Selection and Verification of Workflows in Multimedia Service Compositions

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor-Ingenieur
(abgekürzt: Dr.-Ing.)

genehmigte Dissertation

von **Dipl.-Ing. Ingo Brunkhorst**

geboren am **01.09.1974** in **Hameln**

2010

Referent:	Prof. Dr. Wolfgang Nejdl
Ko-Referenten:	Prof. Dr. Jörn Ostermann
	Prof. Dr. Wolf-Tilo Balke
Tag der Promotion	11.08.2009

Acknowledgments

First and foremost, I would like to thank my advisor Prof. Dr. Wolfgang Nejdl. He introduced me to methodical research and provided an excellent working environment. He gave me the freedom to pursue my research goals, and in short, without his support, this thesis would not have been possible.

I also would like to thank Prof. Dr. Wolf-Tilo Balke for always being the source of encouraging advice and helpful comments.

I am grateful to my colleagues at the University of Torino, Dr. Viviana Patti, Dr. Cristina Baroglio and Dr. Matteo Baldoni. Our collaboration greatly shaped my understanding of not only protocol verification.

The collaboration and discussions with my colleagues at the L3S Research Center was an indispensable source of information. What is more important, our joint work was always a pleasure and I would like to thank all of my colleagues for their cooperation and openness.

Kurzfassung

Selection and Verification of Workflows in Multimedia Service Compositions

in

Deutsch

Große Multimediaapplikation werden typischerweise als monolithische Systeme realisiert worden. Auch die inzwischen weite Verbreitung von Service-orientierten Architekturen hat letztendlich nichts daran ändern können. Hauptgrund dafür ist die Art, in der die digitalen multimedialen Objekte transportiert werden. Kontinuierliche Datenströme, wie sie bei der Übertragung von Filmen und Musik enstehen, lassen sich nicht mit den üblichen, auf kurze Dialoge ausgelegten Web-Services bearbeiten.

In dieser Arbeit wird die Erstellung von Service-Kompositionen für die Personalisierung von Multimediadaten im Rahmen des PUMA Systems beschrieben. Die PUMA Architektur ermöglicht die Entwicklung von Webdiensten zur Adaption kontinuierlicher Datenströme.

Diese Anpassung der Inhalte ist nötig, denn oft liegen die Objekte in den unterschiedlichsten Formaten vor, und auch für die Wiedergabe von Multimediainhalten sind eine Vielzahl an unterschiedlicher Endgeräte im Einsatz. Eine Eigenschaft vieler Multimedia-Adaptionssysteme ist die Modellierung des Adaptionsprozesses als Workflow. Wird ein solcher Workflow mit Hilfe einer Service-orientierten Architektur realisiert, entsteht eine Servicekomposition, in die für jeden der Schritte eines Workflows ein eigener spezieller Service integriert wird.

Am Beispiel des PUMA Systems wird gezeigt, wie durch den Einsatz von Semantic Web Technologie die Kommunikation der Services mit ihren jeweiligen Gesprächspartnern genau spezifiert werden kann. Damit Dienste miteinander kommunizieren können, müssen sie die vorgegebenen Protokolle einhalten. Durch einen “conformance test”, basierend auf Zustandsautomaten, wird sichergestellt, das nur kompatible Dienste ausgewählt werden.

Mit Hilfe einer prototypischen Implementierung wird gezeigt, daß Multimedia-Adaptation mit einer Service-orientierten Architektur realisiert werden kann. Zusätzlich wird das Zeitverhalten des Systems untersucht, wenn ein ausgefallener Service durch einen neuen ersetzt werden muß.

Stichworte: Semantische Web Services, Servicekomposition, Zustandsautomaten

Abstract

Selection and Verification of Workflows in Multimedia Service Compositions

in

English

In the past, large scale multimedia applications have been mostly realized as monolithic systems. Although the Service-oriented Architectures have been accepted as an efficient way to implement large distributed applications, they have not yet been widely used for multimedia systems. One of the reasons for this is the nature of the information involved with digital multimedia items, which usually are transported as a continuous stream of data, which normally can not be processed by currently existing Web Service frameworks.

This work describes the selection and verification of service compositions for Multimedia Adaptation in the context of the PUMA system for personalized universal multimedia access.

Personalizing content is necessary to make the different types of digital items available for the plethora of different client devices currently in use. Just as in similar Digital Item Adaptation systems, a workflow is the starting point of any sequence of processing steps, which is applied to multimedia objects. For each of the tasks specified in the workflows, the system selects adaptation services which will perform the content transformation. The main focus of this thesis is the selection of services such that they match the requirements of the roles in the adaptation workflows, and the verification of the interoperability between services w.r.t to the required communication protocol.

Based on the evaluation of a prototypical implementation, it will be shown that service oriented multimedia adaptation is possible. Furthermore, even for workflows which combine interoperable services for processing the digital items, startup times are shorter than comparable P2P approaches, while guaranteeing that failing services will be replaced fast enough for an uninterrupted user experience.

Keywords: Semantic Web Services, Service Composition, Verification

Contents

Glossary

BPEL	Business Process Execution Language
BPEL4WS	BPEL for Web services[6]
BPEL4WS	Business Process Execution Language for Web Services
BPMN	Business Process Modelling Notation
CFG	Context Free Grammers
CHR	Constraint Handling Rules
CLP	Constraint Logic Programming
DCG	Definite Clause Grammars, extension of CFG
DIA	Digital Item Adaptation
DMCI	Dublin Core Metadata Initiative
DNS	Domain Name Service, the address resolution protocol service of the Internet
DOM	Degree of Match between IOPE attributes of Service Templates and Service Descriptions
DSTP	Data Space Transfer Protocol
FIPA	Foundation for Intelligent Physical Agents [69]
FSA	Finite state automaton (plural: automata), sometimes also called finite state machine
GADS	Grid Access Data Service
I18N	Abbr. for *Internationalization*: "I" + 18 characters + "N"
IOPE	A description consisting of: Input, Output, Preconditions, Effects
IOPR	A tuple of Inputs, Outputs, Preconditions and Results, see IOPE
IP	The Internet Protocol
IPTV	Internet Protocol Television
IRI	Internationalized Resource Identifier [59]
ITIL	IT Infrastructure Library
KQML	Knowledge Query and Manipulation Language
MPEG	Moving Picture Experts Group
N3	Notation 3: A readable RDF syntax [30]
OASIS	Organization for the Advancement of Structured Information Standards

OWL ..... Ontology Web Language [23]
OWL-S ..... Semantic Markup for Web Services (see OWL)
OWL-S ..... Semantic Markup for Web Services [114]
QEL ..... Query Exchange Language of the Edutella System [129]
QoS ..... Quality of Service
RDF ..... Resource Description Framework [24]
RDFS ..... RDF Schema [43]
RDQL ..... A RDF Query Language, implemented in the Jena RDF Framework [144]
RIF ..... Rule Interchange Format [36]
RQL ..... A functional RDF Query Language and predecessor of SPARQL
SeRQL ..... RQL-like implementation of the Sesame Database [45]
SIC ..... Social Integrity Constraints
SLA ..... Service Level Agreements
SOA ..... Service-oriented Architecture
SPARQL ..... The official Query language of the Semantic Web [135]
UM ..... User Model
UP ..... User Preferences
URI ..... Uniform Resource Identifier [29]
VLC ..... VideoLAN Client [150], an open-source multimedia player application
W3C ..... World Wide Web Consortium
WfMC ..... Workflow Management Coalition (see `http://www.wfmc.org/`)
WFMS ..... Workflow Management Systems
WS-BPEL ..... Web Service Business Process Execution Language [99], formerly known as BPEL4WS
WS-BPEL ..... Web Service Business Process Execution Language (see BPEL4WS)
WS-CDL ..... Web Service Choreography Description Language
WSD ..... Web Service Description
WSDL ..... Web Service Description Language
WSFL ..... Web Service Flow Language
WSLA ..... Web Service Level Agreements
WSMO ..... Web Service Modelling Ontology
WSMX ..... Implementation of a service composition engine based on WSMO
XPDL ..... XML Process Definition Language
YAWL ..... Yet Another Workflow Language

Chapter 1

Introduction

In the recent years, building large scale applications —and applications in general— as monolithic architectures has been replaced with a more modular approach. For many of today's large scale applications, specifically —but not limited to— web applications, the functionality is realized as atomic operations distributed over the network via so called *Web Services*. From the business perspective, e.g. online money transactions with credit cards, buying and selling of goods and services, and collection and exchange of customer data. The use of Service-oriented Architectures has actually spawned a new industry. Travel agencies use Web Services to inquire about flights and to book them, using the Web Services which are in turn provided by all the major airlines. Online sellers such as Amazon and online auction houses such as Ebay, use the Web Service interface of banking institutions to validate and charge credit cards and perform transactions for payment. There are many emerging standards for these online marketplaces, which are called, depending on the participants, B2B - business-to-business, B2C - business-to-consumer or even B2G - business-to-government. Of course, the use of Web Services is not limited to business processes, although they are the driving forces behind the emerging standards such as WS-BPEL, the Web Service Business Process Execution Language.

A particular benefit of Web Services is their capacity for enabling interoperability in heterogeneous environments, such as the exchange of information over the Internet independent of the implementation language, operating systems and hardware. This makes Web Services ideally suited for implementing applications on the Web, i.e. information portals, collaborative and community-based web applications, digital library services and distributed search and retrieval engines. Additionally, Web Services play an important part in the realization of Grid Computing [68], which provides reliable, consistent, accessible and cost effective access to the processing power of supercomputers.

There is however, a domain which has not widely adopted Service-oriented computing for their applications yet, namely *Large Scale Multimedia Systems.*

A *Multimedia System* is defined as "the use of a computer system to create, manipulate, present, store and distribute information, which is encoded in at least one continuous and one discrete media" [149].

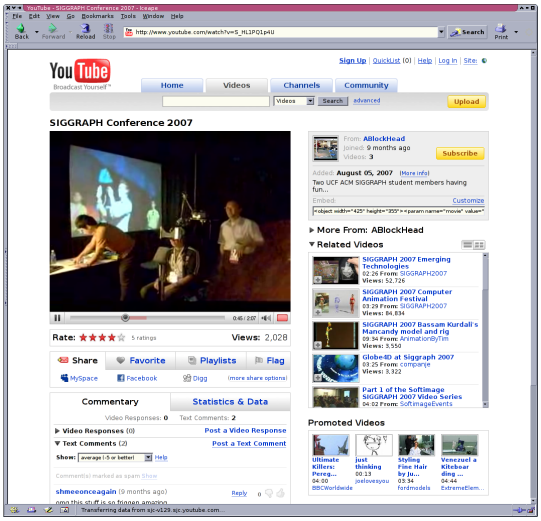

Fig. 1.1: YouTube Portal

Fig. 1.2: Google Video Portal

In the literal sense, “multi-media” refers to the concurrent usage of different media for presentation, i.e. learning material which combines and synchronizes textual information on slides with a video and audio recording of the presentation. With the increasing performance of computers and interfaces, the use of multimedia started to grow rapidly, and interactivity has become a central part of this new media. As a result, besides entertainment, one important use of multimedia is in the pedagogic domain, where the different types of media should improve the learning process by addressing different senses [165].

Simplified, a multimedia object can include a combination of text, audio, still images, animations, video, and other interactive content forms. There exist a number of conferences and workshops, i.e. the ACM Multimedia (MM) and IEEE ICME conference, which are dedicated to this topic.

A large scale multimedia application is a system that attempts to store a large amount of content objects, as well as scale to support a very large number of users. However, as stated earlier, many of the systems currently in use are monolithic systems. The main reason for not using Service-oriented architectures is the type of data associated with multimedia processing. The information exchanged between multimedia systems usually consists of continuous data streams dependent in time and space. This makes the application of existing implementations of Web Service systems nearly impossible. For the same reasons, Peer-to-Peer systems are also not prevalently used for multimedia systems. There are some notable exceptions, where the P2P model is successfully adopted, thereby overcoming some limiting aspect of traditional multimedia systems.

Currently, the most prominent example of multimedia delivery systems are web-based video portals such as YouTube [171], Google Video [90], Yahoo Video [92] (Figure 1.1, 1.2, and 1.3), and P2P systems, such as PPLive [91] (Figure 1.4).

In these scenarios, a P2P infrastructure would have the additional benefit of being able to distribute content quickly. The drawback however, is that the content is actually distributed across the network and

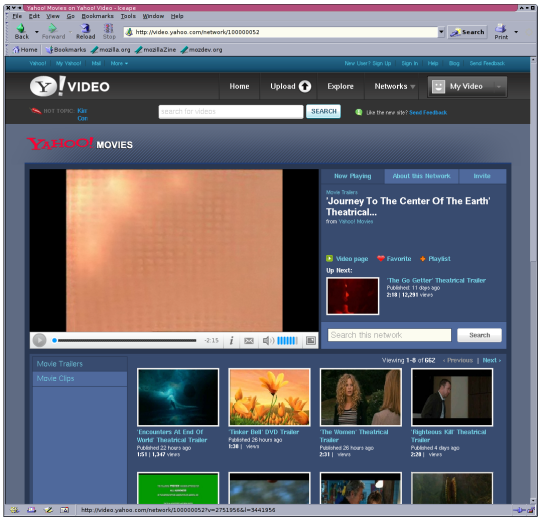

Fig. 1.3: Yahoo Video Portal

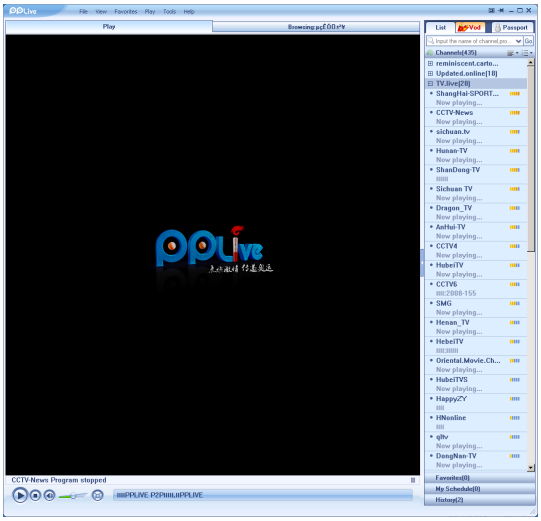

Fig. 1.4: PPLive P2P Video Application

leaves the control of the original author and owner. Moreover, such a system is resilient against individual content provider failures, since the movies, or parts of it, are available from different participants.

Monolithic systems can be scalable, but from a developers point of view, they do not offer the needed adaptability and re-usability to keep up with the advancements in technology. One impact of such limitations is that these portals require a proprietary viewing application and only support very few or even just one single video format. One of the few successful peer-to-peer applications for streaming multimedia content is PPLive [91, 110] (Figure 1.4). PPLive's focus is on the Asian market, with the majority of the content available only in Chinese. Lately, many television networks in Europe started to publish their own streaming services, or allow access to their content library for a limited time. For example, arte [71] allows access to content for up to 7 days after the initial broadcast, while German public television networks such as ARD/ZDF [134, 63] provide their live streams over IPTV. Although there exists a complex payment scheme for those government-coordinated networks, there has been cooperation with companies to create client software for watching these channels over the Internet. Such an example would be zattoo [96], which takes into account the location of the viewer for unlocking those channels which are available in the geographical region of the user.

1.1 Problem Statement and Outline

Distributed Systems and Service-oriented Architectures have been the solution for many application domains, but the implementations that currently exist lack the support for many of the processing steps required in the multimedia domain, starting with the support for continuous data streams and ending with reliable Quality-of-Service mechanisms.

Based on the requirements of large scale multimedia application and the need for a Service-oriented approach to overcome the limitations of monolithic systems, we designed the PUMA architecture.

Motivating Scenario Consider the following use-case (Figure 1.5), where it is the user's goal, to watch a movie. The user's name is "Mira" and she is from Finland. The movie however, is only available in English language, therefore she would like to have Finish subtitles in the movie. Mira's situation is complicated by the fact that she currently only has a pocket computer on which to watch the movies.

For many of the systems currently in existence, delivering a video to a client device is not much of a problem. However, the available systems do not provide the necessary functionality for Mira's requirements, which is the adaptation of the content for her less powerful client device and the integration of subtitles in the movie.

The content provider is unable, or unwilling, to do so, and does not provide a stream which is already in the required format. Modifying the size, color-depth, encoding, modality, and thereby varying the bandwidth and processing requirements of multimedia objects, is called *Digital Item Adaptation*.

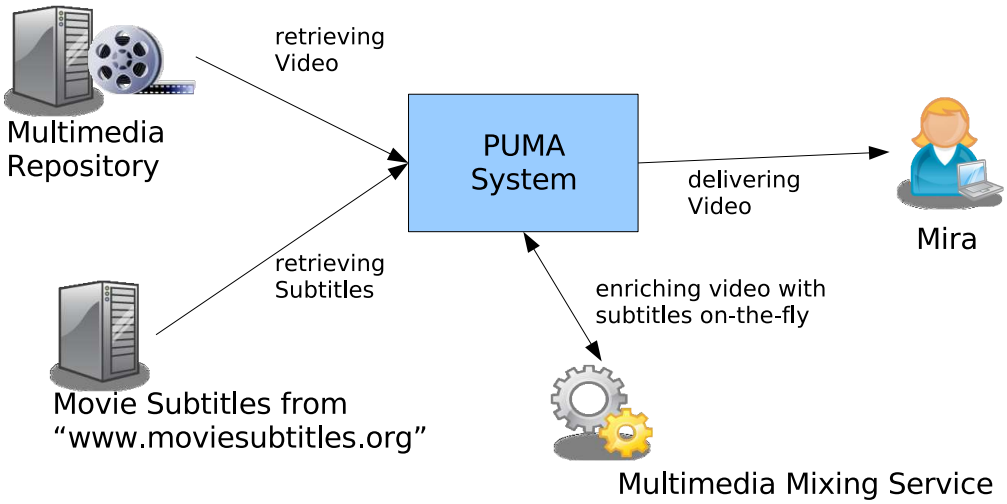

Fig. 1.5: PUMA Scenario 1: Enriching Video on-the-fly with Subtitles

In many of the traditional systems, the delivered content has to be physically available as a file stored on the content provider's system, and will be broadcasted without any on-the-fly modification. Currently, the most common form of embedding subtitles is by using a software plugin on the client system, which takes subtitle information downloaded from a different location, provided by either the publisher of the original content, or created by volunteers and made available for free.

Challenges In traditional systems, some types of adaptation are possible, but the existing monolithic systems are not flexible enough to support the complex *Adaptation Workflows* required for Digital Item Adaptation.

For this scenario, the main focus is the creation of *adapted* content on-the-fly in a distributed manner, without the need for storing semi-finished products. There are additional benefits of creating the content in this way. Copyright violations are less of an issue when the content is delivered using this method, because the involved participants only change the content and do not copy and re-distribute it to others. Additionally, this scenario enables new business models, e.g. the adaptation can be offered by third-party providers, which can charge for their services to a market which is neglected by the content authors.

The second major problem faced in this scenario stems from the dynamic nature of the network. Transmitting data over a wireless media is not much of a problem nowadays, but it is a problem if the network's quality is only measured at the beginning of a session, which possibly will last for hours. Of course, there exist systems which can up- and downgrade between streams of different quality, but again, in many architectures, this is only possible between those streams that have been pre-recorded and stored on the servers. In many cases, this kind of adaptation only switches between a "high" and a "low" quality stream.

Finally, the properties of a network connection will vary greatly over time, and location, and therefore need to be monitored closely for changes so that the playback of content can be adapted. The goal here is to provide an uninterrupted video experience, which requires dynamically changing the adaptation workflow, rather than simply aborting the process and restarting it with different parameters.

Furthermore, it is also important to monitor other states of the user's device, i.e. the battery status. Instead of missing the last minutes of an important video session due to battery failure, it might be more advisable to simply increase battery life by reducing the quality of the content, thereby reducing the CPU load and network utilization.

Digital Item Adaptation requires the creation of workflows. These workflows specify the sequence of adaptation tasks which turn an original multimedia resource into the desired target format.

In the course of this work we will show, how service compositions, which are required for the adaptation of content, will be selected, and verified, in the context of the PUMA system. The PUMA system was designed as a distributed multimedia system in which the content adaptation is performed by Web Services.

Such Service-oriented Architectures (SOA) can help in the creation of extendable content adaptation systems, in which the multimedia data is transformed as it passes through the network to the client. However, for such a system to work correctly, the workflows which control the adaptation, and the Services which perform the content processing, have to be selected carefully, and interoperability between all the involved components, a known problem of distributed heterogeneous systems, has to be ensured.

The research presented in this work is focussed on selecting such adaptation Web Services according to the roles described in such an adaptation workflow, using semantically enriched Web Service descriptions. Furthermore, one important part of creating service compositions from these workflows in a heterogeneous environment is to verify that the selected Services are able to communicate with each other. Traditional interoperability verification requires a complete view of all the participants in the Choreography, whereas the presented verification only requires that a Service implementation is conformant with

the role it should play in a global interoperable protocol. This verification is done *a-priori*, i.e. before the service composition is executed.

For this, service descriptions have to include a so called Process Model, which is a description of the behavior of a service. This model is extended to allow specification of the possible conversations supported by a Semantic Web Service. The traditional Web Service description is not sufficient to model complex protocols, because it only supports atomic service invocations, i.e. request-response dialogs.

Outline

- Chapter 2 starts with a description of the technologies which enable the description and processing of multimedia resources, as well as architectures which can be employed for creating distributed, large-scale applications.

- A brief overview of the PUMA system, the goals, and designed components is given in Chapter 3. This includes a list of requirements which led to the current design of the PUMA system, as well as an overview of PUMA's method of service composition.

- Chapter 4 is dedicated to the validation of interoperability between Web Services in a service composition, based on their capability to create legal conversations. In order to work, the participants in such a composition have to exchange messages in a predefined way, which is usually dictated by a global interaction protocol. Based on a semantic description of a service's behavior, the conformance w.r.t the protocol can be verified even before the execution of the composition.

- An overview of the implementation of the PUMA system is given in Chapter5. This includes a description of the different components of PUMA and highlights the parts which are essential for service selection and validation.

  Furthermore, discussed in this chapter are the evaluation results of the first runs of the prototype, which show that not only adaptation is possible in a Service-oriented way, but that the system is fast enough to cope with failing services and changes in the environment.

- At last, Chapter 6 gives a summary and outlook.

Chapter 2

Technologies for Building Distributed Large Scale Applications

This chapter discusses the required technologies and architectures for building distributed large scale multimedia applications. In Section 2.1, the *enabling technologies* (see Fig. 2.1) for creating metadata and modelling workflows are presented. Multimedia systems use metadata to describe the properties of digital items and to model the behavior and capabilities of Web Services. Workflows are used to describe sequences of individual tasks, but also support many other process structures, such as loops or parallel execution. In multimedia systems, Workflows are used to describe the tasks necessary to convert an original digital item into the desired target format.

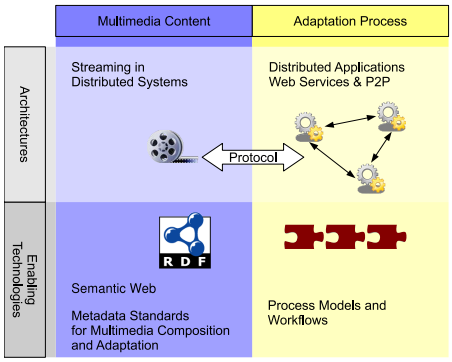

Fig. 2.1: Building Blocks of Large Scale Multimedia Architectures

Section 2.2 comprises of a description of architectures for creating large distributed systems, specifically Service-oriented Architectures and peer-to-peer systems. The capability of managing continuous data streams, required for the on-the-fly adaptation of content, is a required property of such a distributed systems.

2.1 Enabling Technologies for Large Scale Multimedia Applications

In heterogeneous environments it is important that the participants agree on the syntax and semantics for describing the properties of digital resources. The Semantic Web provides exactly this, the possibility to define concepts and vocabularies to name and describe the properties of objects on the Web, and help machines to reason about the meaning of these descriptions (see Section 2.1.1).

For some types of resources, the research community and business partners have already started to create standards to annotate items. For example, the Dublin Core Metadata Initiative (DCMI) created a vocabulary for exchanging information about titles, authors and subjects of generic documents. The Moving Picture Experts Group (MPEG) published standards for describing the composition of complex multimedia objects, for modelling the different processes for adapting digital items, and for managing rights (see Section 2.1.3).

The process of adapting a digital item needs to be carefully planned, to allow for different adaptation goals. Whether a movie is first scaled down, and then encoded in a different format, or the other way around, can have great impact on the quality of the result. Workflows are used in many systems for modelling such complex processes, and Digital Item Adaptation in multimedia systems is no exception (see Section 2.1.5).

2.1.1 Semantic Web

In order to create large distributed applications, the parts which are spread across different systems have to use an interoperable manner for describing the resources, either data or functionality, that they provide. While there exist protocols to allow the open exchange of information in heterogeneous systems, the real problem is to access the data in an *automated* fashion, as this type of descriptions are usually authored by humans. Using the technologies provided by the Semantic Web tries to solve this problem. This vision of the Semantic Web, was formulated by Berners-Lee [31], specifically in context of agents and services:

Definition 1.1 (The Semantic Web)**.** The vision of the Semantic Web is to extend principles of the Web from documents to data. Data should be accessed using the general Web architecture using, e.g., URI's; data should be related to one another just as documents (or portions of documents) are already. This also means creation of a common framework that allows data to be shared and reused across application, enterprise, and community boundaries, to be processed automatically by tools as well as manually, including revealing possible new relationships among pieces of data. (from `http://www.w3.org/`)

Semantic Web Tower

The Semantic Web Tower, as shown in figure 2.2, is a representation of the standards provided by the Semantic Web to formally structure and represent information. The following paragraphs introduce the

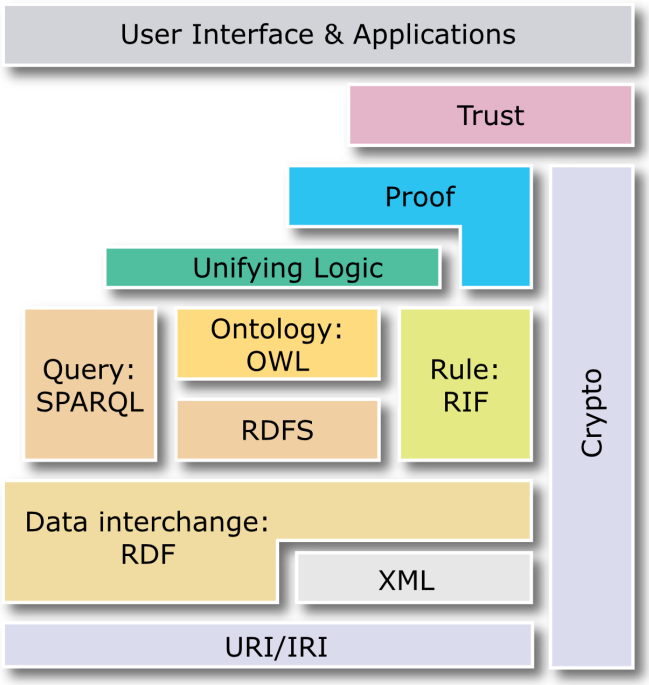

Fig. 2.2: The Semantic Web Tower (source: `http://www.w3.org/`)

different layers of the tower from the bottom upwards.

URI/IRI: Encoding and Addressing Resources The lowest level of the semantic Web tower specifies the mechanics for addressing resources on the Web: Uniform Resource Identifiers (URI [29]) and Internationalized Resource Identifiers (IRI [59]), which are based on Unicode [93], allow to assign unique ids to things on the Web.

Data Interchange: RDF The Resource Description Framework (RDF [24]) is a datamodel of creating statements about objects (*resources*) and relations between them, by providing simple and intuitive semantics.

XML: Syntax The syntactical layer provides the foundation for expressing semantic meaning:

XML [42] provides the syntax for creating structured, machine processable documents, but does not impose any semantic constraints. On top of that, the XML Schema [62, 154, 32] language allows restricting the structure of XML documents, and it allows the specification of data types for the value strings. XML is one notation to write down the statements created by the RDF Data Model. Another syntax, specifically created to be human readable, is Notation 3 (N3) [30].

OWL and RDFS: Sharing a Vocabulary and Concepts The RDF Schema (RDFS) [43] consists of some basic constructs which allow the creation schemas for data which is represented by RDF state-

ments, e.g. to specify classes and their properties, as well as sub-class relationships. For more complex relations, the Ontology Web Language[23, 119] (OWL) can be used. OWL adds Description Logic [11] to the model, and allows to specify full-fledged Ontologies, explicitly representing the meaning of terms in vocabularies and the relationships between those terms.

Query: Retrieving data from a RDF knowledge base For querying RDF data, that is retrieving result sets or graphs, there exist a number of different query languages. The "official" query language, SPARQL [135], contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph The result of a SPARQL query is either a results sets or a RDF graph.

Rules: Logical Constraints and Inference The Rule Interchange Format [36] (RIF) is an ongoing effort to create a specification for defining logic rules for use within the semantic Web. The obvious main goal is to allow the modelling of production rules, logic programming, first-order logic rules, integrity constraints and much more in RDF. In this setting, URIs are used to describe logical constants, predicates and function symbols. RIF specifies a Condition Language for use in the body of rules and queries.

Unifying Logic and Proof For the upper layers of the tower, no final standards are published yet. The unifying nature of the semantic Web has always been a design goal, and the upper logic in the semantic Web tower is a placeholder for a such as system, e.g. a world ontology connecting everything. The proof system on top of the logic allows to check, if the statements which are expressed in the system are valid, sound or true in a given context.

Trust and Cryptography Also still in discussion are the means of Trust and Cryptography for the Semantic web. The pillar reaching from the top to the bottom represents the availability of cryptography in every level of the semantic Web, realized by e.g. a public key infrastructure. A system which enables the users and applications to decide, if a statement can be trusted is the last building block. Proposals include the use of the reputation of an author to decide if a statement can be trusted or not.

Using the Semantic Web In RDF, it is possible to describe resources, identified by URIs, in terms of simple properties and property values. Thereby, RDF creates simple statements about resources in form of graphs of nodes and edges, representing the resources and their properties and values respectively. Statements, as shown in Definition 1.2, consist of *subject*, *predicate* and *object*, always in that order.

Definition 1.2 (RDF Statement)**.**

An RDF Statement is a Triple $T = (S, P, O)$, where

the subject S	is a *Resource*, identified by an URI
the predicate P	is a *Resource*, identified by an URI, specified e.g. in a shared vocabulary (RDFS)
the object O	is either a *Resource*, identified by an URI, or a *Literal*, e.g. a XSD String

The following example is an excerpt from the semantic description of a Digital Item, which is processed by the adaptation processes of the PUMA architecture. The unique identifier of the item is the URI

`http://l3s.de/puma/video1`.

Furthermore, this resource has a title,

A day in the life of the south-american cougar,

and additionally, it is associated with a genre,

`http://imdb.com/genres#Documentary`.

The knowledge about this digital item can now be written in form of statements, as shown in Example 2.1.1.

Example 2.1.1 (Statements about a Movie)**.**

Subject	***Predicate***	***Object***
`http://l3s.de/puma/video1`	*title*	*"A day in the life of the south-american cougar"*
`http://l3s.de/puma/video1`	*genre*	`http://imdb.com/genres#Documentary`

RDF distinguishes two types of nodes: *resources*, which are identified by URIs, and *literals*, which are data items, i.e. text strings (see Def. 1.2) In the above example, the labels of the predicates, e.g. "title", were chosen in an ad-hoc fashion. A different author could have written down his knowledge about the same resource with using other predicates, e.g. using the slightly different label "movieTitle" instead of "title". To uniquely identify the label for such properties and allow authors to agree on the semantics, a shared description scheme is needed. For this, RDF provides the means to create Schemas and Ontologies.

