BACHELOR THESIS

Service Oriented Architectures in Planetary Sciences

carried out at

CAMPUS 02 GRAZ
FACHHOCHSCHULE DER WIRTSCHAFT

Course Programme
Information Technologies and IT-Marketing

By: Florian Topf
PID: 1010319048

Graz, February 8, 2012

....
Signature

Declaration of Authorship

Acknowledgements

The RTD work described in this thesis was done within the workpackage IDIS of the EU FP7 Project EuroPlaNet in close collaboration with the EU FP7 project IMPEx. The author is thankful for all contributions of Baptiste Cecconi and the "IDIS Data Model Working Group" (IDIS-DM-WG) to the described IDIS specifications and the scientific work done with all the included tools. Furthermore the development of the VEX MAG Web service for AMDA and according studies of the architecture were supported by Nicolas André, Elena Budnik, Nataliya Bourrel and Michel Gangloff from the CDPP, Toulouse. The overall process of elaborating this thesis was assisted by Dr. Maxim Khodachenko and DI Tarek Al-Ubaidi from the IWF (Institut für Weltraumforschrung/Space Research Institute) of the Austrian Academy of Sciences, in Graz. Additionally, Prof. DI Manfred Steyer gave guideance with his didactic role at the Campus 02 and as supervisor of this thesis.

Florian Topf

Graz, February 2012

Abstract

In a modern research environment the usage of advanced information technologies becomes increasingly important for both scientists and engineers in order to keep their scientific work organised. NASA[1] and ESA[2] are strongly engaged in developing standards for archiving measurement data from planetary missions since the beginning of space exploration. The increasing amount of information has shown that an implementation of *Service Oriented Architectures* is necessary to ensure long-term interconnectivity among global data centers. Modern state-of-the-art web services are able to achieve this goal by providing a common protocol for communication between different sources with their own internal structure. Several communities have formed in the recent years, which are trying to develop specifications on how data centres should describe their data and services. This work compares common *Service Oriented Architectures* and the work done by the planetary science community in particular the standards provided by the IPDA[3] and the IVOA[4]. In addition two implementation examples from the FP7 Research Infrastructure EuroPlaNet illustrate the usage of the identified standards and their advantages. Finally the future of the developed tools and their way towards global standards will be outlined.

[1]National Aeronautics and Space Administration: `http://www.nasa.gov/`
[2]European Space Agency: `http://www.esa.int/`
[3]International Planetary Data Alliance: `http://planetarydata.org/`
[4]International Virtual Observatory Alliance: `http://www.ivoa.net/`

Contents

1. Introduction

In todays research environment the Internet and its global networking abilities play a significant role especially in providing and distributing data (Topf et al., 2011b). In the field of planetary sciences the amount of measurement data is growing fast, since modern communication technologies allow us to transmit data in high bandwith from one end of the Solar System to the other. What was saved on magnetic tape thirty years ago is now stored in big datacenters around the globe. International Space Agencies like NASA have done a lot of work in digitalizing past and present data to preserve it for the future.

1.1. Present Status and Motivation

The “Planetary Data System” (PDS)[5] operated by NASA states in it's aims that their main goal is to make scientific data more accessible to the research community by maximising its usefulness (NASA, 2011). In reality the amount of data provided is nearly unmanageable for advanced search mechanisms and state-of-the-art Web service technologies. ESA is facing the same problem, altough not having huge archives like NASA in its own “Planetary Science Archive”, (PSA)[6] which also states similar aims to ensure long-term preservation of scientific data (ESA, 2011).

The “International Planetary Data Alliance” (IPDA) has identified this problem and started developing an access standard to planetary data by developing the “Planetary Data Access Protocol” Standard (PDAP). The IPDA has dedicated its first task to internationalize the PDS Archive in order to ensure sharing and distribution of data among all global leading space agencies (IPDA, 2011). Furthermore the main focus of a partner association the “International Virtual Observatory Alliance” (IVOA) is to transform international data centers like PDS and PSA into Web service oriented so called “Virtual Observatories” (VOs) in order to stay up-to-date in the fast growing service driven world of the Internet (IVOA, 2010). Both communities are well embedded in the international Planetary Community, so their objectives meet the global trend of research.

[5]PDS: `http://pds.nasa.gov/`
[6]PSA: `http:/www.rssd.esa.int/PSA/`

The following chapters will describe the basic information technologies behind the so called “Service Oriented Architecture” (SOA) paradigm. In Addition a set of specialized, community based protocol standards, data models, registries and query languages, which form “Semantic Web Services”, will be presented and compared. This most recent service paradigm allows not only to distribute data for human consumption, but also provide data which is easily machine-readable (Fensel et al., 2007, p. 5-6).

In order to be able to identify the advantages and disadvantages of those standards, two adaptions within the community around the FP7 Research Infrastructure EuroPlaNet[7] will be illustrated. In the projects prosposal (Blanc et al., 2008, p. 5) there is an indication on the main objective of EuroPlaNet, and this basically forms the motivation of this thesis:

“The European Planetary Science Community will integrate major parts of its distributed European Infrastructure to be shared, fed and expanded by all planetary scientists.”

To accomplish that, the central part of EuroPlaNet, the “Integrated and Distributed Information Service” (IDIS) is defining the standards required to enable EuroPlaNet services to work in an interoperable fashion as described in (Capria, Chanteur, & Schmidt, 2009, p. 4) and build up a VO prototype for all fields in the Planetary Science Community. In particular, the IDIS Data Model Working Group (IDIS-DM-WG) has dedicated its studies to evaluate a set of IVOA standards and the IPDA protocol PDAP (Cecconi et al., 2011, p. 1-2) whose preliminary results will be presented in this thesis. Finally this paper will describe an online-tool called Automated Multi-Dataset Analysis (AMDA)[8] which is playing a pioneer role in IDIS since it has already connected to and from different data centers with Web Services in a standardized manner.

The following sections summarize the procedure of this thesis and will define the objectives, the theoretical approach and the composition of the content.

1.2. Conceptual Formulation

Based on the present status and motivation, the author has formulated one precise research question for this paper, which will be answered with a survey on existing standards and their first prototype implementations:

Which potential standards of Service Oriented Architectures exist in the scientific community to connect observational data from planetary space missions stored in different data centers to web based analysis tools?

[7]EuroPlaNet: `http://www.europlanet-ri.eu/`

[8]Automated Multi-Dataset Analyis: `http://cdpp-amda.cesr.fr/`

1.3. Objectives of this Thesis

Derived from the concept of this thesis two major objectives were extracted:

- Emphazising the advantages of Service Oriented Architectures (SOA) with the focus on existing implementations in Planetary Sciences.
- Comparison of these specialized implementations with the SOA paradigm and common implementations to identify possible differences and critical issues.

1.4. Approach and Methodology

The scientific findings of this thesis were mainly derived from literature research in the fields of:

- Service Oriented Architectures
- Web services
- The integrated systems of EuroPlaNet RI, IDIS and AMDA
- Specifications within the IPDA and IVOA Community

This also correlates with the professional expertise of the author which will be summarized in section 1.6.

1.5. Composition of this Thesis

This thesis is composed by the following theoretical chapters:

1. Introduction to the topics covered, the approach and expected results
2. Definition of Service Oriented Architectures and Web services
3. Description of common Web service protocols (XML-RPC and SOAP together with WSDL and UDDI)

The practice-oriented chapters will include work done within the Joint Research Activity 4 "Integrated and Distributed Information Service" (IDIS) of the FP7 Research Infrastructure EuroPlaNet:

4. Development of a data model and common access protocol to interconnect different scientific data centers including concrete Web service implementations for the “Automated Multi Dataset Analysis Tool” (AMDA) operated by CDPP, Toulouse.
5. Discussion of the results, their advantages and possible disadvantages.
6. Conclusions to be drawn and outlook to forthcoming research and studies.

1.6. Work done by the Author

The author of has been working on following content by himself, as a part of dedicated working groups:

1. Development of the Europlanet-IDIS Data model, and correlating assessment studies on IPDA and IVOA standards as outlined in section 4.2 on page 29 and the evaluation of a proposed IDIS Architecture.
2. Implementation of the Venus Express MAG Web Service of the AMDA Tool, whose environment and practical implementation will be illustrated in section 4.3 on page 38.

2. Service-Oriented Architectures (SOA)

2.1. Introduction

As stated in the proposal of the Research Infrastructure EuroPlaNet and its core, the "Integrated and Distributed Information Service" (IDIS)[9], the aim is to define common Data Access Protocols for a variety of research fields within the Planetary Science Community, and the interactive tools needed to facilitate retrieval of datasets. In addition to that, interfaces must be established to ensure that data formats can be made compatible with a variety of existing platforms (Blanc et al., 2008, p. 7). According to Capria et al. (2009), IDIS will evolve into an information access system providing interoperability of a wide range of different data sources and access tools.

Up to now, there are a lot of different tools existing for the research work in the thematic fields of Planetary Science, each of them acts as a standalone product without taking advantage of other tools. This is a major drawback, since the present day research work is comprised of interdisciplinary work among different science fields, where different tools and services have to be combined to solve a more complex scientific problem. Therefore scientists and engineers pay significant attention on evolving these tools into a so called "Virtual Observatory" environment, where data centers and tools get interconnected and research work can be performed interactively over the Internet (Blanc et al., 2008, p. 41-42).

The approach of developing software in a *service oriented* manner is recognized to be the best solution for solving the problems of different tools with different architectures and standards. Erard et al. (2011d) indicates this in the proposed architecture of IDIS by illustrating, that the IDIS distributed data services will be implemented either with IVOA[10] protocols whereever relevant, or with the future IPDA[11] protocol, which follow SOA principles. Furthermore Erard et al. (2011d) states, that there is a need of defining a *data model* description layer for the data services, which will be made accessible trough *registries*. This statement already assumes the usage of a

[9]IDIS: `http://www.europlanet-idis.fi/`
[10]International Virtual Observatory Alliance: `http://www.ivoa.net/`
[11]International Planetary Data Alliance: `http://planetarydata.org/`

service-oriented infrastructure in conjunction with a "yellow pages" registry, where a user can search and find useful tools and services.

Erl (2005, sec. 3.1) indicates, that the term *service-oriented* has existed for some time and it has been used in different contexts and for different purposes. But one constant through its existence has been that it represents a distinct approach for separating concerns. So in fact, a logic, which is required to solve a large problem can be better constructed, carried out, and managed if it is decomposed in a collection of smaller, related pieces. Each of these pieces addresses a concern or a specific part of the problem. To constitute a "Service Oriented Architecture" (SOA) Erl (2005, sec. 3.1) further describes, that collectively, these units comprise a larger piece of business automation logic, and individually, these units can be distributed.

Altogether, the path for IDIS in EuroPlaNet was set from the beginning of the project, the next step was to define the exact standards. Besides of the SOA paradigm, the requirements for IDIS include the possibility to easily discover, select and invocate specific tools or services via a registry, usable for machine-to-machine interaction, which will directly led to the term of "Semantic Web Services".

To provide a more clear understanding about SOA, the following section will define the paradigm and its strong relation to the Web. Chapter 3 on page 12 will continue describing "Web Services" based on established standards. Finally chapter 4 on page 27 will give an introduction to different standards used by IDIS and its tools which form a "Semantic Web Service" environment.

2.2. Definition of SOA

The Book of Applied SOA by Rosen et al. (2008, p. 15) describes the problem of information systems and its very promising solution of developing Service Oriented Architectures:

"Over the years companies have developed a lot of business software with different standards and technologies. Due to business process improvements they started to integrate the software into one another. The results were often very fragile and unmaintainable solutions with no common interfaces between different protocols and technologies."

The Service Oriented Architecture based on Web services is aimed at simplify the approach, providing a universal connectivity to existing systems and data. Of course one has to take into account that some existing sofware APIs[12] do not fit the SOA paradigm. Old APIs have to be adapted to fit in the requirements and to be transformed into modern interfaces. Another problem which persisted in the past was

[12]Application Programming Interface (API)

data integration. A lot of resources from enterprises were put into common data model development, where the aim was to provide a singular model for all business processes. This often led to exploding costs, when the data model had to be redefined with the introduction of new software (Rosen et al., 2008, p. 16).

According to the SOA paradigm, there must be a common approach too, business processes involved do not have to agree on every single item of a data model, only the standards for the *shared data* needs to be agreed on. The architecture encourages individual units of logic to exist autonomously but not isolated from each other. Units of logic are still required to conform to a *set of principles* that allow them to evolve independently, while still maintaining a suffiencent amount of commonality and standardization. Within SOA, these units of logic are known as *services* (Erl, 2005, sec. 3.1).

A service is a discrete unit of business functionality that is made available through a *service contract*. The service contract specifies all interactions between the service consumer and service provider. According to Rosen et al. (2008, p. 50), this includes:

- Service interface
- Interface documents
- Service policies
- Service level agreement
- Performance

As one can see, the service contract plays a fundamental role in describing the manner in which services expose functionality, how data types and data models are defined, and how policies are asserted and attached. A constant focus must be drawn on ensuring that service contracts are optimized and standardized to ensure that the endpoints established by services are consistent, reliable and governable (Erl, 2007, p. 71).

Figure 2.1 illustrates the two main aspects of a service: The *service interface* at the top defines the style and details of the interactions, so that a consumer can communicate with the provided service. The *service implementation* at the bottom defines how a particular service offers its capabilities (Rosen et al., 2008, p. 50-51). This leads to two major requirements an architect should meet when implementing a SOA:

1. The consumer should see only what the service does, not how it's implemented.
2. The provider is free to change the implementation, as long that doesn't change the interface or the behavior.

What is most important at the service interface level for both provider and consumer is the information that must be passed between services to enable and complete a specific business process. It is assumed that this process is composed of several interacting services, so a fundamental question has to be raised: *How are these services related to each other?*

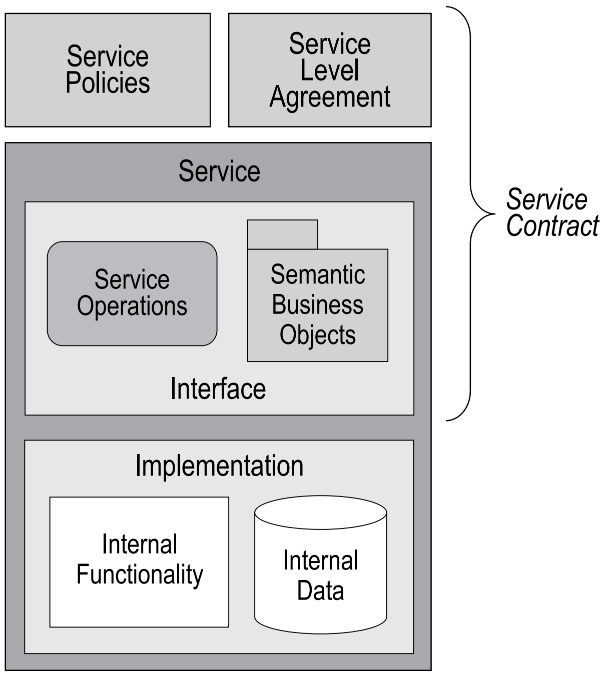

Figure 2.1.: Components of a service (Rosen et al., 2008, p. 51)

In the interaction process of two or more services it is necessary that the services are aware of each other. For this reason a Service Oriented Architecture forsees the use of *service descriptions*. Erl (2005, p. 54) describes way how services use service descriptions as a relationship classified as *loosely coupled*. The principle of "Service Loose Coupling", as defined by Erl (2007, p. 71), promotes the independent design and evolution of a service, while still guaranteeing baseline interoperability. For services to interact and accomplish a business process sucessfully, they need to exchange information. If a service is in possession of another service's description it is basicly able to communicate with the related service. As a natural conclusion to that fact, there only needs to be an agreement on a common messaging protocol.

The dynamic behaviour of messaging between services is realized with the *Request/Response* pattern, the most common interaction pattern in the public Web. According to figure 2.2, the *service client* represents a piece of software, which can issue an invocation request to a service implemented by the *service provider*. This request results in an optional response from the provider.

In order to be able to automatically configure certain aspects of the client that the consumer uses to subscribe to services, a service should be made available through a so called *registry*. As its illustrated in figure 2.3, the service provider publishes service details and service changes to a registry. The registry then notifys a subscribed consumer about change events. The consumer is finally able to reconfigure his or her client to gain more flexibility in their business processes (Governor et al., 2009, p. 121). In Addition to that, a registry makes it much more easier for the client to discover certain services needed for specific processes and tasks.

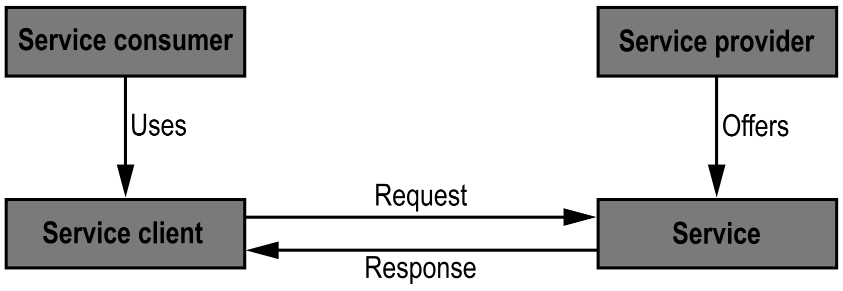

Figure 2.2.: SOA Request/Response pattern (Governor et al., 2009, p. 121)

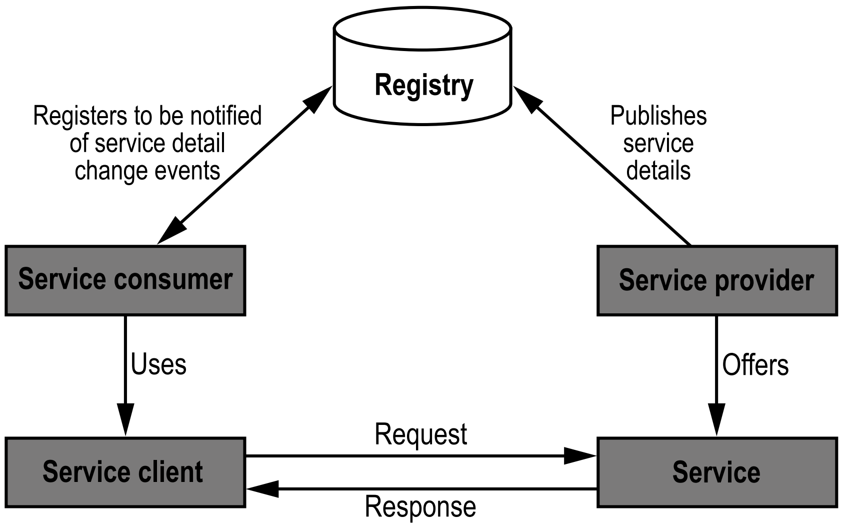

Figure 2.3.: SOA Request/Response pattern with a service registry (Governor et al., 2009, p. 122)

To emphasise the major principle of a Service Oriented Architecture the generic term *interoperability* was introduced (Erl, 2007, p. 74). The basis of this interoperability is formed by the service contracts, which harmonize different data models. The abstraction of details of a service limits all interoperation to the service contract, which increases the long-term consistency of interoperability. The reuse of particular services of a business process can be achieved in a faster way and is still preserving autonomy of each particular service. Finally, as described by Erl (2007, p. 75) the *service discoverability* simply allows services to be more easily located by those who want to potentially interoperate with them.

