:Facsimile>
16              </ows:Phone>
17              <ows:Address>
18                <ows:DeliveryPoint>Weseler Strasse 253</ows:DeliveryPoint>
19                <ows:City>Muenster</ows:City>
20                <ows:PostalCode>48151</ows:PostalCode>
21                <ows:Country>Germany</ows:Country>
22                <ows:ElectronicMailAddress>kristof.lange@uni-muenster.de</ows:
                      ElectronicMailAddress>
23              </ows:Address>
24              <ows:OnlineResource xlin:href="http://www.ifgi.de" xmlns:xlin="
                    http://www.w3.org/1999/xlink"/>
25            </ows:ContactInfo>
26          </wsag-ogc:Contact>
27        </wsag-ogc:Contact>
28      </wsag:AgreementInitiator>
29      <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
30      <wsag:AgreementResponder>
31        <wsag-ogc:Contact>
```

```
32          <wsag-ogc:Name>Institute for Geoinformatics</wsag-ogc:Name>
33          <wsag-ogc:Site xlin:href="http://www.ifgi.de" xmlns:xlin="http://www
                .w3.org/1999/xlink"/>
34          <wsag-ogc:Contact>
35            <ows:IndividualName>Bastian Baranski</ows:IndividualName>
36            <ows:PositionName>Research Associate</ows:PositionName>
37            <ows:ContactInfo>
38              <ows:Phone>
39                <ows:Voice>+49 251 8333071</ows:Voice>
40                <ows:Facsimile>+49 251 8339763</ows:Facsimile>
41              </ows:Phone>
42              <ows:Address>
43                <ows:DeliveryPoint>Weseler Strasse 253</ows:DeliveryPoint>
44                <ows:City>Muenster</ows:City>
45                <ows:PostalCode>48151</ows:PostalCode>
46                <ows:Country>Germany</ows:Country>
47                <ows:ElectronicMailAddress>baranski@uni-muenster.de</ows:
                      ElectronicMailAddress>
48              </ows:Address>
49              <ows:HoursOfService>The hours of service are Monday to Friday
                      from 8 AM to 16 PM.</ows:HoursOfService>
50              <ows:ContactInstructions>Please contact the service desk via
                      phone or mail.</ows:ContactInstructions>
51            </ows:ContactInfo>
52          </wsag-ogc:Contact>
53        </wsag-ogc:Contact>
54      </wsag:AgreementResponder>
55      <wsag:TemplateId>WSAG_DEFAULT_TEMPLATE_8</wsag:TemplateId>
56      <wsag:TemplateName>INSPIRE_VIEW_SERVICE_TEMPLATE</wsag:TemplateName>
57    </wsag:Context>
58    <ws:Terms>
59      <ws:All>
60        <wsag:ServiceDescriptionTerm wsag:Name="SERVICE_DESCRIPTION_SDT" wsag:
              ServiceName="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-
              open.org/wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap
              /2007/03/ws-agreement" xmlns:ows="http://www.opengis.net/ows/2.0"
              xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http
              ://www.ifgi.org/namespaces/wsag/ogc" xmlns:addressing="http://www.
              w3.org/2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.
              org/2009/07/wsag4j-scheduling-extensions">
61          <wsag-ogc:ServiceDescription>
62            <wsag-ogc:Title>INSPIRE View Service</wsag-ogc:Title>
63            <wsag-ogc:Abstract>This service instance is an INSPIRE View
                  Service implementation.</wsag-ogc:Abstract>
64            <wsag-ogc:Keywords>INSPIRE, View Service, OGC, WMS</wsag-ogc:
                  Keywords>
65            <wsag-ogc:Type>urn:ogc:doc:is:wms:1.1.1</wsag-ogc:Type>
66          </wsag-ogc:ServiceDescription>
67        </wsag:ServiceDescriptionTerm>
68        <wsag:ServiceDescriptionTerm wsag:Name="SERVICE_PROPERTIES_SDT" wsag:
              ServiceName="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-
              open.org/wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap
              /2007/03/ws-agreement" xmlns:ows="http://www.opengis.net/ows/2.0"
              xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http
              ://www.ifgi.org/namespaces/wsag/ogc" xmlns:addressing="http://www.
              w3.org/2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.
              org/2009/07/wsag4j-scheduling-extensions">
69          <wsag-ogc:ServiceProperties>
70            <!--RESOURCE-RELATED PROPERTIES-->
71            <wsag-ogc:Property>
72              <wsag-ogc:Name>operations</wsag-ogc:Name>
73              <wsag-ogc:Title>Supported Operations</wsag-ogc:Title>
74              <wsag-ogc:Abstract>The operations that are supported by the
                    service.</wsag-ogc:Abstract>
75              <wsag-ogc:Type>urn:ogc:def:sla:property:resource:operation</wsag
                    -ogc:Type>
76              <wsag-ogc:Monitoring>
77                <wsag-ogc:ActiveMonitoring>
78                  <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
79                  <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
80                  <wsag-ogc:Period>360000</wsag-ogc:Period>
81                </wsag-ogc:ActiveMonitoring>
82              </wsag-ogc:Monitoring>
```