Ontologies

For describing multimedia content and the different types of possible adaptations, e.g. scaling a video down to half its size, there exist a vocabulary created by the MPEG group, it will be introduced in Section 2.1.3). For reasoning about content and adaptation processes, the multimedia items and adaptations have to categorized into a hierarchical structure, e.g. a Taxonomy. Consider as a simple example, if the user is explicitly looking for "nature documentary", but the system can not find any that match the user's interest. However, there are movies annotated with the genre "documentary, wildlife". If there exist a taxonomy, connecting these different concepts, the connection, that "documentary, wildlife" is a subgenre of "nature documentary", can be exploited to find additional relevant content. The Semantic Web standard for describing this knowledge is the Web Ontology Language (OWL). OWL is the successor of the DAML-OIL [56] web ontology language and supports three increasingly-expressive sub-languages: OWL-Lite, OWL-DL, and OWL-Full.

- OWL Lite supports cardinality constraints (but only cardinality values of 0 or 1), and a such it is an ideal tool for modeling thesauri and other taxonomies. Classes needs to be expressed in terms of *named* superclasses, and equivalence/subclass relationships are restricted to named classes as well. OWL Lite does not offer much more than RDF-S, but is decidable in polynomial time.
- OWL DL allows the full expressivity given by description logics, but is imposing some constraints (i.e. type separation) to retain computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL-DL is .
- OWL Full allows all the expressiveness of description logic, but it is not of much practical use, as it is undecidable.

A list of the vocabularies and ontologies used in this work is given in the appendix (see Section A.1).

An extended Example The predicates used in statements are identified by URIs. Using the shorthand notation which follows the XML namespace syntax and semantic. It is possible to avoid ambiguity while still providing a short, human readable notation. For example, the RDF Schema from the Dublin Core Metadata Initiative [107] provides a shared vocabulary for describing documents, then the property identified by the URI

`http://purl.org/dc/terms/title`,

which has the meaning "A name given to the resource", can be abbreviated as:

`dc:title`.

RDF statements have a graph structure, and a such they can be visualized as a directed graph. The common notation depicts resources as ellipses, literals as rectangles, and predicates as directed edges between resources, and between resources and literals. Figure 2.3 shows the RDF Graph of the knowledge in Example 2.1.1, extended with some additional statements.

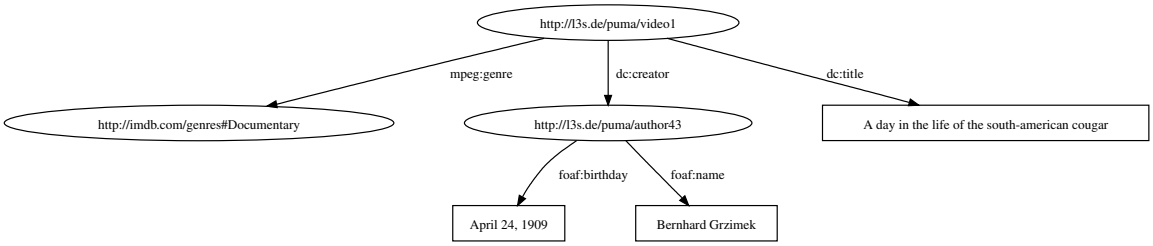

Fig. 2.3: Sample RDF graph of a Multimedia Resource

2.1.2 Querying semantic Web data

In some architectures, the semantic descriptions of resources will be stored in a central repository. Other architectures allow the distribution of metadata across multiple systems. Nevertheless, in any case a method for searching and retrieving the descriptions is required. For this, the semantic Web provides a number of different query languages.

At the time of writing, the "official" query language is SpaRQL [135], which replaced the former RDQL [144] language.

Example 2.1.2 (SparQL Example: Searching for Documentaries)**.**

```
PREFIX dc:    <http://purl.org/dc/terms/>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX mpeg7: <urn:mpeg:mpeg7:schema:2001>

SELECT ?uri ?title
WHERE  {
  ?uri mpeg:Genre ?genre .
  ?service dc:title ?title .
  FILTER (?genre = "http://imdb.com/genres#Documentary")
}
```

The example query 2.1.2 is searching the service registry for services whose profiles support the "Transcode" adaptation role. Given that the database only contains the only matching service from Figure 2.3, the query retrieves the following result set:

?uri	**?title**
`http://l3s.de/puma/video1`	"A day in the life of the south-american cougar"

Other Query Languages Apart from the official query language, which should be supported by query engine implementations, there exist many application specific query languages, which exploit the struc-

ture of the RDF data, statements and graphs. A comparison of different RDF query languages is given in [78]. The Query Exchange Language [129] (QEL) was the language developed for the RDF-based P2P system Edutella.

2.1.3 Multimedia Objects and Digital Item Adaptation

When talking about multimedia content in the digital world, audio and video documents are the most prevalent types of so called *Digital Items*. For example, music encoded in MP3 format, or movies stored in the AVI format are both digital items. More generally, multimedia object can include any combination of text, audio, still images, animations, video, and other interactive content forms.

Although the use of this multimedia for presentation, which was used for example in teaching, uses different types of information encoding, and predates the age of computers. The use of machines for creating and displaying multimedia objects added "interactivity" as another modality of media use.

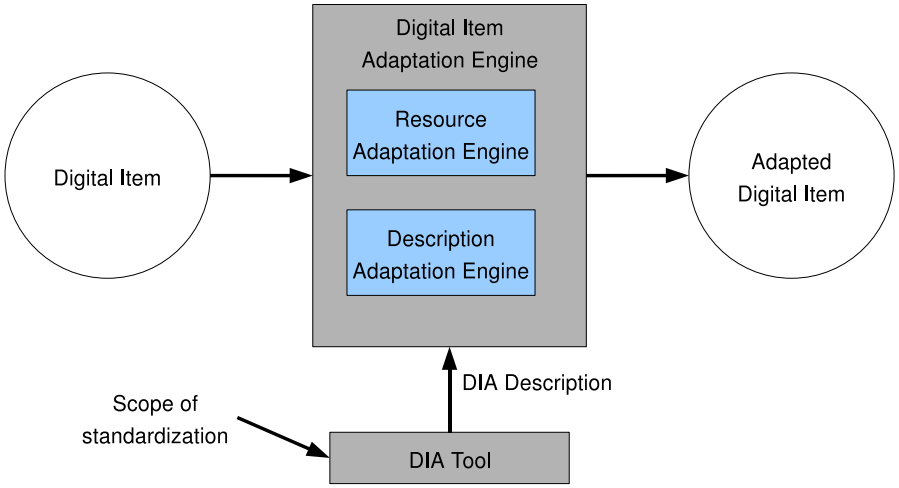

Fig. 2.4: MPEG-21: Concept of Digital Item Adaptation

In section 1.1, we have given the motivation for the development of the PUMA architecture. The main objective is the personalization, that is the modification, or adaptation, of the content to best suit the user's requirements.

Figure 2.4 further illustrates the concept of Digital Item Adaptation. An adaptation engine takes a digital item, and creates an adapted digital item by processing the resources and associated metadata descriptions. The operations performed on the content are described in an adaptation workflow in a standardized way.

The problem of adapting arbitrary content is the unmanageable large number of different multimedia formats and transport protocols. In the early years of emerging multimedia systems, many of these applications for content delivery and adaptation implemented their own formats and protocols, but with

the establishment of the Moving Picture Experts Group (MPEG) in 1990, a more focussed development of generic standards for the annotation of multimedia resources took place.

Whilst all of the MPEG standards are relevant in the context of PUMA, the MPEG-7 and MPEG-21 standards are particularly important.

MPEG-21 is a standard for Multimedia Frameworks, with the goal to describe the technology needed to support Users to exchange, access, consumer, trade and manipulate so called Digital Items in efficient , transparent and interoperable ways. The most important aspect of MPEG-21 for this work is the mechanisms and elements for creating multimedia delivery and adaptation chains as well as the syntax and semantics of operations on multimedia resources, and the relations between them.

MPEG-7, the content description standard, was designed for describing the properties of digital items, such as collections, the structure of documents in different dimensions (2D, 3D, time), colors, textures, shapes, motions, appearing persons, for both video and audio, where applicable [163, 164].

Great attention however was focused on expressing the different intellectual property rights associated with the resources a multimedia object is comprised of. Actually, the Intellectual Property Management, Rights Expression Language and Rights Data Dictionary are the most extensive part of the MPEG-21 standard [94].

Container and Content Encoding for Multimedia Objects There exists a plethora of different video and audio encoding algorithms, and for only a few exist publicly available information about the details of the compression and encoding functions. Whilst the codec (short for coder/decoder) is responsible for digitally coding the information contained in an image or audio sample, the so called *container* formats, e.g. AVI or MP4, are responsible for binding together data streams, e.g. a video encoded with MPEG-4 and multiple MP3-encoded audio streams.

The following list contains some of the container formats used for creating digital items.

- *Quicktime* is one of the most widespread container formats, which is also the basis of the MPEG-4 container format MP4.
- *Ogg* is an open container format, which is comparable to MP4, but designed to be free from patents
- *RealMedia* is a container format for the proprietary RealAudio and RealVideo codecs, but there exist an officially supported open source implementation.
- *Advanced Systems Format* (ASF) is a container format developed by Microsoft and used for all their proprietary codecs.
- *3gp* is a container format used for mobile phones, which contains MPEG-4 encoded video.

Although, a container format can used to encapsulate any combination of codecs, do keep at least some chance for compatibility between applications, the use of different codecs is usually limited. For example, the MP4 container format can video, audio, images and text (for example subtitles).

video: MPEG-4, H.264, MPEG-2, MPEG-1
audio: AAC, MP3, MP2, etc.
images: JPEG, PNG

This flexibility is enough to make adaptation of arbitrary MP4 encoded items a difficult task, as all combinations of content has to be dealt with.

2.1.4 Access to Digital Items

There exists a difference between just broadcasting video and a complex stream control protocol. The MPEG-2 standard is an example of a stream based broadcasting protocol, analogous to the old radio based television and radio systems, sending a continuous flow of data, and with the receiver locking on to the signal and starting the decoding after finding the right marks in the stream. for example, this format is used by the Digital Video Broadcast (DVB) system, which replaced the older analog television.

For more control over the streaming process, and simulating the behavior of a remotely controlled VCR, i.e. "play" and "pause" commands, as well as accessing arbitrary positions in a stream, more complex protocols are required. While there exist proprietary protocols, specifically for the many emerging web video portals with embedded video player applications, the most widespread used standard is the Real-time Transport Protocol [102] (*RTP*) family of protocols. The Microsoft Media Server (*MMS*) protocol, while providing content using a proprietary protocol, is supporting RTP as well.

2.1.5 Using Workflows to model Complex Processes

The adaptation of content as described by the Digital Item Adaptation standard of MPEG-21 requires the execution of complex operations on the involved multimedia objects.

In many of the existing multimedia adaptation systems, adaptation plans describing the necessary operations are created before the actual transformation processing step. Alternative names for these type adaptation plans are *Workflows*, *Chains* [52] or simply *Flows* [113].

All of these applications utilize some type of formalism for describing the necessary adaptation steps, but very few actually create workflow models which can be exchanged with other systems and stored for re-use or optimization.

However, in the PUMA architecture we want to use a more generic approach to workflows, to support many different adaptation scenarios. Creation of such adaptation workflows has to take into account the preferences and environment of the user. A user which prefers optimal video and audio quality requires a different adaptation process than a user who just wants the content played on his device without it consuming all network bandwidth and battery power.

For representing and reasoning about these workflows, there exist a large number of different workflow models, and Workflow Management Systems (WFMS). These approaches, which have been around since the industrialization period of the 19th century, can be traced back to Taylor [153] and Gantt, which

are responsible for starting the *scientific management*, also known as the industrial production management.

As a result, today the field of workflow management systems is very well researched. There not only exist countless different systems and implementations for all types of applications, the Workflow Management Consortium (WfMC) tries to harmonize and standardize the different systems developed by the member institutions.

Workflow Languages

The general notion is that workflows are used to handle *cases* [157, 166]. For this, workflows divide complex processes into steps, which are referred to as *tasks*.

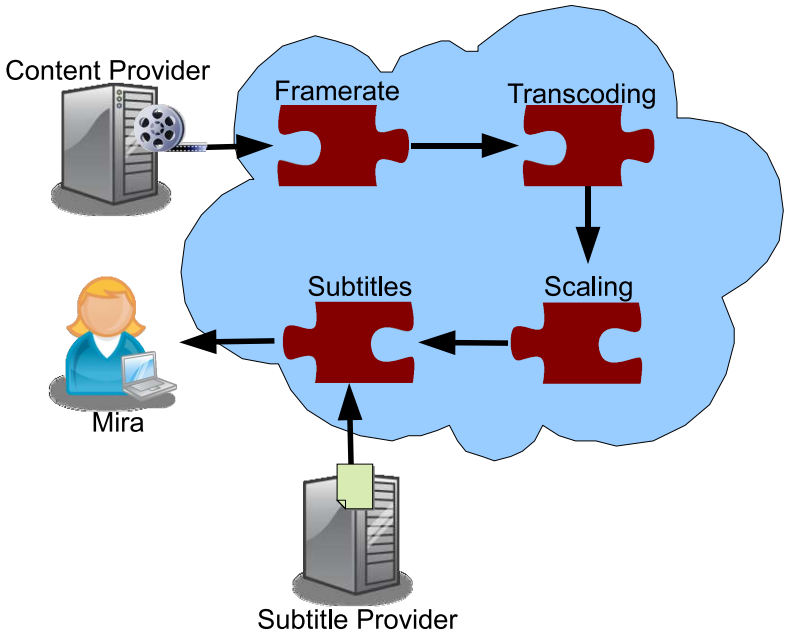

Fig. 2.5: An Adaptation Workflow Example

Figure 2.5 depicts the different task of a workflow which can be used by the PUMA system to solve the example scenario from the *introduction* (see Figure 1.5 on page 8).

The sequence of processing steps starts with scaling, reducing the width and height of the frames, and changing the encoding from mpeg-2, to a format which is supported by the mobile client device, e.g. *H.264*. The last step in the workflow is the creation of an overlay containing the subtitles loaded from a community website, and integrating it into the main video stream.

For representing such workflows in a machine-readable format, there exist a large number of standards. Ignoring the many application specific formats which are not exchangeable between implementations, three widely adopted standards remain: BPEL, BPMN and YAWL. The following paragraphs briefly describe the differences between these three models. Particularly BPEL is of importance for Web

Service compositions (see Section 2.2.1). Furthermore, a high-level comparison of these three workflow modeling languages is given by Vasko and Dustdar in [162].

BPEL The origins of BPEL can be traced back to description languages developed by Microsoft (*Xlang*) and IBM (*WSFL*).

In July 2002, Oasis released version 1.0 of BPEL4WS (Business Process Execution Language for Web Services). BPEL4WS [6] is based on WSDL , the Web Service description language, and is therefore limited to the operations possible within the expressiveness of that language.

Processes are defined by specifying the message exchange behavior of each participating peer. BPEL4WS defines two types of processes:

Abstract Process: abstract processes specify roles rather than services, and model the interactive behavior (message exchange via Web service calls) of such a system.

Executable Process: the more widely used type of process descriptions, as it directly describes the nature and sequence of concrete web service interactions. As such, these type of description is employed by many products to run service compositions (*orchestrations*, using a generalized execution engine which interprets the execution plan and performs the Web service calls.

Although BPEL4WS seems ideally suited for modelling the workflows for the PUMA system, as it is based on the current Web service standards, it does not support the continuous data streams required for multimedia adaptation. There exist many proposals for adding this functionality to BPEL, which could be seen as a requirement for content processing as it is envisioned by the PUMA system. However, no suitable implementations of such execution engines exist. Workflow Management Systems based on BPEL4WS are products like Staffware, Cosa, SAP Workflow and IBM WebSphere MQ Workflow.

BPMN The Business Process Management Notation (BPMN) is a standard notation harmonized by the WfMC, OASIS and W3C. The notation itself is a very high level language, and intended to be mapped to lower level languages and engines for execution, i.e. BPEL4WS. Notable documents in the standard are a large Terminology and Glossary of Workflow-related concepts[166]. The WfMC Workflow Management Coalition lists more than 300 organizations (vendors and users) and more than 80 products which uses their official language and modelling paradigm XPDL/BPMN.

YAWL YAWL is “Yet another Workflow Language”, which was developed to bring together the insights from workflow design patterns, and the benefit of Petri Nets [159, 161]. However, the semantics of the YAWL language are based on transition systems and it is not only an extension of Petri Nets. YAWL can be used to create Web Service Orchestrations.

2.2 Architectures for Distributed Applications

The challenges faced in the PUMA project requires our application to be scalable, easily extendable with new functionality, and the processing of content can only be done by multiple systems working together. The type of software design best suited for this type of application is a *distributed system*.

Definition 2.3 (Distributed Computing)**.** A computer system in which several interconnected computers share the computing tasks assigned to the system [89]

A distributed application is an application that makes use of resources which are distributed across different systems with the goal to provide functionality. The systems can be physically different systems, using different hardware, operating systems, storage media, and networking technology. The resources used by these applications encompass computing power, content data, storage space and network bandwidth. In *Parallel Computing*, which is a form of distributed computing, an algorithm is computed in a decentralized manner, usually simultaneously on multiple processor of the same computer, and often in clusters of machines which are connected with fast and low-latency networks. is also possible for data, and metadata, which allow content to be replicated among the participants to better distribute the load or make the system more resilient against failing network links or individual systems, as resources are available from multiple sources.

The two most prominent models of distributed systems are the *client-server* model and the *peer-to-peer* (P2P) model. The distinction between both however is not clear, as both models can be built on a spectrum of different attributes [124]. Figure 2.6 is a simplified, high level comparison between the two systems, clearly showing the difference in the organization.

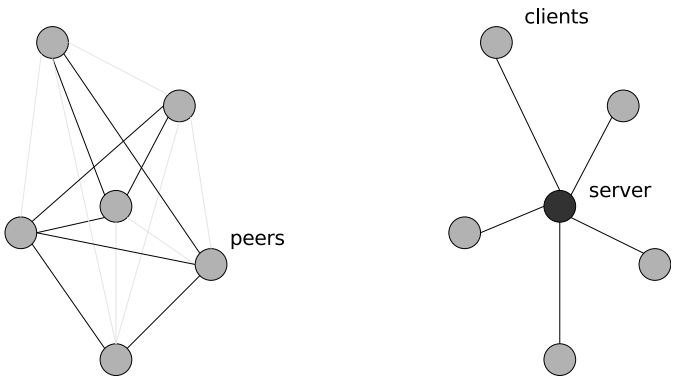

Fig. 2.6: Simplified View of Peer-to-Peer vs. Client Server

The peers in a P2P network are self-organizing, are mobile, and the network structure is created ad-hoc. In client-server models, the structure of the network is managed and configured, usually by a human administrator. Servers are static, and resources are often registered in a central registry, so they can be

found using a fast lookup. Resources are usually identified by unique names, which are resolved by the Internet's DNS system to static IP addresses. In P2P systems, resources have to be found using discovery protocols, e.g. first asking the neighboring peers about the required resource, and then asking nodes which are even further away. Because of the mobile nature of P2P systems, they usually use a custom naming system, as the IP address of a node can change or when a completely different communication protocol is used.

One of these not clearly categorized type of distributed system is the proxy architecture. With the use of a proxy architecture, the load on the servers can be distributed among a cluster of systems [74].

2.2.1 Sematic Web Services and Service-oriented Architectures

The most prominent example of an architecture which uses the client-server model is the Web Service. In principal, Web Services allow the remote execution of a functionality, by sending a request containing the data and metadata to a server, and receiving the results in a message which is returned upon completion of the computation.

Definition 2.4 (Definition of a Web Services)**.**
A Web service is 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. [37]

A short history of Services on the Web

Soon after the first computer systems where inter-connected and later grew into the Internet, companies started to investigate methods to allow the remote execution of code, using the TCP/IP protocol. One of the first standardized approaches was the SUN RPC protocol (later ONC RPC), published in 1988 and discussed and filed by the IETF as "standard" RFC 1057 (replacing the earlier RFC 1050) [152]. The concept of *Remote Procedure Calls* itself is even older, it dates back to 1983 described in works by Birrel and Nelson from Xerox in Palo Alto [33]. Already the early RPC protocol was independent of the underlying transport protocol.

These protocols are all strictly following the client-server paradigm, whereas the server provides a function at a defined address, expects a certain set of parameters, and will respond with a defined message to calls from clients. One of the best-known applications making use of this method is NFS, the network file system, allowing transparent access to files across the network. Transparent in the behavior as if the file was stored on the local machine, without having to rely on applications, such as FTP, to copy the data between the machines. However, most RPC calls are synchronous, with the client having to wait for the

server to finish with the procedure. A second standard is the Distributed Computing Environment (DCE) RPC, which was used by Microsoft as MSRPC for the Windows NT operating system.

Summarizing, the original RPC protocol is a simple, message oriented protocol for requesting the executing of code on a remote system.

With the advent of object-oriented software development, with C++ and Java, not only the remote procedure call was renamed to *remote method invocation*. Additional new implementations for working with "objects" across the network were created, i.e. CORBA, DCOM, and .NET Remoting. These systems are very complex, difficult to master, and as such not really suitable for developing distributed, light-weight applications.

It was in 1998 that XML-RPC was released by Dave Winer, in collaboration with Microsoft. Its successor, *SOAP* [76], has been turned over to a W3C working group and made the de-facto standard for web-based remote procedure calling. Different from the early RPC implementations, i.e. UNC-RPC, SOAP used XML as platform-independent message format and HTTP as transport protocol.

Describing the Functionality of Web Services

By definition, a *Web Service* is a software system designed to support interoperable machine-to-machine interaction over a network [37]. To enable this, it advertises its functionality via a public service registry. These registries are commonly organized as centralized service repositories. A service requester can use the service repository to search for a service which provides the functionality he desires, which only works, if the requester and provider agree on the semantics for describing the service's properties.

A Web Service consists of an abstract interface definition, which must be implemented by a concrete *agent*. As long as the implementation follows the interface of the Web Service, each of the implementations are interchangeable with each other.

The functionality of a Web service is documented in a Web service description (WSD). The most common language for describing a Web service's interface in a machine processable way is the Web Service Description Language [54]. Basically, the Service Description is an agreement which controls the mechanics of interacting with that service. It defines message formats, data types, transport protocols and the serialization formats which are going to be used between the requester and provider. Furthermore, the document usually also contains the locations on the network, where the functionality, which is provided by the service, can be invoked.

In the following Examples illustrate some of the parts of such a WSDL document, written in XML. It describes an imaginary multimedia service, which, when invoked, opens a connection to a stream and modifies it according to a configuration command. The namespaces are simplified, the implementation's namespace is abbreviated as "impl".

The first piece of code (2.2.1) shows the definition of the datatypes which are used in the messages. The first element, "invoke" consists of two strings, one for the location of the video stream, the second for the configuration document. The other type, which will return the status of the operation, is just a string.

Example 2.2.1 (Data Types)**.**

```
<wsdl:types>
  <schema elementFormDefault="qualified"
            targetNamespace="impl"
                      xmlns="http://www.w3.org/2001/XMLSchema">

    <element name="invoke">
      <complexType>
        <sequence>
          <element name="sourceStream" type="xsd:string"/>
          <element name="configuration" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>

    <element name="invokeResponse">
      <complexType>
        <sequence>
          <element name="invokeReturn" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>

  </schema>
</wsdl:types>
```

In the next Example (2.2.2), two messages are defined, “invokeRequest” and “invokeResponse”, which use the defined message types as the keys for parameters. Additionally, this Example also for the first time refers to a class of MultimediaService which defines a single remote operation, simply called “invoke”, which expects to receive the previously defined “invokeRequest” message and will answer upon completion with an “invokeResponse” message. When the Web Service is called, each of the parameter gets assigned a value.

Example 2.2.2 (Messages and PortType)**.**

```
<wsdl:message name="invokeRequest">
  <wsdl:part element="impl:invoke" name="parameters"/>
</wsdl:message>

<wsdl:message name="invokeResponse">
  <wsdl:part element="impl:invokeResponse" name="parameters"/>
</wsdl:message>

<wsdl:portType name="MultimediaService">
  <wsdl:operation name="invoke">
```

```
    <wsdl:input message="impl:invokeRequest" name="invokeRequest"/>
    <wsdl:output message="impl:invokeResponse" name="invokeResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

The next information that needs to be defined is the method of transport for sending and receiving the messages. The code Example below (2.2.3) shows how a binding of the abstract invocation to an actual soap call could look like.

Example 2.2.3 (Binding)**.**

```
<wsdl:binding name="MultimediaServiceSoapBinding"
               type="impl:MultimediaService">

  <wsdlsoap:binding style="document"
              transport="http://schemas.xmlsoap.org/soap/http"/>

  <wsdl:operation name="invoke">
    <wsdlsoap:operation soapAction=""/>

    <wsdl:input name="invokeRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>

    <wsdl:output name="invokeResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>

</wsdl:binding>
```

Now that the parameter types, messages, operation and a binding to a soap message is defined, a potential client needs to know where to call the service. In this Example, the service is running at the given address on a web service engine at the local machine, port 8080,

Example 2.2.4.

```
<wsdl:service name="MultimediaService">
  <wsdl:port binding="impl:MultimediaSoapBinding" name="MultimediaService">
    <wsdlsoap:address location="http://localhost:8080/mmservice"/>
  </wsdl:port>
</wsdl:service>
```

Advanced Semantics for Describing Web Services

For the discovery and communication with Web Services, it is crucial that there exist a somewhat shared knowledge about what behavior is expected from the service.

For this, before the invention of Semantic Web Services, the users of service architectures used different ways for describing the semantics of the operations, parameters and protocols involved in the discovery of Web services and communication with them.

The semantic description of the services in a UDDI registry is very limited, compared to the possibilities of the Semantic Web. Searching for services is possible by matching category keywords, field contents, i.e. names, addresses, and technical data such as message, parameter, and operation names. In many cases, the creation of applications which utilize web services, and the selection of Web services and operations which will be invoked, has to be done manually.

Nevertheless, even in the early days of Web Service architectures, the fields stored in the entries in a service registry have been used to create semantic service descriptions. One of the first proposed extensions for adding semantic information to Web services was SAWSDL [132, 117], which allows to refer from inside a service description, to concepts from a semantic model, i.e. shared vocabularies, which itself is stored somewhere on the web.

Currently, the official standard for creating semantic descriptions for Web services, is OWL-S, which is *a domain ontology for semantic Web services*. The foundation of OWL-S is the Resource Description Framework (RDF, see Section 2.1.1), and it is the successor of DAML-S [115], which was based on the semantic web standards DAML+OIL [56] (DARPA Agent Markup Language plus Ontology Inference Layer). The inference layer was subsequently replaced by the Ontology Web Language (OWL), to form what is now known as OWL-S, the Semantic Markup for Web Services [114]. A more closer look on the differences between SAWSDL and OWL-S is given by Martin et al in [116].

Whilst the WSDL document only covers the syntactical and technical aspects of the message exchange for calling web services, OWL-S adds two more aspects to a service description. As introduced earlier, RDF documents can be rendered as graphs, or written down in an XML notation. In Figure 2.7, the top concepts of a Semantic Web Service description are shown, including the three different notions of semantic models:

- The *Service Grounding* provides information on *how to access* a service, i.e. the transport protocols, parameter types and operations. Furthermore, the grounding allows a mapping of the messages, processes and transport protocols defined in the OWL-S description to the corresponding element in the WSDL description of the service.

- For advertising *what a service provides* for a prospective requester, the *Service Profile* is used. The profile is comparable to the information published in a service registry, such as UDDI, but with build-in semantic Web capabilities, such as inference and logic.

- The third type of knowledge is about *how a service works* and interacts with other participants.

The *Service Model* provides information about the atomic and more complex functionality which is provided by a service.

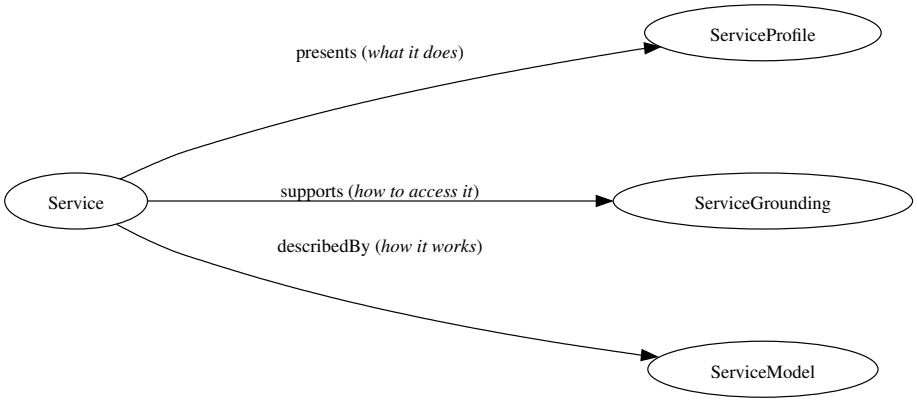

Fig. 2.7: Top Level of the Service Ontology [114]

The service profile, and the service model, both represent the functionality of a Web service operation in terms of Inputs, Outputs, Preconditions and Results (IOPR). However, the profile most often only provides a more general view on the capability and behavior of a service, whereas the model contains a detailed description of each and every produced results and operation possible with the service. The purpose of the profile is the use in the matchmaking process for service selection from a registry. Figure 2.8 shows an excerpt from the service profile ontology.

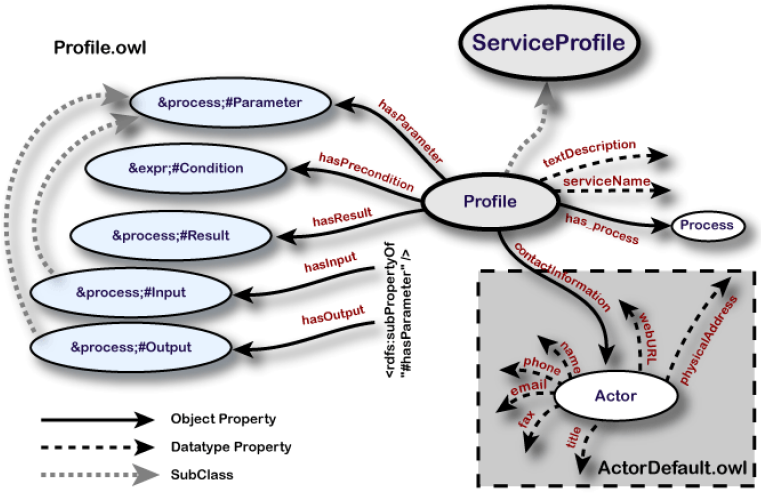

Fig. 2.8: Excerpt from the Service Profile Ontology [114]

The Process class in the service model or process ontology provides the same IOPR structure as the Profile class, with the difference, that processes can be combined to composite actions, i.e. Sequences, Splits, Joins or Loops.

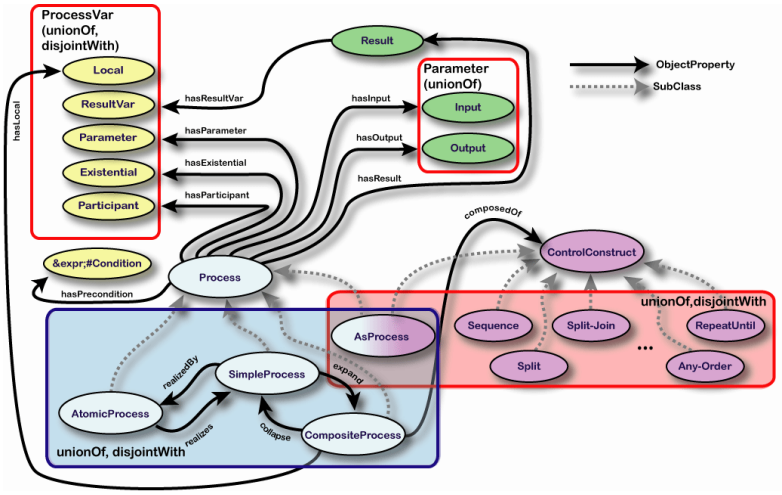

Fig. 2.9: Composite Processes and Control Structures in OWL-S [114]

For accessing a service, a description of the protocols and parameters is needed, which historically was the purpose of WSDL. With a service grounding, it is possible to map a the atomic processes and inputs/outputs of an OWL-S description to the operations and messages defined in a WSDL document (see figure 2.10).

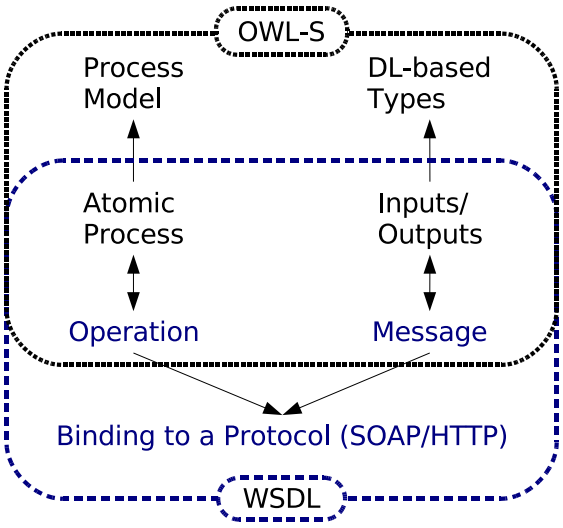

Fig. 2.10: Mapping between OWL-S and WSDL [114]

The next pieces of XML document snippets are an exemplary grounding description, which connects the elements from a WSDL document to the resources in an OWL-S description. The location of the

WSDL document is the same as in the previous MultimediaService Example, and simply abbreviated by "impl".

The RDF statements from the OWL-S are written in XML notation. The first Example opens up the Grounding of a single atomic process "invoke", which corresponds to a single operation invocation in the WSDL description.

Example 2.2.5.