The next section will describe how those principles are transformed into a Web environment. It will help to understand the architectural concept behind common web-based SOA technologies which will then be mentioned in chapter 3.

2.3. SOA and Web Services

As discussed in section 2.2, SOA represents an abstract architectural concept. Its approach consists of loosely coupling components (services) within software systems that have been described in a uniform way and that can be discovered and composed (Weerawarana et al., 2005, chap. 3). Web services in that respect, represent one important approach to realizing SOA. The World Wide Web Consortium (W3C)[13] defines a Web Service as:

"... a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards." (Hass & Brown, 2004)

Therefore, the primitive requirements for a Service Oriented Architecture are fullfilled by a given set of Web technologies including the HTTP Protocol[14]. Erl (2005, sec. 5.2) provides an overview of the most basic Web service design concepts. In particular, every Web service can be associated with:

1. **service roles** – a temporary classification it assumes during the runtime processing of a message
2. **service models** – a permanent classification based on the application logic it provides.

A Service role defines in which context a Web service is used. As already mentioned, a service can act as the initiator, relayer, or the recipient of a message. The Web service is able to change its role more than once within a given business process. The service provider takes its role when another service is invoking a request published through a service description wheras the service requester role is applied to any processing logic issueing a request message (Erl, 2005, sec. 5.2).

Depending on the application logic being provided by Web services and their overall role in a business process they are classified as *service models*. The business service model represents the most fundamental building block. It is fully autonomous and encapsulates a set of business logic. Utility service models are assigned to generic Web services which can be potentially reused. Finally a controller service model describes compositions of independent services that each contribute to the execution of a process (Erl, 2005, sec. 5.2).

In order to be able to interact with other Web Services and all kinds of possible consumers, each Web service needs a service description, which is the main ingredient

[13]The World Wide Web Consortium: `http://www.w3.org/`
[14]Hypertext Transfer Protocol (HTTP) at W3C: `http://www.w3.org/Protocols/`

for communication. This description is most commonly based on the *Web Service Description Language* (WSDL), published by every service provider and where a service requester may subscribe to (Weerawarana et al., 2005, sec. 3.4). The technological aspects of WSDL and the used XML format will be described in section 3.5.

In order to achieve service messaging as described with the *Request/Response* pattern, an interoperable messaging language must be defined too. The W3C explains in Booth et al. (2004) that XML will also be able to solve this key technology requirement, because it's offering a standardized, flexible and inherently extensionable data format. Furthermore, to keep *interoperability* at the highest level, a Web service which is using a standardized XML messaging system, remains not tied to any one operating system or programming language (Cerami, 2002, p. 16). Chapter 3 will illustrate several alternatives for XML messaging between internet-based services including the *Simple Object Access Protocol* (SOAP) mentioned in Booth et al. (2004).

Finally, to solve the problem on how Web Services can be discovered by potential consumers, the W3C introduced several ways on how a discovery mechanism should be conceived (Booth et al., 2004). This work will focus on one solution, already introduced as a principle SOA design pattern in section 2.2 as a *registry* where a service provider may publish its services. To establish this part of a Web services framework Erl (2004, p. 80) notes, that a central directory to host service descriptions is required. This goal is achieved by *Universal Description, Discovery and Integration* (UDDI), an interface based on XML currently representing the discovery layer within the SOA paradigm. Section 3.6 will describe how such an interface can be realized.

The next chapter will help to understand the architectural relation between these basic design principles of Web services by sequentially going trough each characteristic, defining all W3C standards and specifications.

3. Common SOA Technologies

3.1. Introduction to XML

The previous chapter already indicated, that the focus of this work are Service Oriented Architectures with special emphasis on "Web services". To get a better understanding on how each key element of a Web service is working together in an overall architecture, the foundation layer of SOA in the Web, the *eXtensible Markup Language (XML)* will be introduced, including all the necessary XML mechanims frequently used in Web Services.

XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language[15]. It describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. By construction, XML documents are conforming SGML documents (Maler et al., 2006). In a nutshell, XML is not a language, but a metadata language for defining new languages. The definition of XML is platform independent and has rapidly become the de facto format for data interchange between disparate entities (Weerawarana et al., 2005, sec 2.1).

According to Erl (2004, p. 20) XML is implemented using a set of *elements*, which are making up the majority of an XML document. There is only one top-level element in each XML file, which is referred to the *document element* (Skonnard & Gudgin, 2001, p. 1). Each element can have *attributes* and children, which can be again elements. Attributes are used to associate name-value pairs with elements and they are referred to as the specifications of the element (Maler et al., 2006). They can be used to encode data or to provide metadata about an element – that is, provide extra information about the content of the element on which they appear (Skonnard & Gudgin, 2001, p. 5). A set of related XML elements can be classified as a *vocabulary*. Because XML allows designers to chose their own tagnames, elements can have the same local name, but are in fact from different vocabularies. In order to be able to identifiy the right vocabulary Layman, Hollander, and Bray (2009) is introducing XML *namespaces* which are associated with an element by an additional attribute defining its name. A namespace name is a *Uniform Resource Identifier* (URI) and together with the local name of an element it forms a globally unique name known as a *qualified name*. According

[15]Standard Generalized Markup Language (SGML): `http://www.w3.org/MarkUp/SGML/`

to Jacobs and Walsh (2004) URIs are a cornerstone of the Web architecture, providing and identification mechanism for resources that is common across the Web. The most commonly used URI is also referred to as Uniform Resource Link (URL) e.g. `http://www.w3.org/`.

The example in listing 3.1 was taken from Newcomer (2002, p. 20). It illustrates how an XML document can look like. The first line represents an *XML declaration* where the used XML version and the file encoding is indicated. As one can see, the element `<Company>` is associated with a namespace URI attribute `xmlns="urn:example-org:Company"` thus making the document element names globally unique. Furthermore the `<CompanyName>` element has an extra attribute `region` which associates the element here with a specific geographical region. The literal text in between elements is the actual information one would like to express with this document.

Listing 3.1: Basic example of an XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<Company xmlns="urn:example-org:Company">
  <CompanyName region="US">
    Skateboots Manufacturing
  </CompanyName>
  <address>
    <line> 200 High Street </line>
    <line> Springfield, MA 55555 </line>
    <Country> USA </Country>
  </address>
  <phone> +1 781 555 5000 </phone>
</Company>
```

According to this example, one can see the purpose of XML. It was developed to overcome the limitations of HTML, especially to better support dynamic content creating. HTML is suitable for defining and maintaining static content, but as the Web evolves towards a software-enabled platform, in which data has an associated meaning, the content needs to be generated and digested dynamically. Using XML, one can achieve this goal, because an XML *element* associates meaning with data (Newcomer, 2002, p. 20).

Weerawarana et al. (2005, sec. 2.1.1) indicates, that one can define a new language with XML, by simply deciding on a set of element names, their valid content, and the kinds of literal text that are permissible as attribute values and element content. As a logic conclusion, the only requirement by the W3C on an XML document is, that it is *well-formed*, which means, that it matches the production label document, and each of the parsed entities is well-formed, by means of syntax (Maler et al., 2006). For that reason a mechanism must be used to describe a set of elements (a *vocabulary*). As indicated in Erl (2004, p. 21-22) this is done by a *schema definition language*, which is defining the structure of XML documents. XML schemas protect the integrity of XML document data by providing structure, validation rules, type constains, and inter-element relationships. In other words XML schemas dictate what can and cannot be done with XML data. There are different XML schema languages existing, the two most common are the "Document Type Definitions (DTD)" and the XML Schema Definition Language (XSD)" (Walmsley & Fallside, 2004). An example on how an XML Schema is structured will be given later in chapter 4 on page 27, where some real-world examples are introduced.

3.2. The Web Service Architecture Stack

In a nutshell one can assume, that if two parties are exchanging XML data and are also sharing the same definitions (a *schema*), they can be sure to understand the meaning of the same element tags in the same way (Newcomer, 2002, p. 21). This is exactly how Web services work and hence XML features are reflected in figure 3.1 as "Base Technologies". To finally understand the context of XML in the overall Web service architecture and to close the circle of the SOA principles behind Web services the whole "Web Service Architecture Stack" is introduced in figure 3.1.

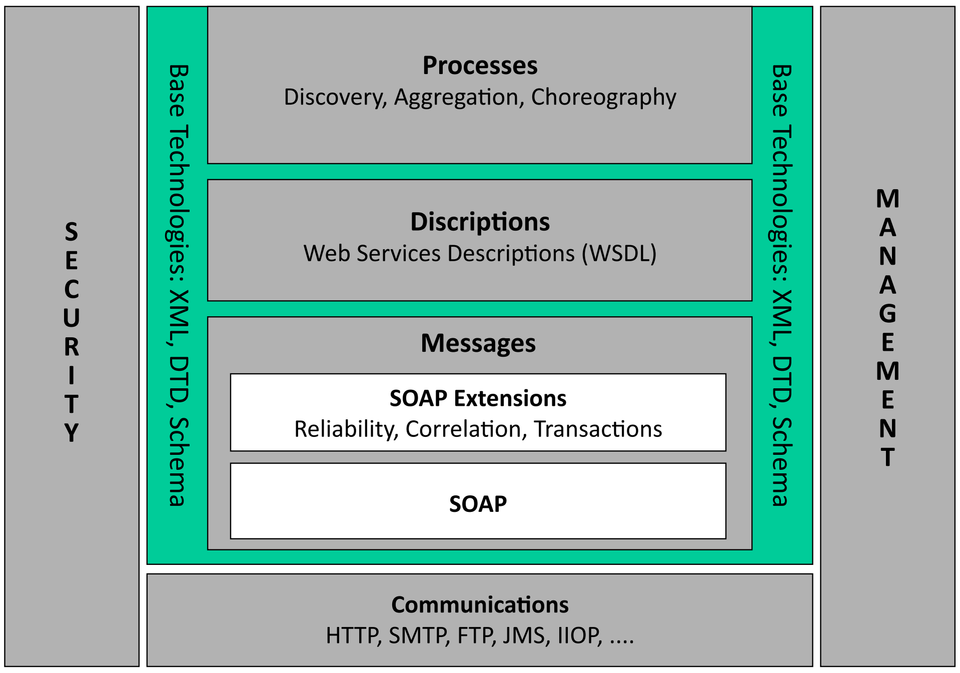
SECURITY
Base Technologies: XML, DTD, Schema
Processes
Discovery, Aggregation, Choreography
Discriptions
Web Services Descriptions (WSDL)
Messages
SOAP Extensions
Reliability, Correlation, Transactions
SOAP
Base Technologies: XML, DTD, Schema
MANAGEMENT
Communications
HTTP, SMTP, FTP, JMS, IIOP, ...

Figure 3.1.: Web Services Architecture Stack (Booth et al., 2004)

The "Web Service Architecture Stack" is still evolving, but currently it has four main layers. Below the description of each layer is provided:

Communications:

Often referred to the *Service transport* layer, this block is responsible for transporting messages between applications over the Internet (Cerami, 2002, p. 11). As explained in Studer, Grimm, and Abecker (2007, p. 25) these technologies are located in the "Application Layer" of the ISO/OSI Protocol Reference Model. Typical examples would be the Hypertext Transfer Protocol (HTTP) the Simple Mail Transfer Protocol (SMTP), the File Transfer Protocol (FTP), and newer protocols, such as the Blocks Extensible Exchange Protocol (BEEP). Studer et al. (2007, p. 26) also clarifies, that each of these protocols provide specific benefits and drawbacks and the optimal choice heavily depends on the specific use case of a Web service.

Messages:

The *Messaging Layer* of the "Web Services Architecture Stack" contains the most fundamental Web services specifications and technologies (Weerawarana et al., 2005, sec. 3.3). Here the XML plays a major role, because this framework provides a standardized format to describe message content between services. Since so much emphasis is placed on message-centric application design within SOAs, the receipt of a message by a service is also the most fundamental action within the Web Service Architecure (Erl, 2005, sec. 5.4). For this purpose the *Simple Object Access Protocol (SOAP)* was introduced by the W3C as "a standard, extensible, composable framework for packaging and exchanging XML messages" (Booth et al., 2004). SOAP defines an XML-based messaging framework: a processing model and an extensibility model. SOAP messages can be carried by a variety of protocols, as listed above or with proprietary messaging protocols (Mitra & Lafon, 2007). Furthermore the SOAP specification contains conventions for adapting its one-way messaging to the *Request/Response* pattern (Newcomer, 2002, p. 25). This protocol and its extensions are studied in detail in section 3.4.

In order to complete the description of XML messaging, it is worth to mention the *XML Remote Procedure Calls (XML-RPC)* which is the easiest way to start web services, since its simpler than SOAP and easier to adopt. Unlike SOAP, the requests are encoded in XML and sent via HTTP-POST. The responses are embedded in the HTTP response (Cerami, 2002, p. 15). XML-RPC has no services description layer like SOAP per se, but Sun Microsystems[16] developed a proprietary grammar for describing HTTP applications in Java[17], which can be used in this context, the so called Web Application Description Language (WADL)[18] (Hadley, 2009).

Descriptions:

The *Service Description Layer* is responsible for describing the public interface to a specific Web service. In a nutshell, the most widely used solution is the *Web Service Description Language (WSDL)*. It provides the service description layer within the

[16]Sun Microsystems are now part of Oracle see: `http://www.oracle.com/`
[17]Java at `http://java.com/`
[18]Web Application Description Language (WADL) at `http://wadl.java.net/`

Web Service Architecture Stack by defining an XML grammar for public interfaces (Cerami, 2002, p. 17). This metadata is important, and it is fundamental in order to achieve the *loose coupling* that is associated with SOA (Weerawarana et al., 2005). However, (Studer et al., 2007) emphasises that WSDL does cover only the technical but not the semantical description of a service. The main ingredients of an WSDL file and an example are shown in section 3.5.

Processes:

In order to make it possible for machines to discover and select a service they need, to accomplish a business process, they aquire additional functional descriptions to agree on the semantics of the interaction (Booth et al., 2004). The following three process scenarios shown in figure 3.1 are providing such mechanisms:

1. **Discovery:** This process is the most important in the field of Web services. The *Universal Description, Discovery and Integration (UDDI)* is providing the main solution for locating a Web Service that fits the needs of a potential consumer. UDDI, as discovered in section 2.3 is implementing the *registry* pattern by providing a *publish/subcribe* scheme. It is building the fundamental basis of other processes since it is describing the interfaces to Web services (Weerawarana et al., 2005, sec. 3.5). Different practical approaches will be presented at a later stage in this work.
2. **Aggregation:** Since Web services are used in a *loosely-coupled* manner according to SOA rules, the aggregation of services to form composite business processes and maximize reuse is a key requirement. There are different approaches existing such as *unconstrained* and *constrained* aggregation. Unconstrained aggregation can be realised by grouping, which is achieved by simply building a collection of related services. WSDL is able to inherit interface descriptions among a collection of Web services to form such a collection (Khalaf & Leymann, 2003, sec. 3.1).
3. **Choreography:** Constrained aggregation in contrast is not driven by the services themselves. In more complex cases of business functionality it is necessary to provide an aggregation mechanism which is conducted externally. Choreography provides this approach by introducing workflow and business modeling schemes. (Khalaf & Leymann, 2003, sec. 3.2).

Finally, two support layers are introduced to harmonize the overall Web services Architecture. Theses mechanisms are jointly used by the four main layers:

Management:

Booth et al. (2004) describes a set of management capabilities that enable monitoring and reporting of, service qualities and service usage. Such service qualities include health qualities, for example availability, performance and accessibility measures. It is naturally considered as part of the "Service contract" between a service requester and a service provider in terms of SOA. The authorities of that contract have to decide on which capabilities must be made available in order to meet their requirements.

Security:

Service-oriented applications need to be capable to handle many of the traditional security demands, of protecting information and ensuring that access logic is only granted to those equipped with sufficient access rights (Cerami, 2002, p. 21). As a starting point the HTTP over SSL protocol (HTTPS) is mentioned to provide confidentdiality and integrity. In addition WS-Security is a SOAP extension that enables identification, authentication and authorization in a *claims* based manner. The access of secured service providers is managed with *tokens* (Newcomer, 2002, p. 158). On top of all Web service security mechanisms, WS-Trust defines an extensible model for setting up and verifying trust relationships. The key concept is a *Security Token Provider (STS)* which is a distinguished Web service, that issues exhanges and validates security tokens. It allows Web services to set up and agree on which security severs they trust (Weerawarana et al., 2005, sec. 3.6.1). The underlying XML security technologies are completely described in the according W3C standard specifications.[19]

The following sections give a detailed insight to common implementations of those Web service layers, which are associated with the XML language from Messaging over Descriptions to Processes.

3.3. XML Remote Procedure Calls (XML-RPC)

XML-RPC was developed by Dave Winer and others at UserLand[20], an Internet publishing company looking for a more efficient way of transferring XML documents across the Web and developing interactive Web sites (Newcomer, 2002, p. 174). According to its definition XML-RPC provides an XML- and HTTP-based mechansim for making method or function calls across the network. It is in particular useful for connecting disparate systems and for publishing machine-readable information (Cerami, 2002, p. 26). A very basic implementation of the *Request/Response* pattern within XML-RPC provides the possibility to send XML messages in the body of an HTTP-POST request. The HTTP-POST request is specified by the IETF[21] as a uniform HTTP method to provide a block of data, such as a result of submitting a form to a data-handling process. The actual function performed by the POST method is determined by the server and is usually dependent on the Request-URI (*Unique Resource Identifier*), which identifies the requested resource. The action performed by the POST method might not result in a resource that can be identified by an URI. In this case

[19]XML Security at W3C: `http://www.w3.org/standards/xml/security`
[20]UserLand Software: `http://www.userland.com/`
[21]Internet Engineering Task Force: `http://www.ietf.org/`

either 200 (OK) or 204 (No Content)[22] is the appropriate HTTP response depending on whether or not the response includes an entitiy that describes the result (Fielding et al., 1999). In case of an XML-RPC request sent over HTTP-POST a resulting HTTP response by the service provider may contain an XML document with results. Winer (1999) provides a simple example of an XML-RPC HTTP-POST request as seen in listing 3.2:

Listing 3.2: Basic example of an XML-RPC request

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
      <value><i4>41</i4></value>
    </param>
  </params>
</methodCall>
```

The first block represents the header of an HTTP-POST request. It includes the URI, in this case `/RPC2`, the User-Agent of the host, the content type and the content length. The second block is representing the actual message of an RPC in XML where a single `<methodCall>` element wraps all the necessary paramaters. According to Cerami (2002, p. 32) this data model must contain a `<methodName>` element, containing the method to be called and a set of elements in `<params>`. A single `<param>` element has a `<value>` which can be from different types. These types include integers (in this axample `<i4>`, a four-byte signed integer), strings (`<string>`), an iso date/time (`<dataTime.iso8601`) and a encoded binary (`<base64>`)[23]. There is also the possibility to define simple structures and arrays (Winer, 1999). It is important to note, that XML-RPC does not define how to map the `<methodName>` to an actual method or procedure; that's up to the implementation (Newcomer, 2002, p. 175).