```
83              </wsag-ogc:Property>
84              <!--RUNTIME-RELATED PROPERTIES-->
85              <wsag-ogc:Property>
86                <wsag-ogc:Name>availability</wsag-ogc:Name>
87                <wsag-ogc:Title>Service Availability</wsag-ogc:Title>
88                <wsag-ogc:Abstract>The general availability of the service.</
                      wsag-ogc:Abstract>
89                <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:availability</
                      wsag-ogc:Type>
90                <wsag-ogc:Monitoring>
91                  <wsag-ogc:ActiveMonitoring>
92                    <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
93                    <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
94                    <wsag-ogc:Period>360000</wsag-ogc:Period>
95                    <wsag-ogc:Request>
96                      <wsag-ogc:Method>GET</wsag-ogc:Method>
97                      <wsag-ogc:Content><![CDATA[service=WMS&version=1.3.0&
                            request=GetMap&layers=topp:tasmania_state_boundaries&
                            styles=&bbox=${__random(142.0,144.0)},${__random
                            (-46.0,-44.0)},${__random(150.0,152.0)},${__random
                            (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&
                            format=image/png]]></wsag-ogc:Content>
98                    </wsag-ogc:Request>
99                    <wsag-ogc:Response>
100                     <wsag-ogc:Status>200</wsag-ogc:Status>
101                   </wsag-ogc:Response>
102                 </wsag-ogc:ActiveMonitoring>
103               </wsag-ogc:Monitoring>
104             </wsag-ogc:Property>
105             <wsag-ogc:Property>
106               <wsag-ogc:Name>response</wsag-ogc:Name>
107               <wsag-ogc:Title>Response Time</wsag-ogc:Title>
108               <wsag-ogc:Abstract>The response time of the service.</wsag-ogc:
                      Abstract>
109               <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:response</wsag-
                      ogc:Type>
110               <wsag-ogc:Monitoring>
111                 <wsag-ogc:ActiveMonitoring>
112                   <wsag-ogc:Start>00:00:00</wsag-ogc:Start>
113                   <wsag-ogc:Stop>23:59:59</wsag-ogc:Stop>
114                   <wsag-ogc:Period>360000</wsag-ogc:Period>
115                   <wsag-ogc:Request>
116                     <wsag-ogc:Method>GET</wsag-ogc:Method>
117                     <wsag-ogc:Content><![CDATA[service=WMS&version=1.3.0&
                            request=GetMap&layers=topp:tasmania_state_boundaries&
                            styles=&bbox=${__random(142.0,144.0)},${__random
                            (-46.0,-44.0)},${__random(150.0,152.0)},${__random
                            (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&
                            format=image/png]]></wsag-ogc:Content>
118                   </wsag-ogc:Request>
119                   <wsag-ogc:Response>
120                     <wsag-ogc:Status>200</wsag-ogc:Status>
121                   </wsag-ogc:Response>
122                 </wsag-ogc:ActiveMonitoring>
123               </wsag-ogc:Monitoring>
124             </wsag-ogc:Property>
125             <wsag-ogc:Property>
126               <wsag-ogc:Name>capacity</wsag-ogc:Name>
127               <wsag-ogc:Title>Service Capacity</wsag-ogc:Title>
128               <wsag-ogc:Abstract>The response time of the service for multiple
                      parallel requests.</wsag-ogc:Abstract>
129               <wsag-ogc:Type>urn:ogc:def:sla:property:runtime:response</wsag-
                      ogc:Type>
130               <wsag-ogc:Monitoring>
131                 <wsag-ogc:ActiveMonitoring>
132                   <wsag-ogc:Start>20:00:00</wsag-ogc:Start>
133                   <wsag-ogc:Stop>04:00:00</wsag-ogc:Stop>
134                   <wsag-ogc:Period>3600000</wsag-ogc:Period>
135                   <wsag-ogc:Session>
136                     <wsag-ogc:Capacity>20</wsag-ogc:Capacity>
137                     <wsag-ogc:Duration>60000</wsag-ogc:Duration>
138                     <wsag-ogc:Period>1000</wsag-ogc:Period>
139                   </wsag-ogc:Session>
```

```
140                 <wsag-ogc:Request>
141                   <wsag-ogc:Chance>10</wsag-ogc:Chance>
142                   <wsag-ogc:Method>GET</wsag-ogc:Method>
143                   <wsag-ogc:Content>service=WMS&amp;version=1.3.0&amp;
                          request=GetCapabilities</wsag-ogc:Content>
144                 </wsag-ogc:Request>
145                 <wsag-ogc:Request>
146                   <wsag-ogc:Chance>90</wsag-ogc:Chance>
147                   <wsag-ogc:Method>GET</wsag-ogc:Method>
148                   <wsag-ogc:Content><![CDATA[service=WMS&version=1.3.0&
                          request=GetMap&layers=topp:tasmania_state_boundaries&
                          styles=&bbox=${__random(142.0,144.0)},${__random
                          (-46.0,-44.0)},${__random(150.0,152.0)},${__random
                          (-38.0,-36.0)}&width=800&height=600&srs=EPSG:4326&
                          format=image/png]]></wsag-ogc:Content>
149                 </wsag-ogc:Request>
150                 <wsag-ogc:Response>
151                   <wsag-ogc:Status>200</wsag-ogc:Status>
152                 </wsag-ogc:Response>
153               </wsag-ogc:ActiveMonitoring>
154             </wsag-ogc:Monitoring>
155           </wsag-ogc:Property>
156           <!--USAGE-RELATED PROPERTIES-->
157           <wsag-ogc:Property>
158             <wsag-ogc:Name>pixel</wsag-ogc:Name>
159             <wsag-ogc:Title>Sum of Pixels</wsag-ogc:Title>
160             <wsag-ogc:Abstract>The accessed number of pixels.</wsag-ogc:
                    Abstract>
161             <wsag-ogc:Type>urn:ogc:def:sla:property:usage:pixel</wsag-ogc:
                    Type>
162             <wsag-ogc:Monitoring>
163               <wsag-ogc:PassiveMonitoring>
164                 <wsag-ogc:Request>
165                   <wsag-ogc:Resource>/state/urn:ogc:def:sla:property:usage:
                          pixel</wsag-ogc:Resource>
166                   <wsag-ogc:Method>GET</wsag-ogc:Method>
167                 </wsag-ogc:Request>
168               </wsag-ogc:PassiveMonitoring>
169             </wsag-ogc:Monitoring>
170           </wsag-ogc:Property>
171           <!--INFRASTRUCTURE-RELATED PROPERTIES-->
172           <wsag-ogc:Property>
173             <wsag-ogc:Name>provider</wsag-ogc:Name>
174             <wsag-ogc:Title>Infrastructure Provider</wsag-ogc:Title>
175             <wsag-ogc:Abstract>The name of the infrastructure provider.</
                    wsag-ogc:Abstract>
176             <wsag-ogc:Type>urn:ogc:def:sla:property:infrastructure:provider:
                    name</wsag-ogc:Type>
177             <wsag-ogc:Value>default</wsag-ogc:Value>
178           </wsag-ogc:Property>
179           <wsag-ogc:Property>
180             <wsag-ogc:Name>image</wsag-ogc:Name>
181             <wsag-ogc:Title>Virtual Machine</wsag-ogc:Title>
182             <wsag-ogc:Abstract>The name of the Virtual Machine (VM) template
                    .</wsag-ogc:Abstract>
183             <wsag-ogc:Type>urn:ogc:def:sla:property:infrastructure:vm:name</
                    wsag-ogc:Type>
184             <wsag-ogc:Value>ami-59f9c62d</wsag-ogc:Value>
185           </wsag-ogc:Property>
186         </wsag-ogc:ServiceProperties>
187       </wsag:ServiceDescriptionTerm>
188       <ws:ServiceDescriptionTerm ws:Name="TIME_CONSTRAINT_SDT" ws:
              ServiceName="INSPIRE_VIEW_SERVICE">
189         <wsag:TimeConstraint xmlns:wsag="http://schemas.wsag4j.org/2009/07/
                wsag4j-scheduling-extensions">
190           <wsag:StartTime>2011-05-27T11:00:00</wsag:StartTime>
191           <wsag:EndTime>2013-05-27T11:00:00</wsag:EndTime>
192         </wsag:TimeConstraint>
193       </ws:ServiceDescriptionTerm>
194       <wsag:ServiceProperties wsag:Name="SERVICE_PROPERTIES" wsag:
              ServiceName="INSPIRE_VIEW_SERVICE" xmlns:wsrf="http://docs.oasis-
              open.org/wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap
              /2007/03/ws-agreement" xmlns:ows="http://www.opengis.net/ows/2.0"
```