```
<grounding:WsdlAtomicProcessGrounding rdf:ID="Grounding1">
  <grounding:wsdlDocument>
    impl
  </grounding:wsdlDocument>

  <grounding:wsdlVersion
         rdf:resource="http://www.w3.org/TR/2001/NOTE-wsdl-20010315" />

  <grounding:owlsProcess rdf:resource="#Invoke">
```

The Atomic process the Grounding refers to, is defined somewhere else in the OWL-S document, and could look like the following code piece. It basically defines the same datatypes and parameters, such as the non-semantic service description (see Example 2.2.1), but this time attached to an OWL-S IOPR process statement, although this particular process does not require preconditions nor has it any effects on the future state of the service.

Example 2.2.6.

```
<process:AtomicProcess rdf:ID="Invoke">
  <process:hasInput>
    <process:Input ref:ID="SourceStream">
      <process:parameterType rdf:about="&xsd;#string">
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Input ref:ID="Configuration">
      <process:parameterType rdf:resource="&xsd;#string">
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output ref:ID="InvokeResponse">
      <process:parameterType rdf:resource="&xsd;#string">
    </process:Output>
  </process:hasOutput>
</process:AtomicProcess>
```

The following part describes the “Invoke” Web service operation and the associated MultimediaService class (see Example 2.2.2).

Example 2.2.7.

```
<grounding:wsdlOperation>
  <grounding:WsdlOperationRef>
    <grounding:portType>
      <xsd:uriReference rdf:value="impl:MultimediaService"/>
    </grounding:portType>
    <grounding:operation>
      <xsd:uriReference rdf:value="impl:invoke"/>
    </grounding:operation>
  </grounding:WsdlOperationRef>
</grounding:wsdlOperation>
```

In OWL-S, a process defines tuples of Input, Output, Preconditions and Effects. The Inputs and Outputs of a such a process have to be mapped to the corresponding operation parameters in the WSDL description. This is done using the MessageMap construct, which has two properties, owlsParameter and wsdlMessagePart, which refer to the OWL-S and WSDL URIs, respectively. The code below shows the definitions of the message send to the Web service containing the two parameters “sourceStream” and “configuration” (see Example 2.2.1). The description for the response message is similar. This closes the Grounding Definition of an OWL-S Atomic Process.

Example 2.2.8.

```
<grounding:wsdlInputMessage rdf:resource="impl:invoke"/>

<grounding:wsdlInput>

  <grounding:wsdlInputMessageMap>
    <grounding:owlsParameter rdf:resource="#SourceStream">
    <grounding:wsdlMessagePart>
      <xsd:uriReference rdf:value="impl:sourceStream">
    </grounding:wsdlMessagePart>
  </grounding:wsdlInputMessageMap>
</grounding:wsdlInput>

<grounding:wsdlInput>
  <grounding:wsdlInputMessageMap>
    <grounding:owlsParameter rdf:resource="#Configuration">
    <grounding:wsdlMessagePart>
      <xsd:uriReference rdf:value="impl:configuration">
    </grounding:wsdlMessagePart>
  </grounding:wsdlInputMessageMap>
</grounding:wsdlInput>
```