The example in listing 3.3 represents the XML-RPC response to the previous shown XML-RPC request. As one can see in the header block, the request responded with 200 (OK), so the procedure of the request was found, executed correctly, and returned results (Cerami, 2002, p. 33). The actual result is located in the content block where a

[22]HTTP Status Code Definitions: `http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`

[23]Definition of Base64 Data Encoding: `http://tools.ietf.org/html/rfc4648`

single `<methodResponse>` represents it. The enclosed XML elements are the same as in the request (Laurent, Johnston, & Dumbill, 2001, p. 28).

Listing 3.3: Basic example of an XML-RPC response

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><string>South Dakota</string></value>
    </param>
  </params>
</methodResponse>
```

It is important to note, that a case could arise sometimes, where the remote method is not returning any response value. (like a `void` function). XML-RPC mandates, that excatly one paramater must be returned from a method invocation, like e.g. `true` in the message body (Laurent et al., 2001, p. 29). If there is an application level error, for example, a value of the request has the wrong type, there is usually a response without the `<params>` element, but with a `<fault>` element, representing an error message. Protocol level errors usually respond with a HTTP response code like 400 (Bad request) or 500 (Internal Server Error) (Newcomer, 2002, p. 176).

The following section will introduce the successor of XML-RPC, the *Simple Object Access Protocol (SOAP)*, which was finally integrated in the Web Service Architecture Stack of W3C as official messaging protocol.

3.4. Simple Object Access Protocol (SOAP)

As already indicated in section 2.3 the *Simple Object Access Protocol (SOAP)* is the most common messaging protocol between internet-based services. The W3C defines SOAP in Gudgin et al. (2007) as follows:

"SOAP Version 1.2 is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible

messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independet of any particular programming model and other implementation specific semantics."

In other words, with SOAP, one can access Web services through a *loosely coupled* infrastructure, following the SOA paradigm, that provides significant resilence, scalability and flexibility in deployment. Weerawarana et al. (2005, chap. 4) emphasizes the four main capabilities provided by SOAP:

- A standardized message structure based on the XML specifications by W3C;
- A processing model, that describes how a service should process the messages;
- A mechanism to bind SOAP message to different network transport protocols;
- A way to attach non-XML encoded information to SOAP messages.

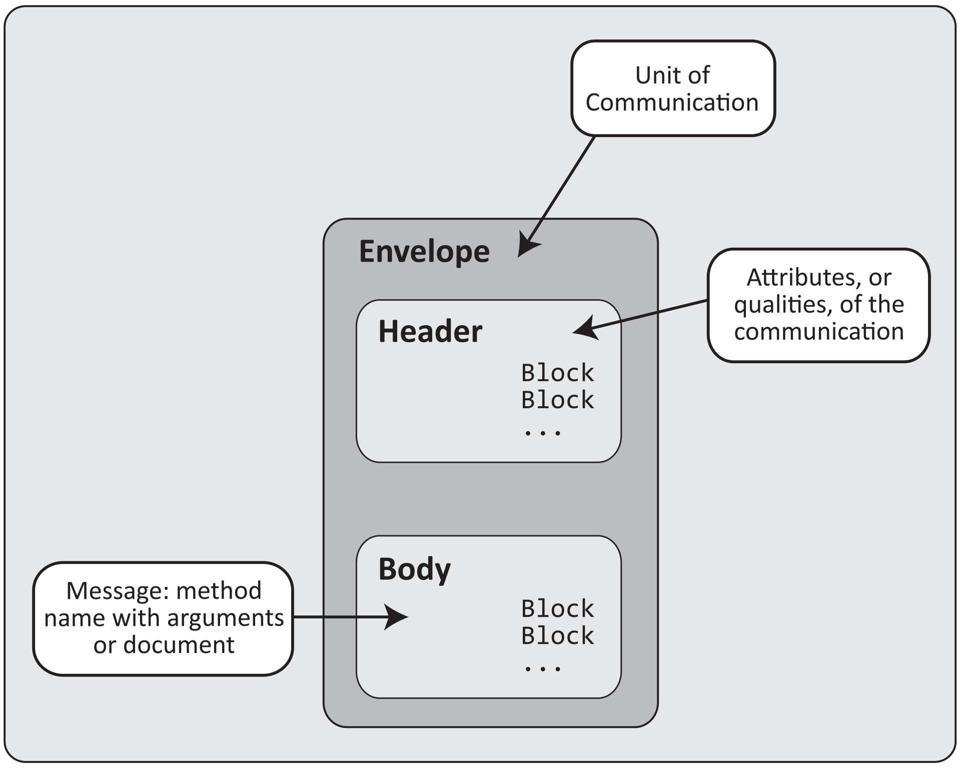
Unit of Communication
Envelope
Attributes, or qualities, of the communication
Header
Block
Block
...
Body
Message: method name with arguments or document
Block
Block
...

Figure 3.2.: SOAP message parts: envelope, header, and body (Newcomer, 2002, p. 83)

In order to get a clear view how the SOAP protocol is working, the structure of a SOAP message as seen in figure 3.2 is outlined below (Tidwell, Snell, & Kulchenko, 2001, p. 17). a SOAP message is covered by an *envelope*, which marks the start and the end of the SOAP message, containing an optional header and a required body. The header contains blocks of information relevant to how the message is processed. This includes routing and deliver settings, authentication or authorization assertions, and transaction contexts also known as *qualities of service*. The body contains the actual message to be delivered and processed. Anything that can be expressed in XML syntax can go in the body of a message (Tidwell et al., 2001, p. 17). Usually the body follows a specific XML format, so that a conforming SOAP protocol implementaion understands how to inteprete such a document and how to map the data to an underlying software implementation of the service.

This work will focus on RPC-style messages of SOAP, where messages usually come in pairs comprising of a request sent by a SOAP client and an optional response sent by the SOAP server thus implementing the *Request/Response* pattern in a similar way as XML-RPC described in section 3.3 on page 17. In order to understand the role of XML and its capabilities in the context of SOAP, one must introduce the *XML namespace* a SOAP envelope is based on as specified by the W3C: `http://www.w3.org/2003/05/soap-envelope` (Gudgin et al., 2007).

According to Skonnard and Gudgin (2001, sec. 10.2) this namespace defines four *elements*, the `envelope` as top-level element, as well as an optional `header` element and a mandatory `body` element as child elements. In Addition to that, a SOAP response may contain a `fault` element in its `body` element. The following two examples in listing 3.4 and listing 3.5 on the following page taken from Tidwell et al. (2001, p. 19) show an implementation of the function `public Float getQuote(String symbol)`, which returns a stock's price, as a SOAP service:

Listing 3.4: RPC-style SOAP request

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header>
    <m:transaction xmlns:m="soap-transaction"
          s:mustUnderstand="true">
      <transactionID>1234</transactionID>
    </m:transaction>
  </s:Header>
  <s:Body>
    <n:getQuoteRequest xmlns:n="urn:QuoteService"
      s:encodingStyle=
        "http://www.w3.org/2003/05/soap-encoding">
      <symbol xsi:type="xsd:string">
        IBM
      </symbol>
    </n:getQuoteRequest>
  </s:Body>
</s:Envelope>
```

As one can see this SOAP envelope is in the namespace specified by W3C. The given SOAP header is containing a single element `<m:transaction>` with a `<transactionID>` identifing the message, in the namespace `soap-transaction`.

In section 3.1 on page 12 it was already discovered, that such a namespace declaration makes this elements and all child elements globally unique, thus forming a *well-formed* XML content. The element `<n:getQuoteRequest>` in the SOAP body represents the remote method to be called, with an additional child element `symbol` containing a string. Note that the element standing for the remote method is again in a specific namespace.

Of course the *XML Schema* coming with the XML namespace illustrated above, also defines a set of *attributes* each element can have. In this case the `s:mustUnderstand` and `s:encodingStyle` are such attributes. The W3C explains `mustUnderstand` as a flag used to indicate whether the processing of a SOAP header block is mandatory (true) or optional (false) (Gudgin et al., 2007). Furthermore the `encodingStyle` attribute, which is again related to a W3C namespace, is a set of rules that define exactly how native application and platform data types are to be encoded in common XML syntax.

In listing 3.5 one can see a possible SOAP response to the previous SOAP request:

Listing 3.5: RPC-style SOAP response

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Body>
    <n:getQuoteRespone xmlns:n="urn:QuoteService"
      s:encodingStyle=
      "http://www.w3.org/2003/05/soap-encoding">
      <value xsi:type="xsd:float">
        98.06
      </value>
    </n:getQuoteResponse>
  </s:Body>
</s:Envelope>
```

Like in XML-RPC, a SOAP response looks very similar to it's counterpart. This time no header is sent, but a single element `<n:getQuoteResponse>` in the body block is representing the response by providing a return value also following the same `encodingStyle` rules as in the request.

In the case if there was an error in processing a request, the response would include a single `<s:fault>` element with a predefined set of child elements, describing the problem in the body of the SOAP message. A detailed description of those elements can be found in (Cerami, 2002, p. 48).

Finally Tidwell et al. (2001, p. 25) introduces a model for message exchange, in order to be able to use the fundamental one-way transmission of an envelope from a provider to a requestor to build up a complex business process. In other words, the message may pass through various processors that each in turn do something with the message. The W3C defines such a model as a so called "SOAP extension" which can be designed to ensure that SOAP header blocks are processed in an appropriate order along a message path (Gudgin et al., 2007). The message path refers to the route taken by a message from when it is first sent until it arrives at its final destination (Erl, 2005, sec. 5.4.3).

A *SOAP intermediary* for example, is a specific Web service which is designed to sit between a service requestor and a service provider and add value or functionality to the transaction between the two. It is resonsible for relaying contents of a message to another intermediary or the final receiver. In most cases the intermediary will process and alter the header information relating to the forwarding logic (Erl, 2005, sec. 5.4.2). For that reason another important *attribute* to describe a forwarding mechanism within the SOAP header is introduced by (Gudgin et al., 2007). The `role` attribute is specifying an *Uniform Resource Identifier* (URI) identifying the intermediary for which the the annotated "SOAP extension" element is intended. The following listing 3.6 shows an example of how such a SOAP header could look like:

Listing 3.6: The actor, a SOAP header attribute

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header>
    <x:signature s:role=
      "http://www.w3.org/2003/05/soap-envelope/role/next">
      <!-- extension detail goes here -->
    </x:signature>
  </s:Header>
  <s:Body>
    <!-- message content goes here -->
  </s:Body>
</s:Envelope>
```

In this particular case the intended intermediary is specified as the first one in the chain and therefore the `role` attribute must be associated with the XML namespace `http://www.w3.org/2003/05/soap-envelope/role/next` (Gudgin et al., 2007). As already defined in the Web Service Architecture Stack a variety of protocols can be used as transport mechanism for SOAP styled messages. This thesis will focus on HTTP based SOAP services, since HTTP is a natural match for the *Request/Response* pattern used with SOAP's RPC conventions.

3.5. Web Service Description Language (WSDL)

The Web Service Description Language (WSDL) as part of the Web Service Architecture Stack is a specification how to describe public interfaces of web services in a common XML vocabulary (or grammar). WSDL describes four critical pieces of data according to Cerami (2002, p. 103):

- Interface information describing all publicly available functions
- Data type information for all message requests and message responses
- Binding information about the transport protocol to be used
- Address information for locating the specified service

In a nutshell, WSDL represents a *service contract* between the service requester and the service provider. It is used for defining how the Web service is accessed, the operations it peforms, how messages are passed and the structure of the message (Tidwell et al., 2001, p. 79). WSDL also meets the requirement that a developer has to provide information about how all procedures are supposed to happen. A client can gather information about what the Web service does, how it is done and how he can use it, but she/he do not need to know any details about the API behind the provided Web service (Richards, 2006).

Basically a WSDL document as explained in Weerawarana et al. (2005) consists of two parts:

- **a reusable abstract part** – describing the operational behaviour of a Web service by recounting the messages that go in and out from services
- **a concrete part** – describing how and where the Web service implementation can be accessed.

Before introducing the XML elements which are building up these two parts of an WSDL document, one has to mention that the *XML Schema Definition (XSD)* language plays a fundamental role in the Web services architecture. The main reason for that is, that XSD can formally define the hierarchical structure of XML documents. Furthermore the structure established wihtin an XSD Schema contains a series of rules and constraints to which XML documents using this Schema have to comply in order to be valid (Erl, 2005). In case of WSDL, Newcomer (2002) mentions, that an XSD Schema is providing a set of non-proprietary data types used to represent information in the WSDL document.

The listings A.1 and A.2 in appendix A represent a complete WSDL document, with it's *abstract* and *concrete* definition parts. As one can see, in each part there are a variety of XML elements describing the whole functionality of a Web service. Note, that this example is using the WSDL 1.1 standard which was published as a W3C Note in 2001 (Christensen et al., 2001). The successor WSDL 2.0 already is a recommendation and some essential details have changed but the scope of describing services remains. Further details can be found in (Chinnici et al., 2007).

The document element `<definitions>`, is defining the start and the end of a WSDL document. It contains a series of attributes in which the service is assigned the name "QuoteService" and in which a number of namespaces are declared. The default namespace in this example is `http://www.w3.org/ns/wsdl`. All elements or attributes assigned to `xmlns:xsd` will follow the *XML Schema* definitions (Walmsley & Fallside, 2004). Alternatively a Web service provider may define it's own schema with type definitions using a single element `<schema>` in `<types>`. For every message a service is designed to receive and transmit, a `message` element must be added. In this case `getQuoteRequest` and `getQuoteResponse`. These elements may contain one or more `part` elements as respective values of the message. This example uses types of the standard XML Schema provided by W3C. Finally the service operation is defined within `portType` referring to the messages previously declared as input and output message.

These four elements of a WSDL document represent the *abstract* part, which makes the service interface complete, wihout referencing to the messaging communication technology (Erl, 2005, sec. 13.3.6). In the *concrete* part of a WSDL document, the `binding` element is defining the implementation of the *Request/Response* pattern (Tidwell et al., 2001, p. 81). In this case, the messaging protocol is defined as SOAP using HTTP as the transport protocol. The actual `operation` style is defined as "rpc" so the SOAP messages have to contain an XML document, with the root element `getQuote` in its body. The `input` and `output` elements of the operation are to be SOAP encoded. The final element of the WSDL document is `<service>` which indicates in this example where the "StockQuote" Web service can be accessed. The service can be invoked by using SOAP messages at the given physical address defined within the `<port>` element relying to the SOAP binding.

After defining the SOAP messages to be transferred and describing the service with WSDL that will receive and process a message, this work continues with the *Discovery process* in the topmost layer of the Web Service Architecture Stack.

3.6. Universal Description, Discovery and Integration (UDDI)

In order to be able to reuse services in the process of *service composition*, developers need to know which existing services are available and how they can be accessed. To fulfill this requirement from the SOA paradigm *Universal Description, Discovery* *and Integration (UDDI)* was specified as a group of web-based registries that expose information about a business process and its technical interfaces (or APIs) (Bellwood et al., 2002). The UDDI framework therefore defines a data model in XML and SOAP APIs for publishing and discovering Web services and related business information in a distributed directory (or *registry*).

According to Cerami (2002, p. 135f) the data captured within UDDI is divided into three main categories:

- **White pages** – including general information about a service provider e.g. a name, address and contact information.
- **Yellow pages** – comprised of categorized information for searching, such as the type and general service functionality
- **Green pages** – containing technical information about a web service including pointer to external specification and addresses for invoking Web services.

Newcomer (2002, p. 27) desribes a scenario of UDDI where Web service providers use SOAP to register themselves in a *registry*. Clients use the query APIs to search registered information and to discover a suitable Web service for their business processes. In a nutshell, the service provider registers a WSDL document using the UDDI API along with the other necessary information. A service consumer may then query the registry via SOAP to obtain the required WSDL description of a service. The service consumer can finally send the appropriate message to the specified operation of a service. As usually the provider and consumer need to agree on a transport protocol, which is naturally HTTP in this case.

This work will cover a different implementation of a *Registry* pattern in terms of SOA which provides specific characteristics in the field of space sciences but nevertheless is following the principles of UDDI (see section 4.2 on page 29).

4. Integrated Service-Oriented Systems

The following sections will introduce two practical approaches of a service-oriented environment within the European Planetary Science Community. The FP7 project EuroPlaNet has founded a part of the development costs of those implementations in order to foster integration of a major distributed European infrastructure. The scientific community was used to develop their own particular data exploitation and analysis tools over the last four decades, often duplicating their work. In order to compete with the growing international pressure on research, a variety of projects were conducted to build up distributed datacenters and "Virtual Observatories" (VOs) for sharing the research efforts. According to EURO-VO (2010) the main purpose of a VO is to allow global electronic access to the available astronomical data archives of space and ground-based observatories. It also aims to enable data analysis techniques through a coordinating entity that will provide common standards and state-of-the-art analysis tools. The "International Planetary Data Alliance" (IPDA) and the "International Virtual Observatory Alliance" (IVOA) are building up such coordinating entities. They are closely cooperating with NASA and ESA who naturally share interest in the development of such standards.

The IVOA is mainly dealing with astronomy data, so in fact there are a lot of different scientific fields covered in their specifications. This thesis is focusing on planetary data since EuroPlaNet is only dealing with planetology, so only a fraction of IVOA standards are suitable for this infrastructure. As stated in Arviset, Gaudet, and the IVOA Technical Coordination Group (2010) the IVOA architecture itself is comprised of a *Resource Layer* provided by large data centres and smaller teams. Consumers are interacting with the envisaged system via a *User Layer* and the *Virtual Observatory Layer* is representing the necessary middleware to connect the providers and consumers. This layer represents a technical framework to share data by providers and to enable the consumer to find, obtain and make use the data. All the functionalities of the IVOA framework are based on previously described XML standards and all capabilities of the Web Service Architecture Stack can be smoothly exploited.