```
            xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http
            ://www.ifgi.org/namespaces/wsag/ogc" xmlns:addressing="http://www.
            w3.org/2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.
            org/2009/07/wsag4j-scheduling-extensions">
195           <wsag:VariableSet>
196             <wsag:Variable wsag:Name="SERVICE_PROPERTIES_STATE" wsag:Metric="
                  xs:string">
197               <wsag:Location>declare namespace ws='http://schemas.ggf.org/
                    graap/2007/03/ws-agreement';declare namespace wsag-ogc='http
                    ://www.ifgi.org/namespaces/wsag/ogc';declare namespace wsag
                    ='http://schemas.ggf.org/graap/2007/03/ws-agreement';/ws:
                    AgreementProperties/ws:ServiceTermState[@termName='
                    SERVICE_PROPERTIES_SDT']/ws:State/text()</wsag:Location>
198             </wsag:Variable>
199           </wsag:VariableSet>
200         </wsag:ServiceProperties>
201         <ws:ServiceReference>
202           <wsag-ogc:ServiceReference xmlns:wsag-ogc="http://www.ifgi.org/
                namespaces/wsag/ogc">
203             <wsag-ogc:URL>http://localhost:8088/sla-proxy/DefaultWMS/246fc372
                  -5aeb-4f76-a718-775e944d1698</wsag-ogc:URL>
204           </wsag-ogc:ServiceReference>
205         </ws:ServiceReference>
206         <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RESOURCE_OPERATIONS" wsag:
              Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
              /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
207           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
208           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
209           <wsag:ServiceLevelObjective>
210             <wsag:CustomServiceLevel>
211               <wsag-ogc:CustomServiceLevel>
212                 <wsag-ogc:Name>InspireOperations</wsag-ogc:Name>
213                 <wsag-ogc:Title>INSPIRE (Operations)</wsag-ogc:Title>
214                 <wsag-ogc:Abstract>The following operations shall be
                      implemented for an INSPIRE View service: GetCapabilities,
                      GetMap.</wsag-ogc:Abstract>
215                 <wsag-ogc:Status>isGetCapabilities = false;
216                   isGetMap = false;
217                   for (item : operations.name)
218                   {
219                     if (item.equalsIgnoreCase('GetCapabilities'))
220                     {
221                       isGetCapabilities = true;
222                     }
223                     if (item.equalsIgnoreCase('GetMap'))
224                     {
225                       isGetMap = true;
226                     }
227                   }
228                   (isGetCapabilities and isGetMap);</wsag-ogc:Status>
229               </wsag-ogc:CustomServiceLevel>
230             </wsag:CustomServiceLevel>
231           </wsag:ServiceLevelObjective>
232           <wsag:BusinessValueList/>
233         </wsag:GuaranteeTerm>
234         <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_AVAILABILITY" wsag:
              Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
              /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
235           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
236           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                QualifyingCondition>
237           <wsag:ServiceLevelObjective>
```