```
  <grounding:wsdlOutputMessage rdf:resource="impl:invokeResponse"/>
  ...
</grounding:WsdlAtomicProcessGrounding>
```

Creating Applications based on Web Services

Realizing complex applications based in a distributed manner with loosely coupled Web Services, within or across a trusted domain, is desirable to reduce operating costs and improve re-usability. For the modelling of service compositions, two conceptually different approaches have manifested

- An *Orchestration* describes the execution logic of Web service compositions based on control flows, such as sequences, parallel processes, conditions, and exception handling. The Web Service Business Process Execution Language [99] (WS-BPEL) is a XML based language to describe the interactions between (business) partners on a high level.

  Most commonly, the Orchestration is executed by an Orchestration engine, e.g. open source systems such as Apache ODE, JBoss jBPM, Sun OpenESB, or commercial engines such as IBM WebSphere Process Server, SAP Exchange Infrastructure, or Oracle BPEL Process Manager.

  The interaction with each partner occurs through Web Service interfaces, and the structure of the relationship at the interface level is encapsulated in what is called a partnerLink. The WS-BPEL process defines how multiple service interactions with these partners are coordinated to achieve a business goal, as well as the state and the logic necessary for this coordination. WS-BPEL also introduces systematic mechanisms for dealing with business exceptions and processing faults. Moreover, WS-BPEL introduces a mechanism to define how individual or composite activities within a unit of work are to be compensated in cases where exceptions occur or a partner requests reversal.

- In a *Choreography*, not service execution sequences, but the collaboration between participants is defined, based on the observable behavior and the actual occurring information exchange. Although, there does exist something similar in BPEL, the abstract process model, in which not instances of services, but abstract services can be specified. However, the majority of application build in a Service-oriented architecture are based on concrete Orchestrations, where the services have been manually selected and the process execution is laid down based on design documents. As a result, apart from research prototypes and experimental implementations, there do not exists a wide-spread implementations of a choreography engine.

  The latest standardized description language for choreographies is WS-CDL [100], the successor of WSCL [22].

Automatic Creation of Service Compositions When Service-oriented architectures are built, one of the steps in the design process is the creation of the Orchestration or choreography which describes the functions that are provided when the different services interact with each other. In some cases, when the system should adapt itself to specific requirements, designing the service composition on the fly might prove more efficient.

- The Web Service Modelling Ontology [139] (WSMO) is a language to model service compositions, with WSMX [79] as a execution engine. It uses mediators which deal with interoperability issues between services, and provides concepts to describe choreographies, as well as Orchestrations. Furthermore, it supports the dynamic selection of Web services based on a desired goal, i.e. a high level description of a task. In WSMO, web service operations, consisting of Inputs, Outputs, Preconditions and Effects (IOPE) can also be interpreted as transitions, such that a reasoning system can create a plan of invocations to reach a desired end state.

- The Internet Reasoning Services [51] (IRS-III) are similar to the WSMO, as the systems is also based on semantic Web services with rich descriptions and a logic-based model which allows the creation of a service composition by defining a desired goal.

Automatic Selection of Services for Service Compositions Selecting Web Services is based on discovering appropriate services in a registry, which used to be UDDI for a long time. With the availability of semantic information, more sophisticated methods of matching services with requirements.

In many application scenarios, the creation of an Orchestration is performed by a human designer, which selects the services by comparing requirements with the descriptions of available services. The service compositions are created for one specific application only, the services stay the same during the whole lifetime of that application, and if one of those services fails or disappears, the application breaks and the composition needs to be adjusted.

Automatic selection of services is used in environments, where the goals, available services, and the data changes, which is the case for large multimedia systems.

Using the semantic descriptions, compared to only having a UDDI registry entry available, automatically finding services which match a defined role of a choreography or Orchestration has become easier.

Without semantic descriptions, relying only on the keywords and arbitrary categories in UDDI registries, it was already possible to exploit semantic similarity measures [136], by relating terms in the descriptions to external ontologies, e.g. wordnet [120]. Other approaches for automatic selection of Web services is based on singular value decomposition (SVD) [140].

With the advent of Semantic Web Services, a more suitable approach, using the Input, Output, Precondition and Effect (IOPE) parameters of OWL-S profiles was devised. However, as it is unrealistic to expect a perfect match of all the IOPE attributes with a request, different measures for the “Degree of Match” (DOM) have been introduced [155]. Approaches, which use semantic similarity measure are provided by Hau et al. [80]

It is also possible to use other elements from the semantic descriptions of Web services for matching, i.e. service categories and capabilities. If terms for categorizing services are selected from an extensive ontology, reasoning about these concepts allow the selection of matching services based on exploiting the relationships between the categorization terms, e.g. choosing a broader concept to find additional services. In [55], Colucci et al. use concept abduction and contraction to narrow down or broaden the scope of when to consider a service to be matching the requirements. As the OWL-S descriptions of services can also include preconditions and effects, finding matching services also requires basic logic reasoning, usually some form of Description Logics, which is the logic used by the Ontology Web Language (OWL).

In [131], Paolucci et al. describe their implementation of a system which provides a public Semantic Web Service registry and a matchmaking service based on jUDDI and Racer [77]. Klusch et al. present in [103] their OWLS-MX Hybrid matchmaker, which uses profile matching and reasoning with external service ontologies. An extensive description of the changes from using only UDDI-based selection of services to OWL-S with profile matching and description logics is given by Srinivasan et al. in [148].

In [20] Balke et al. describe an algorithm for selecting composition creation and monitoring, with a strong emphasis on monitoring. Monitoring is important for the replacement of service in real-time scenarios, such as multimedia content delivery, and as such it is specifically relevant for PUMA. Although, E2MON provides a service selection algorithm based on a cost function, without specifying those functions in detail.

Personal Reader: Personalized Semantic Web Services

Personal Reader was developed as a test-bed, using semantic Web services for content adaptation and personalization in the course of the REWERSE project. Although the applications developed using the Personal Reader Framework [84] were geared toward different domains, E-Learning [13] and Digital Libraries [1].

The framework shown in figure 2.11 is a multi-layer architecture of dedicated services:

- the user interface is provided by so called SynServices (Syndication Services), which are responsible for the rendering of content and the creation of requests for the adaptation services,

- the actual personalization of content is performed by PServices (Personalization Services), which are all sharing the same interface, and communicating with RDF documents as requests and responses only. Syndication services are responsible for interpreting the content.

- the Connector is responsible for the selecting and configuring the right PServices for handling a request. Matchmaking services and requests is based on semantic descriptions of services, and configuration of the services is done dynamically based on the user's profile, stored in the User Model Service (UMService) and parameters embedded in the request message.

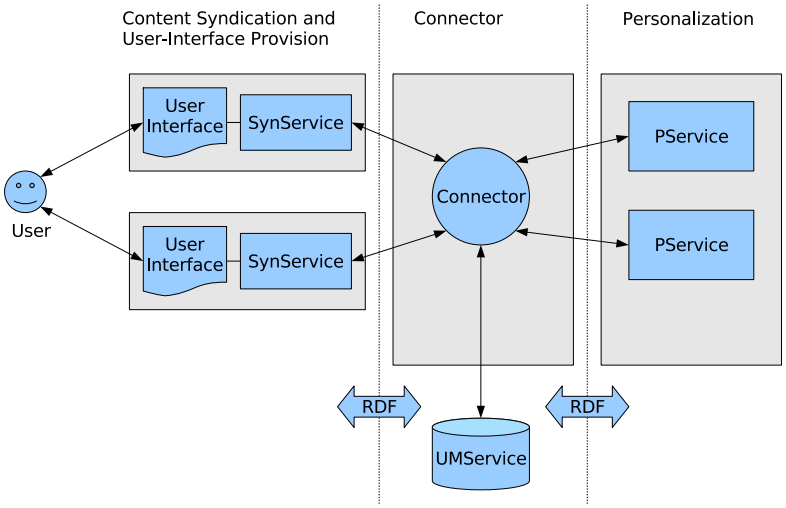

Fig. 2.11: Personal Reader Framework (source: [14, 84])

2.2.2 P2P-based Multimedia Systems

In the context of multimedia applications, P2P systems are often used for content distribution. P2P-based systems have unique properties, which make them resilient against failing, moving and disappearing participants and sometimes even provide anonymity, which makes them ideally for sharing large content objects [8, 124]. This also is a major problem for content owners, which loose the control of their content, if it gets copied, modified and distributed among thousand, even millions of nodes in such a network. However, there exist applications and business models, which allow the use of P2P for multimedia content distribution. One example are TV applications, which use a P2P-based system to distribute the content. Content, which was intended for broadcasting to a broad audience, and as such is freely distributable under certain conditions. Prime example of such a systems are zattoo [96] and PPLive [91].

Whilst the use of Peer-to-Peer for content distribution, and replication, faces some serious legal problems, using it for sharing metadata and thus enabling distributed information retrieval, found acceptance for sharing learning material [129] and in the digital library domain [21]. To improve the retrieval of information on P2P systems, new methods for content lookup [151] and query distribution were developed [47, 48].

2.2.3 Support for Continuous Data in Distributed Systems

Adapting a video which is stored on a harddisk is possible with all types of encoding and compression implementations. However, as soon as the data is streamed over a network, the client, and also any processing step between the client and the delivery system can not simply access any position and material in the digital item it might require for decoding and encoding the information.

A popular and simple approach for accessing large data sets is to split the content up into smaller

chunks of data. What makes this approach complicated in this scenario, is the way information is encoded and transported to the client. Naturally, the mixing of video and audio content to a multiplexed stream requires additional care when modifying the content, for example to not loose the synchronization between the video and audio channels.

In Section 2.1.4 we have briefly introduced some of the common protocols for streaming video over the Internet, namely *RTSP* and *MMS*. However, these protocols were designed for controlling specific content delivery systems, and not for the communication between distributed processes that perform digital item adaptation. Furthermore, they are not particularly well suited for implementation in a Web service architecture, nor in P2P content sharing systems.

- Web Services work in a synchronous way: receiving a request, processing and returning a result, and then waiting for a new request. This type of communication is best suited for small messages, not for large objects such as videos. The solution for processing large data sets has always been "streaming", the continuous sending and receiving of bits. Unfortunately, the current Web service implementations do not support this transport protocol. Using the traditional protocols, sending many message which each contains only a small chunk of data, creates unnecessary overhead, since the chunks need to be encoded into SOAP messages and decoded by the service engine again, together with all the additional SOAP information.

- Peer-to-Peer systems work exceptionally well, if a single item (a video, or a query) needs to be distributed to many recipients. In case of content adaptation, when each task in the workflow is performed by a different peer, this architecture does not provide any benefit over a Web service architectures.

Streaming solutions Black et al. present in [34, 106] the Infopipe system, which is a middleware for creating streaming based applications, based on a model of sources and sinks, but lacking a control protocol which provides functionality such as RTP. Instead, it provides good visibility of QoS parameters of the active streams. For GRID systems, where the middleware is based on Web Services, accessing large amounts of data, i.e. meteorological simulations from super-computers, specialized protocols have been created. GADS [169] by Woolf et al. and Styx [35] by Blower et al. allow the addressing and access to parts of very large data set. Another different type of multimedia application is the proxy architecture, where the adaptation is performed by a intermediate system, which sits between the client and the content provider, e.g. QBIX-G [39].

A simple form of streaming application is content delivery without any active adaptation. The Helix Server [138] is an example of such a system, which is mainly used to deliver pre-generated content to client devices. It even allows on-the-fly encoding of live content directly retrieved from a recording device. Limited processing capabilities usually only allow the production of one or two different formats.

2.2.4 Quality of Service in Distributed Systems

When it comes to providing an uninterrupted service, the business world has already developed a contract model called "*Service Level Agreements*" (SLA). In this corpus of regulation two partners can define the quality of the rendered services in terms of responsibilities, availability, escalation methods, and costs. The IT Infrastructure Library (*ITIL*) provides the de-facto standard for service level management in form of best practices, and corporations can get a certification of their compliance with this standard (ISO/IEC 20000) which as an important attribute of quality for organisations for internal and external customers regarding business processes.

The Global Grid Forum has developed *WS-Agreement* [7] for specifying Service Contracts for jobs performed in a Service Environment. This proposal was pre-dated by an initiative from IBM, which designed the WSLA [112] standard for creating Web Service Level Agreements.

Since then, WS-Agreement has also become part of the service selection process for multimedia systems [170].

The problem with Quality of Service is, available service execution engines lack the implementation of these standards. Currently, the most practical way of monitoring services is the processing of service exceptions which are recorded by orchestration engines and usually lead to failing the complete task, leaving the repeating of the executions as the only option.

Furthermore, the Internet in its current form is an unreliable medium. While there are standards for defining service classes, it is possible that not all the nodes between two points in the network respect these settings. The next generation Internet protocol was specifically designed to create and monitor reliable quality parameters of a connection, but for it to work it must be adopted by all service providers. For a guaranteed level of service, other means are necessary, e.g. leasing dedicated direct high-speed connections.

The Multimedia Operating System and Networking Group (MONET) has focussed on end-to-end Quality of Service (QoS) constraints for distributed multimedia architectures, for P2P-based systems [81, 111] and Service Composition Frameworks [75, 127]

2.3 Summary and Discussion

In this chapter, the technologies available for creating large scale distributed applications have been discussed. The task of realizing such an application requires enabling technologies, such as the descriptive Metadata provided by the Semantic Web and the MPEG Standards. Semantic Web enables applications to describe resources on the Web and the relations between them, using shared vocabularies and ontologies. Resources can be anything: videos, text documents even functionality provided by Web Services. The standards created for Digital Item Adaptation (DIA) allows for the description of all kind of transformations which can be applied to multimedia resources. DIA also includes a semantically rich language for the annotation of content properties, managing digital rights and describing delivery methods.

The second part of this chapter discussed the various architectures for building distributed systems for content adaptation. While these architectures provide good facilities for a large number of possible applications, the particular requirements for multimedia adaptation are not available for Web Services and P2P systems so far. Web service architectures have trouble processing continuous data streams. Furthermore, as it is the nature of the Web service description languages currently in existence, most forms of automatic interoperability checking is neglected. Services can be selected automatically, but the most commonly used WSDL documents for describing service does not allow provide more information than just the message format, data-types and transport protocol for invocation of a remote process. The more complex task of creating applications from sequences of such service invocations has to be done manually, as the required information is only available in form of technical manuals, by looking at the implementation directly, or calling the developer.

In the next chapter we will show in context of the PUMA system, how to achieve a better automatic creation of service compositions, using a detailed description of a process' behavior and an abstract model of the interactions between services (which actually is a type of service choreography). This type of selection not only makes use of the categorization of services using the UDDI information, but also the profile and process models of OWL-S. As such, it can be verified if a service is actually able to communicate with the other participants in a service composition.

Chapter 3

Universal Multimedia Access with the PUMA Architecture

Providing easy and reliable access to multimedia objects of various types from a plethora of different end user devices, is the goal of the PUMA project. For this, we designed a distributed, service-based architecture for adapting and delivering multimedia content. The properties and constraints which governed the design of such a system are described in Section 3.1. Following this, Section 3.2 is dedicated to the architecture and gives an overview over the components of the PUMA system. The third and of this chapter is an introduction to the approach that is responsible for the service composition.

3.1 Design Parameters of the PUMA systems

The *design* and implementation of large scale applications, as already stated in the introduction, "is a complicated, time-consuming and tedious task" [126]. Monolithic architectures have weaknesses, such as the lack of exposed interfaces, which makes extending the system with new functionality hard, and therefore limits the re-use of already implemented code. In a centralized system, there is always the risk that a failure in one small part of the system will bring the complete application to a stop.

Because of this, building service-oriented infrastructures is favored over the construction of monolithic applications. The idea of decentralizing functionality is not new. In the early days of computing, functionality and data was usually provided by central powerful computers which were accessed by much simpler interface terminals.

However, nowadays distributed systems developing tools are as mature as those for monolithic, centralized platforms. And there exist general purpose Web Service frameworks, which make the creation of distributed, service-oriented applications comparable to the development with other software designs.

The key difficulty for the decentralization of multimedia applications is the nature of the information that is managed by these systems. Digital movies with sound, even when encoded with the best and most

advanced compressing algorithms, are too large to be transported from the content repository to the client device in a single message. In many cases, the clients do not even have the capability to store all the content before rendering the video, or the processing power to decompress and decode the information in real time. There are literally hundreds of different media types which from which Digital Items can be composed. In the same order of magnitude is the number of different client devices used for viewing the content at the user's side.

One goal of the PUMA project is to address some of the issues presented in [126]. The acronym PUMA stands for *Personalized Universal Multimedia Access*, which defines a system that allows a user to find and access multimedia resources in a personalized way. That is, the content is adapted to meet the requirements and preferences of the user.

- *Universal* access to any type of multimedia content is twofold. First, the user should be able to use any client device for rendering the content. For example, the client device could be a multimedia enhanced mobile phone. Second, there is no limitation to the variety of media types, from which the user can choose. There can be high quality, high resolution, Hollywood blockbuster movies or just thumbnail-sized videos recorded with a mobile phone. Content might be a streamed RSS textual document or an MP4 video stream with eight audio channels, subtitles and full textual descriptions of all scenes.

- In general, *Personalization* is the automatic adaptation of the system's behavior to match the preferences of the user. The user's preferences are not only honored during the selection of content, regarding actors, the author, or genre, but also the user's bias towards technical details, such as the use of a specific video format. Personalization, sometimes simply called Adaptation, should take care of the necessary transformations to create the experience, that the user desires.

- Furthermore, the goal is to provide an *extendable* platform, which allows the integration of new forms of content adaptation and transformation tools from many different providers. For example, tools for enriching and modifying content for a niche group of customers, such as providing subtitles to movies in additional languages.

3.1.1 Design Decisions

- *Extendability* of the PUMA system is achieved by choosing a modular architecture for implementing the system. The obvious way to allow functionality to be provided by different providers is to use a service oriented architecture, which in turn is able to process multimedia content. Moreover, a verification process must ensure, that the services selected for performing the personalization are able to communicate with each other.

- The *Personalization* of content, in the PUMA system, is performed by small specialized services, each tackling a different aspect of the Digital Item Adaptation. For example, one service is responsible for scaling videos, another service is specialized on reducing the number of frames per

seconds of a movie. Providers may charge for the use of services, and as such there can be more generic services which can perform their operation on many different multimedia formats, while there can also exist highly optimized services which only work on one specific format. As is common in many existing digital item adaptation systems, the first step, before the actual transformation of the content, is the creation of an adaptation workflow, which contains the sequence of transformations applied to a multimedia object [108, 125]. Furthermore, the adaptation must be constantly monitored to detect changes in the environment or failing services, which have to be replaced before the delivery of the content is interrupted.

- In the PUMA system, many different types of resources are used. A heterogeneous system requires the use of a unifying technology, which is able to describe these resources in a interoperable way. This is ensured by using techniques from the Semantic Web, such as RDF and Ontologies. Resources on the PUMA system are Digital Items, Adaptation Workflows, Service Capabilities and Communication Protocols. For Digital Items and Adaptation, there exist the MPEG-7/21 standard. Workflows can be formalized in many different formats, of which BPEL is one of the most common. For describing the capabilities and the behavior of Web Services, the Ontology for Web Services (OWL-S) is used. Protocols, which specify the possible interactions between participants in a service composition, are best modelled with a choreography language, such as WS-CDL [100].

The focus of this work is the creation of adaptation workflows, the selection of services for the roles in such workflows, and the verification of interoperability of the selected services. The description of the PUMA system is included to provide an overview of the context in which the composition of services is performed.

3.2 Components of the PUMA Architecture

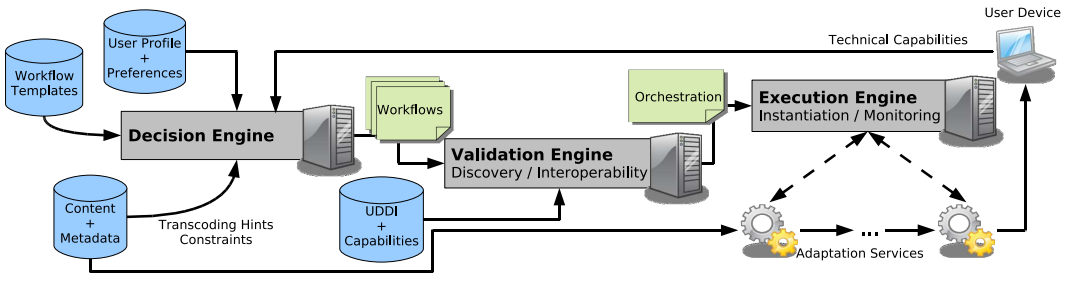

Fig. 3.1: The PUMA System: Architectural Overview

Figure 3.1 shows the different modules of the PUMA *architecture* which we developed. This Section gives a short overview of the core modules of the architecture and the components we designed for the different task that are required in the process of creating, validating and executing service compositions.

- The *User Device* (see Section 3.2.1) is used for rendering the adapted multimedia content. This piece of software is not only a media player application, but contains additional functionality to monitor the state of the client hardware, to quickly detect and broadcast changes of the client's state, e.g. changes in the network connection or warnings of imminent battery failure.

- In every adaptive system, the *User Model* stores the profile and preferences of the users. Depending on how sophisticated the adaptation process is, the model will also keep a history of the user's interactions with the system.

- The *Content Database* (see Section 3.2.3) is the source of Digital Items for adaptation and delivery by the PUMA system. Conceptually, every document on the web accessible via standard retrieval methods can be used as an original source for adaptation, the current design requires the descriptions and annotations of digital items to be stored in a database system.

- The *Decision Engine* and the *Validation Engine* are responsible for creating the adaptation workflows which are used to adapt the content for delivering it to the user. Moreover, these two modules are responsible for selecting services from a service registry, and the validation of the service compositions which are created from the adaptation workflows (see Section 3.3).

- The *Execution Engine* (see Section 3.2.4) takes the workflows and services (now in form of a simple Web Service Orchestration), and starts the service composition by initializing the adaptation services. Furthermore, it continuously monitors the state of the services, including the expectation of notifications from the client device, on which it has to react.

- The *Adaptation Services* (see Section 3.2.5) are the building blocks which provide the processing functionality for the different steps in the adaptation workflows.

3.2.1 Client Application: A Multimedia Player Agent

In systems which are based on a common stream control protocol, such as RTSP (see section 2.1.3), watching a video stream would only require one of the standard multimedia applications included in many modern operating systems. Rendering of the content is typically done by applications such as Windows Media Player or the VideoLAN Client [150] (VLC).

However, the *client* application of the PUMA system (Figure 3.2) is more than just a media player. The task of the client application is threefold.

1. Using locally stored information about the user and the user's preferences, and supported by the technical capabilities of the device, the search interface of the application can be used to find multimedia objects from the available content databases. The request which is sent to the decision engine is also influencing the decision which type of adaptation workflow is created.

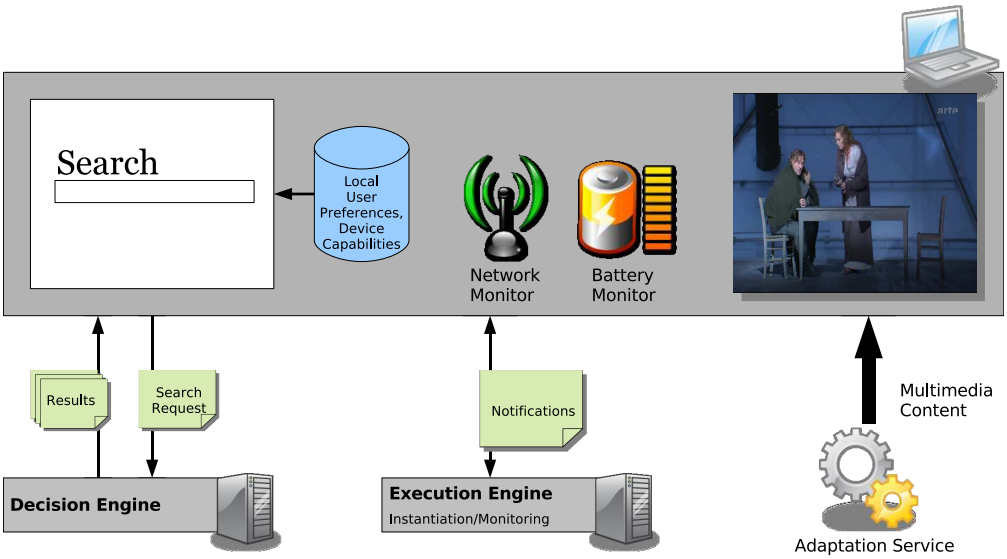

Fig. 3.2: PUMA: Client Application

2. In certain scenarios the application is run on a mobile device, and as such it has to stay in contact with the Execution Engine of the PUMA system. This way, the client application can inform the Monitoring module about changes in its state, e.g. low battery state.

3. The client application is also responsible for rendering the multimedia content it receives from the last service in the adaptation workflow.

3.2.2 User Models

Every application which performs adaptation, uses some form of user modelling. User Models [104] (UM) are not only used for storing user preferences (UP), but also allow to represent past interactions with the system (user history), interests, needs, and characteristics, such as name, age and affiliation. Normally, for every adaptive system a separate model was used. Only in recent years, with the propagation of Semantic Web technologies, and the development of user modelling languages, interoperable representations of user models appeared. Examples for such general-purpose models are the *General User Model Ontology*[82] (GUMO), which is realized as an OWL Ontology, and is used by frameworks such as the Personal Reader [2]. While GUMO tries to cover every aspect of user modelling, other approaches are dedicated towards certain aspects, such as the modelling of social networks, e.g. FOAF [44].

In the context of multimedia applications, the MPEG-7 standard provides properties to describe user preferences. Preferences are described by a key and value pair, plus an additional *importance* value, which is a numerical value between -100 and 100. These types of weights are un-intuitive and not expressive enough, so for practical reasons these should be replaced by a more sound preference model, e.g. qualitative partial order preferences [105]. Preferences of a user can also be distributed among multiple systems [27, 28].

3.2.3 Content Database: Describing and Retrieving Content

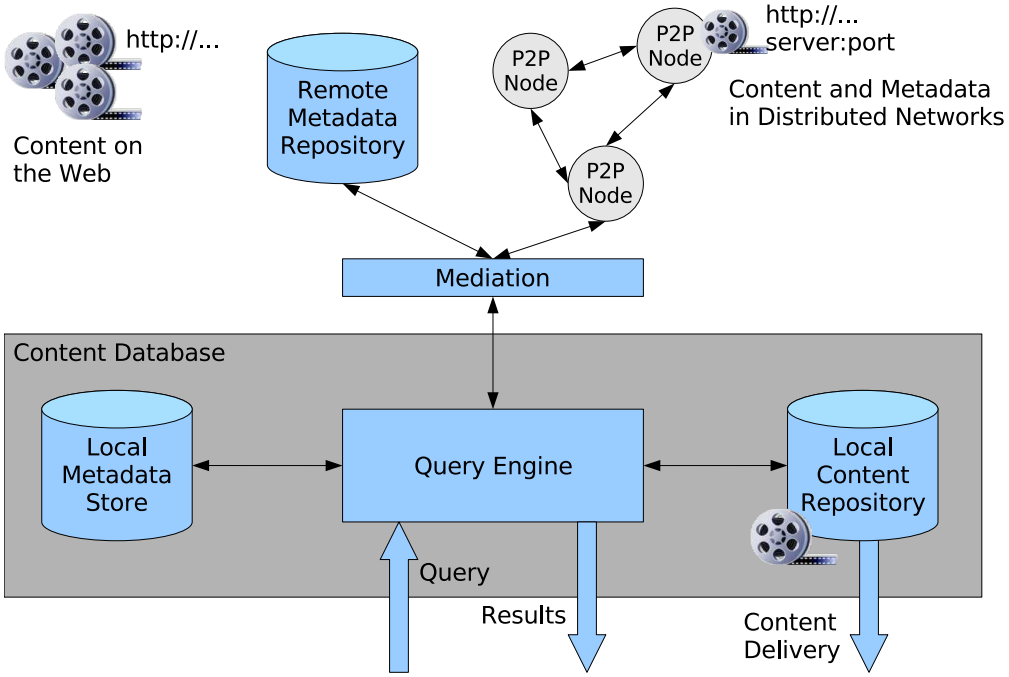

Fig. 3.3: PUMA: Content Database

Compared to encoded music, text and still images, complex multimedia content, and especially movies, are rather large. Although the multimedia data is highly compressed, even a single movie can take away gigabytes of space.

In other domains, such as the digital library domain, the description and annotation of the content is separated from the actual object data. The same process is possible for multimedia resources as well. For example, learning material, which is very often some type of multimedia document, combining slides, audio and sometimes video, can already be found in this way, e.g. by repositories such as Ariadne [60] and REASE [57], which export the metadata to search and retrieval systems, such as Edutella [49, 128, 168]

The separation of the metadata from the content resources allow authors to dictate the terms of access, while allowing their resources to be found by e.g. federated search engines. The semantic Web technologies (c.f. chapter 2.1.1) play an important role in the separation of data and metadata.

In most system however, the technical metadata of the content and is still tied together. Possible additional annotations, e.g. rankings, summaries, translations and other non-technical information is often made available by interested third-party sites. Prominent examples for such sources are the Internet Movie Database (IMDB), and other community collaboration sites which rank, tag and group together resources, without actually providing the content, which in many cases would be a copyright violation.

The PUMA systems should be able to access all content by opening a connection to an URL, in

other cases an IP address and port number is sufficient. Another problem of searching for content across different metadata repositories is that of data integration from heterogeneous sources. However, for the mediation between different metadata standards and query languages, solutions have been developed.

Mediation and Aggregation of information, while manually creating the necessary mapping rules, is a small problem, even when the system needs to bridge between P2P networks and Web services [49]. However, fully automatic integration with the high quality required for information retrieval as needed here, is still not feasible.

The functionality for selecting and ranking content is located in the Decision Engine of the PUMA systems, as it is the primary interface with the client agent. The adaptation must take into account not only the client's goals, but also can only work with the content that is actually available.

The PUMA metadata repository is a database describing resources on the web, using RDF. The database expects a query in one of the many query languages available for the Semantic web, i.e. SERQL, and will return a metadata document with results matching the query. Technically, such a repository can be a P2P system or distributed database, like Edutella [48, 47].

The information contained in such a repository is of technical nature, i.e. resolution, codec, and bitrate. Additionally, to allow the user to select specific content, it contains essential information such as title, genre, and year of recording. However, any additional information suits the selection and ranking process and increases the chance for the users to find the content they want.

3.2.4 Execution Engine: Running and Monitoring the Service Chain