The aspects of sharing data in specific formats, finding data over registries with certain semantics, making use of the data by querying the system and finally transporting the data over data access protocols are presented in the next sections, where the proposed architecture of the "Integrated and Distributed Information System" (IDIS) of EuroPlaNet is introduced. Since IDIS has not yet fixed its way to connect to data resources, another protocol is studied, the "Planetary Data Access Protocol" (PDAP) developed by the IPDA, which is merely developed for planetology. This protocol consists of metadata format standards and related semantics to be able to share scientific information among data archives like PDS and PSA. Furthermore it provides a specification to acesss the data via HTTP protocol similar to XML-RPC with GET/-POST according to Salgado et al. (2009, p. 2). In the meantime the IPDA technical experts group is working on more complex messaging protocols like SOAP. This specification fills the gap of missing semantics for planetology in IVOA, however it stays compatible and can be integrated in the IVOA architecture with a few modifications.

Besides of IDIS, a second approach of a service driven tool in EuroPlaNet is called "Automated Multi-Dataset Analysis" (AMDA). It mostly deals with plasma physics as related to planetology. AMDA is considered in more details in section 4.3.

4.1. The FP7 Project EuroPlaNet

One of the first steps after the official kick-off of EuroPlaNet at the beginning of 2009 was to start up a central portal for external and internal information access in order to pursue the main motivation of this project. The central objectives of the EuroPlaNet project are given by five major science themes which are covered by the twenty-seven intitiating participants dealing with the following research questions (Blanc et al., 2008, p. 2-4):

1. How did the solar system form?
2. How do giant planet systems form and work?
3. How do the terrestrial planets form and work?
4. What is the nature of the sun-planets connection?
5. How do extra-solar planetary systems form and work?

Each of the participants has its certain expertise and role within the project and the common goal is to solve scientific problems within those themes, which is primarily supported by space-based observations. I addition to that, the Research & Technical Development (RTD) facilities of EuroPlaNet have to extend their focus beyond space missions and share their knowledge and skills to the community (Blanc et al., 2008, p. 5).

The EuroPlaNet infrastructure is organized by a combination of Networking Activities, Transnational Access Activities and Joint Research Activities which are responsible for supporting the multi-disciplinary exchange of information. They also provide a variety of strategies for sucessfully launching, maintaining and evolving synergies among the participants. The EuroPlaNet-RI portal gives a brief overview about those activities (EuroPlaNet, 2012):

- The **Networking Activities (NAs)** aim to foster a culture of cooperation between planetary scientists within EuroPlaNet and the broader european community e.g. with coordinating ground-based observations.
- The **Transnational Access Activities (TNAs)** objective is to provide European users a structured access to state-of-the-art facilities e.g. laboratories and planetary field analogues.
- The **Joint Research Activities (JRAs)** purpose is to develop the EuroPlaNet Research Infrastructure further, improving the quality of facilities, models, software tools and services offered.

4.2. Integrated and Distributed Information Service (IDIS)

The main focus of this thesis is on the JRAs of EuroPlaNet where the "Integrated and Distributed Information Service" (IDIS), dealing with software tools and services is playing the central role. IDIS is also responsible for providing access to all results and findings obtained by the NAs and TNAs. Furthermore, IDIS also provides a web-based infrastructure for locating teams and laboratories with special knowledge. The aim is to support own research activities, give access to available data and initiate new research activities (Schmidt, Capria, & Chanteur, 2009, pp. 6). A separate IDIS portal was initiated from the beginning of the project and is evolving towards a "Virtual Observatory" (VO), where an interoperable information access system provides a wide range of different data sources and access tools located in different data centres (EuroPlaNet IDIS Management, 2010, p. 1).

To make a clear allocation of competences, IDIS is also divided into a *Service Activity (SA-IDIS)* and a *Joint Research Activity (JRA-IDIS)*. The SA-IDIS part is responsible for the management of all products related to EuroPlaNet and is providing those products via five different thematic nodes, which correspond to the above specified science themes. Each node has its own special field in planetary sciences, having their own teams and is responsible for providing access to information and data centres related to its area of competence (Capria et al., 2009, pp. 2). Figure figure 4.1 on the next page provides an overview about the thematical structure. As one can see, besides of the scientific nodes, there is also a technical node, which is coordinating the technical development of IDIS. The author of this thesis is technical manager of the

Planetary Plasma Node (PPN)[24] and hence responsible for all software developments in the plasma physics domain at IDIS.

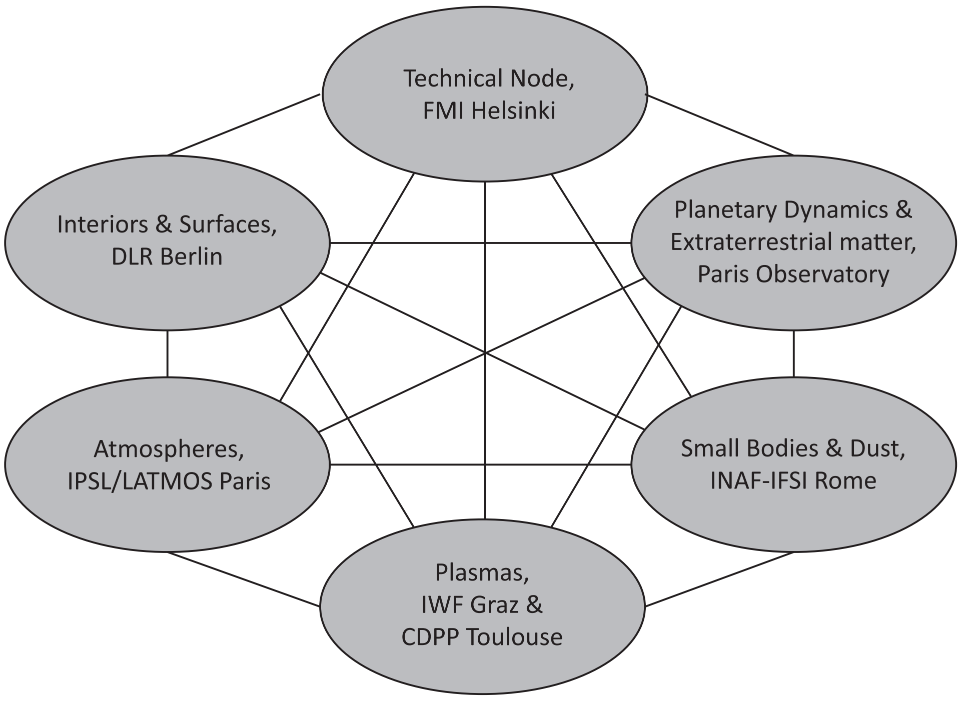

Figure 4.1.: Node Structure of EuroPlaNet IDIS, adapted from Schmidt et al. (2009)

The next sections give an detailed insight into the architecture of IDIS which is common for all nodes and aims to transform the nodes of SA-IDIS into a *Planetary Virtual Observatory*. Furthermore the development of data models by JRA-IDIS for providing a coordinated way of consuming and producing new data services and databases is outlined.

4.2.1. Overall IDIS Architecture

The JRA-IDIS activity with its Task 2 *Interoperable Data Access* will establish the basis allowing the evolution of SA-IDIS towards a future VO for planetary sciences (Capria et al., 2009, p. 3). This foresees the construction of an interoperable architecture where planetary scientists make use of the same common system coming from the distributed nodes of SA-IDIS. After careful studies, the task leaders of JRA-IDIS proposed to start with the existing technologies in space research, transforming them according to the project needs. The Overall IDIS Infrastructure will connect to existing distributed data services supporting IVOA protocols whereever it is necessary, enabling access to already existing data sources. Furthermore JRA-IDIS is participating in evaluating the PDAP protocol by IPDA and proposing the application of

[24]Planetary Plasma Node (PPN): `http://europlanet-plasmanode.oeaw.ac.at/`

specific needed data types for IDIS to the data model of PDAP (Erard et al., 2011d, p. 1).

First of all, the task leaders of JRA-IDIS started to set up a core of data services, which will be made accessible through VO protocols implemented by IDIS. These data services initually include the plasma physics tool AMDA, SSODnet[25] a solar system object database, GhoSST[26] a solid sample spectroscopy service and the PSA, the space mission archive operated by ESA (Erard et al., 2011c, p. 10).

The second step was to provide a general scheme, which demonstrates all steps of a typical working session. The IDIS user, working from the computer, sends request queries to a catalogue system or *registry* and gets responses where to find the data. After identifying the information source, the data may be delegated and processed with visualization tools for plotting images but also in more elaborated tools performing specialized functions. Figure 4.2 shows the way how an envisaged IDIS client discovers data, similar to the principles of UDDI in the *Process Layer* of the Web Service Architcture Stack. At first, the *IDIS User* sends a query to the *IDIS Registry*, requesting a needed data service. The *IDIS Client* retrieves a list of services suitable for the need of the *IDIS User*. The *IDIS User* then selects wanted services and transmits a request using the appropriate protocol needed to communicate with the service. Ultimately this service responds with a set of URLs pointing to datasets along with descriptions about available access methods.

The *IDIS Registry* is based on the IVOA "Resource Metadata for the Virtual Observatory" specification described in The IVOA Resource Registry Working Group et al. (2007), which extensively uses the "Dublin Core"[27] vocabulary, providing metadata for the purpose of service discovery. In IDIS this minimal description includes the name of the resource, its address and the protocols it supports. User queries are sent to a global catalogue containing a description of all accessible services and their capabilities. This can be seen as a first order selection of services matching the users query. In the IVOA specification this is done by having a system of mirrored registries where data providers can publish their services (Erard et al., 2011b, p. 8). The IDIS architecture also foresees the same approach as the *Service Description Layer* of the Web Service Architecture Stack, so the detailed description of the service is stored and maintained locally by the data providers (Erard et al., 2011d, p. 4).

Figure 4.2 also shows the initially planned data sources in context of the overal IDIS architecture, providing information about their own access protocols and data models. These include (Erard et al., 2011b, p. 7):

[25]SSODnet - Solar System Object Database Network: `http://vo.imcce.fr/webservices/ssodnet/`

[26]GhoSST - Grenoble Astrophysics and Planetology Solid Spectroscopy and Thermodynamics database service: `http://ghosst.obs.ujf-grenoble.fr/`

[27]Dublin Core specifications: `http://dublincore.org/specifications/`

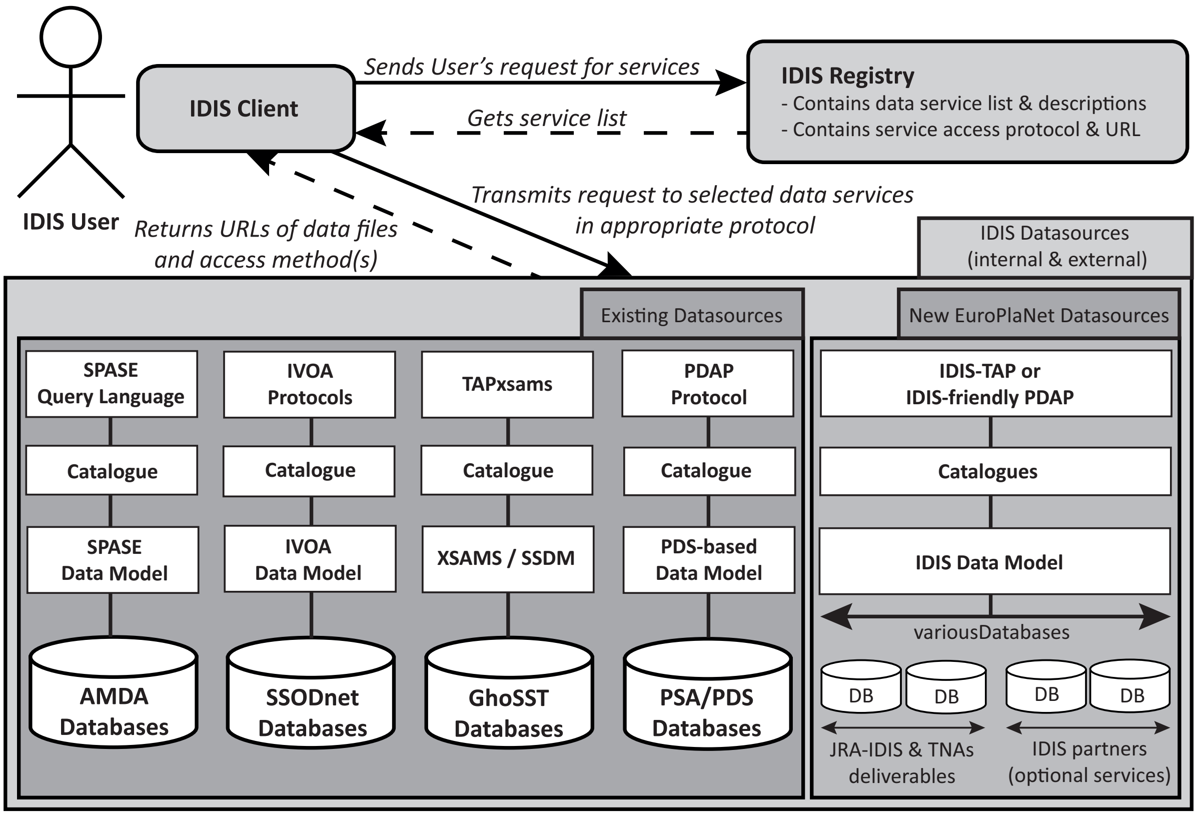
IDIS User
IDIS Client
Sends User's request for services
Gets service list
IDIS Registry
- Contains data service list & descriptions
- Contains service access protocol & URL
Returns URLs of data files and access method(s)
Transmits request to selected data services in appropriate protocol
IDIS Datasources (internal & external)
Existing Datasources
New EuroPlaNet Datasources
SPASE Query Language
IVOA Protocols
TAPxsams
PDAP Protocol
IDIS-TAP or IDIS-friendly PDAP
Catalogue
Catalogues
SPASE Data Model
IVOA Data Model
XSAMS / SSDM
PDS-based Data Model
IDIS Data Model
variousDatabases
AMDA Databases
SSODnet Databases
GhoSST Databases
PSA/PDS Databases
DB
JRA-IDIS & TNAs deliverables
IDIS partners (optional services)

Figure 4.2.: Overall IDIS architecture with envisaged data sources, adapted from Erard et al. (2011d) and Erard et al. (2011c)

- The SPASE Query Language and SPASE Data Model used to access services and data from AMDA which are further described in section 4.3 on page 38.
- The IVOA Protocols and Data Model used to access services and data from SSODnet. Currently the access protocol is the IVOA Observation Table Access Protocol (ObsTAP) together with the Observation Data Model Core Components (ObsCoreDM) which are further described in (Louys et al., 2011).
- The TAPxsams Protocol a special version of the IVOA TAP protocol and the XSAMS/Solid Spectroscopy Data Model (SSDM)[28] used to access services and data from GhoSST.
- The PDAP protocol to access the PSA and PDS archives via a PDS-based Data Model.

Besides of the existing data sources, there will also be new data sources coming from EuroPlaNet project results, like data sets from the TNAs and deliverables from JRA-IDIS Task 4 *New databases*. These databases will be accessible via IDIS-TAP a special version of the IVOA TAP or an IDIS-friendly PDAP protocol based on the *IDIS*

[28]XAMS - an XML Schema for Atoms, Molecules and Solids: `http://www-amdis.iaea.org/xsams/`

Data Model developed by the IDIS Data Model Working Group (IDIS-DM-WG), which is described in section 4.2.2.

A demonstrator version of the *IDIS Client*, which chains the functions described above[29], is currently designed by VO-Paris. The first step, as illustrated in Erard et al. (2011d, p. 6), is to provide an interface to send queries to the *IDIS Registry*. After the initial selection of services, the *IDIS Client* should be able to translate the description and send an appropriate query to the data provider. As a response to this request, there will be an VOTable built according to the IVOA XML fomat standard, where the dataset is represented as a set of tables. Each row in this table is a sequence of table cells and each of these contains either a primitive data type, or an array of such primitives (Ochsenbein & Williams, 2009). VOTable is designed as a flexible storage and exchange format for tabular data, since the XML fabric allows applications to easily validate requests or inputs with XML Schemas (XSD) as well as transformations trough XML Stylesheets (XSLT).

Ultimately the *IDIS Client* will be able to foward selected data to IVOA visualization tools for further processing such as the Aladin Sky Atlas[30] for example. These IVOA visualization tools uses the "Simple Application Messaging Protocol" (SAMP) which primarily enables astronomy software to interoperate and communicate. In fact, this protocol goes beyond simple data exchange between tools and is also designed to let VO-tools share functionality with other VO-tools. The semantics of the SAMP messages also follow the VOTable standard and according to Taylor et al. (2011) it is based on the XML-RPC specification which enables the protocol to use general SOA messaging concepts such as the *Publish/Subcribe* pattern or simple point-to-point communication with *Request/Response*. Altough there is no demonstrator of the proposed application within the IDIS architecture, the AMDA tool has already connected to Aladin using SAMP to obtain Auroral images from Jupiter and Saturn collected by the Hubble Space Telescope. The results of this development are described in André et al. (2011) and will naturally be reused in the development of the *IDIS Client*.

4.2.2. The IDIS Data Model

The IDIS Data Model Working Group (IDIS-DM-WG), comprised of experts from the IDIS nodes who dedicate their development work in elaborating a common access protocol and data model including semantics which enables to describe and share data in their specific science fields of planetology. As already indicated, this common aproach makes extensive use of existing standards from IVOA and IPDA. According to the overall IDIS architecture the *IDIS User*, accessing IDIS with the *IDIS Client* is expected to perform queries to describe a search. Such queries must be translated in a

[29]IDIS Demonstrators at the VO-Paris IDIS Node: `http://voparis-europlanet.obspm.fr`
[30]Aladin Sky Atlas: `http://aladin.u-strasbg.fr/`

standardized form to be used for searching the *IDIS Registry* where all available data are published (Erard et al., 2011d, p. 4). This metadata access is followed by the access to a selected data source via the *IDIS Client* to obtain the needed datasets. There are two main solutions for building queries, processing them and retrieving data related to planetary science products:

- The existing IVOA protocols, which allow discovering, describing and querying the data according to a variety of criteria. Particular interest is on the ObsCoreDM and the related ObsTAP.
- The PDAP protocol, currently developed by the IPDA. PDAP is mainly intended to address the entire contents of the PSA by querying these services as a whole (Salgado et al., 2009, pp. 3).