```
238             <wsag:CustomServiceLevel>
239               <wsag-ogc:CustomServiceLevel>
240                 <wsag-ogc:Name>InspireAvailability</wsag-ogc:Name>
241                 <wsag-ogc:Title>INSPIRE (Availability)</wsag-ogc:Title>
242                 <wsag-ogc:Abstract>The probability of a Network Service to be
                        available shall be 99% of the time.</wsag-ogc:Abstract>
243                 <wsag-ogc:Status>(availability.week >= 0.99) and (availability
                        .month >= 0.99) and (availability.year >= 0.99)</wsag-ogc:
                        Status>
244               </wsag-ogc:CustomServiceLevel>
245             </wsag:CustomServiceLevel>
246           </wsag:ServiceLevelObjective>
247           <wsag:BusinessValueList/>
248         </wsag:GuaranteeTerm>
249         <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_RESPONSE" wsag:
              Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
              /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
250           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
251           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
252           <wsag:ServiceLevelObjective>
253             <wsag:CustomServiceLevel>
254               <wsag-ogc:CustomServiceLevel>
255                 <wsag-ogc:Name>InspirePerformance</wsag-ogc:Name>
256                 <wsag-ogc:Title>INSPIRE (Performance)</wsag-ogc:Title>
257                 <wsag-ogc:Abstract>The response time for sending the initial
                        response to a Get Map Request to a view service shall be
                        maximum 5 seconds in normal situation.</wsag-ogc:Abstract>
258                 <wsag-ogc:Status>fulfilled = 0;
259                   for (item : response.initial.week) {
260                     if (item lt 5000)
261                     {
262                       fulfilled = fulfilled + 1;
263                     }
264                   }
265                   percent = fulfilled / (size(response.initial.week) / 100.0);
266                   percent gt 90.0;</wsag-ogc:Status>
267               </wsag-ogc:CustomServiceLevel>
268             </wsag:CustomServiceLevel>
269           </wsag:ServiceLevelObjective>
270           <wsag:BusinessValueList/>
271         </wsag:GuaranteeTerm>
272         <wsag:GuaranteeTerm wsag:Name="GUARANTEE_RUNTIME_CAPACITY" wsag:
              Obligated="ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org
              /wsrf/bf-2" xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
              agreement" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="
              http://www.w3.org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.
              org/namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org
              /2005/08/addressing" xmlns:res-sla="http://schemas.wsag4j.org
              /2009/07/wsag4j-scheduling-extensions">
273           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
274           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
275           <wsag:ServiceLevelObjective>
276             <wsag:CustomServiceLevel>
277               <wsag-ogc:CustomServiceLevel>
278                 <wsag-ogc:Name>InspireCapacity</wsag-ogc:Name>
279                 <wsag-ogc:Title>INSPIRE (Capacity)</wsag-ogc:Title>
280                 <wsag-ogc:Abstract>The minimum number of served simultaneous
                        service requests to a view service according to the
                        performance quality of service shall be 20 per second.</
                        wsag-ogc:Abstract>
281                 <wsag-ogc:Status>fulfilled = 0;
282                   for (item : capacity.initial.week) {
283                     if (item lt 5000)
284                     {
285                       fulfilled = fulfilled + 1;
286                     }
```

```
287                 }
288                 percent = fulfilled / (size(capacity.initial.week) / 100.0);
289                 percent gt 90.0;</wsag-ogc:Status>
290               </wsag-ogc:CustomServiceLevel>
291             </wsag:CustomServiceLevel>
292           </wsag:ServiceLevelObjective>
293           <wsag:BusinessValueList/>
294         </wsag:GuaranteeTerm>
295         <wsag:GuaranteeTerm wsag:Name="COSTS_PER_YEAR" wsag:Obligated="
              ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org/wsrf/bf-2"
               xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement"
              xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="http://www.w3
              .org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.org/
              namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org/2005/08/
              addressing" xmlns:res-sla="http://schemas.wsag4j.org/2009/07/
              wsag4j-scheduling-extensions">
296           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
297           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
298           <wsag:ServiceLevelObjective/>
299           <wsag:BusinessValueList>
300             <wsag:CustomBusinessValue>
301               <wsag-ogc:CustomBusinessValue>
302                 <wsag-ogc:Name>CostsPerYear</wsag-ogc:Name>
303                 <wsag-ogc:Title>Usage Costs (Year)</wsag-ogc:Title>
304                 <wsag-ogc:Abstract>The cost to be assessed for using the
                        service on a yearly basis (in Euro).</wsag-ogc:Abstract>
305                 <wsag-ogc:Type>urn:ogc:def:sla:business:cost:year</wsag-ogc:
                        Type>
306                 <wsag-ogc:Value>factor;
307                   if (pixel.year lt (1000000 * 1000))
308                   {
309                     factor = 1.0;
310                   } else
311                   if (pixel.year lt (1000000 * 10000))
312                   {
313                     factor = 0.5;
314                   } else
315                   if (pixel.year lt (1000000 * 100000))
316                   {
317                   factor = 0.25;
318                   } else
319                   if (pixel.year lt (1000000 * 1000000))
320                   {
321                     factor = 0.125;
322                   } else
323                   {
324                     factor = 0.0625;
325                   }
326                   (factor * (pixel.year / 1000000.0));</wsag-ogc:Value>
327               </wsag-ogc:CustomBusinessValue>
328             </wsag:CustomBusinessValue>
329           </wsag:BusinessValueList>
330         </wsag:GuaranteeTerm>
331         <wsag:GuaranteeTerm wsag:Name="PENALTY_PER_YEAR" wsag:Obligated="
              ServiceProvider" xmlns:wsrf="http://docs.oasis-open.org/wsrf/bf-2"
               xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement"
              xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xs="http://www.w3
              .org/2001/XMLSchema" xmlns:wsag-ogc="http://www.ifgi.org/
              namespaces/wsag/ogc" xmlns:addressing="http://www.w3.org/2005/08/
              addressing" xmlns:res-sla="http://schemas.wsag4j.org/2009/07/
              wsag4j-scheduling-extensions">
332           <wsag:ServiceScope wsag:ServiceName="INSPIRE_VIEW_SERVICE"/>
333           <wsag:QualifyingCondition>SERVICE_PROPERTIES_STATE eq 'Ready'</wsag:
                  QualifyingCondition>
334           <wsag:ServiceLevelObjective/>
335           <wsag:BusinessValueList>
336             <wsag:CustomBusinessValue>
337               <wsag-ogc:CustomBusinessValue>
338                 <wsag-ogc:Name>PenaltyPerYear</wsag-ogc:Name>
339                 <wsag-ogc:Title>Penalty (Year)</wsag-ogc:Title>
340                 <wsag-ogc:Abstract>The penalty to be assessed for not meeting
                        service level objectives on a yearly basis (in Euro).</
```