A simple way to create service compositions is to create code in a programming language which calls the different services and processes the output. For PUMA, a execution engine for workflows is required. Most available engines only provide support for BPEL orchestrations. Furthermore, the PUMA multimedia workflows require constant monitoring, and changes to the workflow on-the-fly. The use of continuous data is an additional problem. As a result, we implemented a simple execution environment for initializing and monitoring the different adaptation services. Different from a Service Orchestration, the majority of communication in PUMA is directly between the Web Service.

3.2.5 Adaptation of Content with Web Services

The Adaptation Web Services, or Multimedia Services, provide the basic functionality required for Digital Item Adaptation. These services are not processing content objects in one step. Instead, they rely on streaming protocols to exchange content and other information with their partners in the service composition. These services differ from traditional Semantic Web Services, due to their additional persistent streaming connection to other services. Stream control, notifications, and monitoring are done using the standard Web Service protocols. Furthermore, in the OWL-S description the Services expose their communicative behavior to the verification system, so that protocol conformance can be checked. Important

in the DIA context is the specification of service properties, such as costs, network parameters, processing capabilities, and encoding specific details.

A detailed description of the implementation of a specific service is given in Section 5.1.

3.3 Creating Service Compositions in PUMA

The topic of this section is the creation of service compositions in the PUMA system. Any digital item adaptation —using a Service-oriented architecture or not— requires the creation of a adaptation workflow, a plan that defines which transformations or modifications are applied to a resource, and in which order.

In the PUMA architecture, the Decision Engine, and the Validation Engine are responsible for

Workflow Selection: Creating the sequence of processing steps needed in order to create a personalized version of an original multimedia object.

Service Selection: Finding service candidates, which possess the capabilities required by the roles of the workflow (semantic matching).

Interoperability Test: Selecting from the list of candidates those services, which pass a conformance check, and as such are able to communicate with each other.

3.3.1 Creating Workflows for Digital Item Adaptation

As stated before, the PUMA system uses a library of Workflow templates based on typical adaptation scenarios. Such a workflow is an abstract description of an adaptation process, but for the next steps of service composition, information about the actual source content and target device needs to be integrated.

In this section we will now describe how the different adaptation *cases* are handled. For modelling the adaptation process, the PUMA system uses *workflows*. Workflows are well suited to describe the sequence of steps needed to personalize content. Furthermore, workflows are a well tried and tested method of representing processes in many domains, but have been originally developed and applied in the office workplace and in industrial production. The preparation of workflows for content adaptation is the task of the PUMA *Decision Engine*.

There is however an important difference between the PUMA system, and many other systems, which are using reasoning and planning methods to create service compositions.

Interpreting the adaptation steps as actions, and the properties of the original and adapted media item of every step as preconditions and effects, allows the use of standard planning methods, e.g. STRIPS[51, 147]. Then, by providing the initial input and the goal format, a planner can combine those services, where the transformation process allows the content to be transformed into the desired format.

Planning systems omit the first step of creating an abstract plan. These systems directly resort to the available services in the repository when selecting the services to reach their goal. The PUMA system requires the creation of an adaptation plan before the actual service selection process. This is similar to

other approaches [98, 39] for Digital Item Adaptation. First, an adaptation plan is created. Second, the plan is executed by assigning separate processes to each step.

However, many of these systems are proxy-based adaptation architectures, and although a modular approach is used, it is not directly comparable with distributed Web Services architectures

3.3.2 The PUMA Workflow Model

The PUMA system uses an application specific workflow description language. Figure 3.4 shows the two concepts, *Roles* and *Links*, used in the PUMA workflow model (PUMA-WF).

Roles A role models an atomic task in the workflow. A task is defined by a role type, selected from a multimedia adaptation taxonomy. The role's task is to modify certain aspect of the digital item, which is processed in this step. A role only has one output connection, but multiple input links can be specified, as this is necessary for tasks where the output of two or more processes is combined, i.e. adding subtitles.

Links A link is a connection between two roles, symbolizing an interaction. It represents the data transport stream, and the properties associated with the object transmitted over it.

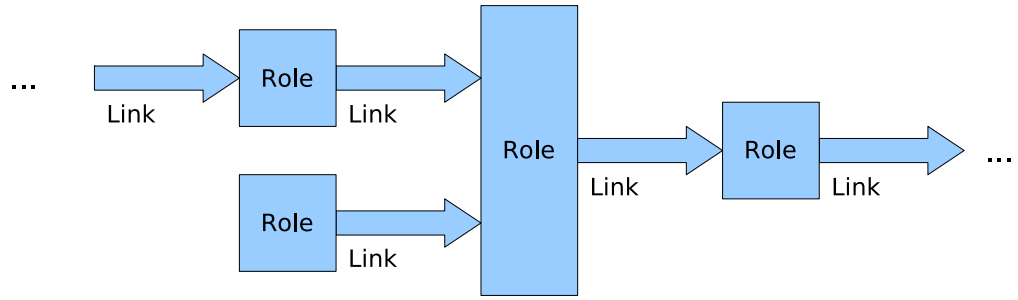

Fig. 3.4: Abstract PUMA Workflow Model: PUMA-WF

Using Workflow Templates for Service Compositions When adaptation is interpreted in an action theory, service compositions and workflow can be generated automatically [97], e.g. by providing a multimedia adaptation ontology, and a set of annotated services. Systems, such as WSMO [139] or IRS-III [51], are generic Web Service engines, which support this type of reasoning. PUMA, however, does not use this approach.

The creation of a workflow in PUMA is based on workflow *templates*, where there exists a specific workflow for each of the different adaptation scenarios. The templates are *instantiated* with concrete adaptation goals for each new user request.

The reason for this decision was based on the assumption, that not a large number of different workflows are needed, but the amount of different use-cases and scenarios is limited. The second, more decisive factor, was the amount of work required to develop and maintain a full *multimedia domain ontology*

for reasoning about adaptation and workflows. Figure 3.5 shows the structure of the PUMA workflow taxonomy.

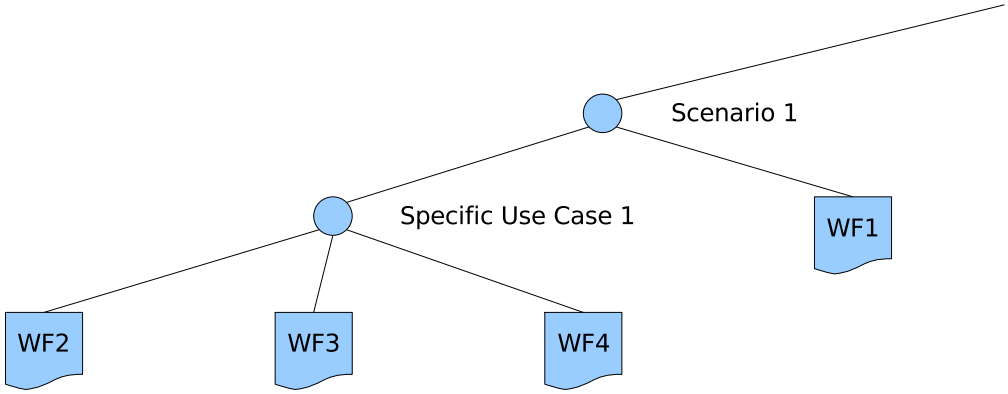

Fig. 3.5: Workflow Taxonomy Structure

The adaptation workflows used in the PUMA system are optimized for these specific scenarios, and the knowledge which enables multimedia experts to create such optimized workflows is very hard to model in an ontology for automatic planning.

The taxonomy represent different types of scenarios. Attached to each of the concepts, which represent the different use-cases, there are workflows which address the specific constraint or task specified in the scenario. On the top level the hierarchy starts with un-adapted content delivery, and then branches into different types of application scenarios, of which the use case shown in section 1.5 (on page 8) is one particular example. It is obvious, that given the constraints imposed by the capabilities of the user's client device, or by the source content objects, there exist multiple possibilities for adaptation.

One example of such a workflow variation, triggered by user preferences, might be the position of a scaling operation in a workflow. A scaling operation at the start of a processing chain can reduce the size of data that needs to be processed by all consecutive steps. While this has a positive effect on the server load and network bandwidth, it might also reduce the effective picture quality, as it reduces the available information for processing steps occurring later in the workflow. The specific knowledge about constructing these workflows remains in the hands of domain experts, and representing it as rules in a possible domain ontology for reasoning about adaptation plans, happens to be beyond the scope of this work.

The typical approach for selecting workflows is to compute a score for each workflow candidate based on a utility function (similar to service selection based on utility functions, e.g. [20, 75], which can be based on processing costs, processing delay or picture quality. The resulting ranked list of workflows will then be used in the process of creating a service composition. Multiple candidate workflows are needed, because at this point it can not be ensured, that the available services can provide all the required capabilities for adapting the content and still maintain interoperability.

In case not all roles can be filled, or if the conformance check can not find an interoperable composition, alternative workflows need to be available as a fall-back solution unless a valid composition can be found.

Service Selection For many applications, which use service compositions for accomplishing their tasks, it is common to use static, manually created orchestrations. This is true for almost any implementation in the business domain, such as the creation of business processes for e-commerce. In these approaches, new applications are generated by manually plugging together Web Services from different providers, adding some additional information about the processing of inputs and outputs of these services. These service compositions are then executed and monitored by an orchestration engine (see section 2.2.1).

Using the techniques shown in the previous section (3.3.1), the PUMA system now has acquired a ranked list of workflows. Now, for each of the tasks defined in a workflow, a suitable Web Service has to be found. However, selecting a Web Service solely based on a single attribute in a service description document is not sufficient. PUMA uses complex protocols for controlling streams and service monitoring, which is different from the "state-less" Web Services found in business processes.

As such, the Web Service candidates found by a semantic matching algorithm are additionally verified for compliance of their communicative behavior w.r.t. the protocols required for e.g. streaming and monitoring.

The selection and verification is done *a-priori*, before running the service composition. As a result, the time required for this step only adds to the startup time, it is only needed to be performed once, and not critical for the timing during the execution phase of the actual adaptation and delivery process.

However, a fast interoperability check is very important, because the network environment and the services are dynamic. Services can fail, and need to be replaced by "equivalent" substitutes. Additionally, events on the client or changes in the network topology, such as mobile network hand-overs require the re-arrangement, or even a replacement, of the adaptation workflow. Therefore, it is important that access to the information that is required for the selection and conformance is fast enough to allow for seamless content delivery.

Traditionally, Web Services are described in the Web Service Description Language (WSDL). Unfortunately, this language was developed to provide a description of the physical access to a service. This description provides information on how the exported functions can be called and specifies the syntax of the input and output data. It contains fields, which allow a description of the functionality, but these are mainly used as hints for human developers, which build applications by manually combining services.

Semantic Web Services, described using OWL-S (see 2.2.1) are more descriptive in providing information of their functionality, as it is possible to use e.g. a shared ontology to classify Web Service operations. Furthermore, this extends to the parameters of the operations as well, so that the calling instance can reason about the input and output parameters.

3.3.3 Validation of Service Interoperability

Before a service composition is started and used to adapt the content, it must be ensured that a set of services is selected for the choreography such that all participants are able to communicate with each other. The Validation Engine ensures that the communicative behavior of the services which are finally selected to form the multimedia adaptation chain, are following the required protocol. The details of this validation process is explained in detail in the next chapter (see chapter 4).

3.4 Summary

In this chapter, an overview of the PUMA architecture was given. The objectives of Service-oriented designs are code re-use and extensibility. If the interfaces of such a system are shared between applications, a service implemented for one system can be re-used by another. Furthermore, services providing new functionality can be developed and integrated easily. Normally, service compositions are created manually, as the traditional specifications do not include information about the interactive behavior of services. In WSDL, only the messages syntax is specified. The Process Model of Semantic Web Services can be used to specify a service's supported message exchanges in detail. Furthermore, the service descriptions provided by OWL-S can contain references to shared vocabularies about capabilities, which makes service selection much easier.

In the PUMA system, for each supported content adaptation scenario, a specific workflow exist. The workflows contain the tasks, which have to be performed for adapting content. For each of these tasks, or roles, a service has to be selected, based on its capabilities.

All components in the PUMA system, and not only the services which perform the content adaptation, are part of a complex choreography, where they interact with each other by exchanging messages.

In the next chapter, the verification method for ensuring interoperability between the participants is presented.

Chapter 4

Verification of Interoperability and Conformance

Service-Oriented Systems can be build upon two different models, Choreographies and Orchestrations. The properties of these models have been already discussed in Section 2.2.1. Nowadays, the Orchestrations and Choreographies are mostly designed manually, although automatic composition systems exist. The PUMA architecture relies on a semi-automatic way of service compositions. The choreography is selected from a library of engineered adaptation workflows. The selection of services, however, is performed fully automatically, and as such, the only information available for this task is the service description. To avoid interoperability problems in the manual composition process, the process designer also relies on written documentation, access to the program code, and experience with existing service implementations.

In Section 3.3, the service composition approach, that is used by PUMA, was presented. As such, the system already has a set of adaptation workflows available for the verification step that is presented in this chapter. For each of the roles defined within the workflows, a ranked list of candidate services was created. The interoperability check can be performed completely independent from the workflow creation and service selection. The only requirement for the verification is a global interaction protocol specification, and the service descriptions which specify the messages a service can send and handle.

However, the verification system expects the service candidate and workflow lists to be ranked according to their utility for the adaptation process, i.e. best services first. For example, depending on the user's preferences, this could either be the service with the lowest costs, or the service providing the best picture quality.

Protocol Example The adaptation workflows, used in the PUMA system, define roles, which need to be filled with service implementations. Similar to a choreography, the roles define the possible ways in which services can interact with other participants. The protocols used in the context of PUMA were

designed for content search and retrieval, the control of multimedia streams, service monitoring and event notification.

To illustrate the details of the verification method, a simplified excerpt from the protocol for stream control will be used. However, the technique presented is not limited to multimedia services and protocols, but can be easily adapted for other application domains.

The messages used for controlling the content exchange between the services, involved in the adaptation workflow are very similar to the requests defined by the RTSP protocol, such as: setup, play, pause and teardown. One of the many possible ways to present the protocols are with sequence diagrams. The UML Sequence Diagram in Figure 4.1 shows the interaction between three roles in the systems: the client device, a single adaptation service, and the monitoring system. At any point after initialization, the monitoring system can check the other services for their status. The client can send commands, which have to be acknowledged by the adaptation service, to start and stop the stream.

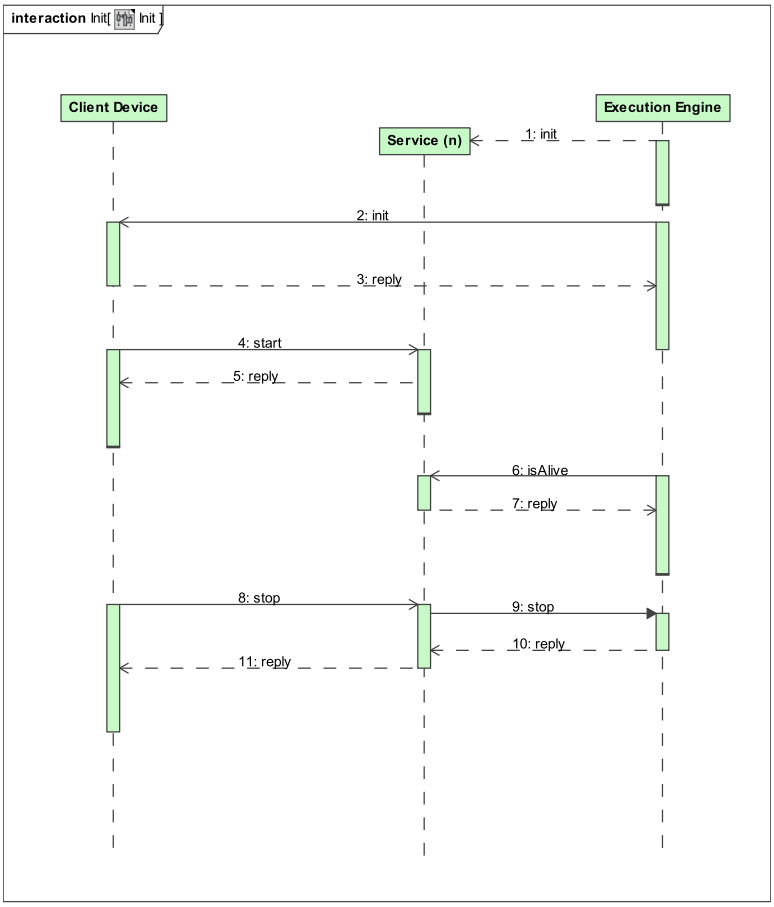

Fig. 4.1: Sequence Diagram of the Initial Communication between Client and other Services.

4.1 Interactions between Services

The interaction between participants, either services in a service composition, or agents in a multi-agent system, involves the exchange of messages. In other literature, i.e. in philosophy of language, a sequence of such messages is often called a *speech act* [10, 145, 146, 16, 17, 17, 15]. The exchange of messages is also the foundation of the two Agent Communication Languages FIPA [69] and KQML [64]. Modelling the conversations between the peers in a distributed system, is essential for checking if they are interoperable with each other. In the context of the PUMA system, and for Web Services in general, remote method invocations, e.g. via SOAP, can be seen as the exchange of such messages as well. The interactive behavior of the communicating participants is reflected in conversation *protocols* and *policies* [58]

Definition 1.5 (Conversation Policy)**.** The *conversation policy* is a program that describes the actual communicative behavior of an interactive entity, such as a Web Service.

Definition 1.6 (Conversation Protocols)**.** The *conversation protocol* specifies the *desired* communicative behavior of the interactive entities. The protocol is an exact specification of the sequences of messages that can be exchanged between the involved parties and that is considered legal.

Both protocols and policies can be modelled using speech acts. For the analysis of conversations, it is important to distinguish between incoming and outgoing messages.

Definition 1.7 (Speech acts)**.** A speech act or message exchange has the form $m(a_s, a_r, l)$ where

- m is the type of message, event or other performative,
- a_s is the sender,
- a_r is the receiver, and
- l is the message content.

The following notation for conversations will be used when the receiver or sender of a message is clear from the context, or is not relevant for the observation:

incoming message	$m?$
outgoing message	$m!$

A message exchange $m(a_s, a_r, l)$ will then be written as $m?$ from the point of view of the receiver a_r, and as $m!$ from the point of view of the sender a_s. A sequence of such speech acts, which is a dialog between a set of parties, will be denoted by the term *Conversation*.

In a dialog, a message m, *sent* by one system, is *received* by another system, which is the *opposite* behavior, and can be written as $\overline{m}$.

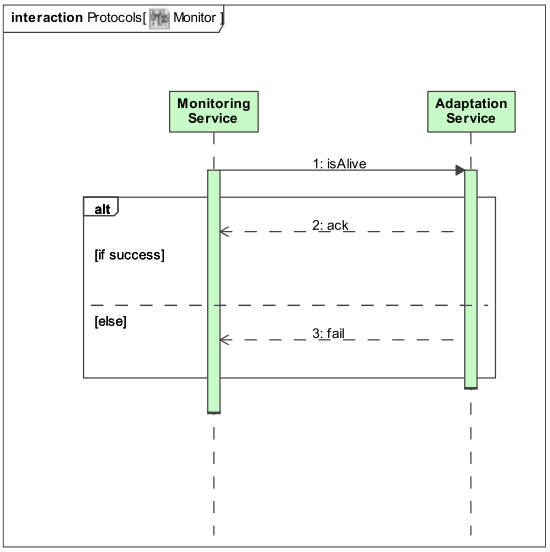

Fig. 4.2: Dialog: The Monitor inquiring the Status of a Service

Figure 4.2 shows a very simple conversation between the two participants, “Monitoring Service” and “Adaptation Service”. In this example, the adaptation service expects to receive a message “isAlive”, and will respond to such an event by sending a “reply” to its partner. Policies and Protocols are *collections* of conversations.

For protocols, the set of conversations corresponds to all the possible message exchange sequences allowed between the participants. A policy is reflecting the implementation of a system. It is a specification of which messages a participant can handle and send. Then, the set of policies are specifying all the possible conversation supported by the system, but during *run-time* of the system, some of them might not be performed at all.

4.2 Finite State Automata

Labelled transition systems (LTS) [9]. are often used for modelling the interactions, and consecutively the state of a dialog between systems.

A labelled transition system is a tuple $(S, L, \rightarrow)$, where S is a set of states, L is a set of labels and $\rightarrow \subseteq S \times L \times S$ is a *ternary* relation of labelled transitions. The labels in this definition can represent many things, depending on the language of interest. In our context, this includes expected incoming messages and messages sent during a transition. For two states $s_1, s_2 \in S$, and a label $l \in L$, the transition $(s_1, l, s_2) \in \rightarrow$ is written as $s_1 \xrightarrow{l} s_2$. This represents the fact that there is a transition (an incoming or outgoing message labelled l), such that it advances the protocol into the next state.

When modelling conversations in PUMA, *finite state automata* (FSA) are used to represent the speech

acts, for both the service behavior (policies), and the interaction protocols. FSAs differ from state transition systems in several ways:

- In a FSA, the set of states is finite, whereas in LTS, the set of states is not necessarily finite, or even countable.
- In a FSA, the set of transitions is finite, whereas in LTS, the set of transitions is not necessarily finite, or even countable.

Definition 2.8 (Finite State Automaton (FSA))**.** A finite state automaton is a tuple (S, s_0, Σ, T, F), where

S	is a finite set of *states*,
s_0	is a distinguished initial state, $s_0 \in S$,
Σ	is the *alphabet*, a finite set of *labels*,
T	is a set of *transitions*, $T \subseteq (S \times \Sigma \times S)$, and
F	is a set of *final* states, $F \subseteq S$.

The labels Σ, representing the messages, are build from two sets of speech acts, corresponding to the two different types of messages associated with each conversation, incoming and outgoing messages.

The notation for finite state automata, used in this work corresponds to the one used by SPIN [86], a popular FSA-based model checker used for this type of verification [12].

Additionally, [86] defines a *dot* notation for components of a FSA. For example, the notation $A.s$ is used to denote that a state s belongs to an automaton A. This is not to be confused with the notation used by π-calculus, which is the foundation for modelling processes in languages such as WS-CDL.

The corresponding automaton for the role of the monitoring service, created from the protocol example given in Figure 4.2, is shown in Figure 4.3. This particular protocol role is part of a larger global protocol, and can be initiated as often as deemed necessary by a monitoring service.

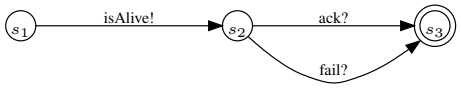

Fig. 4.3: FSA of the role of the Monitoring Service

Definition 2.9 (Strings and Runs)**.** A *run* σ of a finite state automaton $A = (S, s_0, \Sigma, T, F)$ is an ordered, possibly infinite, set of transitions (a sequence)

$$\{(s_0, l_0, s_1), (s_1, l_1, s_2), (s_2, l_2, s_3), \ldots\}, \text{such that}$$

$\forall i, (i \geq 0), (s_i, l_i, s_{i+1}) \in A.T$, while the *sequence*

$$l_0 l_1 l_2, \ldots$$

is the corresponding string $\bar{\sigma}$, $l_i \in A.\Sigma$.

The finite set of labels $A.\Sigma$ is a set of speech acts, and the *strings* produced by the FSA, represent the set of possible *Conversations*. A sequence of such speech acts is called an *accepting* run, if after all the input is processed, the automaton has reached one of the final states.

Definition 2.10 (Standard Acceptance)**.** An *accepting run* σ of finite state automaton $A = (S, s_0, \Sigma, T, F)$ is a *finite* run in which the final transition (s_{n-1}, l_{n-1}, s_n) has the property that $s_n \in A.F$.

Given an FSA A, every state $s_i \in A.S$ is considered *alive*, if s_i has occurred as an element of an accepting run σ, such that there is a finite run $\sigma' = \{(s_i, l_i, s_{i+1}), \ldots, (s_{n-1}, l_n, s_n)\}$ and $s_n \in A.F$.

4.3 Representing Protocols and Policies

Usually, protocols and policies are not directly specified in state automata. The first step in verification approaches is to model the protocols and policies of communicating systems in an expressive language.

The languages for describing the communicative behavior of entities found wide acceptance in the Multi-agent systems domain [58]. Among the languages that can be interpreted as state automata are the so called process algebras, such as the *Calculus of Communicating Systems* (CCS) [121], π-calculus [141], the *Algebra of Communicating Processes* (ACP_τ) [26], and *Communicating Sequential Processes* (CSP) [85].

When extending these approaches to Web Service systems [17], and specifically in the context of PUMA, the languages used for describing the communicative behavior of Web Services become the source for the transformation into finite state automata. However, this exceeds the capabilities of WSDL, which is not sufficient to model the different states of more complex protocols. The basic communication with a Web Service is limited to the request-response dialog defined by the client-server paradigm.

For the simple type of interactions, conformance between protocols and policies can be ensured without the need of OWL-S, Sequences and Choreographies. This is the level of the Web Service operations, which explicitly follow the client-server paradigm. Always, one partner, the client, is starting the conversation by sending a request message, and the server is responding with a result message. For traditional web services, the protocol *has finished* after this simple message exchange, which is also called an *Invocation*.

However, if services are no longer state-less, but advance in the protocol by exchanging messages, the continuation of the conversations has to be planned and validated. As long as the standard Web Service mechanisms are used, conversations are always performed in the form of a request-response dialog, with a possible role change after each dialog. In other environments, such as peer-to-peer systems or by using other transport protocols, this property is no longer true.

As already stated, the FSAs which are required for the validation of interoperability in the context of PUMA can be derived from the different specification languages used for Web Services and Service Compositions. All of the Orchestration and Choreography languages are founded on some form of process algebra, e.g. WS-CDL is based on π-calculus. Outside of these methods, there exist other generic approaches from software engineering, such as UML Sequence and Activity diagrams, which are used to describe the implementation of a system, and can be transformed to state machines representing a protocol or policy.

- Sources for policy specifications:

  OWL-S Process Model: The Process Model of OWL-S [114], can be used to describe the behavior of a Semantic Web Service. OWL-S supports the creation of composite actions, such as *Sequences* and *Choices*. With this construct, it is possible to describe the message exchanges implemented by the concrete Web Service.

  WS-BPEL: With BPEL it is possible to specify structured Activities, such as *Sequences*, *Loops* and *Choices*. Together with the basic activities, such as invoking and providing Web service operations, it is possible to describe the implemented conversation policies.

- Specification of conversation protocols:

  WS-CDL: A choreography is a description of the interactions between entities. In WS-CDL [100] it is possible to specify abstract roles and communication channels. Ordering structures such as *Sequence*, *Parallel*, and *Choice* can be used to describe complex message exchanges between participants (roles) via channels.

  (A)UML: A UML *Sequence* diagram (see Figure 4.1 for an example) which describes the message exchange between roles [130], can be converted to finite state automata.

However, transforming such specification into automata can be done in different ways. The related work about verification and conformance in Web Service architectures is not very specific on how the mapping between web services *operations* and *messages* and the speech acts required for analyzing the properties of the systems is exactly performed.

One interpretation, used by Schifanella [142], and Baldoni et al. [18], suggests the use of semantic matchmaking and other methods for mapping a “signature”, based on Web Service operations and the IOPE elements to a form of “capability”.

4.3.1 Specifying Policies with OWL-S

In the PUMA system, the Web services' communicative behavior (their interaction policies) is stored together with the rest of the information about the services. Usually, a WSDL document is used for describing these properties. As already stated in chapter 3.3, PUMA uses the Ontology for Web Services (OWL-S). For selecting service candidates for the roles in a workflow, the Ontology for Web Services was already extended with Profiles which allow the selection of services based on a Taxonomy of Service capabilities. The possibility to extend RDF Ontologies enables different approaches to define interactivity.

We decided on a notation which re-uses the Service Model of OWL-S to model the interactive behavior a Web service, based on Processes, IOPE elements, and control structures to build composite processes, e.g. *Sequences* (see Figure 2.9 on page 30).

An atomic process from the OWL-S ontology, corresponds to a message (Web Service operation) of the described Web Service. The inputs, outputs, preconditions, results (IOPE) and other available properties are considered when mapping a process to a messages $m(a_s, a_r, l)$. to simplify this mapping step, processes can refer to a shared taxonomy of standard protocol message names and categories.

However, the process model of generic OWL-S description, only specifies the behavior from the view of the client that interacts with the service. This can be used to model the different states of a service w.r.t a protocol, but only includes one direction of message flow, that is the *incoming messages* expected by the service. For the validation to work, the complete communicative behavior of the service needs to be available, that is, the messages the service will send to other services.

Basically, an atomic process representing an incoming message, has to be mapped to the elements in a WSDL document, the so called *Grounding*. However, atomic processes which correspond to *outgoing message* must be associated with abstract service definitions in remote WSDL documents. For clearly distinguishing the difference between ingoing and outgoing messages in a policy, the property policyPart is used.

Figure 4.5 depicts the state automata for the sequence defined in the OWL-S Process model given in Fig. 4.4. This policy specifies, that after sending the "Setup" message, the service expected the "Ok" message from the other participant.

4.3.2 Modelling Conversation Protocols

For specifying communication protocols, there exist many description languages. In the Web Service domain, the message exchange between services is described on the lowest level of the orchestration and choreography languages. In the PUMA project, we derive these service compositions from the adaptation workflows.

Choreographies are best suited to describe the conversation protocols required for verification. The officially recommended description language for specifying choreographies is WS-CDL (see 2.2.1). What differentiates orchestrations and choreographies is the different view on interactions between entities. In orchestrations, there is a single point of view on the execution of a process, i.e. a central single instance

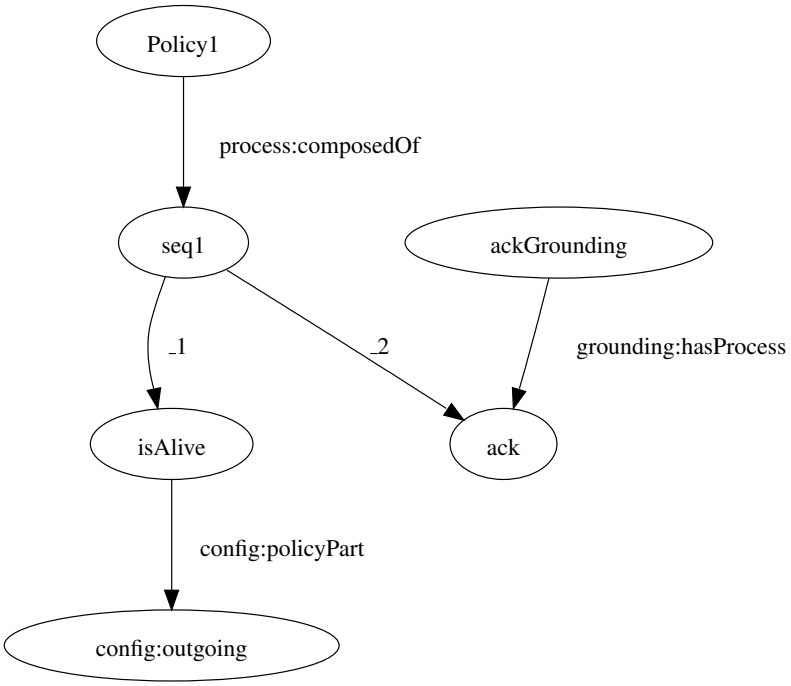

Fig. 4.4: Modelling Policies with OWL-S Processes

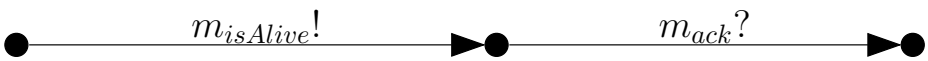

Fig. 4.5: FSA of the OWL-S Process Model (see Fig. 4.4)

controlling the exchange of information. In a Choreography, the interactions between participants are modelled globally.

The basic building blocks of a WS-CDL choreographies are *Interactions*. Interactions capture the information exchanged between collaborating participants, but also additional data about changes in an entity's behavior, or the concrete values of the content of the exchanged messages. For simplification, certain aspects of modelling interactions in Choreographies, such as information alignment, will be skipped.

The lowest level component in a choreography are Activities, which provide, among other ordering structures, notations for sequences of interactions [101].

The most important information contained in a WS-CDL Interaction is the Operation, which specifies what the recipient of the message should do with a message that is received. The operation and the message itself will be the keys for checking the conformance of protocol interactions and implemented policies. Furthermore, the interaction specifies a *From Role* and a *To Role*, defining the two roles which are involved in the message exchanges as sender and receiver.

The translation of WS-CDL Choreographies into state transitions systems has been explored in var-

ious related work. In [67], Foster et al. use a model checker for comparing the message exchange implemented by an orchestration given in WS-BPEL, with the interactions defined by a Choreography in WS-CDL.

In Example 4.3.1, excerpts from a Choreography specification are shown, which is based on the dialog between the Monitoring Service and an Adaptation Service (see Figure 4.2).

Example 4.3.1 (Example interactions in WS-CDL)**.**

```
<roleType name="Monitor"> ... </roleType>
<roleType name="Service"> ... </roleType>

<relationshipType name="MonitoringServiceRelationship" />
<variableDefinitions> ... </variableDefinitions>

...