The IDIS-DM-WG objectives were separated in three tasks in order to be able to extract all usefull definitions from these existing approaches (Cecconi & the IDIS-DM-SWG, 2011, p. 2):

1. List the various data types and propose a set of query criteria that are relevant for data search in each discipline of IDIS.
2. Discuss how possible PDAP extensions for IDIS are to be proposed to the IPDA. This is based on the research done in previous task.
3. Propose, define, and select metadata dicionaries for the *IDIS Data Model* (i.e. select standard sources for the possible information describing the data)

During the first year of EuroPlaNet, the IDIS nodes developed a "General Resource Inventory"[31] in order to identify existing resources and services needed for the particular studies in the thematic fields. These resources were taken as starting point for task 1 of the IDIS-DM-WG in order to provide an overview about the needed data types and search criteria.

According to the architectural specifications by The IVOA Resource Registry Working Group et al. (2007) the general metadata is managed in a hierarchical system. The top-level type of metadata is referred to a *Resource*. It is a general VO element which has an unique idendifier and describes an entity e.g. a data provider or data collection. A *Service* is any VO resource which can be invoked by the user to perform an action on behalf of the user. Associated to this *Service* is descriptive metadata providing information how to use it. The additionally *Query Service* supports a *Request/Response* protocol, where a user can send a query to the service to define characteristics of interest. The service then returns a set of information to the user. In the "Web Service Architecture Stack" the layer *Descriptions* and *Messaging* is responsible for this part of the infrastructure. All the specified industry standards in these two layers will be incoporated in the IVOA scheme according to Schaaff and Graham (2010) including SOAP-based messaging, WSDL and UDDI. Naturally, this hierarchy of *Resources*

[31]IDIS General Resource Inventory: `http://www.europlanet-ri.eu/idis/res`

also fits perfectly into the overall IDIS architecture providing *registries* and *data access services*.

Since the focus of the IDIS-DM-WG is on building compliant data resources and query services in the field of planetology, the first data exchange protocol and data model studied is PDAP specified by IPDA. This protocol is specially designed for data from planetary missions. It supports the needed datatypes and search criteria for IDIS. For instance, it allows the user to perform queries related to coordinates on a planetary surface, to atmospheric profiles, or to internal structures, which are missing in the IVOA architecture (Erard et al., 2011d, p. 3). Furthermore, the primary coordinate of PDAP is time (e.g. the start and stop time of an observation), whereas the primary coordinate in the IVOA architecture are sky-coordinates (Cecconi et al., 2011, p. 10). As already indicated in the IDIS overall architecture, PDAP follows the above mentioned IVOA scheme, so it can be integrated in the general IVOA architecture. The PDAP protocol has a similar structure than general SOA technologies by means of data access. The first step of the protocol obtains metadata about a specific service and the second step retrieves the real data which is provided by the service (Salgado et al., 2009, p. 2).

In addition to the IVOA metadata hierarchy, the IDIS-DM-WG has set up four levels of descriptions for a data resource, each defining a particular granularity (Cecconi & the IDIS-DM-SWG, 2011, p. 3):

- A *Dataset*, which is a series of homogenous data, so that all data records have the same structure.
- A *Granule*, which is a homogenous group of records in a dataset.
- A *Parameter*, which is a series of data identified by a specific quantity.
- A *Record*, which is an individual set of values, that cannot be split without loosing the homogenity of the dataset.

According to these definitions, the metadata description of a resource in PDAP follows the same hierarchy, starting from a general description about the dataset and provider, describing the involved instrument of a space mission, followed by the target of the mission and ending with parameter metadata of the measured quantitites. An example below shows the PDAP metadata descriptor for the *Venus Express Magnetometer*[32] dataset which was assembled by the author for the IDIS Plasma Node. The required selection criteria for magnetospheric measurements, as well as technical information about the resource are covered. According to the general SOA approach with Web services, the descriptor is formatted in XML following the rules and constrains of an XML Schema definition file.[33] The whole descriptor source of PDAP VEX MAG can be found in appendix B.

[32]Venus Express: `http://www.esa.int/esaMI/Venus_Express/`

[33]Target Namespace of the XSD defintion: `http://typhon.obspm.fr/idis/docs/EPNResource-v117.xsd`

General Metadata:

The first element within <GeneralMetadata> contains "Dublin Core" metadata. By definition, this metadata is a part of an interoperable standard to support a broad range of purposes and business models (The Dublin Core Metadata Initiative, 2012). In this case, the definition of the resource contains a title, an unique identifier, the name of creator and publisher, as well as the publishing date. The Dublin Core vocabulary provides more possible elements, but the XML Schema definition file of <EPNResource> requires only those mentioned above. In addition to that, contact information to the resource and general information about the data are provided. These include the decriptions about the format, the access rights and an access url. The <Format> element is also mandatory by definition and contains a set of possible format types. The whole list of possible elements, a detailed description, and formating rules can be found in (Cecconi et al., 2011, p. 12).

Instrument Metadata:

Since the physical parameters of a dataset are usually physical measurements, the <Instrument> metadata describes the instruments (one or more) linked with the dataset. The first element indicates the name of the mission. According to Cecconi et al. (2011, p. 14) the <MissionName> and the subsequent <InstrumentName> element shall be checked with the IPDA/PDS dictionaries available in Rye and the PDS Object Review Commitee (2008). The <InstrumentType> element describes the purpose of the instrument and allowed elements are taken from the SPASE data model dictionary available in The SPASE Consortium (2011). Finally <InstrumentKey> provides a unique identifier for the instrument. All the mentioned elements are mandatory, only the <ReferenceURL> at the end of the instrument is optional. It provides a possible link to additional information about the instrument.

Target Metadata:

The physical parameters of a dataset usually aim at one or several targets. The <Target> metadata describes targets (one or more) associated with the dataset. The first element <TargetType> can be for example a star, a planet, a spacecraft, a region or a feature. The full list of possible target types can be found in Cecconi et al. (2011, p. 61). It is extracted from the target type list of the PDS dictionary and the observed region type list from the SPASE dictionary. The region type is used to define a part of the parent target, such as a layer of an atmosphere or a part of a magnetosphere. The feature type is used to define a feature or phenomenon, such as radio emissions or a type of atmospheric clouds. Additionally to these elements, the <Target> element also has to contain a name, which shall be taken from authoritative lists such as the official IAU[34] thesaurus, together with an unique identifier. The example considered in appendix B includes two targets, secondarily the Venus magnetosphere, having a <ParentKey> which associates it with the primary target, the planet Venus.

[34]International Astronomical Union: http://www.iau.org/

AxisFrame Metadata:

The `<AxisFrame>` metadata describes the various axes or frames of the physical parameters of a dataset. It is identified by an unique `<AxisKey>`. There is at least one axis required and the dimension of the axis shall be defined in terms of type (time, space, frequency), range (min, max), resolution (min, max), accuracy and units. The full list of axis types can be found in Cecconi et al. (2011, pp. 15). Note, that this list is derived from the IVOA "Characterization Data Model" described in The IVOA Data Model Working Group (2008), with a few extensions. The example in appendix B provides one `<AxisFrame>` representing the time in the units are according to the ISO-8601[35] standard. The `<DimensionMode>` element additionally gives information if the axis is in absolut values or relative to a reference point. Finally, the `<DimensionRange>` provides a start and stop time (`<MinValue>` and `<MaxValue>`).

Parameter Metadata:

Parameters are represented as a set of numerical or text values with an arbitrary number of dimensions. Each parameter is described in terms of: parameter type, associated instrument, associated axis, units, processing level and parameter sensing. In addition to that, a name and a description of the parameter is given. The `<ParameterType>` element describes the generic type of the parameter and its concept is derived from the "Unified Content Descriptors" (UCD) by IVOA (see: Martinez et al., 2007)). However, the original UCD list cannot be used, since it does not cover space physics measurements. Therefore a combined list of UCDs from PDS dictionary and the SPASE dictionary is proposed which can be found in Cecconi et al. (2011, p. 62-65). The example in appendix B has two `<Parameter>` elements which are representing a type of $dc.mag$. This type corresponds to a DC field (dc) in general and magnetic field (mag) in specific. The `<IntrumentKey>`, `<AxisKey` as well as the `<TargetKey>` refer to the previously described elements in each of the listed `<Parameter>` elements. The `<Units>` element is a generic type and describes the unit of the axis with three sub elements. This example defines the expression of the unit $Nanotesla(nT)$, the according transformation in the SI derived unit $Tesla(T)$[36] and the dimensional equation[37] with respect to time. Further information on the `<Parameter>` elements are listed in Cecconi et al. (2011, pp. 21).

SupportParameter Metadata:

The `<SupportParameter>` elements follows a similar scheme as the physical parameters. The example shown in appendix B specifies the parameter as the UCD type $loc.ephem$ which refers to location in general and ephemeris of the spacecraft in specific. The `<Units>` element also contains three sub elements defining the parameter in $Kilometres(km)$, the according SI scale and the dimensional equation. A full list of `<ParameterType>` elements is given in Cecconi et al. (2011, p. 23).

[35]ISO-8601 standard for date and time: `http://en.wikipedia.org/wiki/ISO_8601`
[36]$Tesla$ (SI derived unit): `http://en.wikipedia.org/wiki/Tesla_(unit)`
[37]Dimensional Analysis: `http://en.wikipedia.org/wiki/Dimensional_analysis`

According to the studies in Cecconi and the IDIS-DM-SWG (2011, p. 5), the equivalent data model in the IVOA architecture is the ObsCoreDM, which can handle the described metadata groups only with very few extensions. This will finally enable the *IDIS Data Model* to be more flexible in connection with the proposed IVOA discovery process in the overall IDIS architecture. Another advantage of providing a translation to the IVOA standards will be the integration of the ObsTAP protocol to exchange tabular data via common messaging protocols such as SOAP. In this case the data provider has to implement a mechanism for providing the data in `VOTable` format. In addition to that, the IDIS-DM-WG will conduct tests with the current *IDIS Data Model* on the first version of PDAP which will be released in early 2012. The corresponding assessment studies on PDAP to extend the protocol for specific needs of the IDIS nodes are described further in Erard et al. (2011a).

4.3. Automated Multi Dataset Analysis Tool (AMDA)

The "Automated Multi-Dataset Analysis" tool (AMDA)[38] is a modern information system for planetary research in the field of plasma physics developed at the CDPP in Toulouse. It's main function is to provide a web-facility for online analysis of physics data coming from its local and remote databases which are connected via Web services (Jacquey et al., 2010, p. 1). The tool provides classical manipulations on data, which are broadly used in the scientific community of planetary research. The user can visualize data, extract data and make parameter computation. There is also a possibility for specific event search in the content of the data, either with visualisation or in computational manner. In addition to that, the user can create time-tables and catalogues in VOTable format for continuing specific research work in other VO-like tools. Comparative studies within AMDA and its different connected databases is also possible. The AMDA tool is an integral part of the IDIS Plasma Node and the overall IDIS architecture and the presented work is partly resulting from these EuroPlaNet activities. This tool has developed an infrastructure according to the Web Service Architecture Stack which is able to provide access to planetary plasma data coming from different sources and already has a fully interoperable environment in terms of organization, description and formatting of datasets (Topf et al., 2011b).

The usual way a scientist is doing research is covered with preceding case studies in the needs for data exploitation, computation and visualisation (e.g, André et al. (2009, pp 238); André et al. (2011, p. 4-10)).
Five steps were identified as main ingredient for an automatic web-based solution:

1. Event search
2. Collection of the data

[38]Automated Multi-Dataset Analysis: `http://cdpp-amda.cesr.fr/`

3. Formatting the data
4. Preparing the data
5. Data analysis

Following the studies in Jacquey et al. (2010, p. 3) the most crucial step is the formatting and the preparation of data in order to be able to apply to them the inbuilt data analysis routines.

4.3.1. Overall AMDA Architecture

According to Gangloff et al. (2009, p. 2) AMDA is devided into three main components:

- The *AMDA-Client*, which is a web-based graphical interface accessible via a common Web-browser.
- The *AMDA-Server* – the core of AMDA implemented in PHP, which handles all the functionalities offered through the graphical interface.
- The *DD-Server* (Data Distribution Server), which feeds the *AMDA-Server* with local or remote data.

This thesis focuses on certain functionalities of the *AMDA-Server* and the data access to remote databases via the *AMDA-Registry* and the *DD-Server* according to standardized data access protocols. The essential parts of the *AMDA-Client* and its access to data exploitation and visualisation capabilities of the *AMDA-Server* are further described in Jacquey et al. (2010).

Figure 4.3 provides an overview of the general AMDA architecture and the two basic ways, a *AMDA-User* may take advantage of the functionalities of the system. The most common way is via the *AMDA-Client*, a web-based graphical interface, where the user can both access and analyse the available datasets on-the-fly. Another possibility to obtain access to the databases is via the *AMDA-Registry*, which simply provides a searchable list of all the data available from the local and remote databases. This *AMDA-Registry* is comprised of a WSDL[39] descriptor providing the necessary access methods for discovery of the needed datasets. As one can see in figure 4.3, two other web-based systems "HELIO"[40] (a heliophysics VO) and "3D View Multimission"[41] (a JAVA based 3D visualisation tool for spacecraft and solar system object ephemeris) use the *AMDA-Registry* for the access to the services of AMDA.

[39]WSDL descriptor of the AMDA-Registry: `http://cdpp-amda.cesr.fr/BASE/DDService/amda.wsdl`

[40]The FP7 project HELIO: `http://www.helio-vo.eu/`

[41]3D View Multimission: `http://mm3dview.cnes.fr/`

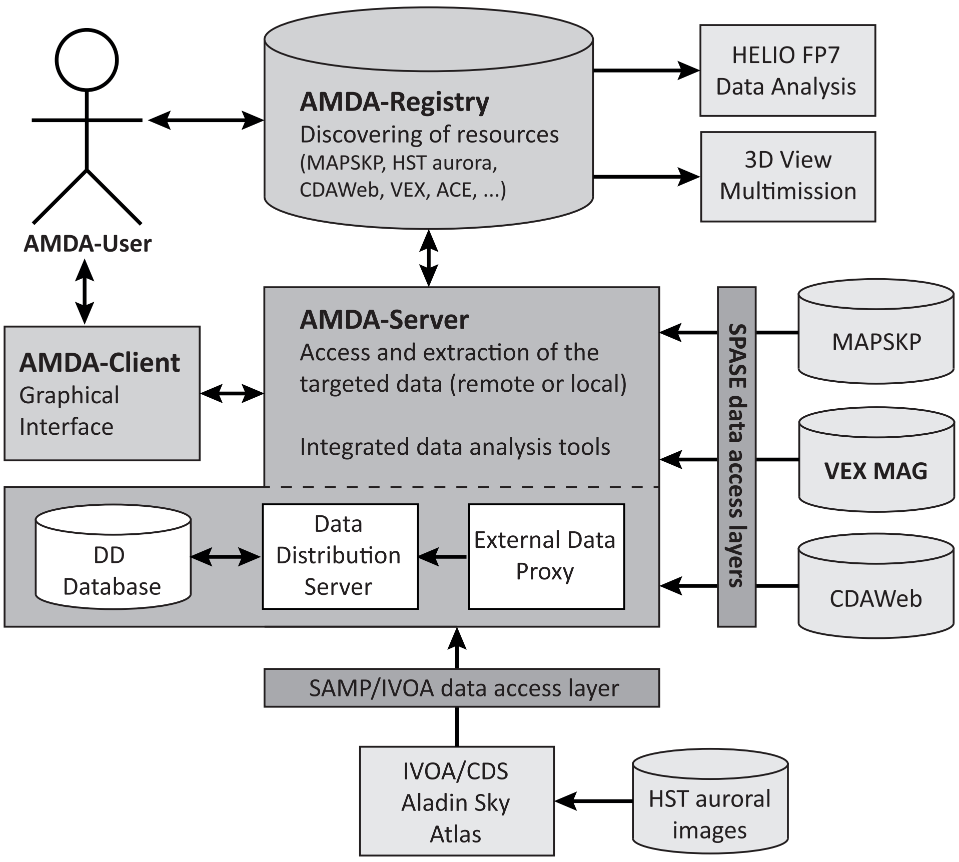

Figure 4.3.: Overall AMDA Architecture accessing remote databases, adapted from Topf et al. (2011b) and Gangloff et al. (2009)

The needed metadata for such services is described with an XML data model coming from the heliophysics research, called "Space Physics Archive Search and Extract" (SPASE), which also provides semantics for plasma and magnetospheric physics to a certain extend. According to The SPASE Consortium (2011, p. 2) the data model is capable of describing datasets, but has also the possibility to describe repositories, registries and services holding scientific data with SPASE metadata. The *AMDA-Registry* is not providing SPASE descriptions of the datasets themselves, this metadata is directly stored at the location of the databases, either locally or remotely. From the *AMDA-Server*s point of view, the access to different datasources is homogenous. The external data is mapped with a proxy service into the internal *DD-Server* and the *DD-Database* respectively. The last holds primarily local datasets put also caches frequently used external data. The actual exchange of data between AMDA and data sources is accomplished via a SPASE data access layer using the SOAP protocol. In order to be able to use the semantics of the SPASE datamodel for searching and identifying datasets and their physical quantities, the remote databases and the *AMDA-Registry* are providing the SPASE metadata via eXist[42] XML databases. This open source database management system provides an search interface for XML files and associated XML Schemas following the querying standards XQuery and XPath by

[42]eXist-db available from: `http://exist.sourceforge.net/`

W3C, according to Meier (2011). XQuery and XPath are providing a specific grammar for extracting information from hierarchical organised XML files in both human-readable syntax and XML-based syntax as stated in Boag et al. (2010) where a detailed description of the semantics is provided.

In addition to that, the AMDA tool is also connected to the "Aladin Sky Atlas" maintained by CDS Strasbourg via the IVOA SAMP protocol which is a subject of other publications.

The following section will provide a complete dataset description of the *Venus Express Magnetometer* data already addressed in section 4.2.2 on page 33 in the context of a *Repository* according to the SPASE hierarchy.

4.3.2. The SPASE Datamodel

The SPASE datamodel, as described in The SPASE Consortium (2011, pp. 5) is intended to enable the sharing of knowledge through structured metadata which can be exchanged in requests (or queries) and responses between "loosely coupled" and highly distributed systems. This approach exactly follows the SOA paradigm and the needs for an interoperable architecture like AMDA. In order to be able to describe all parts of the system, a top level entity, the *Resource* is introduced in the SPASE XML datamodel. There are three main resource types:

- **Data Resources**, describing datasets e.g *Numerical Data* or *Display Data*
- **Origination Resources**, describing the sources or generators of datasets e.g. *Observatory*, *Instrument* or *Person*
- **Infrastructure Resources**, describing system components that are part of the exchange and use of data e.g *Repository*, *Registry* or *Service*

In the case of the VEX MAG Web service, there are five resource types to be provided along with the real dataset files (Topf et al., 2011b): *Repository*, *Person*, *Observatory*, *Instrument* and *Numerical Data*. AMDA for example, obtains a summary of all available observatories in a *Repository* and provides this information either directly to the *AMDA-User* (in order to refine her/his search) or to the *AMDA-Registry*.