```
                    wsag-ogc:Abstract>
341             <wsag-ogc:Type>urn:ogc:def:sla:business:penalty:year</wsag-ogc
                    :Type>
342             <wsag-ogc:Value>factor;
343               if (InspireAvailability.status == true)
344               {
345                 factor = 0;
346               }
347               else
348               {
349                 factor = 0.25;
350               }
351               (factor * CostsPerYear.value);</wsag-ogc:Value>
352           </wsag-ogc:CustomBusinessValue>
353         </wsag:CustomBusinessValue>
354       </wsag:BusinessValueList>
355     </wsag:GuaranteeTerm>
356   </ws:All>
357 </ws:Terms>
358 <ws:AgreementState>
359   <ws:State>Observed</ws:State>
360 </ws:AgreementState>
361 <ws:ServiceTermState ws:termName="SERVICE_DESCRIPTION_SDT">
362   <ws:State>Ready</ws:State>
363 </ws:ServiceTermState>
364 <ws:ServiceTermState ws:termName="SERVICE_PROPERTIES_SDT">
365   <ws:State>Ready</ws:State>
366 </ws:ServiceTermState>
367 <ws:ServiceTermState ws:termName="TIME_CONSTRAINT_SDT">
368   <ws:State>Ready</ws:State>
369 </ws:ServiceTermState>
370 <ws:GuaranteeTermState ws:termName="GUARANTEE_RESOURCE_OPERATIONS">
371   <ws:State>Fulfilled</ws:State>
372 </ws:GuaranteeTermState>
373 <ws:GuaranteeTermState ws:termName="GUARANTEE_RUNTIME_AVAILABILITY">
374   <ws:State>Fulfilled</ws:State>
375 </ws:GuaranteeTermState>
376 <ws:GuaranteeTermState ws:termName="GUARANTEE_RUNTIME_RESPONSE">
377   <ws:State>Violated</ws:State>
378 </ws:GuaranteeTermState>
379 <ws:GuaranteeTermState ws:termName="GUARANTEE_RUNTIME_CAPACITY">
380   <ws:State>Fulfilled</ws:State>
381 </ws:GuaranteeTermState>
382 <ws:GuaranteeTermState ws:termName="COSTS_PER_YEAR">
383   <ws:State>Fulfilled</ws:State>
384   <ogc:CustomBusinessValue xmlns:ogc="http://www.ifgi.org/namespaces/wsag/
        ogc">
385     <ogc:Name>CostsPerYear</ogc:Name>
386     <ogc:Type>urn:ogc:def:sla:business:cost:year</ogc:Type>
387     <ogc:Value>0.001234</ogc:Value>
388   </ogc:CustomBusinessValue>
389 </ws:GuaranteeTermState>
390 <ws:GuaranteeTermState ws:termName="PENALTY_PER_YEAR">
391   <ws:State>Fulfilled</ws:State>
392   <ogc:CustomBusinessValue xmlns:ogc="http://www.ifgi.org/namespaces/wsag/
        ogc">
393     <ogc:Name>PenaltyPerYear</ogc:Name>
394     <ogc:Type>urn:ogc:def:sla:business:penalty:year</ogc:Type>
395     <ogc:Value>0.0</ogc:Value>
396   </ogc:CustomBusinessValue>
397 </ws:GuaranteeTermState>
398 </ws:AgreementProperties>
```

## C.2 Service Interfaces

The XSDs in Listing C.12 and C.13 define the XML input and output format of the Agreement Manager component.

Listing C.12: XSD for Agreement Manager

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<xs:schema xmlns:wsag-rest="http://www.ifgi.org/namespaces/wsag/rest" xmlns:
    xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.ifgi.
    org/namespaces/wsag/rest" elementFormDefault="qualified">
  <!-- ######################################## -->
  <!-- ELEMENT DEFINITIONS                      -->
  <!-- ######################################## -->
  <xs:element name="TemplateList" type="wsag-rest:TemplateListType"/>
  <xs:element name="AgreementList" type="wsag-rest:AgreementListType"/>
  <!-- ######################################## -->
  <!-- TEMPLATE LIST TYPE                       -->
  <!-- ######################################## -->
  <xs:complexType name="TemplateListType">
    <xs:sequence>
      <xs:element name="TemplateURI" minOccurs="0" maxOccurs="unbounded"
          type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <!-- ######################################## -->
  <!-- AGREEMENT LIST TAPE                      -->
  <!-- ######################################## -->
  <xs:complexType name="AgreementListType">
    <xs:sequence>
      <xs:element name="AgreementURI" minOccurs="0" maxOccurs="unbounded"
          type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Listing C.13: XSD for Measurements

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<xs:schema xmlns:wsag-rest="http://www.ifgi.org/namespaces/wsag/rest" xmlns:
    wsag-ogc="http://www.ifgi.org/namespaces/wsag/ogc" xmlns:xs="http://www.
    w3.org/2001/XMLSchema" targetNamespace="http://www.ifgi.org/namespaces/
    wsag/rest" elementFormDefault="qualified">
  <!-- ######################################## -->
  <!-- IMPORT SCHEMA                            -->
  <!-- ######################################## -->
  <xs:import namespace="http://www.ifgi.org/namespaces/wsag/ogc"
      schemaLocation="../../../../wsag-ogc-xml/src/main/xsd/ogc-service-
      properties.xsd"/>
  <!-- ######################################## -->
  <!-- ELEMENT DEFINITIONS                      -->
  <!-- ######################################## -->
  <xs:element name="MeasurementList" type="wsag-rest:MeasurementListType"/>
  <xs:element name="MeasurementHistoryList" type="wsag-rest:
      MeasurementHistoryListType"/>
  <xs:element name="Measurement" type="wsag-rest:MeasurementType"/>
  <!-- ######################################## -->
  <!-- MEASUREMENT LIST TYPE                    -->
  <!-- ######################################## -->
  <xs:complexType name="MeasurementListType">
    <xs:sequence>
      <xs:element name="MeasurementURI" minOccurs="0" maxOccurs="unbounded"
          type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <!-- ######################################## -->
  <!-- MEASUREMENT HISTORY LIST TYPE            -->
  <!-- ######################################## -->
  <xs:complexType name="MeasurementHistoryListType">
    <xs:sequence>
      <xs:element ref="wsag-rest:Measurement" minOccurs="0" maxOccurs="
          unbounded"/>
    </xs:sequence>
  </xs:complexType>
```

```
29    <!-- ######################################## -->
30    <!-- MEASUREMENT TYPE                         -->
31    <!-- ######################################## -->
32    <xs:complexType name="MeasurementType">
33      <xs:sequence>
34        <xs:element minOccurs="1" maxOccurs="1" ref="wsag-ogc:
              ServiceProperties"/>
35      </xs:sequence>
36      <xs:attribute name="timestamp" type="xs:dateTime"/>
37    </xs:complexType>
38  </xs:schema>
```