<sequence>
  <interaction name="monitoringRequest"
    channelVariable="monitoringChannel" operation="aliveRequest">
    <participate relationship="MonitoringServiceRelationship"
                 fromRole="Monitor" toRole="Service" />
    <exchange name="isAliveMessage" informationType="isAliveType"
            action="request"> ...
    </exchange>
  </interaction>

  <choice>

  <interaction name="serviceAck"
    channelVariable="serviceChannel" operation="ackReply">
    <participate relationship="MonitoringServiceRelationship"
                 fromRole="Service" toRole="Monitor" />
    <exchange name="responseMessage" informationType="ackResponseType"
            action="request"> ...
    </exchange>
  </interaction>
  <interaction name="serviceFail"
    channelVariable="serviceChannel" operation="failReply">
    <participate relationship="MonitoringServiceRelationship"
                 fromRole="Service" toRole="Monitor" />
    <exchange name="failMessage" informationType="failResponseType"
            action="request"> ...
    </exchange>
  </interaction>

  </choice>

</sequence>
```

4.4 Interoperability and Conformance Checking

The goal of the verification step in the PUMA architecture, is to check, whether the services which form a service composition are following the same protocol when communicating with each other. This check actually requires two different properties of the system to hold:

- the *conformance* of an implemented message exchange with a global interaction protocol, and
- the *interoperability* between services.

For verifying the interoperability and conformance of a service's behavior w.r.t. a required protocol, different methods have been developed by various authors. In the previous section 4.1 the model for modelling the conversation policies and interaction protocols in the PUMA system, which is based on finite state automata, has been described. Before any type of verification can be performed, the intuition behind the terms conformance and interoperability needs to be explained and formalized.

The PUMA system was designed to allow its extension with services created by external providers. Interoperability between the participants could have been enforced. For example, this is possible by requiring each service to implement the same interface, as was done by the Personal Reader Framework [2, 84]. Instead, the system uses the observable behavior of the available services. Unfortunately, the observable information required for this type of conformance checking is not included in the WSDL descriptions of Web Services. For representing the message exchange between services the process model of OWL-S service descriptions is required.

Conformance checking (and model checking) has a long tradition in Multi-agent Systems (MAS) [53] and Software Engineering [25].

For a system of interactive entities, *Interoperability* is a desired property. The verification of interoperability is essential for the understanding of how a particular system works [16, 17].

Normally, the communication between the participants of such a composition is driven by, usually published, protocols. However, for the verification of agent interoperability, only the behavior of the agent themselves need to be examined.

Unfortunately, this way of checking for agent interoperability requires knowledge of the states and conversations of *all agents* involved in the interaction.

However, in a large-scale heterogeneous distributed environment, such as PUMA, creating this global view of all entities is time consuming, subject to constant changes and not practical. Also, the actual computation for checking the interoperability usually exceeds the user's expected wait-time [133].

To allow fast verification of interoperability and protocol conformance, we adopt the idea of not checking the interoperability by analyzing the complete system of interacting services. Instead, a check is made for each role in an interoperable protocol. If the role can be *substituted* with a service, i.e. a *conformance* test of the service implementation against the protocol role succeeds, then the service is interoperable w.r.t. the protocol.

The roles in the protocol are, by definition, interoperable, which of course has be to checked with a separate verification. The PUMA systems re-uses ideas from existing multimedia transport protocols, e.g. RTSP [143], as such, the design of the protocol is not discussed here.

In order to ascertain the interoperability of a service, it should be sufficient to prove that it fulfills the requirements of the respective role in the protocol.

Definition 4.11 (Interoperability w.r.t. an interaction protocol)**.** Interoperability w.r.t. an interaction protocol is the capability of a set of entities to produce a conversation that is legal w.r.t. the rules of the system (i.e. against an interaction protocol)

Furthermore, there exist two types of verification, depending on *when* the conformance check between policies and protocols is performed:

1. The *a-priori* conformance verification is performed before the actual execution of the service composition. For this, all the policies and protocols must be evaluated.

2. The *run-time* conformance validation only checks, if the messages exchanged are legal in the, at that time, active state of the protocol.

The run-time validation is performed for the *compliance* check, to see if the actual implementation of a service corresponds to the advertised behavior.

In the PUMA system, the conformance check is done a-priori, before starting the content adaptation and delivery. Moreover, the environment in which the content adaptation is performed is subject to changes during the life-time of the session. These changes may require modifications to the workflow, such that additional checking has to be performed during run-time of the service composition. When a service fails, it has to be exchanged with an equivalent replacement service. The conformance check will ensure that the new service is able to fill the role of the failed service. However, although the conformance check is performed during run-time of the service composition, it can still be considered an a-priori verification, because it is performed *before* the new service is integrated and actively starts exchanging messages with the other participants.

The foundation for modelling the interactions between entities, which is required for checking the interoperability is given by Arnold and Holtzman [86]. Both authors define two *products* of state machines to model the composition of conversations, and the actual exchange of messages between entities. These products are important for model checking, with systems such as SPIN [86]

The approach presented here is based on work by Baldoni et al. [17, 16], Singh et al. [53], their joint work [15], and personal communication with Matteo Baldoni. Furthermore, ideas from conformance checking for asynchronous message passing systems [137, 121] will be used to define *interoperability*, *conformance*, *alignment*, and *compatibility* between services and protocols.

Free Product The operator "$\times$" defines the *asynchronous* or *free* product of multiple automata [9, 86]. This product is required for representing all the possible conversations which are created by a set of policies, implemented by a service, or defined by a combination of individual protocol roles.

Definition 4.12 (Free Product)**.** Let $A_1, \ldots, A_n$ be n FSAs. The free product, written $\prod_{i=1}^{n} A_i$, is the new finite state automata (S, s_0, Σ, T, F), where

S	is the set $A_1.S \times \cdots \times A_n.S$;
s_0	is the tuple $(A_1.s_0, \ldots, A_n.s_0)$;
Σ	is the set $A_1.\Sigma \times \cdots \times A_n.\Sigma$;
T	is the set of tuples $(\langle A_1.s_1, \ldots, A_n.s_n\rangle, \langle l_1, \ldots, l_n\rangle, \langle A_1.s'_1, \ldots, A_n.s'_n\rangle)$, such that $(A_i.s_i, l_i, A_i.s'_i) \in A_i.T$, for $i = 1, \ldots, n$; and
F	is the set of tuples $(A_1.s_1, \ldots, A_n.s_n) \in A.S$ such that $s_i \in A_i.F$ for $i = 1, \ldots, n$.

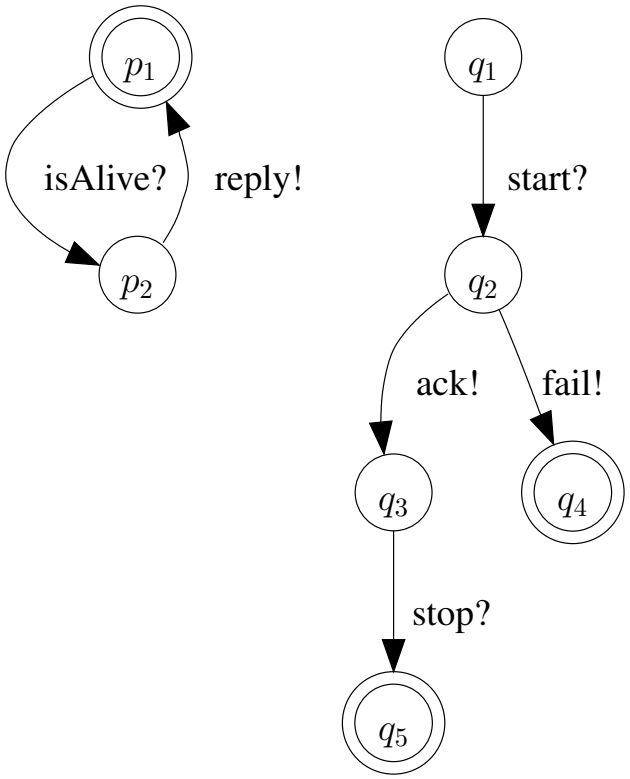

Fig. 4.6: Free Product: Monitoring and Start/Stop Protocol

As an example, consider a service which precisely implements the two protocols (see Fig. 4.6) required for monitoring, $A_M = \{p_1, p_2\}$, and the simple start/stop control protocol for stream control,

$A_C = \{q_1, q_2, q_3, q_4, q_5\}$. All possible conversations generated by this service can be expressed by the free product $A = A_M \times A_C$ (see Figure 4.7).

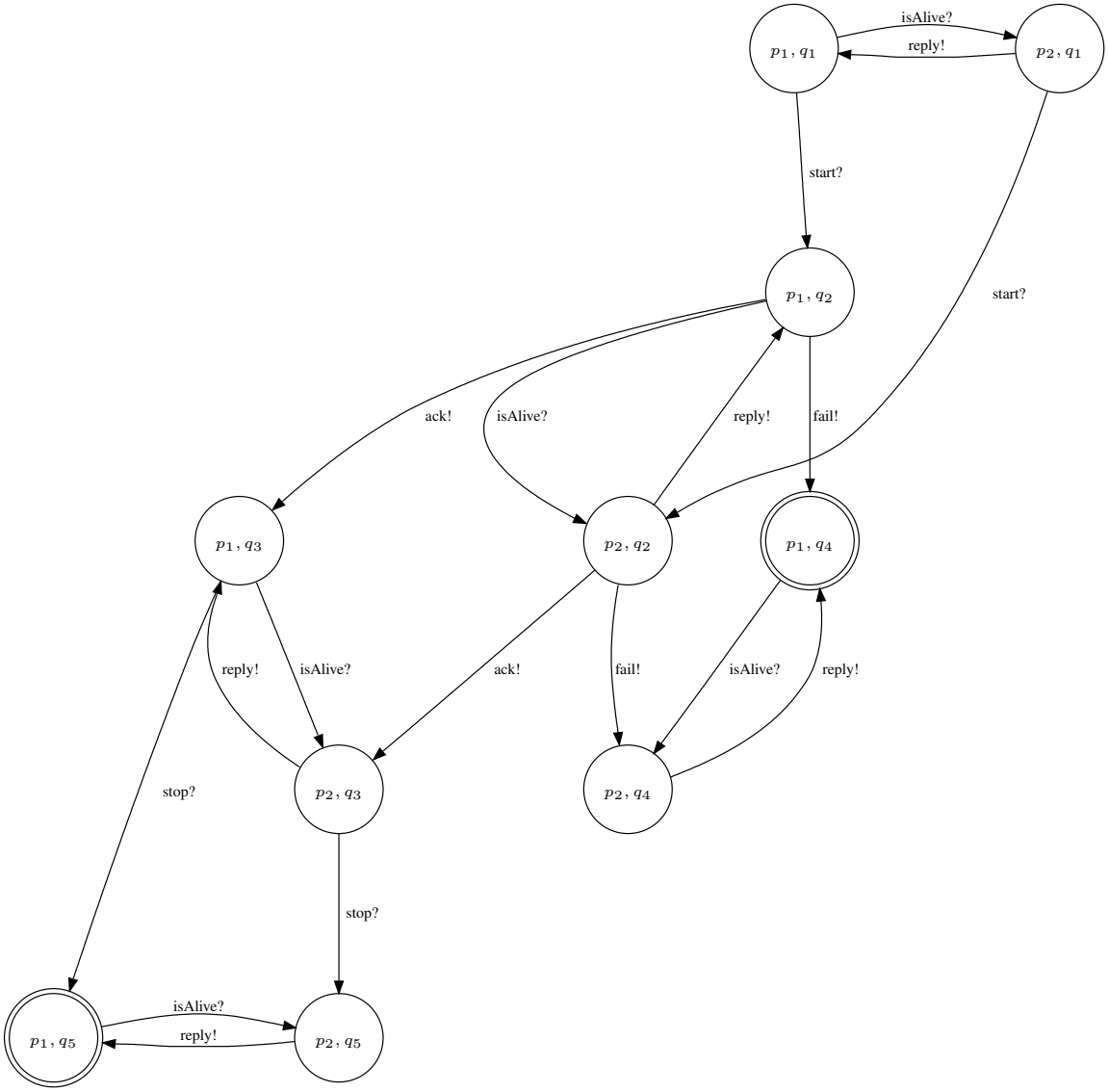

Fig. 4.7: Free Product: Result Automata representing all possible conversations

To support modelling concurrency of entities, with state automata, where only one system is allowed to be active at a time, while the other systems remain in their current state, an *empty* transition $t = (s, \epsilon, s)$ is added to every state $s \in A.S$ of every FSA.

However, there exist different semantics on how transitions between the global states of the combined system are modeled. *Interleaving* only allows one system to be active at a time. In the case of SPIN [86],

the transitions $A.T$ are driven by a single action $l \in A.\Sigma$, such that $A.T$ is a set of tuples of the type $(\langle s_1, \ldots, s_n \rangle, l, \langle s'_1, \ldots, s'_n \rangle)$ and $s_i, s'_i \in A_i.S$.

In this work, the Cartesian product is used to create a *global action* vector $L = A_1.L \times \ldots \times A_n.L$, to keep track which of the entities are active during a transition.

4.4.1 Interoperability

For modelling the interactions between systems, an additional type of composite state automaton is needed. Typically, the *synchronous* product, defined by Holzmann [86] and Arnold [9], written $\otimes$, is used. The difference between the free product and the synchronous product is in the set of *transitions* of the product automata.

The synchronous product, and the causal product given by Singh et al. [53] provide the required notion of *transition of the system* as the result of a successful communication, i.e. the sending of a message $m!$ by one entity, *joint* with its reception $m?$, which is handled by another participant.

Definition 4.13 (Transitions)**.** Let A_1 and A_2 be two FSAs. The function

$$\text{transition}(\langle A_1.s_i, A_2.s_j \rangle),$$

returns the set of transitions which lead to the follow-up states,
$\{\langle A_1.s_{i+1}, A_2.s_{j+1} \rangle | \exists m, (A_1.s_i, m, A_1.s_{i+1}) \text{ and } (A_2.s_j, \bar{m}, A_2.s_{j+1})\}$.

Similar to Def. 2.9, a sequence of transitions is a *run*. If such a sequence of transitions allows both component automata to go from their *initial state* to one of their *final states*, the resulting run is called a *successful communication*. This is similar to the definition of acceptance 2.10

Definition 4.14 (Runs)**.** Let A_1 and A_2 be two FSAs. The run σ is a sequence of transitions

$$(\langle A_1.s_i, A_2.s_j \rangle, m_1, \langle A_1.s_{i+1}, A_2.s_{j+1} \rangle, \ldots, \langle A_1.s_{i+n-1}, A_2.s_{j+n-1} \rangle, m_n, \langle A_1.s_{i+n}, A_2.s_{j+n} \rangle), \text{such that} \quad (4.1)$$

$$\langle A_1.s_{i+k+1}, A_2.s_{j+k+1} \rangle \in \text{transitions}(\langle P.s_{i+k}, Q.s_{j+k} \rangle)$$

Intuitively, interoperability for a set of services requires that the communication is *stuck-free*. That is, whatever state was reached in the interaction, communication is not blocked, and each of the services in the composition will reach one of its final states.

Definition 4.15 (Successful communication)**.** Given the two FSAs A_1 and A_2, a *successful communication* is a run σ from the global initial state $\langle A_1.s_0, A_2.s_0\rangle$ to the state $\langle A_1.s_i, A_2.s_j\rangle$, such that both $A_1.s_i$ and $A_2.s_j$ are final states, $A_1.s_i \in A_1.F$ and $A_2.s_j \in A_2.F$.

Decision to Lead or Follow The additional conditions which guarantee a successful communication can be described as follows.

- At every point in the conversation, the participant which is the *sender* of a message, had to make a *choice*. It must not select an alternative which can not be handled by its interlocutor.
- On the opposite side, the participant which has to *follow* this choice, must be able to handle all the alternatives that can be chosen by its interlocutor.

When both of these assumptions hold, the two participants are in *compatible* states. Figure 4.8 sketches the different notions of compatibility and alignment between states, which lead to conformance and interoperability between protocols and policies.

In order to model the notion of a participant leading or following another entity in the system, the finite state automata (see Def. 2.8) needs to be extended with additional information about the type of a state, i.e. if it is a regular state, leading "$\odot$", or following "$\oplus$".

Definition 4.16 (Extended Finite State Automaton)**.** An extended finite state automaton is the tuple $(S, s_o, \gamma, \Sigma, T, F)$, where

S	is a finite set of states,
s_0	is a distinguished initial state, $s_0 \in S$,
γ	is a function from S to the set $\{\odot, \oplus, \epsilon\}$,
Σ	is the alphabet,
T	is the a set of transitions, $T \subseteq (S \times \Sigma \times S)$, and
F	is a set of final states, $F \subseteq S$

In [15], the interpretation of a state as leading or following is defined to be orthogonal to the actual behavior of the state. That is, the type of state is independent from the transitions originating from the state, be it sending $(s_i, m!, s_j)$, or receiving $(s_i, m?, s_j)$ ones.

However, this would require to extend the available description languages with additional semantic annotation for specifying the choice for leading or following a conversation in the protocols and policies. Instead, a definition closer to the original notion of *emission* and *reception*, given by Bordeaux et al. [38] will be used to assign a label l to each state, $l \in (\odot, \oplus, \epsilon)$.

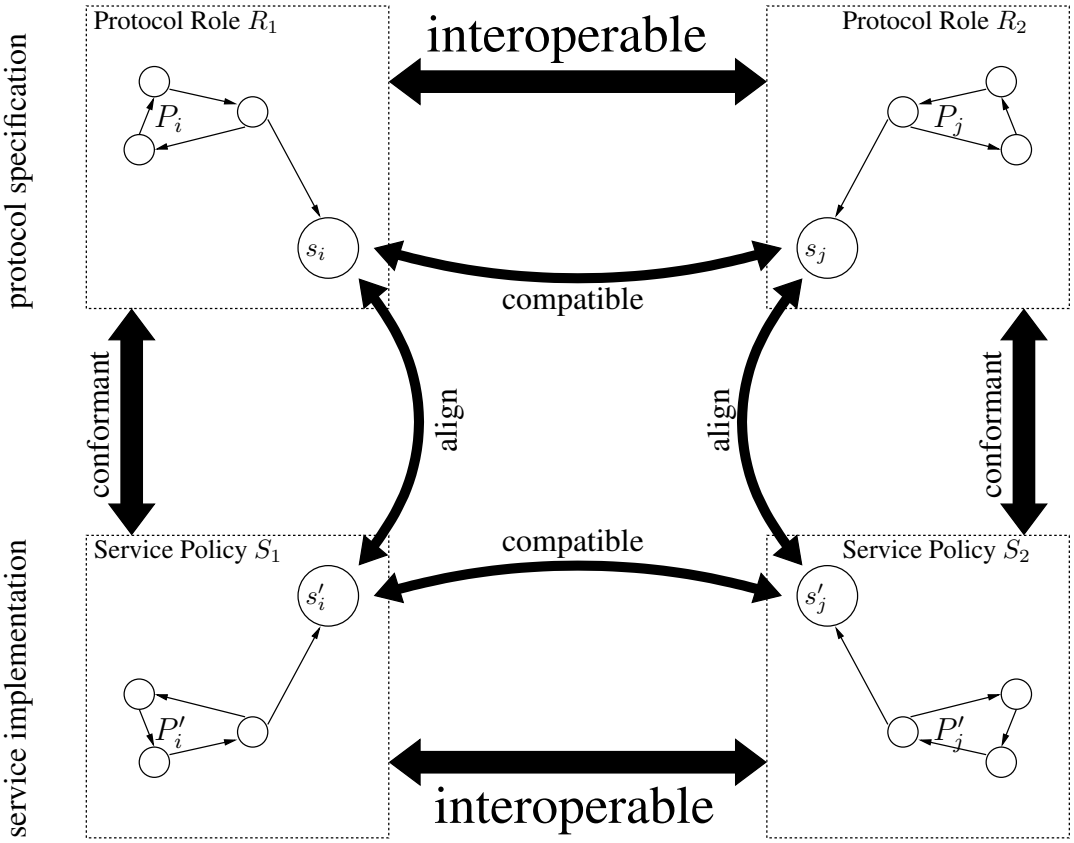

Fig. 4.8: The relations between interoperability and conformance (source: [15]).

However, this requires the protocols and policies to fulfill additional constraints. The simple protocols used in the PUMA system use only one type of message at each state of the conversation, such that the each state can be mapped to one of the three different categories.

Remark 1. **Choice**.

Let A be a finite state automata $(S, s_o, \gamma, \Sigma, T, F)$. The function $\gamma(s_i), s_i \in A.S$ assigns to s_i a label $l_i \in \{\odot, \oplus, \epsilon\}$, such that

- s_i is a $\odot$-state (lead), if all transitions originating from state s_i are *emissions* $(m!)$, the set of leading states is denoted $A.S^{\odot}$,
- s_i is a $\oplus$-state (follow), if all transitions originating from state s_i are *receptions* $m?$, the set of following states is denote $A.S^{\oplus}$, or
- a state is a regular state, if it has no emissions or receptions.

Definition 4.17 (Compatibility)**.** Let R_1 and R_2 be two roles or services, and A_1 and A_2 their corresponding FSAs. The state $A_1.s_i$ is *compatible* with $A_2.s_j$, iff:

$$A_1.s_i \in A_1.S^{\odot}, A_2.s_j \in A_2.S^{\oplus}: \quad \text{message}(A_1.s_i) \subseteq \text{message}(A_2.s_j)$$
$$A_1.s_i \in A_1.S^{\oplus}, A_2.s_j \in A_2.S^{\odot}: \quad \text{message}(A_1.s_i) \supseteq \text{message}(A_2.s_j)$$

Compatibility also holds, whenever both $A_1.s_i$ and $A_2.s_j$ are regular states and have no emissions or receptions.

From the intuitive definition of interoperability, it is now possible to define interoperability as a verification of the compatibility of two component systems. This is done by verifying compatibility for every pair of its states, that can be reached by the defined transitions.

Definition 4.18 (Interoperability)**.** Let R_1 and R_2 be two roles or services and A_1 and A_2 their corresponding FSAs. R_1 and R_2 are interoperable, iff there exists a binary relation $\mathcal{R}$, such that:

1. $A_1.s_0 \mathcal{R} A_2.s_0$;
2. if $A_1.s_i \mathcal{R} A_2.s_j$, then:
   - $A_1.s_i$ is *compatible* with $A_2.s_j$;
   - $\forall \langle A_1.s_{i+1}, A_2.s_{j+1} \rangle \in \text{transitions}(A_1.s_i, A_2.s_j)$, such that $A_1.s_{i+1} \mathcal{R} A_2.s_{j+1}$;
   - $\langle A_1.s_i, A_2.s_j \rangle$ is *alive*.

The choice on what to do is taken by the system which is in the $\odot$-state, while the other system will be in a $\oplus$ state. For two services this behavior is natural, at every step, one of the services will act as the sender, and the other will act as the receiver. Regardless of the interaction started by the two participants, if they are interoperable, they will be able to carry that conversation to the end, each one arriving in one of their final states.

Theorem 4.19 (Interoperability)**.** Let R_1 and R_1 be two interoperable protocol roles or services, and A_1 and A_2 their corresponding FSAs. Now, let $A_1.s_i$ and $A_1.s_j$ be two *reachable* states after a certain run σ. Then, there is a run σ', such that $\sigma\sigma'$ is a successfull communication.

Figure 4.9 shows finite state automata derived from two service implementations (or two interacting roles in a protocol), $P = \{p_1, p_2, p_3, p_4\}$ and $Q = \{q_1, q_2, q_3, q_4, q_5\}$.

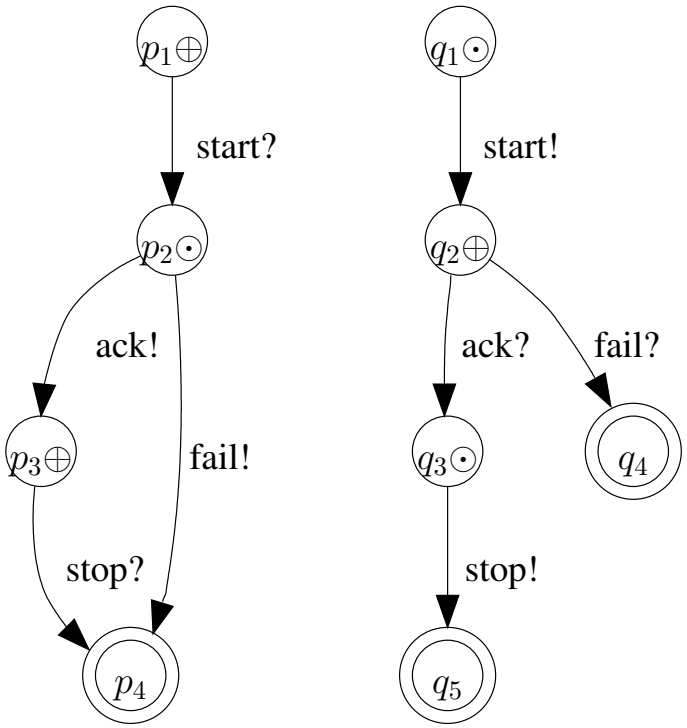

Fig. 4.9: Interoperability: Two service instances interacting

The resulting product system (see Fig. 4.10) consists only of the global states, of tuples in the form $\langle P.s_i, Q.s_j \rangle$, which are connected by synchronized transition, $\langle (p_i, q_j), (m, \overline{m}), (p_{i+1}, q_{j+1}) \rangle$.

Included in the figure are also the states which would have been reachable, if the two systems could have acted independently of each other (free product).

4.4.2 Conformance

Conformance, as shown in Fig. 4.8, is the preservation of interoperability, after substitution of e.g. a protocol role with a concrete service. This conformance check is needed when a service is selected to fill in a role in the adaptation workflow. For the same reason that the automata do not have to complement each other perfectly to ensure interoperability, a service implementation does not have to be a precise copy of the protocol role to be conformant:

- the service, particularly its conversation policies, should not *send* messages which are not intended by the role, and as such are not understood by other participants,
- all messages which are *received* by the service, coming from the other participants, must be handled by the service, as defined by the role,
- whatever point in the conversation is reached, the service must be able to bring it to an end.

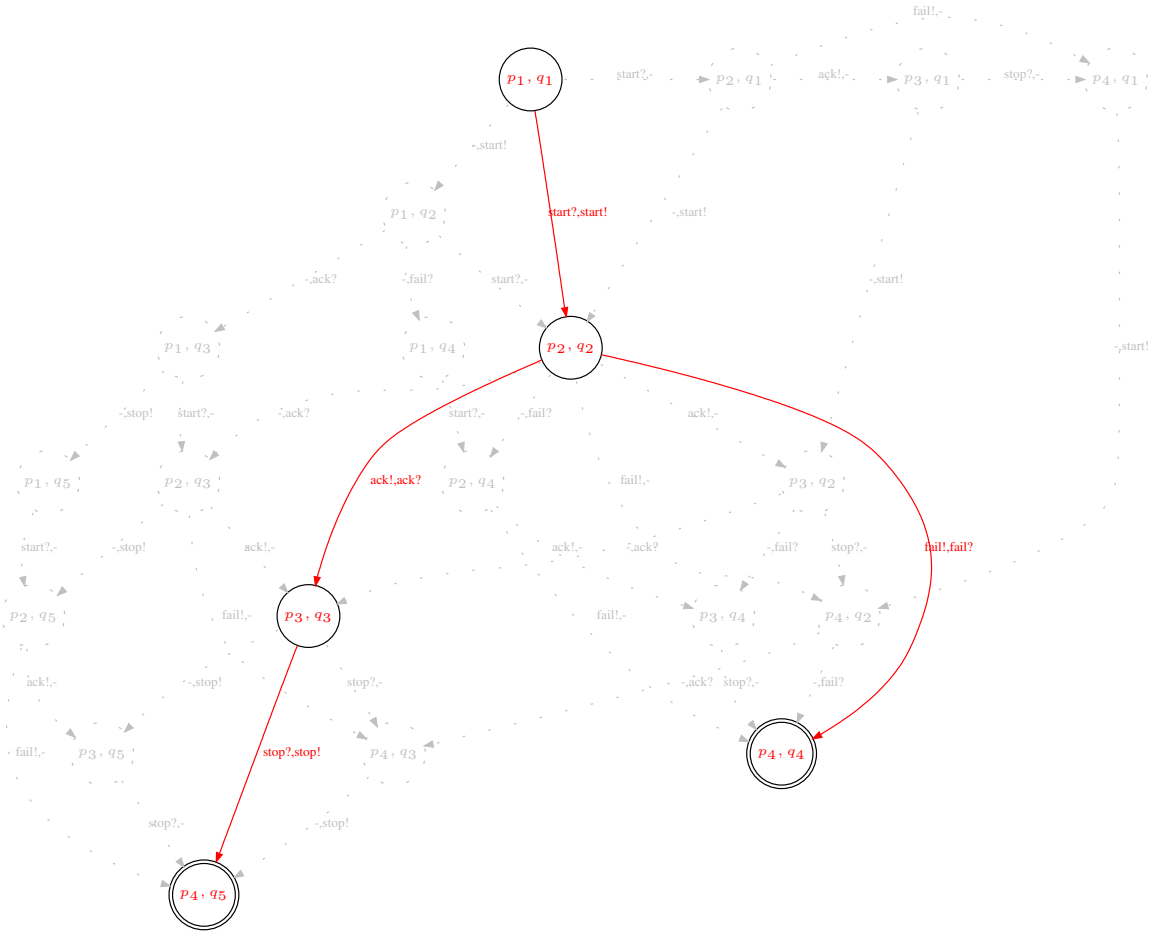

Fig. 4.10: Interoperability: Matching Emissions and Receptions

To summarize, if a service is *conformant w.r.t. the role in the workflow*, it will be able to interact with the other services, which, independently and separately, have been proven conformant with their respective roles.

However, the *conformance* tests used in related literature are too strict here. The conformance test is *not an inclusion test w.r.t. the possible conversations* [17]. Furthermore, the test is *not a bisimulation test w.r.t the protocol role*, because the test defined in concurrency theory [9, 122, 123] is too strict, and does not successfully distinguish conversations which follow the same message exchanges, but have a different branching structure [73, 17].

Additionally, in multi-party message exchanges, i.e. interactions between more than two partners, non-deterministic behavior will make the conformance verification fail, although the services are perfectly interoperable with each other. As an example, consider the three Protocol Roles R_1, R_2 and R_3. R_1 is sending m_1 to R_2, R_2 is waiting for the message m_1, and then waiting for m_2. R_3 is sending m_2 to R_2.

A service S_2 which first waits for Message m_2 and then waits for m_1, is put into the workflow to fill in Role R_2. The three entities are interoperable and can conclude their conversation. However, the created conversation is not legal w.r.t. the protocol [17, 15].

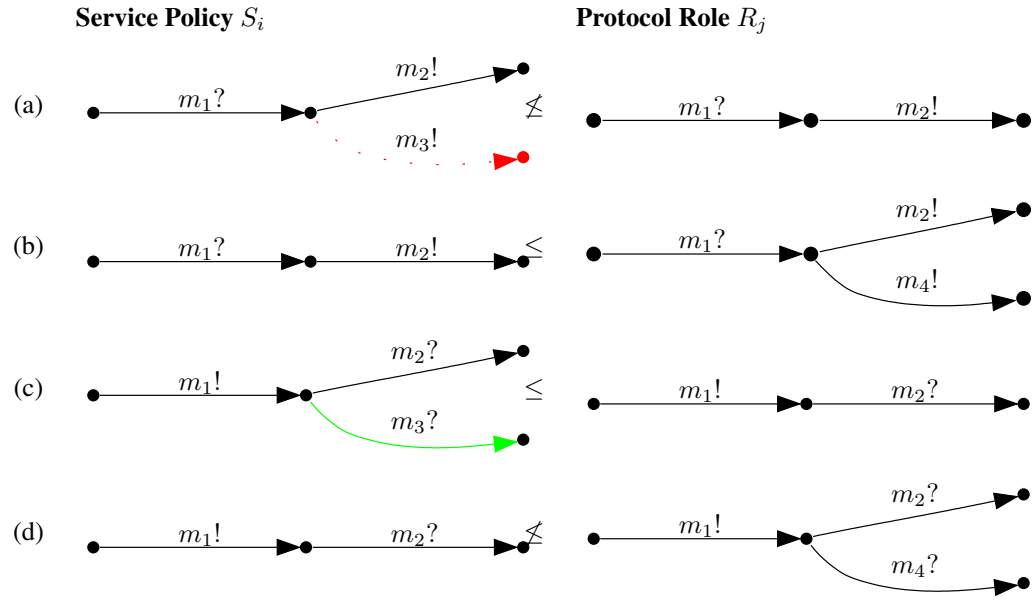

Fig. 4.11: Conformance: Expectations about conformant policies

In figure 4.11, the expectations of conformant policies, which are given by the implementation of a service, w.r.t their roles in the protocol is shown.

Fig. 4.11(a): the policy is a *non-conformant* implementation of the protocol role, because it can send a message m_3, which is not expected by the protocol and as such, is not handled by the interlocutor in the service composition.

Fig. 4.11(b): the policy is conformant with the protocol, as it never sends a message that is not expected by the role. The protocol allows the service to send more messages than are actually implemented by the given policy.

Fig. 4.11(c): the policy passes the conformance test, because the service's ability to receive more messages ($m_3?$) than specified in the protocol does not compromise interoperability.

Fig. 4.11(d): the last policy in this comparison is not a conformant policy, because it does not handle all the expected messages defined in the protocol ($m_4?$).

The conformance check of two state automata, e.g. a protocol and a policy, is needed to check if the interoperability is a system is preserved, if one of the participants is substituted [15, 38, 137]. In [15], the conformance test is defined as *state alignment*.

Definition 4.20 (Alignment)**.** Let A_1 and A_2 be two FSAs. A state $A_1.s_i$ aligns with $A_2.s_j$, written $A_1.s_i$ aligns to $A_2.s_j$, if:

$$A_2.s_i \in A_2.S^{\odot}, A_1.s_j \in A_1.S^{\odot}: \quad \text{message}(A_2.s_i) \subseteq \text{message}(A_1.s_j)$$
$$A_2.s_i \in A_2.S^{\oplus}, A_1.s_j \in A_1.S^{\oplus}: \quad \text{message}(A_2.s_i) \supseteq \text{message}(A_1.s_j)$$

Whenever $A_2.s_i$ and $A_1.s_j$ are regular states and they have no emissions or receptions, A_1 aligns with A_2.

The properties of alignment and compatibility, as shown in Fig. 4.8 can now be used to formalize the notion of a conformance between implementation and specification which does not violate the interoperability between the participants in the composition.

Proposition 4.21. Given to FSAs A_1 and A_2 such that the state $A_1.s_i$ is compatible with $A_2.s_j$. Additionally, let A'_1 and A'_2 be another two FSAs, such that

- $A'_1.s'_i$ aligns to $A_1.s_i$, and
- $A'_2.s'_j$ aligns to $A_2.s_j$.

Then, $A'_1.s'_i$ is compatible with $A'_2.s'_j$

The conformance check implemented by the PUMA system [19], is inspired by bisimulation, as proposed by Baldoni et al. [17] and further refined in [15].

Definition 4.22 (Conformance)**.** Let R_1 and R_2 be two systems and A_1 and A_2 their FSAs. R_1 is conformant to R_2, written $R_1 \leq R_2$, iff there is a binary relation $\mathcal{R}$, such that

1. $A_2.s_0 \mathcal{R} A_1.s_0$
2. if $A_2.s_i \mathcal{R} A_1.s_j$, then:
   - $A_1.s_j$ aligns to $A_2.s_i$;
   - $\forall A_1.s_{j+1}$, such that $(A_1.s_j, m, A_1.s_{j+1})$, there is $A_2.s_{i+1}$ such that $(A_2.s_i, m, A_2.s_{i+1})$ and $A_2.s_{i+1} \mathcal{R} A_1.s_{j+1}$;
3. $\forall A_1.s_j \in A_1.F$, and for which there is a state $A_2.s_i$ such that $A_2.s_i \mathcal{R} A_1.s_j, A_2.s_i \in A_2.F$;
4. $\forall A_2.s_i$ that are *alive* and there exists a state $A_1.s_j$ such that $A_1.s_i \mathcal{R} A_2.s_j$ and $A_2.s_j$ is *alive*

The conditions in Definition 4.22 have the following effects:

- both the FSAs start in their initial state,
- the *alignment* condition captures the notion of "less emissions, more receptions" [38], which allows a service to not implement all messages $m!$ for sending, as well as to implement the handling of more messages $m?$ for reception, different from what is specified by the protocol,
- when one system has transitioned into a final state, the other must also have done so.
- for each foreseen prefix of the system Q, the execution of a transition in system P must progress towards a final state.

Finally, it is possible to formalize the fundamental theorem of the relation between conformance and interoperability:

Theorem 4.23 (Substitutability)**.** Let R_1 and R_2 be two *interoperable* protocol roles, and S_1 and S_2 be two services, such that $S_1 \leq R_1$ and $S_2 \leq R_2$. Then, S_1 and S_2 are *interoperable*.

4.5 Verification of Service Compositions: Related Work

While the main conformance checking originated mostly in Multi-agent systems (and of course, software engineering), it has found an application in Web services as well. However, so far, there most prominent example of an actual *implementation* of a system for such verification of protocol conformance is *LTSA-Eclipse* [67, 66]. Furthermore, there exist a small number of approaches which extend approaches taken from the multi-agent domain and apply them to Web service choreographies and orchestrations.

DecSerFlow [158], by van der Aalst et al. is another example of how a conformance of an conversation policy against a protocol definition can be verified, in case of DecSerFlow, the definition of protocols is based on a Declarative Service Flow Language. The authors provide an easy graphical notation for specifying protocols, but the foundation of their language is Linear Temporal Logic (LTL).

Bisimulation Milner and Park introduced the concept of bisimulation as one of the simplest way to define the *equivalence* of two transition systems. As a result, there exist a large number of approaches, which rely on some form of bisimulation to check the equivalence of transition systems [17, 40, 41, 50, 70, 137].

Many authors introduce a *weak bisimulation*, which allows, that a service can deviate from the required protocol, as long as the alternative message exchanges preserve interoperability.

One example is the work by Endriss et al. [61], which defines three different levels of conformance: *weak*, *exhaustive*, and *robust*. Their work describes a system, where the agents themselves ensure that

their utterances conform to specified policies, rather than having a third party watch for violations of said protocols. The restriction to allowing only *two agents* involved in *sequentially alternating* message exchanges allows the use of DFAs to represent the interactions.

The three levels of conformance are:

Weak conformance: An agent is *weakly conformant* to a protocol $\mathcal{P}$, iff it never utters any illegal dialogue moves w.r.t. $\mathcal{P}$.

Exhaustive conformance: An agent is *exhaustively conformant* to a protocol $\mathcal{P}$, iff it is weakly conformant to $\mathcal{P}$ and it will utter at least one legal output move for any legal input of $\mathcal{P}$ it receives.

Robust conformance: An agent is *robust conformant* to a protocol $\mathcal{P}$, iff it is exhaustively conformant to $\mathcal{P}$ and for any illegal input move it will respond with an error message.

However, the authors only provide an algorithm for a-priori checking of weak conformance, by checking if all the conversations produced by an agent (response space) constrained by private conditions, i.e. agent beliefs, comply to the required protocol.

SPIN SPIN is a model checker, which uses Büchi automata for modelling communicating systems [86]. Büchi-Automata accept ω-regular languages, which are exactly those languages definable in a particular monadic second-order logic called $S1S$ (monadic second order logic over infinite words). *Linear Temporal Logic* (LTL) is a fragment of $S1S$. Therefore, SPIN is capable of verify interoperability and conformance based on temporal constraints [160, 72].

4.5.1 Logic based approaches

Social Integrity Constraints In [4], based on their previous work in [3] and [5], Alberti et al. present a verification system, Java-Prolog-CHR, which uses protocols specified in a logic-base formalism: *Social Integrity Constraints* (SIC). Similar to other work which uses the interaction between agents, the validation is based on the *observable* behavior of the entities, and not on some hidden internal state or policy. In Java-Prolog-CHR, the behavior of an agent corresponds to *events*, while the desired behavior is expressed as *expectations*. Social Integrity Constraints are used to express which expectations are generated as a consequence to occurring events. Events are of the form $\mathbf{H}(\mathrm{Description}, \mathrm{Time})$. Expectations can include constraints based on Constraint Logic Programming (CLP) [95]. A positive expectation (events that are supposed to happen) that is matched by an event is considered fulfilled. A negative expectation (events that are not supposed to happen) matched by an event is violated.

Example 4.5.1 (Social Integrity Constraints)**.**

$$\mathbf{E}(\mathrm{accept}(a_k, a_j, \mathrm{give}(M), d_2), T_a) : M \geq 10, T_a \leq 15$$

Expectation $\mathbf{E}$ *requires an agent* a_k *to* accept *giving agent* a_j *an amount* M *of money, in the context of interaction* d_2*, at time* T_a*. Furthermore, the time when the interaction takes place must be before or exactly at time 15 and the amount of money is expected to be greater or equal the amount 10.*

$$\mathbf{H}(\mathrm{accept}(a_k, a_j, \mathrm{give}(20), d_2), 15)$$

The particular event $\mathbf{H}$ *fulfills the expectation* $\mathbf{E}$ *(4.5.1) and its constraints*

Constraint Handling Rules, implemented in a Prolog system, are used to reason about the state of conjuctions of expectations. An expectation can be verified, if in the history of interactions there exists a matching event. Fulfillment or violation of such expectation is checked, i.e. if a constraint on the timestamp represents an expired deadline.

Agent Languages In [109], Labrou and Finin introduce semantics and conversations for *agent communication languages*, in particular the Knowledge Query Manipulation Language (KQML) [65]. The language KQML consists of primitives (also called *Performatives*), which allow to express attitudes about the content of exchanges between agents and also to exchange these attitudes with other participants.

The authors defined the semantics for a small set of KQML performatives, and use them to reason about messages exchanged between agents. However, similar to FIPA [69], the agent communication languages refer to some kind of internal state of the agent, which is hidden from the outside. The agent's *belief*, *desire*, and *intentions* as core part of the exchange message add additional unwanted complexity to the verification process.

In their approach, they use a Definite Clause Grammars (DCG), which are an extension of Context Free Grammers (CFG). They have chosen DCGs over finite state automata and CFGs, as DCGs can be directly expressed in general purpose programming languages, i.e. Prolog. However, they do not propose any techniques for checking conversation policies against formal protocol definitions, only list possible applications of having such detailed semantic description of the agent's communicate behavior.

4.6 Discussion

In this section, the language for specifying the conversations between services were presented. Furthermore, the foundation for the verification of interoperability between services and conformance checking between the implemented behavior of a service and the protocol role given by the adaptation workflow was discussed. The model for representing the interactions is founded on *speech acts*. This leads to the definition of *conversation policies*, which are representing the implemented communication rules of the services, and *protocol roles*, which specify the expected communicative behavior of the services which is selected for a role in an adaptation workflow.

From the language to describe the policies and protocols, to check certain properties of the conversations which are possible between services in a service composition, a formal model was needed.

In the literature, there exist many possibilities to specify and represent protocols:

- *process algebras*, such as π-calculus, μ-calculus, and ACP τ
- *operational semantics*,
- *Petri Nets*, including their subgroups, such as *state transitions systems*, and *finite state automata*

Finite state machines have a long history in the modelling and verification of interactive processes, including dedicated tools for analyzing the properties of such systems, such as SPIN [86].

Nevertheless, verification of the properties of a system is a time-consuming and complex task, which also requires complete knowledge about every entity in the composition. In the context of PUMA, fast verification of service properties is required. As such, a *conformance* verification, where a single entity is verified against a protocol specification is better suited than a global interoperability check. If a service passes the conformance check, which has to be performed separately for each of the services, it can be put in the place of a role in the global protocol, without harming *interoperability*.

However, a service does not have to implement a precise copy of the protocol to be conformant. In many cases it is sufficient for a service to only send a subset of the possible messages, which can be understood by its conversational partner. Furthermore, a service which can handle more messages than specified by the protocol, does not break interoperability. As such, the traditional equivalence test for finite state automata, such as (bi-)simulation, are not adequate for checking conformance.

For verifying conformance in the PUMA system, a conformance check, which is a form of *weak* bisimulation, was implemented. However, this chapter explores some additional notions of interoperability and conformance checking, which is applicable for Web Services, Multi-Agent Systems, and other distributed message-based architectures.

Chapter 5

Implementation and Evaluation

To prove the usability of the PUMA approach to Web Service composition for multimedia adaptation, a prototypic implementation of the system described in Chapter 3 was created. The first part of this chapter gives a short description of the implementation and the environment. The second part is dedicated to the results retrieved from the first evaluation of the system. This includes delays and duration for processing content with services, transporting it between different nodes, replacing failing services, and selection and verification of services.

5.1 Implementation of the Puma Architecture

For implementing the PUMA system, the Java programming language was used. The J2EE (Enterprise Edition) includes support for XML and Web services, and with Jena and Sesame, there exist two capable tools for storing and managing RDF data. The source code and documentation is available from locations listed on `www.l3s.de/~puma`. The description of the implementation is focussed on the parts which are relevant for selection and verification of workflows and services.

5.1.1 Dependencies

PUMA was designed as a modular and extendable architecture (see 5.1). For providing functionality, existing code was re-used wherever possible. As such, PUMA depends on open-source libraries and tools. The following tools are general purpose implementations which have been implemented and re-used for this project.

l3s-config: The configuration management library (`de.l3s.config`) is a configuring Java applications based on parameter files.

l3s-rdf: The OpenRDF utility library (`de.l3s.openrdf`) is used for simplifying the access to RDF data stored in a sesame database. The OpenRDF API is faster, but also simpler than the Jena

API. Therefore, this code collection was created to provide often-used functionality for loading and storing RDF statements from files, and to allow a simple persistence mechanism for directly storing Java object attributes as RDF statements. Also, a maven plugin is provided to compile RDF Schema files into Java Classes, to allow direct access to RDF Properties and RDF Classes within Java programs.

l3s-webservice The Web Service API (`de.l3s.wsdb`) provides an interface for easy access to Semantic Web Service descriptions. The `l3s-rdf` library is used to allow the use of OWL-S. PUMA uses this package for selection and verification of Web Services, and to realize a simple Web Service registry.

External Dependencies For Java, there exist many frameworks for implementing and invoking Web Services. The two most prominent examples are:

- Java Web Service Development Platform (JWSDP) and its successor, Project GlassFish from Sun, and
- CXF from the Apache foundation.

As an execution environment, which also provides the HTTP transport protocol for SOAP interactions (see Section 2.2.1), the de-facto standard has been Apache tomcat. However, there also exist smaller HTTP-only service engines, such as Jetty, which do not provide as much functionality as an application server such as tomcat. The benefit of lighter engines is, that they can be used in environments with limited capabilities, e.g. on mobile devices.

Java-based RDF frameworks PUMA is a Semantic Web Service framework, and as such requires a persistent repository for storying, retrieving and querying RDF statements and graphs.

The two most prominent open-source implementations for realizing Semantic Web applications in the Java programming language are the open source packages Jena and OpenRDF.

Jena Jena [118, 167] is a Java framework, supporting RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. Initial development was done by the HP Labs Semantic Web Programme, but later was moved to open source. Unfortunately, Jena had severe performance problems when the inference engine and the persistent storage layer were used.

OpenRDF Sesame [46] is a very fast RDF database, and includes an API (OpenRDF) for managing RDF data in Java programs. It was developed by Aduna software, but is available as open source.

The production release of the Sesame database, at the time of this writing, did not yet support the SPARQL language.

While OpenRDF is not providing the same level of inference and reasoning capability as realized in Jena, it is *significantly faster*, which becomes important for large databases, such as a PUMA's centralized Web Service repository.

After performance comparisons were published on the Jena developers mailing list, the developers focussed on improving performance and also made it possible to replace the in-memory data-storage layer of Jena with a Sesame database.

Because of previous experience, it was decided to use OpenRDF and Sesame as the RDF knowledge base for this project.

Therefore, in this work, the SerQL query language will be used to realize the required functionality. There is no conceptual difference between the SerQL and SparQL language, as both query languages are designed to improve on RQL. However, the syntax differs, which can be seen in the structure of the SerQL query shown in Example 5.1.1, and the SparQL query shown in Example 2.1.2 on page 17.

Example 5.1.1 (SerQL Example: Retrieving a Transcoding Services)**.**