Each of the obove listed resources has an unique identifier (an URI) in the `<ResourceID>` element, so that the resources can be tracked and referenced in other resources within the system. The descriptor file representing the *Repository* where the VEX MAG datasets are stored can be found in listing C.1 in appendix C together with all the other required descriptors. The URI of the *Repository* is `spase://iwf/repository/Vex`. It indicates the overall schema with `spase` as data model, the naming authority (`iwf`), the type of *Resource* (`repository`) and the name of the resource (vex). Furthermore, there is an element representing the full name of the *Repository* and a hyperlink to detailed information.

The second descriptor in the listing C.2 provides an information about the source of the datasets, in the case of VEX MAG it is an <Observatory> with the name $VenusExpress$. The <Observatory> element also references to a person with <PersonID> in the <Contact> element. The person responsible for the observation is described in a separate SPASE descriptor, which is shown in listing the C.3. The <Instrument> element in the listing C.4 gives information about the measurement device. It is linked to the observatory with <ObservatoryID> and also to the responsible person.

The most important SPASE descriptor in the listing C.5 contains the information about the associated data. In the case of VEX MAG this is <NumericalData>. It is linked to the *Repository*, the instrument and the person with an URI reference. This descriptor provides information about the four physical quantities stored in the dataset within elements of <PhysicalParameter> and information about the time frame, similar to the axis in the *IDIS Data Model*. Futhermore, the physical parameters provide an information about the measured quantities (e.g. $Magnetic$) or which type of support values are given (e.g. $Positional$). Note, that each parameter has also a <ParameterKey> element, giving it a unique identifier. In case of more complex physical quantities like vectors, each coordinate has such a <ParameterKey>.

Unfortunately, the SPASE data model does not civer all the necessary information for AMDA. The data provider needs to store an additional XML file in the location of the data files (see listing C.6). It duplicates the information of the physical quantities, but adds the elements <DATA_TYPE> and <DISPLAY_TYPE>, which refer to the data and display types needed to make computations and visualisations at the *AMDA-Server* (Gangloff et al., 2009, p. 5).

4.3.3. Venus Express MAG Web Service

Besides of describing the resources with SPASE, the data provider has to implement three *Remote Procedure Calls* (RPC) (see figure 4.4) via the SOAP protocol in order to make the actual data available, according to the sercvice requirements of AMDA. The overall architecture in 4.4 clearly illustrates the principle of the *Request/Response* pattern in these calls. Note, that in this pattern AMDA represents the client or consumer of this Web service.

The first RPC call getAvailableData made by AMDA asks the SOAP server which data is available in the repository (Topf et al., 2011b). On the server side, the service interface extracts all the *Originating Resources* described in SPASE from the eXist database via XQuery. The XML response in this case, is comprised by information of the observatory (*Venus Express*), the associated instrument (*Venus Express Magnetometer*) and its contained physical quantities (magnetic field and spacecraft position). Note, that this Web service is flexible enough to handle different combinations of resources in the SPASE hierarchy, e.g. more than one observatory, instrument or data source.

In the second call the consumer can specify a `<ResourceID>` of the previously received data resources in order to refine her/his search. There is only one option in this Web service – the `<NumericalData>` element with the URI `spase://iwf/numericalData/Vex_Mag_VSO`. The SOAP request `getDatasetInfoUrl` with the parameter $dataSetId$ (containing the previously selected URI) responds with two URLs to XML descriptor files. The first file contains the SPASE elements `<NumericalData>` extracted with XQuery and the second file lists `<DATA_TYPE>` and `<DISPLAY_TYPE>` from the additional XML descriptor. Note, that the descriptors also contain the start and the end date of the dataset and the classification as time series. This information is needed for the next step.

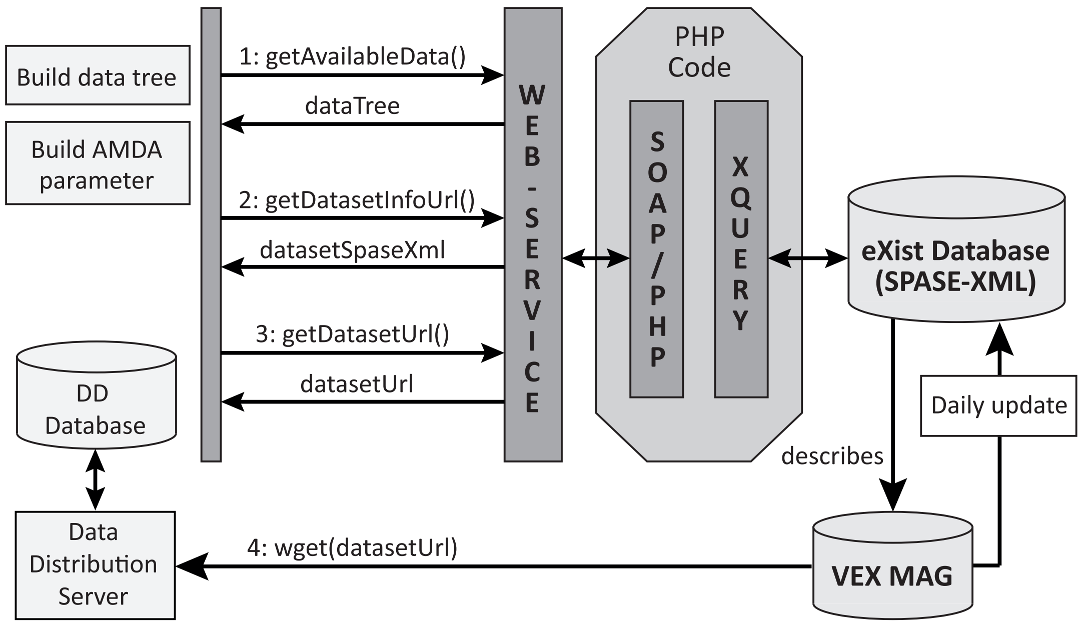

Figure 4.4.: Architecture of the VEX MAG Web service, adapted from Topf et al. (2011a)

The final RPC call, `getDatasetUrl`, requires three parameters: the previously used $dataSetId$, $dateStart$ and $dateStop$. The last two indicated the requested timeframe from the dataset. They will respond with URLs to the according ASCII files, which are accessible via the HTTP protocol. On the server side, the service interface checks the validity of the dates by querying the according SPASE descriptor. If the input times are within the range, the service interface maps the three parameters to the organizational structure of the ASCII files on the Web server.

As already indicated in section 3.4, the SOAP protocol does not provide information about available RPC calls to the client. In case of AMDA the developers requested a direct URL to a SOAP server written in PHP. In order to be more flexible with other technologies implementing SOAP, the author described the available Web service with an additional WSDL file. The content of this descriptor along with links to a simple SOAP client written in PHP and a complete DOXYGEN[43] documentation about the PHP SOAP server is provided in appendix D.

[43]DOXYGEN documentation system: `http://www.doxygen.org/`

5. Results and Discussion

According to the overall composition of this thesis, a general approach to the "Service Oriented Architecture" paradigm was made in order to identify the key elements of infrastructures defined as *service-oriented*. The reason for these studies was to show the reader, why the implementation of SOA is fundamental in succeeding with the interconnection of data sources having different data access standards and formats. One of the main aspects of SOA was identified as the "separation of concerns" in complex business processes, where the process is subdivided into smaller pieces, each with a certain functionality to distribute. In case of the here presented examples within the planetary science community, with focus on the EuroPlaNet project, a clear separation of an entitiy archiving data, and an entity processing and using the data was made. A crucial step after the separation of responsiblities was to agree on parts of the envisaged architecture to be built of in order to sucessfully conduct scientfic research. According to the SOA paradigm, each part of the system can exist autonomously while providing a standardized interface to the other components by enrolling a *service contract*. In order to be able to make *services* aware on each other, another aspect of SOA was identified, which is of significant importance for the projects conducted in EuroPlaNet. It consists of the "loose coupling" relationship between services with *service descriptions*. They provide a simple scheme of roles and messaging semantics for both *service provider* and *service consumer* in order to share resources and making them accessible. Since the term *interoperability* was indicated in the workpackage description of EuroPlaNet IDIS, the primary goal of the RTD there was to specifiy a common standard to achieve the "loose coupling" of different services.

The proposal of EuroPlaNet also indicated the extensive use of the internet for communication. For that reason, a precise evaluation of the industry standards of SOA in the World Wide Web is provided by introducing XML and the "Web Service Architecture Stack". The advantages of the XML format by providing *vocabularies* for the semantic description of resources and according validation schemes show, that it is extensible and robust enough to handle scientific information. The comparison of the four main layers of the "Web Service Architecture Stack" with the overall *IDIS-Architecture* reveals, that all the capabilties are well known by the scientific community. The whole spectrum of messaging protocols with *Request/Response* between service partners and discovery processes with *Publish/Subcribe* over *Registry* systems is covered in the described specifications.

The main challenge of the *IDIS-Architecture* remains to provide all necessary semantics via an *IDIS Data Model* for description of services and resources. The according *vocabularies* and *dictionaries* have to be extended and further studies on PDAP and the IVOA specifications are needed to solve remaining contradictions. Currently, the *IDIS Data model* only provides a fraction of semantics needed to cover all the thematical fields of IDIS and is sharing no syntactic information about how to extract the data, which remains to be a task for the potential data within the EuroPlaNet project. The IVOA architecture clearly is an already well established standard within astronomy, but it remains a challenge for planetary science research workflows since semantics for the main physical features are not the same.

Recent work presented at the "IDIS General Meeting" in early 2012 shows the progess made with the *IDIS Data Model* development. The IDIS-DM-WG integrated it into the IVOA *VOResource* XML Schemes, which are part of the IVOA registry interfaces for searching and publishing services. These *VOResource* element types are based on the described general metadata hierarchy from IVOA. They are providing XML validation mechanisms specialized for resource types such as a *registry* and a *data access service* in the IDIS architecture. However it is still an issue to be discussed among the IDIS-DM-WG members to which extend the IVOA data model ObsCoreDM and the data access protocol ObsTAP are useful to be included into the *IDIS Data Model*. The prototype version of the *IDIS Registry* service[44] was also presented in Graz. It provides a search interface, which is able to query all the included metadata of the *IDIS Data Model* and connects to several distributed datacenters currently supporting the PDAP protocol. In addition to that, the CDPP in Toulouse shares all PDAP descriptors defined by the IDIS-DM-WG to this service, including the VEX MAG dataset description.

As an integral part of IDIS, the AMDA tool already provides a fully functional Web service infrastructure. The fundamental *semantics* and *vocabularies* needed for describing data in the plasma physics domain are implemented with the SPASE data model. So AMDA already has made a step beyond defining a metadata description language and began focussing on the messaging capabilities between service providers and consumers. The use of the described XML standards enabled the AMDA tool to access distributed datasources by providing Web service interfaces with SOAP. There have been efforts to study the various W3C specifications for querying *registries* in the discovery process, so that the user can easily find required datasets. After selection of a specific resource, the AMDA architecture adds value to the consumed Web service by providing a variety of data analysis capabilities. As one can see, the AMDA architecture is highly extensible with its *registry* where published services may be used by AMDA itself or other remote tools and services. External data providers willing to publish their own datasets to AMDA can easily provide a SOAP server interface including SPASE metadata descriptions for their shared scientifc data. This is shown by means of the VEX MAG dataset in this thesis.

[44]The IDIS Registry – A PDAP search interface: `http://manunja.cesr.fr/PDAP_CDPP_WS/WS/pdapRequest.php`

6. Conclusions and Outlook

The work done within the IDIS-DM-WG goes on towards building up a bridge between the protocols and standards of VO infrastructures by the IVOA and the planetary data preservation efforts of the IPDA. The separation of responsibilties will help that the implemented technologies stay interoperable yet providing the needed flexibility to conduct interdisciplinary research. The use of industry standards from the W3C will ensure, that the provided services can be consumed by different software architectures and will stay sustainable in the future. The big datacenters PSA and PDS are able to continue focusing on archiving the data, whereas the IPDA is providing the interfaces for communication among data providers and consumers.

In the next years a significant amount of resources is invested by the FP7-SPACE programme[45] to support the development of integrated solutions in the field of space sciences. The already elaborated AMDA tool and its flexible SOAP interface enriched with SPASE metadata, becomes a part of the "Integrated Medium for Planetary Exploration" (IMPEx)[46], founded within EU FP7 as a collaborative project. IMPEx aims at a similar architecture as IDIS by creating a interactive framework to interconnect observational data with data produced by numerical models in the plasma physics domain (Khodachenko et al., 2011). The communication between tools will be of importance as well as the description of simulation data. The foreseen architecture may implement more complex mechanisms of the "Web Service Architecture Stack" incldiuing certain *Security* aspects and *Processes* such as aggregation and choreography.

[45]FP7-SPACE call:
`http://cordis.europa.eu/fp7/dc/index.cfm?fuseaction=UserSite.FP7CallSummaryPage&call_id=38`

[46]IMPEx FP7: `http://impex-fp7.oeaw.ac.at/`

A. Web Service Description Language (WSDL) Example

The following listings represent an *abstract* and a *concrete* part of a WSDL which is precisely described in section 3.5 on page 24.

Listing A.1: The abstract part of a WSDL

```
<definitions name="QuoteService"
  targetNamespace="urn:QuoteService"
  xmlns:n="urn:QuoteService"
  xmlns="http://www.w3.org/ns/wsdl">
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://www.w3.org/ns/wsdl/soap"
  xmlns:http="http://www.w3.org/ns/wsdl/http"
<types>
  <!-- schema details go here -->
</types>
<message name="getQuoteRequest">
  <part name="symbol" xsi:type="xsd:string"/>
</message>
<message name="getQuoteResponse">
  <part name="value" xsi:type="xsd:float"/>
</message>
<portType name="stockQuoteService">
  <operation name="getQuote">
    <input message="n:getQuoteRequest"/>
    <output message="n:getQuoteResponse"/>
  </operation>
</portType>
```

Listing A.2: The concrete part of a WSDL

```
<binding name="stockQuoteBinding" type="↩
   n:stockQuoteService">
  <soap transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getQuote">
```

```
      <soap:operation soapAction="urn:QuoteService" style="↩
         rpc"/>
      <input>
        <soap:body use="encoded"
          encodingStyle="http://www.w3.org/2003/05/soap-↩
             encoding"/>
      </input>
      <output>
        <soap:body use="encoded"
          encodingStyle="http://www.w3.org/2003/05/soap-↩
             encoding"/>
      </output>
    </operation>
  </binding>
  <service name="StockQuote">
    <port name="stockQuoteService"
      binding="n:stockQuoteBinding">
      <address location="http://example.org/getQuote"/>
    </port>
  </service>
  </definitions>
```

B. Venus Express Magnetic Field Data descriptor for PDAP

The *Venus Express Magnetometer* (VEX MAG) measures the magnetic field vector around Venus. The according observational data is hosted by the IWF (Institut für Weltraumforschung/Space Research Institute) of the Austrian Academy of Sciences, in Graz, Austria. The data are freely available through FTP, as text files. The following XML file represents the corresponding PDAP descriptor, which is explained in section 4.2.2 on page 33. It was generated with a JAXFRONT[47] XML form available from `http://oberoi.cesr.fr:8080/jaxfront/JAXFrontServlet?app=jaxfront&action=loadResource&resource=jumpStart/jumpStart.html`

Listing B.1: The VEX MAG PDAP descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<EPNResource xmlns:xsi="http://www.w3.org/2001/XMLSchema-↩
   instance" xsi:schemaLocation="EPNResource_1_17.xsd" ↩
   xmlns="http://www.europlanet-ri.eu.namespace">
<Dataset>
  <GeneralMetadata>
    <DublinCore>
      <Title>
        Venus Express Magnetic Field in VSO Coordinates
      </Title>
      <Identifier>VEX_MAG_VSO</Identifier>
      <Publisher>IWF-OeAW</Publisher>
      <Creator>IWF-OeAW</Creator>
      <Date>2011-02-04</Date>
    </DublinCore>
    <Contact>
      <Name>Florian Topf</Name>
      <Address>
        IWF-OeAW, Schmiedlstrasse 6, A-8042 Graz, Austria
      </Address>
      <Email>florian.topfoeaw.ac.at</Email>
    </Contact>
```

[47]JAXFRONT GUI solutions: `http://www.jaxfront.org/`

```
    <Rights>public</Rights>
    <Format>ASCII</Format>
    <AccessURL>
      ftp://amda-idis.oeaw.ac.at/MAG/VSO/
    </AccessURL>
  </GeneralMetadata>
  <Instrument>
    <MissionName>Venus Express</MissionName>
    <InstrumentName>MAG</InstrumentName>
    <InstrumentType>Magnetometer</InstrumentType>
    <InstrumentKey>VEX\_MAG</InstrumentKey>
    <ReferenceURL/>
  </Instrument>
  <Target>
    <TargetType>Region</TargetType>
    <TargetName>Magnetosphere</TargetName>
    <TargetKey>magnetosphere</TargetKey>
    <ParentTargetKey>venus</ParentTargetKey>
  </Target>
  <Target>
    <TargetType>Planet</TargetType>
    <TargetName>Venus</TargetName>
    <TargetKey>venus</TargetKey>
    <ParentTargetKey/>
  </Target>
  <AxisFrame>
    <AxisKey>time</AxisKey>
    <AxisName>time</AxisName>
    <DimensionType>time</DimensionType>
    <Units>
      <Time>ISO-8601</Time>
    </Units>
    <DimensionMode>
      <Absolute/>
    </DimensionMode>
    <DimensionRange>
      <MinValue>2006-04-24T00:00:00</MinValue>
      <MaxValue>2009-12-27T23:59:56</MaxValue>
    </DimensionRange>
  </AxisFrame>
  <Parameter>
    <ParameterName>B\_SC</ParameterName>
    <ParameterDescription>
      Magnetic Field Vector in SC Coordinates
    </ParameterDescription>
    <ParameterType>dc.mag</ParameterType>
    <ParameterKey/>
    <InstrumentKey>VEX\_MAG</InstrumentKey>
    <AxisKey>time</AxisKey>
    <Units>
      <Expression>nT</Expression>
      <ScaleSI>1e-9</ScaleSI>
      <DimEquation>TM-1Q-1</DimEquation>
    </Units>
    <ProcessingLevel>
      calibrated (fully calibrated data)
    </ProcessingLevel>
    <SensingMode>in situ</SensingMode>
    <SensingType>passive</SensingType>
    <DataType>measurement</DataType>
    <ObservationDescription>
      <TargetKey>magnetosphere</TargetKey>
      <ParameterType/>
    </ObservationDescription>
  </Parameter>
  <Parameter>
    <ParameterName>B\_VSO</ParameterName>
    <ParameterDescription>
      Magnetic Field Vector in VSO Coordinates
    </ParameterDescription>
    <ParameterType>dc.mag</ParameterType>
    <ParameterKey/>
    <InstrumentKey>VEX\_MAG</InstrumentKey>
    <AxisKey>time</AxisKey>
    <Units>
      <Expression>nT</Expression>
      <ScaleSI>1e-9</ScaleSI>
      <DimEquation>TM-1Q-1</DimEquation>
    </Units>
    <ProcessingLevel>
      calibrated (fully calibrated data)
    </ProcessingLevel>
    <SensingMode>in situ</SensingMode>
    <SensingType>passive</SensingType>
    <DataType>measurement</DataType>
    <ObservationDescription>
      <TargetKey>magnetosphere</TargetKey>
      <ParameterType/>
    </ObservationDescription>
  </Parameter>
  <SupportParameter>
    <ParameterName>POS\_SC\_VSO</ParameterName>
    <ParameterDescription>
      Position of Spacecraft in VSO Coordinates
    </ParameterDescription>
    <ParameterType>loc.ephem</ParameterType>
    <ParameterKey/>
    <InstrumentKey>VEX\_MAG</InstrumentKey>
    <AxisKey>time</AxisKey>
    <Units>
      <Expression>km</Expression>
      <ScaleSI>1000</ScaleSI>
      <DimEquation>L</DimEquation>
    </Units>
  </SupportParameter>
</Dataset>
</EPNResource>
```

C. Venus Express Magnetic Field Data descriptor for SPASE

The following five XML files represent the full description of the SPASE repository for the VEX MAG datasets maintained by the IWF in Graz. They are needed for AMDA to be able to remotely connect to this datasource, according to the users search request which is explained in section 4.3.2 on page 41.

Listing C.1: The Repository description of VEX MAG