The XSD in Listing C.13 imports the XSD for the *Non-Functional Service Description* section of the WS-Agreement Application Profile for OGC Web Services as defined in Listing C.3.

## C.3 Workflow

This section exemplifies a template discovery, agreement creation, service consumption and agreement monitoring workflow from the service consumer perspective. The workflow description is based on the default setup of the SLA4OWS framework. The default setup contains an example agreement template and offers relevant services at port 8088. The `curl` command[McElhearn, 2004] used for the workflow description is a command line tool that is part of most UNIX and Mac OS systems. The command can be used to download anything that can be referenced by an URL.

### C.3.1 Show Template

To show all available templates, use the following command.

```
1  curl -v -X GET -H"Accept: application/xml" http://localhost:8088/sla-manager
       /templates
```

The response should look like this.

```
1  * About to connect() to localhost port 8088 (#0)
2  *   Trying ::1... connected
3  > GET /sla-manager/templates HTTP/1.1
4  > User-Agent: curl/7.23.1 (x86_64-apple-darwin11.2.0) libcurl/7.23.1 OpenSSL
       /1.0.0c zlib/1.2.5 libidn/1.19
5  > Host: localhost:8088
6  > Accept: application/xml
7  >
8  < HTTP/1.1 200 OK
9  < Content-Type: application/xml
10 < Content-Length: 194
11 < Server: Jetty(6.1.26)
12 <
13 * Connection #0 to host localhost left intact
14 <rest:TemplateList xmlns:rest="http://www.ifgi.org/namespaces/wsag/rest">
15   <rest:TemplateURI>http://localhost:8088/sla-manager/template/
         WSAG_DEFAULT_TEMPLATE</rest:TemplateURI>
16 </rest:TemplateList>
17 * Closing connection #0
```

To show template details, use the following command.

```
curl -v -X GET -H"Accept: application/xml" http://localhost:8088/sla-manager
    /template/WSAG_DEFAULT_TEMPLATE
```

The response should look like this.

```
* About to connect() to localhost port 8088 (#0)
*   Trying ::1... connected
> GET /sla-manager/template/WSAG_DEFAULT_TEMPLATE HTTP/1.1
> User-Agent: curl/7.23.1 (x86_64-apple-darwin11.2.0) libcurl/7.23.1 OpenSSL
    /1.0.0c zlib/1.2.5 libidn/1.19
> Host: localhost:8088
> Accept: application/xml
>
< HTTP/1.1 200 OK
< Content-Type: application/xml
< Content-Length: 17111
< Server: Jetty(6.1.26)
<
<wsag:Template wsag:TemplateId="WSAG_DEFAULT_TEMPLATE" (...) >
  (...)
</wsag:Template>
* Closing connection #0
```

An example template can be found in Appendix C.1.2.

### C.3.2 Create Agreement

To make an agreement offer, use the following command.

```
curl -v -X POST -d @agreement-offer.xml -H"Content-Type: application/xml" -H
    "Accept: application/xml" http://localhost:8088/sla-manager/agreements
```

The response should look like this.

```
* About to connect() to localhost port 8088 (#0)
*   Trying ::1... connected
> POST /sla-manager/agreements HTTP/1.1
> User-Agent: curl/7.23.1 (x86_64-apple-darwin11.2.0) libcurl/7.23.1 OpenSSL
    /1.0.0c zlib/1.2.5 libidn/1.19
> Host: localhost:8088
> Content-Type: application/xml
> Accept: application/xml
> Content-Length: 19923
> Expect: 100-continue
>
< HTTP/1.1 100 Continue
< HTTP/1.1 201 Created
< Location: http://localhost:8088/sla-manager/agreement/e97c0b24-3fca-493e-8
    d24-6ce5f3fdbe6a
< Content-Length: 0
< Server: Jetty(6.1.26)
<
* Connection #0 to host localhost left intact
* Closing connection #0
```

An example agreement offer can be found in Appendix C.1.2.