```
SELECT
  Uri
FROM
  {Uri} service:profile {} profile:serviceCategory {stype}
WHERE
  stype = "Transcode"
USING NAMESPACE
   service = <http://www.daml.org/services/owl-s/1.2/Service.owl#>,
   profile = <http://www.daml.org/services/owl-s/1.2/Profile.owl#>
}
```

5.1.2 Package Overview

- The PUMA Model, `de.l3s.puma.model`, contains a mapping of the PUMA Ontologies for Workflows, Digital Item Adaptation, Policy and Protocol Description to static Java objects, which simplifies the use of concepts from those ontologies for the programmer. During compile-time, the RDF and OWL documents are retrieved from their publishing location and concepts, classes and properties are turned into Java Objects which can be used instead of their full encoded URIs.

- The PUMA Core code, `de.l3s.puma.core` is common code shared between all modules, e.g. classes for Session Management, Containers for Digital Items, Users, Services, Workflows, Policies and Protocols. It also is the place of the main class which starts a PUMA system and it contains utilities for profiling and logging operations done within PUMA.

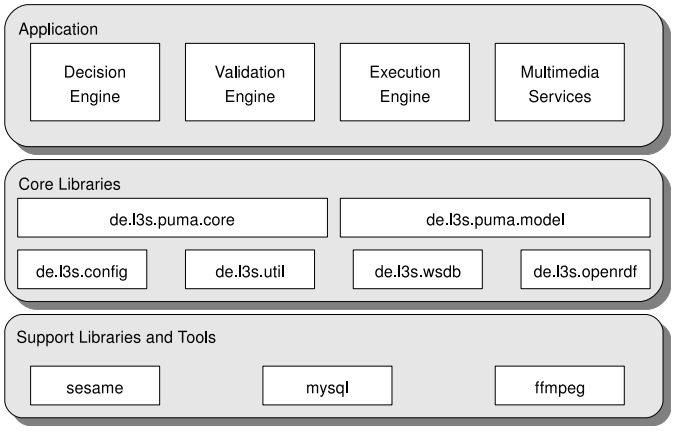

Fig. 5.1: Puma Software Packages and Dependencies

5.1.3 The PUMA Video Client

For testing purposes, a simple video player application, `de.l3s.puma.userdevice`, was created, based on the Java wrapper of the VLC client. This allows to embed the VLC video display window into a Java application. The functionality of the prototype client is currently limited to

- connecting to the PUMA system to send a video request,
- connecting to the delivery service, and
- reassembling the segmented video content for playing.

The code also includes profiling facilities for recording various parameters of the delivery process used in the evaluation.

5.1.4 Publishing and Retrieving Services from the Service Broker

The Web Service registry, and the Content Metadata repositories of PUMA are based on Semantic Web technology. As such, information in that databases can be searched using SeRQL queries.

Figure 5.3 depicts the layout of the service repository `de.l3s.puma.database.service`, which is sharing a tomcat web application server with an instance of the sesame RDF database. To create a persistent repository where semantic web services can be registered, sesame stores the RDF statements about semantic services in a relational database system, in the current puma prototype postgres was chosen. The repository allows to publish services by issuing a OWL-S description, or to retrieve a list of services which match a role described in terms of the PUMA Workflow Vocabulary.

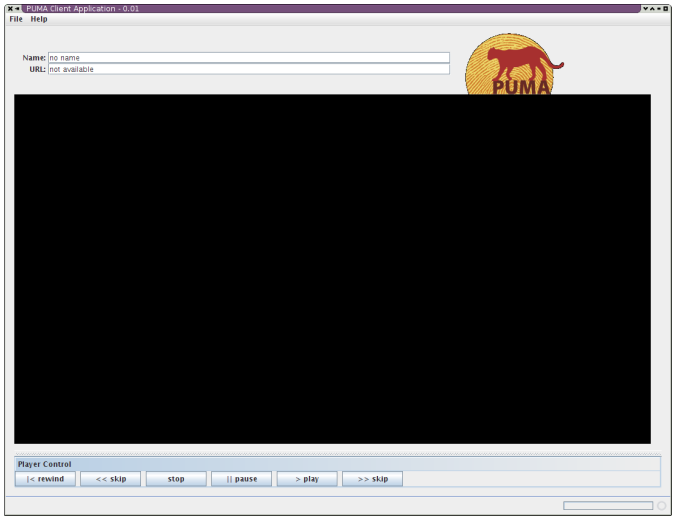

Fig. 5.2: Puma Client Window

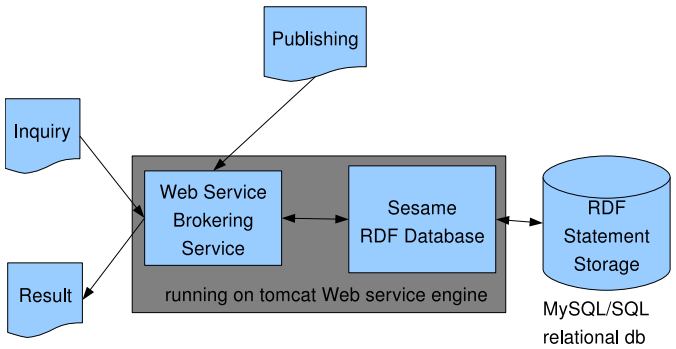

Fig. 5.3: Service Repository

5.1.5 Decision Engine

The three engine packages, Decision Engine `de.l3s.puma.decengine`, Validation Engine `de.l3s.puma.valengine` and the Execution Engine `de.l3s.puma.exeengine` provide the services which are used for workflow management, service composition and execution.

Figure 5.4 shows the sequence of actions required for the selection and verification. All communication between the different modules is performed using Web Service invocations using SOAP request and response messages, which contain RDF/XML-data, such as Workflows and OWL-S service descriptions.

- When requested by a client, the decision engine will start the selection of matching workflow templates from the template repository. These templates are instantiated with the input and output parameters of the requested adaptation. In the current implementation, the client specifies in the request, which template is required.

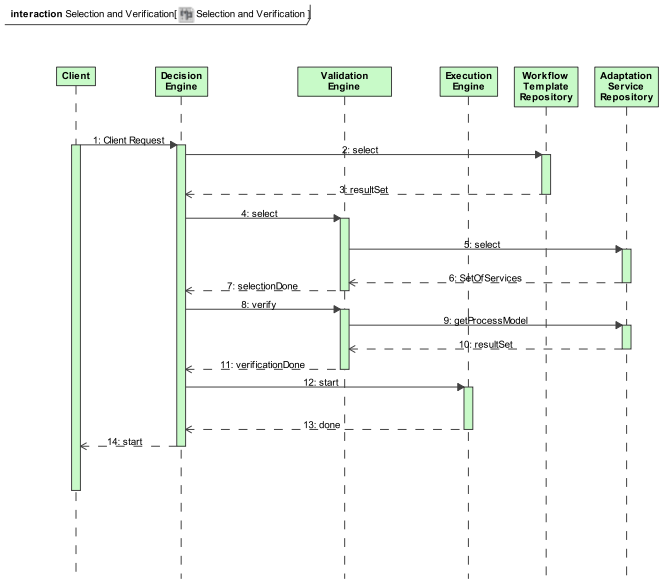

Fig. 5.4: Collaboration between Decision and Validation Engine

- Once the template is instantiated, it is submitted to the validation engine.

- First, the validation engine selects a set of candidate workflows from the service repository. The OWL-S Profile is used to match services to the query. The current implementation only selects perfect matches.

- After the services are selected for each role, the workflows and services are submitted to the conformance check. Each protocol role in the workflow contains a WS-CDL-like interaction description, which is used to create the states and edges of the protocol automata. In the next step, the OWL-S Process Model of the service, that is going to be verified, is retrieved from the service repository. The process model is turned into a state automaton, and using a simple depth-first algorithm, the alignment (*see* Sction 4.4.2) of each pair of states, starting from initial state, is checked. If the conformance check is successful, the service is selected for the role and the system continues with the next role of the workflow.

- The set of verified services is initialized, the channel-information is provided to the client, such that it can open a connection to the last service in the adaptation chain.

After this interaction is complete, then all services are ready to begin the adaptation of content. The client connects to the last service in the sequence of adaptation services and will receive the adapted content.

5.1.6 Implementing Adaptation Services

Adapting multimedia objects, which are stored on a local storage device is possible with most of the currently available implementations of adaptation systems. However, as soon as the data is streamed over a network, the client, and also any processing step between the client and the delivery system, is no longer able to simply access any required location in the material it requires for decoding and encoding the content.

Some of the stream transport protocols allow to skip or rewind to an different locations in a stream. This is a consequence of the type of encoding which is used to provide the strong compression required for multimedia data. For example, to decode an arbitrary frame in an MPEG stream, information from previously transmitted frames is needed.

A popular approach for accessing large data sets is to split the content up into smaller, separately addressable *chunks* of data (see Section 2.2).

It is the nature of this type of data, that requires additional information, such as timestamps for combining multiple sources to a multiplexed stream, to avoid loosing the synchronization between the video and audio channels.

PUMA uses CXF Web Services to provide functionality for adaptation, and therefore requires a method for transporting frames, and other media content, between those services.

The obvious solution is the partitioning of the data into smaller pieces, which is know as chunking.

Interestingly, this also allows the “streaming” of formats which are not originally designed for continuous data transfers. However, splitting such a stream might result in synchronization problems when the chunks are re-assembled for rendering.

5.1.7 Adaptation with Multimedia Processing Tools

For PUMA, a freely available solution for multimedia adaptation is required. Furthermore, the software has to be embedded into a Web Service. The libavcodec library, which is part of ffmpeg, is the most popular open source library for content adaptation, as it supports a very large number of codecs and container formats. Most of the freely available player applications, such as VLC are using this particular library. Many of the applications that are based on ffmpeg, also provide tools to access the decoding, encoding and transformation functions of the libavcodec library.

The situation for creating applications in the Java language is worse. Multimedia support is restricted to very few products, and as a result, developers created interface wrappers (JNI) for the native libraries.

The Java Media Framework (JMF) has, for a long time, been the only reasonable possibility for media processing in Java. Unfortunately, Sun stopped the development in 2004, sadly before their RTP/RTSP implementation was finished. However, third party developers continued to create plugins for JMF, such as Fobs4JMF and IBM’s MPEG-4 Decoder (see Appendix A.3 for a detailed list).

Package de.l3s.puma.mmservices The PUMA Multimedia Services are consisting of two components. The first one, written in Java, realizes the Web Service and Stream Communication. The second part is ffmpeg, which is executed from inside the Java program (similar to [14]).

The transport of content data to the external process is done over UNIX pipes and temporary files.

For realizing the content adaptation, we also considered other applications, such as mencoder and vlc. The ffmpeg implementation was chosen, because it was more stable and provided more functionality than the other variants.

Communicating with other adaptation services, and components of the PUMA architecture is performed by two different methods The first method used is the Web Service framework, which is required for the stream control protocol, event notifications and service monitoring. The content data is transported over a separate direct TCP connection.

The processing inside an adaptation service (see figure 5.5) is implemented as a pipeline, working on a queues containing the multimedia chunks. The reader stores the received data into the input queue, the worker takes the newly received chunks for processing with the external adaptation process (ffmpeg). The results are then moved into the output queue. The writer threads then sends those chunks to the next service in the workflow. Reader and Writer Threads use a simple SOAP based polling protocol to request or push new chunks to other services.

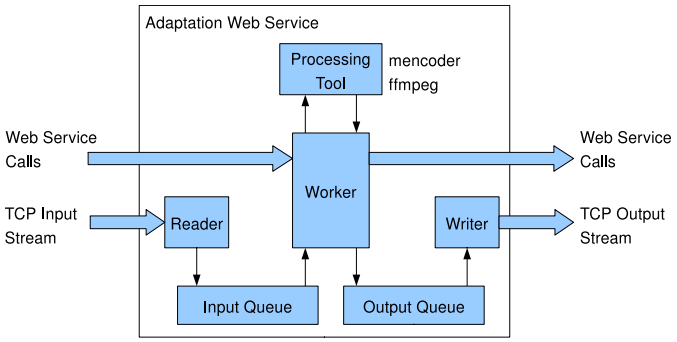

Fig. 5.5: The Internal Structure of an Adaptation Service

5.2 Evaluation

Using the prototype implementation of the PUMA infrastructure, we performed a series of test to evaluate the performance of the system [156, 19].

5.2.1 Experimental Setup

For performing the experiments, a set of interconnected servers were selected. The machines are connected with a high-speed local Ethernet network. Table 5.1 gives a list of the three servers, their processor

Server	Processor Type	Speed	Installed Memory
1	4× Intel Xeon	2.80 GHz	6 GB
2	4× AMD Opteron	2.40 GHz	35 GB
3	4× Intel Xeon	2.80 GHz	4.4 GB

Tab. 5.1: List of Servers used for the Experiment

architecture, and installed main memory.

The experiments for service selection and conformance checking have been performed on a IBM T43 with an Intel 2.0 GHz CPU and 1.5 GB of *installed* main memory.

All measurements are averages over one hundred independent runs, if not stated otherwise. The workflows used in the experiments are based on the example workflow (see Fig 2.5 on page 21).

5.2.2 Estimating Server Load

Multimedia Adaptation is processing intensive, and as such, a server can only run a limited number of consecutive adaptation request before timing constraints are violated. The first experiment measured the runtime of different adaptation tasks: scaling, frame rate adaptation and transcoding. For the test, 12s long video sequences (MPEG-PS, MPEG1, approx. 2MB long) from a short movie were repeatedly processed by each of the different services on server 2, with all required data locally stored.

The result (see Fig. 5.6) indicates, that scaling is the most processing intensive application for. This suggests an upper limit of 14 concurrent scaling processes on server 2. For more processes, the adaptation time would exceed the length of the video, which would make continuous streaming impossible.

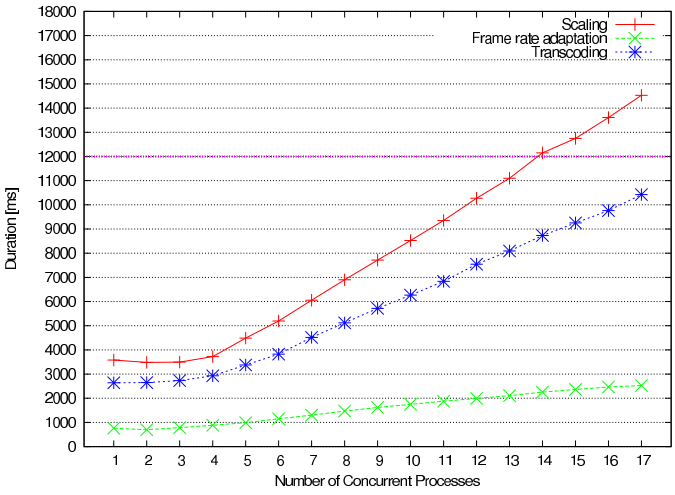

Fig. 5.6: Adaptation of 12s long chunks (approx 2MB per chunk)

Additionally visible from the results is, that up to four processes can run in parallel without any noticeable delay, which is due to each of the processes being run on a different processor of the server.

Number of parallel services	10	11	12	13	14	15	16	17
Processing time $<$ Video length	100	100	99	90	55	36	20	9
Processing time $\geq$ Video length	0	0	1	10	45	64	80	91

Tab. 5.2: The percentage of processes that violate real-time constraints

However, server load is not constant over time, and must be taken into account when estimating the number of parallel processes. For this server and content object, detailed analysis (see Tab. 5.2) of the complete dataset suggests a maximum number of 11 parallel processes. For more than 11 parallel running services, there are instances, where the adaptation time overruns the video length.

5.2.3 Arrival Rate Variance

Simplified, for uninterrupted rendering of the video, consecutive chunks must arrive faster than the duration of the playback. In this experiment, no buffering of chunks was done, so the protocol overhead for requesting and sending data between services is included in the measurements. A 60s long test video was split into chunks of three different sizes: 20 chunks of 3s, 10 chunks of 6s, and 5 chunks of 12s. The complete set of chunks is then processed using a service composition with 4 services: frame-rate adaptation, transcoding, scaling, Table 5.3 shows the delay for receiving the first segment (startup), and then the time between consecutive chunks.

Chunk no.	500 KB	1 MB	2 MB
Startup	4434	7255	14794
02	2388	6070	3880
03	4354	1838	4454
04	4685	1658	5012
05	921	1738	5761
06	1111	1832	
07	954	1807	
08	936	1618	
09	1006	1965	
10	887	1934	
11	1101		
12	969		
13	951		
14	1048		
15	901		
16	904		
17	1040		
18	914		
19	933		
20	952		

Tab. 5.3: Average Arrival Rates of Chunks

For chunks of 6s size, the startup time is reasonably short, and all following chunks arrive in time. For 3s chunks, a longer startup buffering time is needed before rendering can begin. The startup for 12s chunks is longer than wanted, because the delay for workflow selection, service selection and verification must still be added to this value.

5.2.4 Startup Time for different Workflows

A fast startup time is required of a system to be attractive to users. To check the quality of our approach we constructed workflows of different length, starting with the example workflow. All services that were added to the system have been scaling services. As seen in Fig. 5.6, scaling services have the highest processing requirement, by adding only this service types to the workflow, a worst case scenario can be simulated. The maximal length of workflows used was 12 services, spread over the 3 servers in a way so that every communication had to be done over the LAN.

The startup time has been determined as the time passed from pressing the play button on the display device until the first chunk was completely received. The service composition was already verified but not initialized at this point.

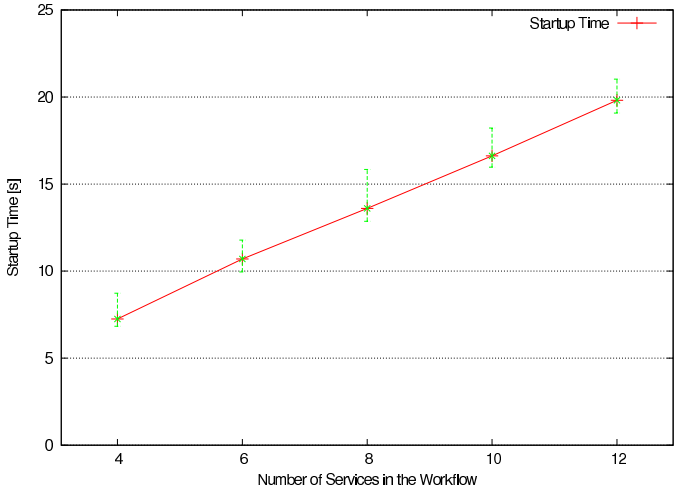

Fig. 5.7: Startup Time for different Workflows

Figure 5.7 shows the variation in startup time as a result of different workflow length. The results show that every extension of the workflow length adds a constant additional delay to the system, which is comprised of processing time, network delay, and IO delay.

The results shown in Fig. 5.8 are the aggregated network and IO delays of all services in the different workflows. For a workflow composed of 4 services, 98% of the time (7075ms) is required for the processing of content, the other 180ms are for communicating and buffering.

Overall, a startup time of around 15 seconds is fast compared to startup times of about 2 minutes as recently reported in P2P streaming applications [83].

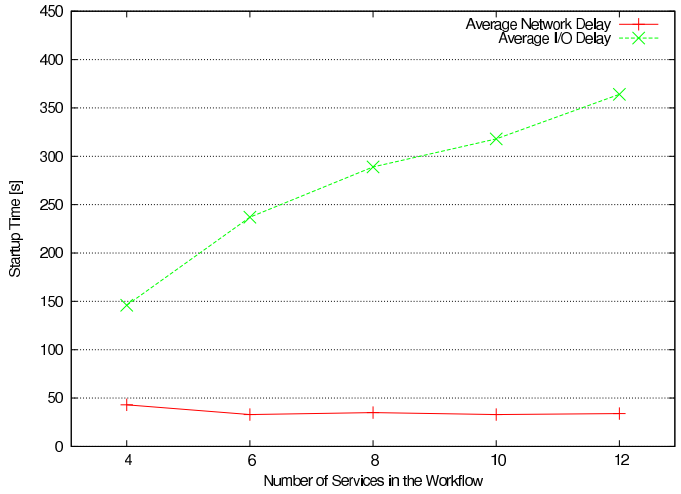

Fig. 5.8: Influence of I/O and Network Latency

An additional experiment was performed for the scaling service, to measure the influence of chunk size and load on the processing time. The result in Figure 5.9 indicate, that the there is an almost linear dependency between chunk size and processing time.

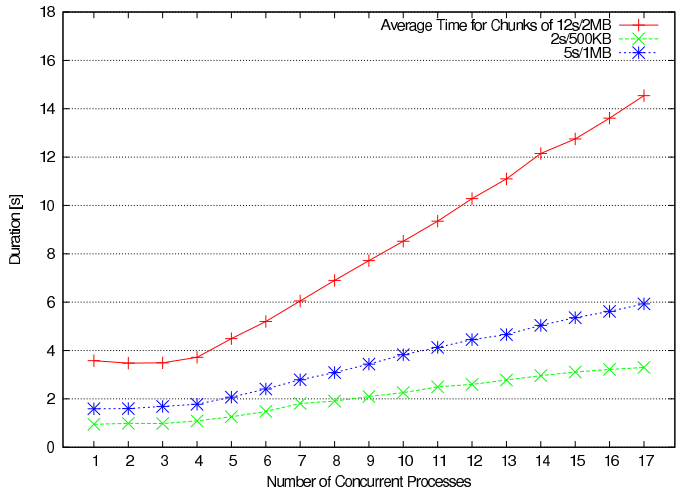

Fig. 5.9: Influence of Chunk Size on Processing Time

Knowing that the content processing is responsible for 98% of the delay in this case we conducted another experiment measuring the processing time depending on chunk size and load. Similar to the previous load experiment we used a single four CPU machine running up to 17 parallel scaling processes with varying chunk sizes.

5.2.5 Recovery from service failures

The objective of this experiment is to measure the time needed to replace a failed service. In an instantiated and running workflow, again based on the known example (see Fig. 2.5), the failure of one of the intermediate services was simulated, in this case the transcoding service. The duration from the failure until the replacement service starts sending out new data to its successor in the sequence, was considered as the recovery time.

Figure 5.10 shows the results (in ms) for each recovery from a failure, simulating 180 service failures in total. The average time for recovery is 3.8 seconds, including network delays and latency. With the buffering strategies prevalent in todays multimedia streaming frameworks this should be short enough for seamless recovery. The time required for recovery is much shorter than the 6s duration of the chunks used for this experiment.

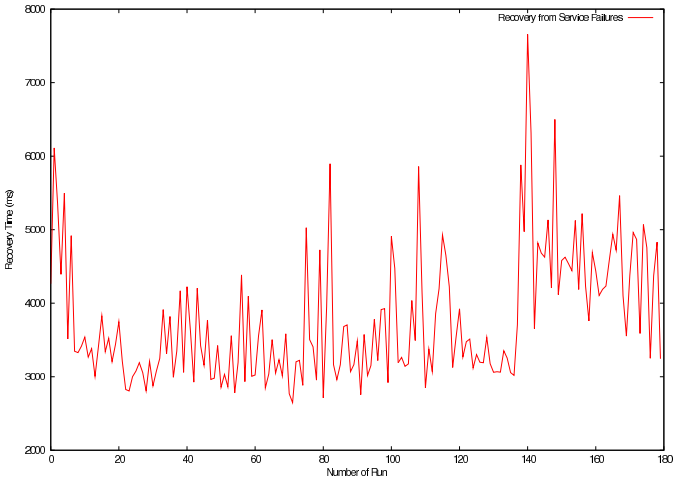

Fig. 5.10: Measured times for replacing the Transcoding Service

5.2.6 Service Selection and Validation

For each of the roles in the workflow, a set of candidate services needs to be discovered and retrieved from the service registry. In the current implementation, this filtering is based around a service capability attribute (e.g. transcoding) from a shared vocabulary.

Figure 5.11 shows the overall time necessary for accessing suitable services inside the service repository. This time is composed of locating the service in the database and retrieving its service description. The overall times are reasonably small for repositories containing up to 5000 services, and selecting between 5 and 10 candidates for each role in the workflow. This leads to a delay of under 8s for service selection.

A more detailed analysis has shown, that locating the service candidates is actually very fast, even for repositories (average $<$ 200ms) containing a large amount of services.

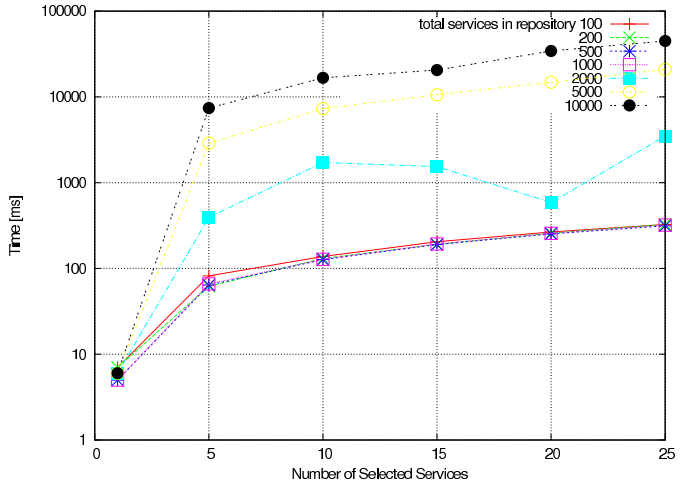

Fig. 5.11: Retrieving Service Candidates from the Repository

However, to perform the verification, the OWL-S Process Model has to be retrieved and transformed into FSA. The retrieval of the complete RDF graph of a service is a costly function.

Therefore, selecting new services on-the-fly for replacing failing services is not an option in the current implementation. For finding a replacement service, only those in the already retrieved set of service candidates can be used.

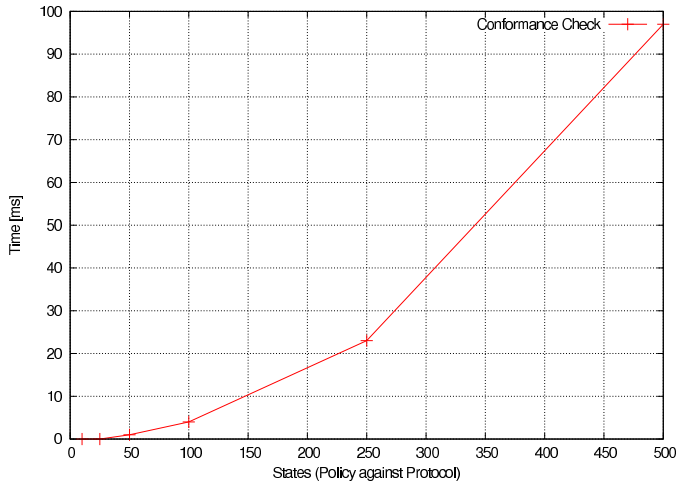

Fig. 5.12: Conformance Check

Figure 5.12 shows the time required for validating a single policy against a protocol role.

The validation only takes milliseconds for each state, so even for complex protocols, a conformance check can be done very fast. Of course, it is required that the roles of the protocol are interoperable, which can be checked separately and preemptive, as the workflows in the PUMA systems are static w.r.t

protocols.

However, the policies which are stored as part of the OWL-S service description need to be retrieved from the database and transformed into Java Objects, which takes considerably longer then the check itself, depending on the size of the repository. To optimize this, the state machine representation of policies could already be generated and stored at publishing time of the service.

5.2.7 Discussion

In the first section of this chapter, a short overview over the implemented components of the PUMA system was given. Simple versions of the different decision, validation, and execution engine were implemented, and used as a test-bed for first evaluations.

The first runs of the system were used to collect data about the behavior of the services in different conditions. Furthermore, workflows of containing a varying number of adaptation tasks were started, and the startup times and delays were measured.

The time for finding service profiles and extracting RDF graphs from a repository, and the conformance check were measured separately.

Results The first measurement, running different types of adaptation services, was used to find out which adaptation task was the most processing intensive. This information was later used to create a worst-case scenario, where the workflow consisted of mainly slow adaptation tasks. Furthermore, the test was conducted for different chunk sizes, and for different workflow lengths.

The startup time is an important characteristic of a content delivery system. It is the time from starting the fully prepared adaptation workflow, until the first chunk arrived at the client. With approx. 7.2s, the startup time for the example workflow (see Fig. 2.5), is shorter than reported for P2P systems. [83].

A key feature of the PUMA system is the capability to recover from failing services. The recovery time, that is the time from the failure of the service, until the new service starts sending out data, has to be short enough to allow uninterrupted delivery of the content. In the measurement the average recovery time from a service failure was 3.8s, which is shorter than the 6s chunk duration that were used.

Furthermore, the conformance check, to see if an already selected service candidate will be interoperable with the other participants of the workflow, is very fast. Even for protocols consisting of 500 states, the conformance check was finished in less than 100ms.

However, while the selection of services from the service repository, based on their Profile, is comparably fast, the actual retrieval of the Process Model and the transformation into state automata, does not scale well with increasing repository size. For a repositories containing 2000 services, the retrieval of the complete RDF graph of a service took around 1s.

Summarizing, content adaptation with Web Services is possible and fast enough in a controlled environment. The conformance check required for ensuring interoperability is very fast, as expected. However, the service selection needs to be optimized.

Chapter 6

Summary and Future Work

6.1 Summary

Web Service infrastructures, and service-oriented architectures (SOA) have become the widely adopted solution for creating large scale applications. Exception to this evolution are large scale multimedia applications, which still widely using monolithic approaches. The goal of the PUMA architecture is to show that *large scale multimedia applications can be realized using service-oriented approaches*. Furthermore, we want to show that these types of applications can benefit from the flexibility of Web service architectures, and allow complex adaptation workflows, which not only adapt the source content format for the plethora of different devices, but additionally allow the creation of new content, on-the-fly, e.g. by enriching a movie with subtitles from third-party providers. In a business scenario, this allows service providers to create new applications specifically tailored to the requirements of each user. Quality-of-Service plays an important part in multimedia content delivery. The service compositions need constant maintenance, as services are failing, the network topology can change, and the user requirements can evolve. All of these changes must be dealt with in a timely fashion, and optimally done, without interrupting or aborting the content delivery.

For this, constantly monitoring the state of the service composition components is necessary, as well as modifying the workflow, and replacing services to ensure a seamless content delivery.

The main contribution of this thesis is a comprehensive proposal of an infrastructure for

1. selecting services for a service composition —defined by a complex workflow, or choreography— using reasoning about service capabilities, and

2. verification of the interoperability between participants of such a service composition w.r.t. to complex interaction protocols.

Service Selection For performing the content adaptation, the systems creates an adaptation workflow which consists of the single steps (or roles) that are necessary to transform the content as desired by the user. To create a service composition, for each of the roles in the workflow a suitable service needs to be found. Commonly, a central repository is used to search for services, but for PUMA the description of services was extended with semantic annotations. The semantic annotations, provided by OWL-S, are used to match services with roles, and is the first step for creating a sound service composition.

The contribution of this work is the extension of the semantic web service description language (OWL-S) with service *capabilities*, describing the type of adaptation a service provides, and *policies*, which are used to specify the interactive behavior of a service, i.e. the sequence of messages it receives and sends. While the field of semantic matchmaking is well researched, we actually provide an implementation of such an algorithm, based on our extensions to OWL-S, using state-of-the-art semantic web technologies, and that is fast enough to not delay the startup of the content delivery for too long.

Validation of Service Interoperability A service composition is only able to work, if the possible conversations (message exchanges) produced by the participants follow the required protocols. We developed and implemented protocols based on Web service message exchange for quality-of-service monitoring and transport of continuous data over Web services. The validation of service interoperability stems from techniques used in multi-agent systems. Finite state automata (FSA) are a popular model to represent communicating processes. Bisimulation is one method to check whether a service's policy conforms to the protocol specification of the role the service plays in the composition. We extended the semantic Web service description language OWL-S with the required vocabulary to represent conversation policies. Bisimulation checks for the equivalence of two automata. However, to be conformant, the implementation (interaction policy) does not have to be a *precise* copy of the protocol. That is, as service can provide the means to handle more types of messages than are actually required by the protocol. Functionality, that is not required by a protocol role, and thus will never be called upon, does not disrupt interoperability. For this, we implemented a form of *weak bisimulation* that takes into account the incoming and outgoing messages and verifies conformance w.r.t. a protocol, rather than equivalence. The initial interoperability verification has to be done a-priori, before the actual instantiation of the service chain. However, the dynamic nature of the application requires the replacement of services on-the-fly. For this, a new *compatible* service has to be found and seamlessly integrated into the service composition. The conformance check ensures that the newly integrated service is interoperable with the other components in the system.

Evaluation We have implemented a prototype of the PUMA architecture, and used it as a test-bed and evaluation platform for workflow selection, multimedia adaptation services, the client application, the protocols for continuous data transport (streaming) and the algorithms for selecting and validating service compositions. The first results retrieved in this setting indicate, that service oriented adaptation of streaming content is possible. Furthermore, as recovery from service failures is performed fast enough in

certain scenarios, this allows the system to recover from this interruption. The startup times are comparable, if not faster, than similar solutions which are based on P2P or monolithic approaches, which are only providing content delivery without adaptation.

6.2 Future work

Authors from both the Multi-agent Systems and Web Service community have noted a convergence between the two fields [58, 16]. Model checking approaches from the domain of Agent Communication Languages have been adopted for use in Web Service compositions.

- The protocols, which are used in the implementation of the PUMA system are simple, and have the properties required by the verification method shown in Chapter 4. The approach shown in [17], while ideal for communication between just two partners, is not suitable for multi-party protocols. Although the most recent proposal solves this problem, the protocols and policies need to be annotated with extra information [15]. For more complex protocols, the possibility of using a different model for conformance checking needs to be investigated, e.g. Petri Nets.
- Speed of the verification is important for seamless content delivery, and as such, there exist many opportunities to investigate optimizations of the workflow creation, selection and verification step. Pre-compilation of Choreography and Web Service description into state machines is only one possibility
- Quality of Service needs to be addressed on a much broader scale, not only in terms of the simple cost function currently used for service selection. An implementation is needed, which makes use of the available techniques for controlling quality in a multimedia environment. For this, the monitoring facilities already in place in the PUMA framework need to be extended with QoS functionality.
- Currently, PUMA-adapted content requires a proprietary client application. By creating a service for protocol adaptation, and adding it to the workflow, other clients will be able to connect to the PUMA system.
- Recent work on interoperability verification [15] presents an update and patch mechanism for agents and services. A compatible update of a service is a modification to the service's implementation, that does not break interoperability. This is relevant for replacing services which fail, as a different, possibly faster interoperability check can be used.
- The adaptation services used in the evaluation of the prototype are based on generic adaptation tools. The PUMA framework was designed to allow the use of highly specialized services, were each service only provides few capabilities, but does so in a very efficient and effective way. The goal is to provide a set of perfect services for the most important tasks in content adaptation.

Appendix A

Appendix

A.1 Common Vocabularies and Ontologies

RDF and XML basic vocabularies

- *short description*
  abbrev: `<namespace>`
- Resource Description Framework - Syntax [24]
  `rdf:` `<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
- Resource Description Framework (RDF) - Schema [43]
  `rdfs:` `<http://www.w3.org/2000/01/rdf-schema#>`
- Web Ontology Language (OWL) [23]
  `owl:` `<http://www.w3.org/2002/07/owl#>`
- XML Schema Datatype Definitions (XSD) [62]
  `xsd:` `<http://www.w3.org/2001/XMLSchema#>`
- XML Schema Datatype Instances [154]
  `xsi:` `<http://www.w3.org/2001/XMLSchema-instance#>`

Web Standards

- Dublin Core (DC) - Element Set - Version 1.1 [107]
  `dc:` `<http://purl.org/dc/elements/1.1/>`
- Friend Of a Friend (FOAF) [44], Schema for People and Social Networks.
  `foaf:` `<http://xmlns.com/foaf/0.1/>`
- Semantic Web Status Ontology: stable, unstable, testing
  `vs:` `<http://www.w3.org/2003/06/sw-vocab-status/ns>`
- Web Of Trust RDF Ontology
  `wot:` `<http://xmlns.com/wot/0.1/>`
- Web Service Description Language [54]
  `wsdl:` `<http://schemas.xmlsoap.org/wsdl/>`

- BPEL - Variables and Properties
  `vprop: <http://docs.oasis-open.org/wsbpel/2.0/varprop>`
- BPEL - Services
  `sref: <http://docs.oasis-open.org/wsbpel/2.0/serviceref>`
- BPEL - Partner Link Types
  `plnk: <http://docs.oasis-open.org/wsbpel/2.0/plnktype>`
- BPEL - Executable Process Schema
  `bpel: <http://docs.oasis-open.org/wsbpel/2.0/process/executable>`
- BPEL - Abstract Process Schema
  `abstract:<http://docs.oasis-open.org/wsbpel/2.0/process/abstract>`
- Semantic Web for Researchers Ontology
  `swrc: <http://swrc.ontoware.org/ontology#>`
- OWL-S - Ontology for Semantic Web Services
  `service: <http://www.daml.org/services/owl-s/1.2/Service.owl#>`
- OWL-S - Service Grounding Ontology
  `grounding:<http://www.daml.org/services/owl-s/1.2/Grounding.owl#>`
- OWL-S - Process Model Ontology
  `process: <http://www.daml.org/services/owl-s/1.2/Process.owl#>`
- OWL-S - Service Profile Ontology
  `profile: <http://www.daml.org/services/owl-s/1.2/Profile.owl#>`
- A Semantic Web Rule Language Combining OWL and RuleML [88]
  `swrl: <http://www.w3.org/2003/11/swrl#>`

MPEG 7 / MPEG 21 - Multimedia Standards This is just a small selection of the existing schemas, there exist more than 50 vocabularies for digital rights management (DRM) alone.

The files are available from the ISO Standard Website: `http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-7_schema_files/`, `http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/`

- MPEG 7 - Schema
  `mpeg7: <urn:mpeg:mpeg7:schema:2001>`
- MPEG 21 - Digital Item Description
  `did: <urn:mpeg:mpeg21:2002:02-DIDL-NS>`
- MPEG 21 - Digital Item Adapatation
  `dia: <urn:mpeg:mpeg21:2003:01-DIA-NS>`

A.2 The PUMA Workflow Vocabulary

The RDF vocabularies and ontologies created for the PUMA project can be found in the package "de.l3s.puma.model" in the PUMA maven repository: `http://www.l3s.de/~puma/de.l3s.puma.model/`

- Taxonomy for Service Categorization: Digital Item Adaptation
  `adapt: <http://www.l3s.de/puma/Adaptation#>`
- Simple RDF Binding for Elements from WS-CDL
  `config: <http://www.l3s.de/puma/Orchestration#>`
- Simple RDF Binding for Concepts from MPEG-7/21
  `pumamm: <http://www.l3s.de/puma/Multimedia#>`
- Vocabulary for Modelling Adaptation Workflows (borrows concepts from YAWL and BPEL
  `pumawf: <http://www.l3s.de/puma/Workflow#>`

A.3 Software Tools

PUMA is depending on the following tools and libraries

1. Maven (Available at `http://maven.apache.org/`)

   Maven is a build manager for Java Projects. It simplifies the compiling of code and documentation, testing, deploying and versioning. PUMA supplies a plugin for maven that creates Java Classes from RDF Vocabularies.

2. GlassFish (Available at `http://java.sun.com/javaee/community/glassfish/`)

   Project GlassFish is an open-source implementation of the Java EE 5 specifications.

3. CXF (Available at `http://cxf.apache.org/`)

   Apache CXF is an open source service framework, that supports SOAP, Basic Profiles, WSDL, WS-Addressing, WS-Policy, WS-ReliableMessaging and WS-Security.

4. Sesame (Available at `http://www.openrdf.org/`)

   Sesame is an open source RDF framework with support for RDF Schema inferencing and querying.

5. MySQL (Available at `http://www.mysql.de/`)

   MySQL is a freely available open source relational database system. In PUMA it is used to store the data of the Sesame RDF repository.