```
<Spase>
<Version>1.2.1</Version>
<Repository>
  <ResourceID>spase://iwf/repository/Vex</ResourceID>
  <ResourceHeader>
    <ResourceName>
      Space Research Institute
      (Austrian Academy of Sciences) - Graz
    </ResourceName>
    <AlternateName>IWF-OeAW</AlternateName>
    <Description>
      <!-- a brief description goes here -->
    </Description>
    <InformationURL>
      <URL>http://www.iwf.oeaw.ac.at/en/</URL>
    </InformationURL>
  </ResourceHeader>
</Repository>
</Spase>
```

Listing C.2: The Observatory description of VEX MAG

```
<Spase>
<Version>1.2.1</Version>
<Observatory>
  <ResourceID>spase://iwf/observatory/Vex</ResourceID>
  <ResourceHeader>
    <ResourceName>Venus Express</ResourceName>
```

```
      <ReleaseDate>2008-11-26T00:00:00Z</ReleaseDate>
      <Description>
        <!-- a brief description goes here -->
      </Description>
      <Contact>
        <PersonID>spase://iwf/person/zhang</PersonID>
        <Role>PrincipalInvestigator</Role>
      </Contact>
      <InformationURL>
        <URL>http://sci.esa.int/venusexpress/</URL>
      </InformationURL>
    </ResourceHeader>
    <Location>
      <ObservatoryRegion>Venus</ObservatoryRegion>
    </Location>
  </Observatory>
</Spase>
```

Listing C.3: The Person description of VEX MAG

```
<Spase>
<Version>1.2.0</Version>
<Person>
  <ResourceID>spase://iwf/person/zhang</ResourceID>
  <ReleaseDate>2008-11-26T14:40:00</ReleaseDate>
  <PersonName>Dr. Tielong Zhang</PersonName>
  <OrganizationName>
    Space Research Institute Graz
  </OrganizationName>
</Person>
</Spase>
```

Listing C.4: The Instrument description of VEX MAG

```
<Spase>
<Version>1.2.1</Version>
<Instrument>
  <ResourceID>
    spase://iwf/instrument/Vex_mag</ResourceID>
  <ResourceHeader>
    <ResourceName>
      Venus Express Magnetometer
    </ResourceName>
    <ReleaseDate>2008-11-26T14:40:00</ReleaseDate>
    <Description>
      <!-- a brief description goes here -->
    </Description>
    <Contact>
      <PersonID>spase://iwf/person/zhang</PersonID>
      <Role>PrincipalInvestigator</Role>
    </Contact>
  </ResourceHeader>
  <InstrumentType>Magnetometer</InstrumentType>
  <InvestigationName/>
  <ObservatoryID>
    spase://iwf/observatory/Vex
  </ObservatoryID>
</Instrument>
</Spase>
```

Listing C.5: The NumericalData description of VEX MAG

```
<Spase>
<Version>1.2.1</Version>
<NumericalData>
  <ResourceID>
    spase://iwf/numericalData/Vex_Mag_VSO
  </ResourceID>
  <ResourceHeader>
    <ResourceName>
      Venus Express Magnetic Field in VSO coordinates
    </ResourceName>
    <AlternateName>MAG_VSO</AlternateName>
    <ReleaseDate>2008-11-26T14:00:00</ReleaseDate>
    <Description>
      Magnetic Field in VSO coordinates
    </Description>
    <Contact>
      <PersonID>spase://iwf/person/zhang</PersonID>
      <Role>PrincipalInvestigator</Role>
    </Contact>
  </ResourceHeader>
  <AccessInformation>
    <RepositoryID>
      spase://iwf/repository/Vex
    </RepositoryID>
    <AccessURL>
      <URL/>
    </AccessURL>
    <Format>Text</Format>
    <Encoding>ASCII</Encoding>
  </AccessInformation>
```

```
<ProviderProcessingLevel>
  calibrated
</ProviderProcessingLevel>
<InstrumentID>
  spase://iwf/instrument/Vex_mag
</InstrumentID>
<MeasurementType>MagneticField</MeasurementType>
<TemporalDescription>
  <TimeSpan>
    <StartDate>2006-04-24T00:00:00</StartDate>
    <EndDate>2010-12-31T23:59:56</EndDate>
  </TimeSpan>
  <Cadence>PT4S</Cadence>
</TemporalDescription>
<ObservedRegion>Venus</ObservedRegion>
<PhysicalParameter>
  <Name>Time_UTC</Name>
  <ParameterKey>Field0</ParameterKey>
  <Description>
    Sample UTC in the form yyyy-mm-ddThh:mm:ss
  </Description>
  <Units/>
  <ValidMin>2006-01-01T00:00:00</ValidMin>
  <ValidMax>2010-12-31T23:59:56</ValidMax>
  <Support>Temporal</Support>
</PhysicalParameter>
<PhysicalParameter>
  <Name>MAG_VEX_SC</Name>
  <Description>
    Magnetic Field Vector in
    Spacecraft Coordinates
  </Description>
  <ParameterKey>Field1</ParameterKey>
  <Units>nT</Units>
  <CoordinateSystem>
    <CoordinateRepresentation>
      Cartesian
    </CoordinateRepresentation>
    <CoordinateSystemName>
      SC
    </CoordinateSystemName>
  </CoordinateSystem>
  <Structure>
    <StructureType>Vector</StructureType>
    <Size>3</Size>
```

```
        <Description/>
        <Element>
          <Name>Bx</Name>
          <Index>1</Index>
          <ParameterKey>Field1</ParameterKey>
        </Element>
        <Element>
          <Name>By</Name>
          <Index>2</Index>
          <ParameterKey>Field2</ParameterKey>
        </Element>
        <Element>
          <Name>Bz</Name>
          <Index>3</Index>
          <ParameterKey>Field3</ParameterKey>
        </Element>
      </Structure>
      <FillValue>99999.999</FillValue>
      <Measured>
        <Field>
          <FieldQuantity>Magnetic</FieldQuantity>
        </Field>
      </Measured>
    </PhysicalParameter>
    <PhysicalParameter>
      <Name>MAG_VEX_VSO</Name>
      <Description>
        Magnetic Field Vector in
        Venus Solar Orbital Coordinates
      </Description>
      <ParameterKey>Field4</ParameterKey>
      <Units>nT</Units>
      <CoordinateSystem>
        <CoordinateRepresentation>
          Cartesian
        </CoordinateRepresentation>
      </CoordinateSystem>
      <Structure>
        <StructureType>Vector</StructureType>
        <Size>3</Size>
        <Description/>
        <Element>
          <Name>Bx</Name>
          <Index>1</Index>
          <ParameterKey>Field4</ParameterKey>
        </Element>
        <Element>
          <Name>By</Name>
          <Index>2</Index>
          <ParameterKey>Field5</ParameterKey>
        </Element>
        <Element>
          <Name>Bz</Name>
          <Index>3</Index>
          <ParameterKey>Field6</ParameterKey>
        </Element>
      </Structure>
      <FillValue>99999.999</FillValue>
      <Measured>
        <Field>
          <FieldQuantity>Magnetic</FieldQuantity>
        </Field>
      </Measured>
    </PhysicalParameter>
    <PhysicalParameter>
      <Name>SC_POS_VSO</Name>
      <Description>
        Spacecraft Position in
        Venus Solar Orbital Coordinates
      </Description>
      <ParameterKey>Field7</ParameterKey>
      <Units>km</Units>
      <CoordinateSystem>
        <CoordinateRepresentation>
          Cartesian
        </CoordinateRepresentation>
      </CoordinateSystem>
      <Structure>
        <StructureType>Vector</StructureType>
        <Size>3</Size>
        <Description/>
        <Element>
          <Name>X</Name>
          <Index>1</Index>
          <ParameterKey>Field7</ParameterKey>
        </Element>
        <Element>
          <Name>Y</Name>
          <Index>2</Index>
          <ParameterKey>Field8</ParameterKey>
        </Element>
        <Element>
          <Name>Z</Name>
          <Index>3</Index>
          <ParameterKey>Field9</ParameterKey>
        </Element>
      </Structure>
      <Support>Positional</Support>
    </PhysicalParameter>
  </NumericalData>
</Spase>
```

Listing C.6: The auxiliary description of the VEX MAG service

```
<DATASET>
  <DATASET_METADATA>
    <DATASET_ID>MAG_VSO</DATASET_ID>
    <PARENT_MISSION>VEX</PARENT_MISSION>
    <PARENT_INSTRUMENT>MAG</PARENT_INSTRUMENT>
    <TIME_RESOLUTION>4.0</TIME_RESOLUTION>
    <ENCODING_TYPE>ASCII</ENCODING_TYPE>
    <START_DATE>2006-04-24T00:00:00</START_DATE>
    <STOP_DATE>2009-12-27T23:59:56</STOP_DATE>
  </DATASET_METADATA>
  <TIME_METADATA>
    <PARAM_ID>TIME_UTC</PARAM_ID>
    <PARAMETER_SHORT_DESCRIPTION>
      Coordinated Universal Time
    </PARAMETER_SHORT_DESCRIPTION>
    <FORMAT>YYYY-DDDTHH:MM:SS</FORMAT>
  </TIME_METADATA>
  <PARAM_METADATA>
    <PARAM_ID>MAG_VEX_SC</PARAM_ID>
    <PARAMETER_SHORT_DESCRIPTION>
      Magnetic field vector in Spacecraft Coordinates
    </PARAMETER_SHORT_DESCRIPTION>
    <DATA_TYPE>FLOAT</DATA_TYPE>
    <SIZES>3</SIZES>
    <TENSOR_ORDER_VALUE>1</TENSOR_ORDER_VALUE>
    <COORDINATE_SYSTEM>SC</COORDINATE_SYSTEM>
    <UNITS>nT</UNITS>
    <DISPLAY_TYPE>TIME_SERIES</DISPLAY_TYPE>
    <LABEL_I>Bx</LABEL_I>
    <LABEL_I>By</LABEL_I>
    <LABEL_I>Bz</LABEL_I>
    <FIELDNAM>MAG SC</FIELDNAM>
  </PARAM_METADATA>
  <PARAM_METADATA>
    <PARAM_ID>MAG_VEX_VSO</PARAM_ID>
    <PARAMETER_SHORT_DESCRIPTION>
      Magnetic field vector in Venus Solar Orbital ↩
         Coordinates
    </PARAMETER_SHORT_DESCRIPTION>
    <DATA_TYPE>FLOAT</DATA_TYPE>
    <SIZES>3</SIZES>
    <TENSOR_ORDER_VALUE>1</TENSOR_ORDER_VALUE>
    <COORDINATE_SYSTEM>VSO</COORDINATE_SYSTEM>
    <UNITS>nT</UNITS>
    <DISPLAY_TYPE>TIME_SERIES</DISPLAY_TYPE>
    <LABEL_I>Bx</LABEL_I>
    <LABEL_I>By</LABEL_I>
    <LABEL_I>Bz</LABEL_I>
    <FIELDNAM>MAG VSO</FIELDNAM>
  </PARAM_METADATA>
  <PARAM_METADATA>
    <PARAM_ID>SC_POS_VSO</PARAM_ID>
    <PARAMETER_SHORT_DESCRIPTION>
      Spacecraft Position in Venus Solar Orbital ↩
         Coordinates
    </PARAMETER_SHORT_DESCRIPTION>
    <DATA_TYPE>FLOAT</DATA_TYPE>
    <SIZES>3</SIZES>
    <TENSOR_ORDER_VALUE>1</TENSOR_ORDER_VALUE>
    <COORDINATE_SYSTEM>VSO</COORDINATE_SYSTEM>
    <UNITS>km</UNITS>
    <DISPLAY_TYPE>TIME_SERIES</DISPLAY_TYPE>
    <LABEL_I>X</LABEL_I>
    <LABEL_I>Y</LABEL_I>
    <LABEL_I>Z</LABEL_I>
    <FIELDNAM>SC VSO</FIELDNAM>
  </PARAM_METADATA>
</DATASET>
```

D. Venus Express Web Service descriptor in WSDL

The following Listing represents the Web Service Description Language (WSDL) file provided at the public URL `http://amda-idis.oeaw.ac.at/WSDL/vexmag.wsdl` which makes the VEX MAG Web Service accessible to all technologies supporting the SOAP protocol. An additional documentation of the SOAP server implemented in PHP can be found at `http://amda-idis.oeaw.ac.at/Doc/`. The Web service can also be easily tested with a simple PHP SOAP client which can be downloaded at `http://amda-idis.oeaw.ac.at/WSDL/VexMagWebService.zip`.

Listing D.1: The WSDL descriptor of the VEX MAG service

```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions name='vexmag'
  targetNamespace='http://amda-idis.oeaw.ac.at'
  xmlns:tns='http://amda-idis.oeaw.ac.at'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
  xmlns:xsd='http://www.w3.org/2001/XMLSchema'
  xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/↩
     '
  xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
  xmlns='http://schemas.xmlsoap.org/wsdl/'>
<message name='getAvailableDataRequest'>
</message>
<message name='getAvailableDataResponse'>
  <part name='Result' type='xsd:string'/>
</message>
<message name='getDatasetInfoUrlRequest'>
  <part name='datasetId' type='xsd:string'/>
</message>
<message name='getDatasetInfoUrlResponse'>
  <part name='Result' type='xsd:string'/>
</message>
<message name='getDatasetUrlRequest'>
  <part name='datasetId' type="xsd:string"/>
  <part name='dateStart' type="xsd:string"/>
  <part name='dateStop' type="xsd:string"/>
```