### C.3.3 Show Agreement

To show all available agreements, use the following command.

```
curl -v -X GET -H"Accept: application/xml" http://localhost:8088/sla-manager
    /agreements
```

The response should look like this.

```
* About to connect() to localhost port 8088 (#0)
*   Trying ::1... connected
> GET /sla-manager/agreements HTTP/1.1
> User-Agent: curl/7.23.1 (x86_64-apple-darwin11.2.0) libcurl/7.23.1 OpenSSL
    /1.0.0c zlib/1.2.5 libidn/1.19
> Host: localhost:8088
> Accept: application/xml
>
< HTTP/1.1 200 OK
< Content-Type: application/xml
< Content-Length: 333
< Server: Jetty(6.1.26)
<
* Connection #0 to host localhost left intact
<rest:AgreementList xmlns:rest="http://www.ifgi.org/namespaces/wsag/rest">
  <rest:AgreementURI>http://localhost:8088/sla-manager/agreement/e97c0b24-3
      fca-493e-8d24-6ce5f3fdbe6a</rest:AgreementURI>
</rest:AgreementList>
* Closing connection #0
```

To show agreement details, use the following command.

```
curl -v -X GET -H"Accept: application/xml" http://localhost:8088/sla-manager
    /agreement/e97c0b24-3fca-493e-8d24-6ce5f3fdbe6a
```

The response should look like this.

```
* About to connect() to localhost port 8088 (#0)
*   Trying ::1... connected
> GET /sla-manager/agreement/e97c0b24-3fca-493e-8d24-6ce5f3fdbe6a HTTP/1.1
> User-Agent: curl/7.23.1 (x86_64-apple-darwin11.2.0) libcurl/7.23.1 OpenSSL
    /1.0.0c zlib/1.2.5 libidn/1.19
> Host: localhost:8088
> Accept: application/xml
>
< HTTP/1.1 200 OK
< Content-Type: application/xml
< Content-Length: 19804
< Server: Jetty(6.1.26)
<
<ws:Agreement ws:AgreementId="e97c0b24-3fca-493e-8d24-6ce5f3fdbe6a" (...) >
  (...)
</ws:Agreement>
* Closing connection #0
```

An example agreement can be found in Appendix C.1.2.

### C.3.4 Service Consumption

To execute a service through the Agreement Proxy, use the following command.

```
curl -v -X GET "http://localhost:8088/sla-proxy/DefaultWMS/e97c0b24-3fca-493
    e-8d24-6ce5f3fdbe6a?service=wms&version=1.1.0&request=GetCapabilities"
```

The response should look like this.

```
* About to connect() to localhost port 8088 (#0)
*   Trying ::1... connected
> POST /sla-manager/agreements HTTP/1.1
> User-Agent: curl/7.23.1 (x86_64-apple-darwin11.2.0) libcurl/7.23.1 OpenSSL
    /1.0.0c zlib/1.2.5 libidn/1.19
> Host: localhost:8088
> Content-Type: application/xml
> Accept: application/xml
> Content-Length: 19923
> Expect: 100-continue
>
< HTTP/1.1 100 Continue
< HTTP/1.1 201 Created
< Location: http://localhost:8088/sla-manager/agreement/e97c0b24-3fca-493e-8
    d24-6ce5f3fdbe6a
< Content-Length: 0
< Server: Jetty(6.1.26)
<
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE WMT_MS_Capabilities SYSTEM "http://localhost:8080/geoserver/
    schemas/wms/1.1.1/WMS_MS_Capabilities.dtd">
<WMT_MS_Capabilities version="1.1.1" (...) >
  (...)
</WMT_MS_Capabilities>
* Closing connection #0
```

### C.3.5 Monitor Agreement

To show the agreement state, use the following command.

```
curl -v -X GET -H"Accept: application/xml" http://localhost:8088/sla-manager
    /agreement/e97c0b24-3fca-493e-8d24-6ce5f3fdbe6a/state
```

The response should look like this.

```
* About to connect() to localhost port 8088 (#0)
*   Trying 127.0.0.1... connected
> GET /sla-manager/agreement/ff129050-7f62-482f-9947-5a2ecc12aa71/state HTTP
    /1.1
> User-Agent: curl/7.23.1 (x86_64-apple-darwin11.2.0) libcurl/7.23.1 OpenSSL
    /1.0.0c zlib/1.2.5 libidn/1.19
> Host: localhost:8088
> Accept: application/xml
>
< HTTP/1.1 200 OK
< Content-Type: application/xml
< Content-Length: 21528
< Server: Jetty(6.1.26)
<
<ws:AgreementProperties (...) >
```

```
14      (...)
15    </ws:AgreementProperties>
16    * Closing connection #0
```

An example agreement properties document can be found in Appendix C.1.2.

# Appendix D

# Implementation

This chapter provides implementation-specific information about the web-based SLA management architecture as realized by the SLA4OWS framework.

## D.1 XML Schema

This section describes the XSDs for defining the domain-specific content in the web-based SLA management architecture as realized by the SLA4OWS framework.

### D.1.1 Agreement Reporter

The XSD in Listing D.1 defines the XML output format of the Agreement Reporter component.

Listing D.1: XSD for Agreement Reporter

```
1  <?xml version="1.0" encoding="utf-8" standalone="yes"?>
2  <xs:schema targetNamespace="http://www.ifgi.org/namespaces/sla/reporter"
       xmlns:sla-reporter="http://www.ifgi.org/namespaces/sla/reporter" xmlns:
       wsag-rest="http://www.ifgi.org/namespaces/wsag/rest" xmlns:xs="http://
       www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
3    <!-- ######################################## -->
4    <!-- ELEMENT DEFINITIONS                      -->
5    <!-- ######################################## -->
6    <xs:element name="ReportList" type="sla-reporter:ReportListType"/>
7    <!-- ######################################## -->
8    <!-- REPORT LIST TYPE                         -->
9    <!-- ######################################## -->
10   <xs:complexType name="ReportListType">
11     <xs:sequence>
12       <xs:element name="ReportURI" minOccurs="0" maxOccurs="unbounded" type
             ="xs:string"/>
13     </xs:sequence>
14   </xs:complexType>
15 </xs:schema>
```

### D.1.2 Infrastructure Manager

The XSD in Listing D.2 defines the XML output format of the Infrastructure Manager component.

Listing D.2: XSD for Infrastructure Manager

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<xs:schema targetNamespace="http://www.ifgi.org/namespaces/sla/
    infrastructure" xmlns:sla-infrastructure="http://www.ifgi.org/namespaces
    /sla/infrastructure" xmlns:wsag-rest="http://www.ifgi.org/namespaces/
    wsag/rest" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
  <!-- ######################################## -->
  <!-- ELEMENT DEFINITIONS                      -->
  <!-- ######################################## -->
  <xs:element name="ScheduleList" type="sla-infrastructure:ScheduleListType
      "/>
  <!-- ######################################## -->
  <!-- SCHEDULE LIST TYPE                         -->
  <!-- ######################################## -->
  <xs:complexType name="ScheduleListType">
    <xs:sequence>
      <xs:element name="ScheduleURI" minOccurs="0" maxOccurs="unbounded"
          type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

## D.2 Service Interfaces

This section describes the service interfaces that are implemented in the SLA4OWS framework.

### D.2.1 Agreement Manager

Table D.1 describes all resources and methods of the Agreement Manager component that are supported by the SLA4OWS framework in addition to standardized interface.

Table D.1: Additional Agreement Manager Resources

| Resource [a] | Description |
|---|---|
| `/measurements/{id}` | With the `POST` method, this `URI` allows the creation of a new measurement for an agreement with the unique identifier `{id}`. The input format is defined by the `ServiceProperties` element as defined in Listing C.3. |
| `/measurement/{id}` | With the `GET` method, this URI returns a representation of the measurements for an agreement with the unique identifier `{id}`. The output format is defined by the `MeasurementList` element as defined in Listing C.3. |

[a] In case of successful or invalid requests, all resources produce HTTP Status Codes as defined in [Fielding et al., 1999] (e.g. `201 CREATED` after a new resources being created or `400 BAD REQUEST` for malformed request syntax).

### D.2.2 Agreement Client

The SLA4OWS framework provides a web-based client that enables service consumers to search for templates, and to create and monitor agreements.

### D.2.3 Agreement Proxy

Table D.2 describes all resources and methods of the Agreement Proxy component that are supported by the SLA4OWS framework in addition to standardized interface.

Table D.2: Additional Agreement Proxy Resources

| Resource [a] | Description |
|---|---|
| `/{service}/{agreement}/state/{urn}` | This URI references the passive monitoring information for the service with the unique name `{service}` and the corresponding agreement with the unique identifier `{id}`.<br><br>With the `GET` method, this URI returns the passive monitoring measurements for the service property from type `{urn}`. A comprehensive dictionary of all available service property types can be found in Appendix B.2.1. The output format is defined by the `MeasurementHistoryList` element (Listing C.13). |

[a] In case of successful or invalid requests, all resources produce HTTP Status Codes as defined in [Fielding et al., 1999] (e.g. `201 CREATED` after a new resources being created or `400 BAD REQUEST` for malformed request syntax).

### D.2.4 Agreement Monitor

The Agreement Monitor permanently runs as a background process and does not have a public accessible service interface.

### D.2.5 Agreement Evaluator

The Agreement Evaluator permanently runs as a background process and does not have a public accessible service interface.

### D.2.6 Agreement Reporter

Table D.3 describes all resources and methods of the Agreement Reporter component that are supported by the SLA4OWS framework.

Table D.3: Agreement Reporter Resources

| Resource [a] | Description |
|---|---|
| `/reports` | With the `GET` method, this URI returns a list of all available reports. The output format is defined by the `ReportList` element (Listing D.1).<br>With the `POST` method, this `URI` allows the creation of a new report. The input format is defined by the Agreement Properties format as defined in the WS-Agreement specification. An example XML document can be found in Listing C.11. |
| `/report/{id}` | With the `GET` method, this URI returns a representation of the report with the unique identifier `{id}`. The output format is defined by the Agreement Properties format as defined in the WS-Agreement specification. An example XML document can be found in Listing C.11. |

[a] In case of successful or invalid requests, all resources produce HTTP Status Codes as defined in [Fielding et al., 1999] (e.g. `201 CREATED` after a new resources beeing created or `400 BAD REQUEST` for malformed request syntax).

### D.2.7 Infrastructure Manager

Table D.4 describes all resources and methods of the Infrastructure Manager component that are supported by the SLA4OWS framework.

Table D.4: Infrastructure Manager Resources

| Resource [a] | Description |
|---|---|
| `/schedules` | With the `GET` method, this URI returns a list of all available schedules. The output format is defined by the `ScheduleList` element (Listing D.2).<br>With the `POST` method, this `URI` allows the creation of a new schedule. The input format is defined by the Agreement Properties format as defined in the WS-Agreement specification. An example XML document can be found in Listing C.11. |
| `/schedule/{id}` | With the `GET` method, this URI returns a representation of the schedule with the unique identifier `{id}`. The output format is defined by the Agreement Properties format as defined in the WS-Agreement specification. An example XML document can be found in Listing C.11. |

Table D.4 – Continued on next page

Table D.4 – Continued from previous page

| Resource [a] | Description |
|---|---|

[a] In case of successful or invalid requests, all resources produce HTTP Status Codes as defined in [Fielding et al., 1999] (e.g. `201 CREATED` after a new resources beeing created or `400 BAD REQUEST` for malformed request syntax).

# Versicherung

Hiermit versichere ich, dass ich bisher noch keinen Promotionsversuch unternommen habe.

Moers, den ..... .....

Bastian Baranski

Hiermit versichere ich, dass ich die vorgelegte Dissertation selbst und ohne unerlaubte Hilfe angefertigt, alle in Anspruch genommenen Quellen und Hilfsmittel in der Dissertation angegeben habe und die Dissertation nicht bereits anderweitig als Prüfungsarbeit vorgelegen hat.

Moers, den ..... .....

Bastian Baranski

Hiermit erkläre ich, nicht wegen einer Straftat rechtskräftig verurteilt worden zu sein, zu der ich meine wissenschaftliche Qualifikation missbraucht habe.

Moers, den ..... .....

Bastian Baranski