```
</message>
<message name='getDatasetUrlResponse'>
  <part name='Result' type="xsd:string"/>
</message>
<portType name='VexMagPortType'>
  <operation name='getAvailableData'>
    <output message='tns:getAvailableDataResponse'/>
  </operation>
  <operation name='getDatasetInfoUrl'>
    <input message='tns:getDatasetInfoUrlRequest'/>
    <output message='tns:getDatasetInfoUrlResponse'/>
  </operation>
  <operation name='getDatasetUrl'>
    <input message='tns:getDatasetUrlRequest'/>
    <output message='tns:getDatasetUrlResponse'/>
  </operation>
</portType>
<binding name='VexMagBinding' type='tns:VexMagPortType'>
<soap:binding style='rpc'
  transport='http://schemas.xmlsoap.org/soap/http'/>
  <operation name='getAvailableData'>
    <soap:operation soapAction='getAvailableData'/>
    <input>
      <soap:body use='encoded'
        encodingStyle='http://schemas.xmlsoap.org/soap/↩
           encoding/'/>
    </input>
    <output>
      <soap:body use='encoded'
        encodingStyle='http://schemas.xmlsoap.org/soap/↩
           encoding/'/>
    </output>
  </operation>
  <operation name='getDatasetInfoUrl'>
    <soap:operation soapAction='getDatasetInfoUrl'/>
    <input>
      <soap:body use='encoded'
        encodingStyle='http://schemas.xmlsoap.org/soap/↩
           encoding/'/>
    </input>
    <output>
      <soap:body use='encoded'
        encodingStyle='http://schemas.xmlsoap.org/soap/↩
           encoding/'/>
    </output>
  </operation>
  <operation name='getDatasetUrl'>
    <soap:operation soapAction='getDatasetUrl'/>
    <input>
      <soap:body use='encoded'
        encodingStyle='http://schemas.xmlsoap.org/soap/↩
           encoding/'/>
    </input>
    <output>
      <soap:body use='encoded'
        encodingStyle='http://schemas.xmlsoap.org/soap/↩
           encoding/'/>
    </output>
  </operation>
</binding>
<service name='VexMagWebService'>
  <port name='VexMagPort' binding='tns:VexMagBinding'>
    <soap:address
      location='http://amda-idis.oeaw.ac.at/↩
         VexMagWebServices/VexMagSoapServer.php'/>
  </port>
</service>
</definitions>
```

Glossary

Aladin	The Aladin Sky Atlas, an interactive astronomy database, developed by CDS, Strasbourg, p. 33.
API	The Application Programming Interface is a specification intended to be used as an interface by software components to communicate with each other, p. 24.
ASCII	The American Standard Code for Information Interchange, a 7-bit character encoding scheme, p. 43.
CDPP	The Centre de Données de Physique des Plasmas, a french data center for natural plasmas of the solar system, located in Toulouse, p. iii.
CDS	The Centre de Données astronomiques de Strasbourg, a french data center for astronomical data, p. 41.
DC	Direct Current, p. 37.
eXist	The eXist-db is an open source database management system built using XML technology., p. 42.
GhoSST	The Grenoble Astrophysics and Planetology Solid Spectroscopy and Thermodynamics database service, p. 32.
HTML	The Hypertext Markup Language is a markup language, designed for the creation of static web content, p. 13.
HTTP	The Hypertext Transfer Protocol, the foundation of data communication for the World Wide Web, p. 15.
IDIS-DM-WG	The IDIS Data Model Working Group is a taskforce of IDIS for developing a common IDIS data model and access protocol, p. 34.
IWF	The Institut für Weltraumforschung or Space Research Institute of the Austrian Academy of Sciences located in Graz, p. 54.

JAVA	An object-oriented programming language, originally developed by Sub Microsystems, p. 39.
JRA	Joint Research Activities of EU FP7 projects, p. 29.
JRA-IDIS	Joint Research Activity of the EuroPlaNet IDIS activity, p. 30.
MAG	The abbreviation for Magnetometer, p. iii.
NA	Networking Activities within EU FP7 projects, p. 29.
IDIS	The Integrated and Distributed Information System is a workpackage within the EuroPlanet RI, p. iii.
FP7	7th Framework Programm for Research and Technological Development, a funding programme created by the European Union, p. iii.
EuroPlaNet	The European Planetary Network, a FP7 Research Infrastructure, p. iii.
IMPEx	Integrated Medium for Planetary Exploration, a FP7 Collaborative Project, p. iii.
AMDA	The Automated Multi-Dataset Analysis Tool, a web-based scientific analysis tool developed by CDPP, Toulouse, p. iii.
NASA	National Aeronautics and Space Administration, the Space Agency of the United States, p. 1.
ESA	The European Space Agency, p. 1.
IPDA	The International Planetary Data Alliance, a community dedicated to the development of a common data access protocol, p. 1.
PDS	The Planetary Data System hosted and maintained by NASA, p. 1.
PSA	The Planetary Science Archive, hosted by ESA, p. 1.
IVOA	The International Virtual Observatory Alliance, a community dedicated to the development of a common VO, p. 2.
PDAP	The Planetary Data Access Protocol, a specialised protocol for Planetary Sciences defined by IDPA, p. 2.

SOA	Service oriented Architecture, a type of software architectur for creating and using business processes, packaged as services, p. 3.
XML-RPC	A Remote Procedure Call Protocol based on XML, p. 3.
SOAP	The Simple Object Access Protocol, an XML based messaging protocol for Web services, p. 3.
WSDL	The Web Service Description Language, an XML based describtion vocabulary for Web services, p. 3.
UDDI	Universal Description, Discovery and Integration, an XML based vocabulary for defining registries for Web services, p. 3.
TAP	The Table Access Protocol, a specialised protocol for exchange of tabular data between service endpoints, defined by the IVOA., p. 32.
SAMP	The Simple Application Messaging Protocol, an IVOA standard based on XML-RPC, p. 33.
ObsCoreDM	The Observation Core Data Model specified by IVOA which is able to describe astronomical data, p. 34.
ObsTAP	The Observation Table Access protocol, a special version of the IVOA TAP protocol, p. 34.
PHP	The PHP Hypertext Preprocessor, a server-side scripting language for web development to produce dynamic web pages, p. 39.
RI	Research Infrastructure, p. 29.
RPC	Remote Procedure Call, p. 23.
RTD	Research & Technical Development, p. iii.
SA-IDIS	Service Activity of an EuroPlaNet IDIS activity, p. 29.
SGML	The Standard Generalized Markup Language for docments, a basis for HTML and XML, p. 12.
SPASE	Space Physics Archive Search and Extract, a data model for describing and accessing heliophysics data., p. 32.
SSODnet	The Solar System Object Database Network, p. 32.

TNA	Transnational Access Activities within EU FP7 projects, p. 29.
UCD	The Unified Content Descriptors, a vocabulary for data quantities in astronomy and space physics, p. 37.
URI	The Uniform Resource Identifier, a unique identifier for an abstract or physical resource, p. 13.
URL	The Uniform Resource Locator, a special version of an URI used in the World Wide Web to address Web resources, p. 31.
VEX	Venus Express, an ESA planetary mission, p. iii.
VO	Virtual Observatory, often referred to an online data analysis tool for space sciences, p. 27.
VOTable	An XML standard developed by the IVOA for the interchange of data represented as a set of tables, p. 33.
W3C	The World Wide Web Consortium, defining Standards and Specifications for the Internet, p. 11.
XML	The eXtensible Markup Language, a set of rules for encoding documents in machine-readable format, p. 11.
XPath	A simple subset of the Xquery XML language, p. 40.
XQuery	An XML language designed for querying collections of XML data, p. 40.
XSD	The XML Schema Definition Language, which represents a set of rules an XML document must conform in order to be considered valid., p. 24.
XSLT	The Extensible Stylesheet Language Transformations, an XML language used for the transformation in other formats such as HTML, p. 33.

List of Figures

Listings

References

André, N., Cecconi, B., Budnik, E., Jacquey, C., Génot, V., Fedorov, A., … Topf, F. (2009, November). Connecting the CDPP/AMDA service to planetary plasma data: Venus, Earth, Mars, Saturn (Jupiter and comets). In *Proceedings SF2A 2009* (p. 231). Besançon, France. Retrieved from http://hal.archives-ouvertes.fr/hal-00442838/PDF/sf2a_09_andre.pdf

André, N., et al. (2011, November). HST Auroral Campaign Observations of Jupiter and Saturn enabled by the CDPP/AMDA and IVOA/Aladin tools. In *Proceedings of the PV 2011 Conference, 2011, Toulouse, France.* Retrieved from http://typhon.obspm.fr/idis/docs/Andre_PV2011.pdf

Arviset, C., Gaudet, S., & the IVOA Technical Coordination Group. (2010, November). *IVOA Architecture Version 1.0* (Note). IVOA. Retrieved from http://www.ivoa.net/Documents/Notes/IVOAArchitecture/

Bellwood, T., et al. (2002, July). *UDDI: Version 2.04 API Specification* (Specification). OASIS. Retrieved from http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm

Blanc, M., et al. (2008, December). *EuroPlaNet Research Infrastructure ANNEX I - Description of Work.* Seventh Framework Programm Capacities Research Infrastructure - Proposal Grant Aggreement No. 228319. (Project Summary: http://cordis.europa.eu/fetch?CALLER=FP7_PROJ_EN&ACTION=D&DOC=1&CAT=PROJ&QUERY=0130ad0c101f:b682:561aca31&RCN=92505)

Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J., & Siméon, J. (2010, December). *XQuery 1.0: An XML Query Language (Second Edition)* (W3C Recommendation). W3C. Retrieved from http://www.w3.org/TR/xquery/

Booth, D., Haas, H., McCabe, F., , Newcomer, E., Champion, M., … Orchard, D. (2004, February). *Web Services Architecture* (W3C Working Group Note). W3C. Retrieved from http://www.w3.org/TR/ws-arch/

Capria, M. T., Chanteur, G., & Schmidt, W. (2009, December). Europlanet Integrated and Distributed Information System (IDIS). In *Proceedings of the PV 2009 Conference, 2009, Madrid, Spain.* Retrieved from http://www.sciops.esa.int/SYS/CONFERENCE/include/pv2009/papers/40_Capria_EuroPlanet-IDIS.pdf

Cecconi, B., et al. (2011, February). *Interoperable Data Access Data Model: Initial Prototype for Plasmas* (Specification). EuroPlaNet. Retrieved from http://typhon.obspm.fr/idis/docs/Data_Model_v118a.pdf

Cecconi, B., & the IDIS-DM-SWG. (2011, November). Europlanet-IDIS Data model: a Data Model for a Planetology Virtual Observatory. In *Proceedings of the PV 2011*

Conference, 2011, Toulouse, France. Retrieved from `http://typhon.obspm.fr/idis/docs/Cecconi_PV2011.pdf`

Cerami, E. (2002). *Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI & WSDL* (First ed.). O'Reilly Media.

Chinnici, R., et al. (2007, June). *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language* (W3C Recommendation). W3C. Retrieved from `http://www.w3.org/TR/wsdl20/`

Christensen, E., et al. (2001, March). *Web Services Description Language (WSDL) 1.1* (W3C Note). W3C. Retrieved from `http://www.w3.org/TR/wsdl`

Erard, S., et al. (2011a, February). *PDAP assessment study for use in EuroPlaNet-IDIS VO system* (Draft). EuroPlaNet.

Erard, S., et al. (2011b, February). *Proposed architecture for EuroPlaNet-IDIS Virtual Observatory* (Draft). EuroPlaNet.

Erard, S., et al. (2011c, October). *Proposed architecture for EuroPlaNet-IDIS Virtual Observatory.* Talk at the session MT1: e-infrastructures for planetary sciences at the European Planetary Science Congress 2011, Nantes, France. Retrieved from `http://typhon.obspm.fr/idis/docs/Erard_EPSC2011.pdf`

Erard, S., et al. (2011d, November). Towards a Planetary VO: EuroPlaNet-IDIS Activities in VO-Paris. In *Proceedings of the PV 2011 Conference, 2011, Toulouse, France.* Retrieved from `http://typhon.obspm.fr/idis/docs/Erard_PV2011.pdf`

Erl, T. (2004). *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services* (First ed.). Prentice Hall PTR.

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology and Design* (First ed.). Prentice Hall PTR.

Erl, T. (2007). *SOA Principles of Service Design* (First ed.). Prentice Hall PTR.

ESA. (2011, January). *The Planetary Science Archive.* Retrieved November 1, 2011, from `http://www.rssd.esa.int/index.php?project=PSA&page=introduction`

EuroPlaNet. (2012, January). *EuroPlaNet-RI Portal.* Retrieved January 12, 2012, from `http://www.europlanet-ri.eu/`

EuroPlaNet IDIS Management. (2010, October). *EuroPlaNet IDIS Leaflet.* Retrieved January 12, 2012, from `http://www.europlanet-idis.fi/documents/public_documents/Publications/IDIS_leaflet.pdf`

EURO-VO. (2010, May). *European Virtual Observatory.* Retrieved November 1, 2011, from `http://www.euro-vo.org/pub/general/intro.html`

Fensel, D., et al. (2007). *Enabling Semantic Web Services* (First ed.). Springer, Berlin.

Fielding, R. T., et al. (1999, June). *Hypertext Transfer Protocol – HTTP/1.1* (RFC No. 2616). Internet Engineering Task Force (IETF). Retrieved from `http://www.ietf.org/rfc/rfc2616.txt`

Gangloff, M., et al. (2009, December). An interoperable architecture using the CDPP/AMDA service and the SPASE Model. In *Proceedings of the PV 2009 Conference, 2009, Madrid, Spain.* Retrieved from `http://europlanet-plasmanode.oeaw.ac.at/fileadmin/documents/publications/18_Gangloff_interoperable_architecture_w_Amda_SpaseProceedings.pdf`

Governor, J., et al. (2009). *Web 2.0 Architectures* (First ed.). O'Reilly Media & Adobe Dev Library.

Gudgin, M., et al. (2007, April). *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)* (W3C Recommendation). W3C. Retrieved from `http://www.w3.org/TR/soap12-part1/`

Hadley, J. M. (2009, February). *Web Application Description Language (WADL)* (Specification). Sun Microsystems. Retrieved from `http://wadl.java.net/wadl20090202.pdf`

Hass, H., & Brown, A. (2004, November). *Web Services Glossary* (W3C Working Group Note). W3C. Retrieved from `http://www.w3.org/TR/ws-gloss/`

IPDA. (2011, June). *International Planetary Data Alliance.* Retrieved November 1, 2011, from `http://planetarydata.org/`

IVOA. (2010, May). *International Virtual Observatory Alliance.* Retrieved November 1, 2011, from `http://www.ivoa.net/pub/info/`

Jacobs, I., & Walsh, N. (2004, December). *Architecture of the World Wide Web, Volume One* (W3C Recommendation). W3C. Retrieved from `http://www.w3.org/TR/webarch/`

Jacquey, C., Génot, V., Budnik, E., Hitier, R., Bouchemit, M., Gangloff, M., ... Pinçon, J. L. (2010). AMDA, Automated Multi-Dataset Analysis: A Web-Based Service Provided by the CDPP. In Laakso, H., Taylor, M. & Escoubet, C. P. (Ed.), *The Cluster Active Archive, Studying the Earth's Space Plasma Environment* (p. 239-247). Springer, Berlin. Retrieved from `http://hal.archives-ouvertes.fr/docs/00/45/54/56/PDF/CAABook_Amda_jacquey.pdf` (Abstract: `http://adsabs.harvard.edu/abs/2010caa..book..239J`) doi: 10.1007/978-90-481-3499-1_16

Khalaf, R., & Leymann, F. (2003). On Web Services Aggregation. In *Technologies for E-Services. Lecture Notes in Computer Sciences 2819* (pp. 1 – 13). Springer, Berlin. Retrieved from `http://www.research.ibm.com/people/r/rkhalaf/aggr.pdf`

Khodachenko, M., et al. (2011, January). *IMPEx Collaborative Project ANNEX I - Description of Work.* Seventh Framework Programm Capacities Collaborative Project - Proposal Grant Aggreement No. 262863. (Project Summary: `http://cordis.europa.eu/fetch?CALLER=FP7_PROJ_EN&ACTION=D&DOC=1&CAT=PROJ&QUERY=01309ac26125:224b:4536ecf7&RCN=99058`)

Laurent, S., Johnston, J., & Dumbill, E. (2001). *Programming Web Services with XML-RPC* (First ed.). O'Reilly Media.

Layman, A., Hollander, D., & Bray, T. (2009, December). *Namespaces in XML* (W3C Recommendation). W3C. Retrieved from `http://www.w3.org/TR/REC-xml-names/`

Louys, M., et al. (2011, October). *Observation Data Model Core Components and its Implementation in the Table Access Protocol Version 1.0* (Proposed Recommendation). IVOA. Retrieved from `http://www.ivoa.net/Documents/ObsCore/`

Maler, E., Cowan, J., Paoli, J., Yergeau, F., Bray, T., & Sperberg-McQueen, C. M. (2006, August). *Extensible Markup Language (XML) 1.1 (Second Edition)* (W3C Recommendation). W3C. Retrieved from `http://www.w3.org/TR/xml11`

Martinez, A. P., et al. (2007, April). *The UCD1+ controlled vocabulary Version 1.23* (Recommendation). IVOA. Retrieved from http://www.ivoa.net/Documents/latest/UCDlist.html

Meier, W. M. (2011, September). *eXist-db Open Source Native XML Database.* Retrieved November 1, 2011, from http://exist.sourceforge.net/

Mitra, N., & Lafon, Y. (2007, April). *SOAP Version 1.2 Part 0: Primer (Second Edition)* (Tech. Rep.). W3C. Retrieved from http://www.w3.org/TR/soap12-part0/

NASA. (2011, June). *The Planetary Data System.* Retrieved November 1, 2011, from http://pds.nasa.gov/about/about.shtml

Newcomer, E. (2002). *Understanding Web Services: XML, WSDL, SOAP and UDDI* (First ed.). Addison-Wesley.

Ochsenbein, F., & Williams, R. (2009, November). *VOTable Format Definition* (Recommendation). IVOA. Retrieved from http://www.ivoa.net/Documents/VOTable/

Richards, R. (2006). *Pro PHP XML and Web Services* (First ed.). Apress.

Rosen, M., et al. (2008). *Applied SOA: Service-Oriented Architecture and Design Strategies* (First ed.). Wiley Publishing, Inc.

Rye, E., & the PDS Object Review Commitee. (2008, October). *Planetary Science Data Dictionary Document* (Dictionary). Jet Propulsion Laboratory - California Institute of Technology. Retrieved from http://pds.nasa.gov/documents/psdd/PSDDmain_1r71.pdf (Interactive Dictionary Lookup: http://pds.nasa.gov/tools/ddlookup/data_dictionary_lookup.cfm)

Salgado, J., et al. (2009, December). IPDA Planetary Data Access Protocol (PDAP): an effort to share planetary scientific data. In *Proceedings of the PV 2009 Conference, 2009, Madrid, Spain.* Retrieved from http://www.sciops.esa.int/SYS/CONFERENCE/include/pv2009/papers/41_Salgado_PlanetaryDataAccessProtocol.pdf

Schaaff, A., & Graham, M. (2010, December). *IVOA Web Services Basic Profile Version 1.0* (Recommendation). IVOA GWS. Retrieved from http://www.ivoa.net/Documents/WSBasicProfile/

Schmidt, W., Capria, M. T., & Chanteur, G. (2009, April). *Europlanet Integrated and Distributed Information System (IDIS).* Talk at the session GI4: General System Design, Image Processing and Data Infrastructures at the EGU General Assembly 2009, Vienna, Austria. (Abstract: http://meetingorganizer.copernicus.org/EGU2009/EGU2009-9366.pdf)

Skonnard, A., & Gudgin, M. (2001). *Essential XML Quick Reference: A Programmers Reference to XML, XPath, XSLT, XML Schema, SOAP, and More* (First ed.). Addison-Wesley.

Studer, R., Grimm, S., & Abecker, A. (2007). *Semantic Web Services: Concepts, Technologies and Applications* (First ed.). Springer, Berlin.

Taylor, M., et al. (2011, September). *SAMP - Simple Application Messaging Protocol Version 1.3* (Proposed Recommendation). IVOA. Retrieved from http://www.ivoa.net/Documents/SAMP/

The Dublin Core Metadata Initiative. (2012, January). *Dublin Core.* Retrieved January

21, 2012, from http://dublincore.org/about-us/

The IVOA Data Model Working Group. (2008, March). *Data Model for Astronomical DataSet Characterization* (Recommendation). IVOA. Retrieved from http://www.ivoa.net/Documents/latest/CharacterisationDM.html

The IVOA Resource Registry Working Group, et al. (2007, March). *Resource Metadata for the Virtual Observatory Version 1.12* (Recommendation). IVOA. Retrieved from http://www.ivoa.net/Documents/latest/RM.html

The SPASE Consortium. (2011, October). *A Space and Solar Physics Data Model* (Specification). SPASE. Retrieved from http://www.spase-group.org/docs/dictionary/spase-2_2_2.pdf

Tidwell, D., Snell, J., & Kulchenko, P. (2001). *Programming Web Services with SOAP* (First ed.). O'Reilly Media.

Topf, F., et al. (2011a, February). *Query a distant database using a SPASE-based connector.* Talk at the EuroPlaNet IDIS General Meeting 2011, Paris, France. Retrieved from http://europlanet-plasmanode.oeaw.ac.at/fileadmin/documents/publications/JRA-IDIS-Presentation_FT_2011.pdf

Topf, F., et al. (2011b, October). *SOAP based web services and their future role in VO projects.* Talk at the session MT1: e-infrastructures for planetary sciences at the European Planetary Science Congress 2011, Nantes, France. Retrieved from http://europlanet-plasmanode.oeaw.ac.at/fileadmin/documents/publications/Topf_EPSC2011.pdf (Abstract: http://meetingorganizer.copernicus.org/EPSC-DPS2011/EPSC-DPS2011-205-2.pdf)

Walmsley, P., & Fallside, D. C. (2004, October). *XML Schema Part 0: Primer Second Edition* (W3C Recommendation). W3C. Retrieved from http://www.w3.org/TR/xmlschema-0/

Weerawarana, S., et al. (2005). *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More* (First ed.). Prentice Hall PTR.

Winer, D. (1999, June). *XML-RPC Specification* (Specification). UserLand Software. Retrieved from http://www.xmlrpc.com/spec

Index