Multimedia Frameworks

6. ffmpeg (Available at `http://ffmpeg.mplayerhq.hu/`)

   FFmpeg is a collection of open source tools, to record, convert and stream audio and video. It *includes* the **libavcodec** library.

7. mplayer, mencoder (Available at `http://www.mplayerhq.hu/`)

   MPlayer is an open source video player application, that is bundled with MEncoder, a powerful tool for transcoding, adaptation and encoding.

8. jmf (Available at `http://java.sun.com/javase/technologies/desktop/media/jmf/index.jsp`)

   The Java Media Framework is a platform-independent system for displaying time-based media, such as video and audio. The latest version, JMF 2.1.1e was released in 2003.

9. vlc, jvlc (Available at `http://www.videolan.org/vlc/`)

   The VLC media player is multimedia player, that can also be used as a streaming server. JVLC is a Java binding for accessing the playback, transcoding and streaming capabilities of the VLC library `http://trac.videolan.org/jvlc/`

10. Jffmpeg (Available at `http://jffmpeg.sourceforge.net/`)

    Jffmpeg is a wrapper that enables using the codecs provided by ffmpeg in JMF.

11. Fobs4JMF (Available at `http://fobs.sourceforge.net/f4jmf_first.html`)

    Fobs4JMF is a different wrapper implementation for the use of ffmpeg codecs in JMF.

12. IBM Toolkit for MPEG-4 (Available at `http://www.alphaworks.ibm.com/tech/tk4mpeg4`)

    IBM provides a collection of Java Classes and sample applications for viewing and generating MPEG-4 content.

List of Publications

Journal articles

1. **Super-peer-based routing strategies for RDF-based peer-to-peer networks**,
   Journal of Web Semantics 1(2), pp. 177-186, 2004
   Co-authors: Wolfgang Nejdl, Martin Wolpers, Wolf Siberski, Christoph Schmitz, Mario Schlosser, Alexander Löser

Conference publications

2. **Ranking Categories for Web Search**,
   30th European Conference on IR Research (ECIR 2008), Glasgow, UK, March 30-April 3, Proceedings, pp. 564-569 Springer, 978-3-540-78645-0
   Co-authors: Gianluca Demartini, Paul-Alexandru Chirita, Wolfgang Nejdl

3. **Multimedia Content Provisioning using Service Oriented Architectures**,
   6th International Conference on Web Services (ICWS 2008),
   to appear
   Co-authors: Sascha Tönnies, Wolf-Tilo Balke

4. **Reasoning-based Curriculum Sequencing and Validation: Integration in a Service-Oriented Architecture**,
   Second European Conference on Technology Enhanced Learning (EC-TEL 2007), Crete, Greece, September 17-20, 2007
   Co-authors: Matteo Baldoni, Cristina Baroglio, Elisa Marengo, Viviana Patti

5. **Interoperability for peer-to-peer networks: Opening p2p to the rest of the world**,
   European Conference on Technology Enhanced Learning (EC-TEL 2006), Heraklion, Greece, Oct 2006. Springer
   Co-authors: Daniel Olmedilla

6. **Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-To-Peer Networks**,
   12th International World Wide Web Conference (WWW2003), Budapest, Hungary, May 2003
   Co-authors: Wolfgang Nejdl, Martin Wolpers, Wolf Siberski, Christoph Schmitz, Mario Schlosser, Alexander Löser

Workshop publications

7. **A Service-Oriented Approach for Curriculum Planning and Validation**,
   Multi-Agent Logics, Languages, and Organisations, Federated Workshops, MALLOW'007 Agent, Web Services and Ontologies, Integrated Methodologies (MALLOW-AWESOME'007) workshop, Durham, GB, September 2007
   Co-authors: Matteo Baldoni, Cristina Baroglio, Elisa Marengo, Viviana Patti

8. **Personal Reader Agent: Personalized Access to Configurable Web Services**,
   14th Workshop on Adaptivity and User Modeling in Interactive Systems (ABIS 2006), Hildesheim, October 9-11, 2006
   Co-authors: Fabian Abel, Nicola Henze, Daniel Krause, Kashif Mushtaq, Peyman Nasirifard and Kai Tomaschewski

9. **A Personalization Service for Curriculum Planning**,
   14th Workshop on Adaptivity and User Modeling in Interactive Systems (ABIS 2006), Hildesheim, October 9-11, 2006
   Co-authors: Matteo Baldoni, Cristina Baroglio, Nicola Henze, Elisa Marengo and Viviana Patti

10. **The Beagle++ Toolbox: Towards an Extendable Desktop Search Architecture**,
    2nd Semantic Desktop Workshop held at the 5th International Semantic Web Conference (ISWC 2005), Athens, GA, United States
    Co-authors: Paul-Alexandru Chirita, Stefania Costache, Julien Gaugaz, Ekaterini Ioannou, Tereza Iofciu, Enrico Minack, Wolfgang Nejdl and Raluca Paiu

11. **User Awareness in Semantic Portals**,
    International Workshop on Personalization on the Semantic Web (PerSWeb 2005), July24-25 Edinburgh, UK
    Co-authors: Nicola Henze

12. **Semantic Caching in Schema-Based Peer-to-Peer Networks**,
    Workshop on Databases, Information Systems and P2P (DBISP2P), August 28/29, Trondheim, Norway, in conjunction with 31st International Conference on Very Large Data Bases (VLDB 2005)
    Co-authors: Hadhami Dhraief

13. **Distributed Queries and Query Optimization in Schema-Based P2P-Systems**,
    Workshop on Databases, Information Systems and P2P (DBISP2P), September 7-8, Humboldt University, Berlin, Germany in conjucntion with 29st International Conference on Very Large Data Bases (VLDB 2003)
    Co-authors: Hadhami Dhraief, Alfons Kemper, Wolfgang Nejdl, Christian Wiesner

14. **An O-Telos provider peer for the RDF-based Edutella P2P-network**,
    Proceedings of Semantic Authoring, Annotation and Knowledge Markup Workshop (SAAKM 2002) at 15th European Conf. on Artificial Intelligence, July 2002, Lyon, France
    Co-authors: Martin Wolpers, Wolfgang Nejdl

15. **Using an O-Telos Peer to Provide Reasoning Capabilities in an RDF-based P2P-Environment**,
    Proceedings of International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2002) at International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), July 2002, Bologna, Italy
    Co-authors: Martin Wolpers, Wolfgang Nejdl

Technical Reports, Posters

16. **A personalization web service for curricula planning and validation**,
4th European Semantic Web Conference (ESWC 2007) June 3-7 Innsbruck, Austria. Poster Session
Co-authors: Viviana Patti, Matteo Baldoni, Cristina Baroglio and Elisa Marengo,

List of Figures

List of Tables

Index

Bibliography

[1] **Fabian Abel, Robert Baumgartner, Adrian Brooks, Christian Enzi, Georg Gottlob, Nicola Henze, Marcus Herzog, Matthias Kriesell, Wolfgang Nejdl, and Kai Tomaschewski**. The Personal Publication Reader. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 1050–1053. Springer, 2005.

[2] **Fabian Abel, Ingo Brunkhorst, Nicola Henze, Daniel Krause, K. Mushtaq, P. Nasirifard, and Kai Tomaschewski**. Personal Reader Agent : Personalized Access to Configurable Web Services. In Klaus-Dieter Althoff and Martin Schaaf, editors, *LWA*, volume 1/2006 of *Hildesheimer Informatik-Berichte*, pages 12–13. University of Hildesheim, Institute of Computer Science, 2006.

[3] **Marco Alberti, Anna Ciampolini, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni**. A Social ACL Semantics by Deontic Constraints. In Vladimír Marík, Jörg P. Müller, and Michal Pechoucek, editors, *CEEMAS*, volume 2691 of *Lecture Notes in Computer Science*, pages 204–213. Springer, 2003.

[4] **Marco Alberti, Davide Daolio, Paolo Torroni, Marco Gavanelli, Evelina Lamma, and Paola Mello**. Specification and verification of agent interaction protocols in a logic-based system. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 72–78, New York, NY, USA, 2004. ACM.

[5] **Marco Alberti, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni**. Specification and Verification of Agent Interaction using Social Integrity Constraints. In *LCMAS'03: Logic and Communication in Multi-Agent Systems*, volume 85(2) of *ENTCS*, pages 94–116, Eindhoven, the Netherlands, 29 June 2003. Elsevier.

[6] **Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana**. Business Process Execution Language for Web Services Version 1.1. Technical report, BEA Systems, International Business Machines Corporation, Microsoft Corporation, SAP AG, Siebel Systems, May 2003.

[7] **Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Kim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu**. Web Services Agreement Specification (WS-Agreement). Technical report, Global Grid Forum (GGF), September 2005.

[8] **Stephanos Androutsellis-Theotokis and Diomidis Spinellis**. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4):335–371, December 2004.

[9] **André Arnold**. *Finite transition systems: semantics of communicating systems*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1994. Translator-John Plaice.

[10] **John Langshaw Austin**. *How to Do Things with Words*. Harvard University Press, Cambridge, MA, 1975.

[11] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, New York, NY, USA, 2003.

[12] **Matteo Baldoni, Christina Baroglio, Ingo Brunkhorst, Elisa Marengo, and Viviana Patti**. A Service-Oriented Approach for Curriculum Planning and Validation. In *Multi-Agent Logics, Languages, and Organisations, Federated Workshops, MALLOW'007, Agent, Web Services and Ontologies, Integrated Methodologies (MALLOW-AWESOME'007)*, Durham, GB, September 2007.

[13] **Matteo Baldoni, Cristina Baroglio, Ingo Brunkhorst, Nicola Henze, Elisa Marengo, and Viviana Patti**. A Personalization Service for Curriculum Planning. In Martin Schaaf and Klaus-Dieter Althoff, editors, *Lernen - Wissensentdeckung - Adaptivit"at*, pages 17–21, Hildesheim, Germany, October 2006. Gesellschaft für Informatik e.V. (GI).

[14] **Matteo Baldoni, Cristina Baroglio, Ingo Brunkhorst, Elisa Marengo, and Viviana Patti**. Reasoning-Based Curriculum Sequencing and Validation: Integration in a Service-Oriented Architecture. In Erik Duval, Ralf Klamma, and Martin Wolpers, editors, *EC-TEL*, volume 4753 of *Lecture Notes in Computer Science*, pages 426–431. Springer, 2007.

[15] **Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, Nirmit Desai, Viviana Patti, and Munindar P. Singh**. Choice, Interoperability, and Conformance in Service Choreographies. Technical report, University di Torino, North Carolina State University, and IBM India Research Labs, October 2008. in submission.

[16] **Matteo Baldoni, Cristina Baroglio, Alberto Martelli, and Viviana Patti**. Verification of Protocol Conformance and Agent Interoperability. In Francesca Toni and Paolo Torroni, editors, *CLIMA VI*, volume 3900 of *Lecture Notes in Computer Science*, pages 265–283. Springer, 2005.

[17] **Matteo Baldoni, Cristina Baroglio, Alberto Martelli, and Viviana Patti**. A Priori Conformance Verification for Guaranteeing Interoperability in Open Environments. In Asit Dan and Winfried Lamersdorf, editors, *ICSOC*, volume 4294 of *Lecture Notes in Computer Science*, pages 339–351. Springer, 2006.

[18] **Matteo Baldoni, Cristina Baroglio, Alberto Martelli, Viviana Patti, and Claudio Schifanella**. Interaction Protocols and Capabilities: A Preliminary Report. In José Júlio Alferes, James Bailey, Wolfgang May, and Uta Schwertel, editors, *PPSWR*, volume 4187 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2006.

[19] **Wolf-Tilo Balke, Ingo Brunkhorst, and Sascha Tönnies**. Multimedia Content Provisioning using Service Oriented Architectures. In *6th International Conference on Web Services (ICWS)*, Beijing, China, September 2008.

[20] **Wolf-Tilo Balke and Jorg Diederich**. A Quality- and Cost-based Selection Model for Multimedia Service Composition in Mobile Environments. In *ICWS '06: Proceedings of the IEEE International Conference on Web Services*, pages 621–628, Washington, DC, USA, 2006. IEEE Computer Society.

[21] **Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden**. DL Meets P2P - Distributed Document Retrieval Based on Classification and Content. In Andreas Rauber, Stavros Christodoulakis, and A. Min Tjoa, editors, *ECDL*, volume 3652 of *Lecture Notes in Computer Science*, pages 379–390. Springer, 2005.

[22] **Arindam Banerji, Claudio Bartolini, Dorothea Beringer, Venkatesh Chopella, Kannan Govindarajan, Alan Karp, Harumi Kuno, Mike Lemon, Gregory Pogossiants, Shamik Sharma, and Scott Williams**. Web Services Conversation Language (WSCL) 1.0. W3c note, World Wide Web Consortium (W3C), March 2002.
`http://www.w3.org/TR/wscl10/`.

[23] **Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein**. OWL Web Ontology Language: Reference. Technical report, World Wide Web Consortium (W3C), 2004.

[24] **Dave Beckett**. RDF/XML Syntax Specification. W3C Recommendation, World Wide Web Consortium (W3C), February 2004.
`http://www.w3.org/TR/rdf-mt/`.

[25] **B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen**. *Systems and Software Verification Model-Checking Techniques and Tools*, volume XII. Vuibert, Paris, 2001.

[26] **J. A. Bergstra and J. W. Klop**. ACPτ: a universal axiom system for process specification. In *Algebraic methods: theory, tools and applications*, pages 447–463. Springer-Verlag New York, Inc., New York, NY, USA, 1989.

[27] **Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci**. Cross-Technique Mediation of User Models. In Vincent P. Wade, Helen Ashman, and Barry Smyth, editors, *AH*, volume 4018 of *Lecture Notes in Computer Science*, pages 21–30. Springer, 2006.

[28] **Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci**. Mediation of user models for enhanced personalization in recommender systems. *User Model. User-Adapt. Interact.*, 18(3):245–286, 2008.

[29] **T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter**. Uniform Resource Identifiers (URI): Generic Syntax. Request for comments, Network Working Group, August 1998.

[30] **Tim Berners-Lee**. Notation 3: An readable language for data on the Web. Introduction, World Wide Web Consortium (W3C), 1998-2006.

[31] **Tim Berners-Lee, James Hendler, and Ora Lassila**. The Semantic Web (Berners-Lee et. al 2001). May 2001.

[32] **Paul V. Biron and Ashok Malhotra**. XML Schema Part 2: Datatypes. Technical Report Second Edition, World Wide Web Consortium (W3C), October 2004.

[33] **Andrew D. Birrell and Bruce Jay Nelson**. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1):39–59, February 1984.

[34] **Andrew P. Black, Jie Huang, Rainer Koster, Jonathan Walpole, and Calton Pu**. Infopipes: an abstraction for multimedia streaming. *Multimedia Syst.*, 8(5):406–419, 2002.

[35] **J. D. Blower, A. B. Harrison, and K. Haines**. Styx Grid Services: Lightweight, Easy-to-Use Middleware for Scientific Workflows. In Vassil N. Alexandrov, G. Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, *International Conference on Computational Science (3)*, volume 3993 of *Lecture Notes in Computer Science*, pages 996–1003. Springer, 2006.

[36] **Harold Boley and Michael Kifer**. RIF Basic Logic Dialect. Technical Report Working Draft, World Wide Web Consortium (W3C), October 2007.

[37] **David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard**. Web Services Architecture. Technical report, Web Services Architecture Working Group, 11 February 2004.

[38] **Lucas Bordeaux, Gwen Salaün, Daniela Berardi, and Massimo Mecella**. When are Two Web Services Compatible? In Ming-Chien Shan, Umeshwar Dayal, and Meichun Hsu, editors, *TES*, volume 3324 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 2004.

[39] **László Böszörményi, Hermann Hellwagner, and Peter Schojer**. Metadata-driven optimal transcoding in a multimedia proxy. *Multimedia Syst.*, 13(1):51–68, 2007.

[40] **Mario Bravetti and Gianluigi Zavattaro**. Contract Based Multi-party Service Composition. In Farhad Arbab and Marjan Sirjani, editors, *FSEN*, volume 4767 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2007.

[41] **Mario Bravetti and Gianluigi Zavattaro**. A Theory for Strong Service Compliance. In Amy L. Murphy and Jan Vitek, editors, *COORDINATION*, volume 4467 of *Lecture Notes in Computer Science*, pages 96–112. Springer, 2007.

[42] **Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau**. Extensible Markup Language (XML) 1.0. Technical report, World Wide Web Consortium (W3C), September 2006.

[43] **Dan Brickley and R.V. Guha**. RDF Vocabulary Description Language 1.0: RDF Schema, February 2004.
`http://www.w3.org/TR/rdf-schema/`.

[44] **Dan Brickley and Libby Miller**. FOAF Vocabulary Specification. Technical Report 0.91, November 2007.

[45] **Jeen Broekstra and Arjohn Kampman**. SeRQL: A Second Generation RDF Query Language. In *In Proc. SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, pages 13–14, 2003.

[46] **Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen**. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In Horrocks and Hendler [87], pages 54–68.

[47] **Ingo Brunkhorst and Hadhami Dhraief**. Semantic Caching in Schema-Based Peer-to-Peer Networks. In Gianluca Moro, Sonia Bergamaschi, Sam Joseph, Jean-Henry Morin, and Aris M. Ouksel, editors, *DBISP2P*, volume 4125 of *Lecture Notes in Computer Science*, pages 179–186, Trondheim, Norway, August 2005. Springer.
`http://dx.doi.org/10.1007/978-3-540-71661-7_17`.

[48] **Ingo Brunkhorst, Hadhami Dhraief, Alfons Kemper, Wolfgang Nejdl, and Christian Wiesner**. Distributed Queries and Query Optimization in Schema-Based P2P-Systems. In Karl Aberer, Vana Kalogeraki, and Manolis Koubarakis, editors, *DBISP2P*, volume 2944 of *Lecture Notes in Computer Science*, pages 184–199, Berlin, Germany, September 7-8 2003. Springer.

[49] **Ingo Brunkhorst and Daniel Olmedilla**. Interoperability for Peer-to-Peer Networks: Opening P2P to the Rest of the World. In Wolfgang Nejdl and Klaus Tochtermann, editors, *EC-TEL*, volume 4227 of *Lecture Notes in Computer Science*, pages 45–60. Springer, 2006.

[50] **Nadia Busi, Roberto Gorrieri, Claudio Guidi, Roberto Lucchi, and Gianluigi Zavattaro**. Choreography and Orchestration: A Synergic Approach for System Design. In Boualem Benatallah, Fabio Casati, and Paolo Traverso, editors, *ICSOC*, volume 3826 of *Lecture Notes in Computer Science*, pages 228–240. Springer, 2005.

[51] **Liliana Cabral, John Domingue, Stefania Galizia, Alessio Gugliotta, Vlad Tanasescu, Carlos Pedrinaci, and Barry Norton**. IRS-III: A Broker for Semantic Web Services Based Applications. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 201–214. Springer, 2006.

[52] **Darren Carlson and Andreas Schrader**. Seamless media adaptation with simultaneous processing chains. In *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*, pages 279–282, New York, NY, USA, 2002. ACM.

[53] **Amit K. Chopra and Munindar P. Singh**. Producing Compliant Interactions: Conformance, Coverage, and Interoperability. In Matteo Baldoni and Ulle Endriss, editors, *DALT*, volume 4327 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006.

[54] **Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana**. Web Services Description Language (WSDL) 1.1. Technical report, World Wide Web Consortium (W3C), 15 March 2001.

[55] **Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Marina Mongiello, and Francesco M. Donini**. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. In *ICEC '04: Proceedings of the 6th international conference on Electronic commerce*, pages 41–50, New York, NY, USA, 2004. ACM.

[56] **Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein**. DAML+OIL Reference Description. Technical report, World Wide Web Consortium (W3C), 2001.

[57] **Jörg Diederich, Martin Dzbor, and Diana Maynard**. REASE: the repository for learning units about the semantic web. *New Rev. Hypermedia Multimedia*, 13(2):211–237, 2007.

[58] Frank Dignum, editor. *Advances in Agent Communication, International Workshop on Agent Communication Languages, ACL 2003, Melbourne, Australia, July 14, 2003*, volume 2922 of *Lecture Notes in Computer Science*. Springer, 2004.

[59] **M. Duerst and M. Suignard**. RFC 3987 - Internationalized Resource Identifiers (IRIs). Technical report, IETF, January 2005.

[60] **Erik Duval, Eddy Forte, Kris Cardinaels, Bart Verhoeven, Rafaël Van Durm, Koen Hendrikx, Maria Wentland Forte, Norbert Ebel, Maciej Macowicz, Ken Warkentyne, and Florence Haenni**. The Ariadne knowledge pool system. *Commun. ACM*, 44(5):72–78, 2001.

[61] **Ulrich Endriss, Nicolas Maudet, Fariba Sadri, and Francesca Toni**. Protocol Conformance for Logic-based Agents. In G. Gottlob and T. Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, pages 679–684. Morgan Kaufmann Publishers, August 2003.

[62] **David C. Fallside and Priscilla Walmsley**. XML Schema Part 0: Primer. Technical Report Second Edition, World Wide Web Consortium (W3C), October 2004.

[63] **Zweites Deutsches Fernsehen**. ZDFmediathek, 2008. http://www.zdf.de/ZDFmediathek/content/9602?inPopup=true.

[64] **Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire**. KQML as an agent communication language. In *CIKM '94: Proceedings of the third international conference on Information and knowledge management*, pages 456–463, New York, NY, USA, 1994. ACM.

[65] **Tim Finin, Jay Weber, Gio Wiederhold, Michael Genesereth, Richard Fritzson, Donald McKay, James McGuire, Richard Pelavin, Stuart Shapiro, and Chris Beck**. Specification of the KQML Agent-Communication Language – plus example agent policies and architectures. Technical report, The DARPA Knowledge Sharing Initiative External Interfaces Working Group, 1993.

[66] **Howard Foster, Sebastián Uchitel, Jeff Magee, and Jeff Kramer**. LTSA-WS: a tool for model-based verification of web service compositions and choreography. In Leon J. Osterweil, H. Dieter Rombach, and Mary Lou Soffa, editors, *ICSE*, pages 771–774. ACM, 2006.

[67] **Howard Foster, Sebastián Uchitel, Jeff Magee, and Jeff Kramer**. Model-Based Analysis of Obligations in Web Service Choreography. In *AICT/ICIW*, page 149. IEEE Computer Society, 2006.

[68] **Ian Foster, Carl Kesselman, and Steven Tuecke**. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, 2001.

[69] Foundation for Intelligent Physical Agents, Geneva, Switzerland. *FIPA Communicative Act Library Specification*, December 2002. http://www.fipa.org/specs/fipa00037/SC00037J.html.

[70] **Cédric Fournet, C. A. R. Hoare, Sriram K. Rajamani, and Jakob Rehof**. Stuck-Free Conformance. In Rajeev Alur and Doron Peled, editors, *CAV*, volume 3114 of *Lecture Notes in Computer Science*, pages 242–254. Springer, 2004.

[71] **ARTE G.E.I.E.** Arte+7, 2008. http://plus7.arte.tv/de/1697480.html.

[72] **Laura Giordano, Alberto Martelli, and Camilla Schwind**. Specifying and verifying interaction protocols in a temporal action logic. *J. Applied Logic*, 5(2):214–234, 2007.

[73] **R.J. van Glabbeek**. Bisimulation. Scheduled to appear in the forgotten *Encyclopedia of Distributed Computing* (J.E. Urban & P. Dasgupta, eds.), Kluwer, 2000. Available at http://boole.stanford.edu/pub/bis.ps.gz.

[74] **Stephane Gruber, Jennifer Rexford, and Andrea Basso**. Protocol considerations for a prefix-caching proxy for multimedia streams. In *Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications networking*, pages 657–668, Amsterdam, The Netherlands, The Netherlands, 2000. North-Holland Publishing Co.

[75] **Xiaohui Gu and Klara Nahrstedt**. Distributed multimedia service composition with statistical QoS assurances. *IEEE Transactions on Multimedia*, 8(1):141–151, 2006.

[76] **Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon**. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). Technical report, W3C XML Protocol Working Group, 27 April 2007.

[77] **Volker Haarslev and Ralf Möller**. Racer: An owl reasoning agent for the semantic web. In *In Proc. of the International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with 2003 IEEE/WIC International Conference on Web Intelligence*, pages 91–95. Society Press, 2003.

[78] **Peter Haase, Jeen Broekstra, Andreas Eberhart, and Raphael Volz**. A Comparison of RDF Query Languages. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 502–517. Springer, 2004.

[79] **Armin Haller, Emilia Cimpian, Adrian Mocan, Eyal Oren, and Christoph Bussler**. WSMX - A Semantic Service-Oriented Architecture. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services*, pages 321–328, Washington, DC, USA, 2005. IEEE Computer Society.

[80] **Jeffrey Hau, William Lee, and John Darlington**. A Semantic Similarity Measure for Semantic Web Services.

[81] **Wenbo He and Klara Nahrstedt**. Impact of Upper Layer Adaptation on End-to-end Delay Management in Wireless Ad Hoc Networks. In *RTAS '06: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 59–70, Washington, DC, USA, 2006. IEEE Computer Society.

[82] **Dominik Heckmann, Tim Schwartz, Boris Brandherm, Michael Schmitz, and Margeritta von Wilamowitz-Moellendorff**. Gumo - The General User Model Ontology. In Liliana Ardissono, Paul Brna, and Antonija Mitrovic, editors, *User Modeling*, volume 3538 of *Lecture Notes in Computer Science*, pages 428–432. Springer, 2005.

[83] **Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W. Ross**. A Measurement Study of a Large-Scale P2P IPTV System. *IEEE Transactions on Multimedia*, 9(8):1672–1687, 2007.

[84] **Nicola Henze and Daniel Krause**. Personalized Access to Web Services in the Semantic Web. In *SWUI 2006 - 3rd International Semantic Web User Interaction Workshop, collocated with ISWC 2006*, 2006.

[85] **C. A. R. Hoare**. *Communicating sequential processes*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985.

[86] **Gerard J. Holzmann**. *The SPIN Model Checker. Primer and Reference Manual*, volume 3. Printing. Pearson Education, 75 Airlington Street, Suite 300, Boston, MA 02116, December 2006.

[87] Ian Horrocks and James A. Hendler, editors. *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*, volume 2342 of *Lecture Notes in Computer Science*. Springer, 2002.

[88] **Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean**. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3c member submission, World Wide Web Consortium (W3C), May 2004.

[89] **IEEE**. IEEE standard computer dictionary. A compilation of IEEE standard computer glossaries, Jan 1991.

[90] **Google Inc**. Google Video. `http://video.google.com`.

[91] **PPLive Inc**. PPLive.
http://www.pplive.com/en/index.html.

[92] **Yahoo! Inc**. Yahoo! Video. 701 First Avenue, Sunnyvale, CA 94089
http://video.yahoo.com.

[93] **ISO/IEC**. Universal Multiple-Octet Coded Character Set (UCS), December 2003. International Standard 10646.

[94] **ISO/IEC**. ISO/IEC JTC 1/SC 29 - Programme of Work, 2008.
http://www.itscj.ipsj.or.jp/sc29/29w42911.htm.

[95] **Joxan Jaffar and Michael J. Maher**. Constraint Logic Programming: A Survey. *J. Log. Program.*, 19/20:503–581, 1994.

[96] **Sugih Jamin and Beat Knecht und Wenjie Wang**. Zattoo. Zattoo Europe Ltd.
http://zattoo.com/.

[97] **Dietmar Jannach and Klaus Leopold**. Knowledge-based multimedia adaptation for ubiquitous multimedia consumption. *J. Network and Computer Applications*, 30(3):958–982, 2007.

[98] **Dietmar Jannach, Klaus Leopold, Christian Timmerer, and Hermann Hellwagner**. A knowledge-based framework for multimedia adaptation. *Applied Intelligence*, 24(2):109–125, 2006.

[99] **Diane Jordan, John Evdemon, Alexandre Alves, Assaf Arkin, Sid Askary, Charlton Barreto, Ben Bloch, Francisco Curbera, Mark Ford, Yaron Goland, Alejandro Guízar, Neelakantan Kartha, Canyang Kevin Liu, Rania Khalaf, Dieter König, Mike Marin, Vinkesh Mehta, Satish Thatte, Danny van der Rijn, Prasad Yendluri, and Alex Yiu**. Web Services Business Process Execution Language Version 2.0. Oasis standard, April 2007.

[100] **Nickolaos Kavantzas, David Burdett, and Greg Ritzinger**. Web Services Choreography Description Language Version 1.0. W3c working draft, World Wide Web Consortium (W3C), April 2004.
http://www.w3.org/TR/ws-cdl-10/.

[101] **Nickolas Kavantzas, David Burdett, Gregory Ritzinger, Tony Fletcher, and Yves Lafon**. Web Services Choreography Description Language Version 1.0. Technical report, World Wide Web Consortium (W3C), December 2004.

[102] **Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, and H. Kimata**. RFC 3016 - RTP Payload Format for MPEG-4 Audio/Visual Streams. Request For Comment, Internet Engineering Task Force (IETF), November 2000.

[103] **Matthias Klusch, Benedikt Fries, and Katia Sycara**. Automated semantic web service discovery with OWLS-MX. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 915–922, New York, NY, USA, 2006. ACM.

[104] **Alfred Kobsa**. Generic User Modeling Systems. *User Modeling and User-Adapted Interaction*, 11(1-2):49–63, 2001.

[105] **Benjamin Köhncke and Wolf-Tilo Balke**. Preference-driven personalization for flexible digital item adaptation. *Multimedia Syst.*, 13(2):119–130, 2007.

[106] **Rainer Koster, Andrew P. Black, Jie Huang, Jonathan Walpole, and Calton Pu**. Infopipes for composing distributed information flows. In *In Proceedings of the International Workshop on Multimedia Middleware*, pages 44–47. ACM, 2001.

[107] **J. Kunze and T. Baker**. RFC 5013 – The Dublin Core Metadata Element Set. Technical report, Dublin Core Metadata Initiative, 2007.

[108] **Benjamin Köhnke and Wolf-Tilo Balke**. Preference-Driven Personalization for Flexible Digital Item Adaptation. *SMAP*, 2006.

[109] **Yannis Labrou and Tim Finin**. Semantics and conversations for an agent communication language. pages 235–242, 1998.

[110] **Bo Li, Susu Xie, G.Y. Keung, Jiangchuan Liu, I. Stoica, Hui Zhang, and Xinyan Zhang**. An Empirical Study of the Coolstreaming+ System. *Selected Areas in Communications, IEEE Journal on*, 25(9):1627–1639, December 2007.

[111] **Jin Liang and Klara Nahrstedt**. RandPeer: Membership Management for QoS Sensitive Peer-to-Peer Applications. In *INFOCOM*. IEEE, 2006.

[112] **Heiko Ludwig, Alexander Keller, Asit Dan, Richard P. King, and Richard Franck**. Web Service Level Agreement (WSLA): Language Specification. Technical report, IBM Corporation, January 2003.

[113] **D.-A. Manolescu and K. Nahrstedt**. A Scalable Approach to Continuous-Media Processing. In *RIDE '98: Proceedings of the Workshop on Research Issues in Database Engineering*, page 84, Washington, DC, USA, 1998. IEEE Computer Society.

[114] **David Martin, Mark Burstein, Grit Denker, Daniel Elenius, Joseph Giampapa, Drew McDermott, Deborah McGuinness, Sheila McIlraith, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara**. OWL-S: Semantic Markup for Web Services. Technical report, The OWL-S Coalition, 2005.

[115] **David Martin, Mark Burstein, Grit Denker, Jerry Hobbs, Lalana Kagal, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara**. DAML-S: Semantic Markup for Web Services. Technical report, The DAML Services Coalition, May 2003.

[116] **David Martin, Massimo Paolucci, and Matthias Wagner**. Bringing Semantic Annotations to Web Services: OWL-S from the SAWSDL Perspective. In Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 340–352. Springer, 2007.

[117] **David L. Martin, Mark H. Burstein, Drew V. McDermott, Sheila A. McIlraith, Massimo Paolucci, Katia P. Sycara, Deborah L. McGuinness, Evren Sirin, and Naveen Srinivasan**. Bringing Semantics to Web Services with OWL-S. In *World Wide Web*, pages 243–277, 2007. `http://dx.doi.org/10.1007/s11280-007-0033-x`.

[118] **Brian McBride**. Jena: Implementing the RDF Model and Syntax Specification. In *SemWeb*, 2001.

[119] **Deborah L. McGuinness and Frank van Harmelen**. OWL Web Ontology Language: Overview. Technical report, World Wide Web Consortium (W3C), February 2004.

[120] **George A. Miller**. WordNet: a lexical database for English. *Commun. ACM*, 38(11):39–41, 1995.

[121] **R. Milner**. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[122] **Robin Milner, Joaching Parrow, and David Walker**. A Calculus of Mobile Processes Pt.1. *Information and Computation*, 100(1):1–40, September 1992.

[123] **Robin Milner, Joaching Parrow, and David Walker**. A Calculus of Mobile Processes Pt.2. *Information and Computation*, 100(1):41–77, September 1992.

[124] **Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu**. Peer-to-Peer Computing. Technical report, HP Labs, July 2003.

[125] **Klara Nahrstedt and Wolf-Tilo Balke**. A taxonomy for multimedia service composition. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 88–95, New York, NY, USA, 2004. ACM. General Chair-Henning Schulzrinne and General Chair-Nevenka Dimitrova and Program Chair-Angela Sasse and Program Chair-Sue Moon and Program Chair-Rainer Lienhart.

[126] **Klara Nahrstedt and Wolf-Tilo Balke**. Towards building large scale multimedia systems and applications: challenges and status. In *MSC '05: Proceedings of the first ACM international workshop on Multimedia service composition*, pages 3–10, New York, NY, USA, 2005. ACM.

[127] **Klara Nahrstedt, Bin Yu, Jin Liang, and Yi Cui**. Hourglass multimedia content and service composition framework for smart room environments. *Pervasive Mob. Comput.*, 1(1):43–75, 2005.

[128] **W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser**. Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-to-Peer Networks. In *12th International World Wide Web Conference (WWW'03)*, Budapest, Hungary, may 2003.

[129] **Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch**. EDUTELLA: a P2P networking infrastructure based on RDF. In *WWW*, pages 604–615, 2002.

[130] **James Odell, H. Van Dyke Parunak, and Bernhard Bauer**. Representing agent interaction protocols in UML. In *First international workshop, AOSE 2000 on Agent-oriented software engineering*, pages 121–140, Secaucus, NJ, USA, 2001. Springer-Verlag New York, Inc.

[131] **Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara**. Semantic Matching of Web Services Capabilities. In Horrocks and Hendler [87], pages 333–347.

[132] **Massimo Paolucci, Matthias Wagner, and David Martin**. Grounding OWL-S in SAWSDL. In Bernd J. Krämer, Kwei-Jay Lin, and Priya Narasimhan, editors, *ICSOC*, volume 4749 of *Lecture Notes in Computer Science*, pages 416–421. Springer, 2007.

[133] **M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso**. Planning and Monitoring Web Service Composition. In *In: Workshop on Planning and Scheduling for Web and Grid Services (held in conjunction with The 14th International Conference on Automated Planning and Scheduling. (2004) 70 Ž013 71*, pages 106–115, 2004.

[134] **ARD Play-Out-Center**. ARD und ZDF als IPTV über DSL-Netze zu empfangen, October 2006. `http://www.ard-digital.de/14026`.

[135] **Eric Prud'hommeaux and Andy Seaborne**. SPARQL Query Language for RDF. W3c recommendation, World Wide Web Consortium (W3C), January 2008.

[136] **Tian Qiu, Lei Li, and Pin Lin**. Web Service Discovery with UDDI Based on Semantic Similarity of Service Properties. In *SKG '07: Proceedings of the Third International Conference on Semantics, Knowledge and Grid*, pages 454–457, Washington, DC, USA, 2007. IEEE Computer Society.

[137] **Sriram K. Rajamani and Jakob Rehof**. Conformance Checking for Models of Asynchronous Message Passing Software. In Ed Brinksma and Kim Guldstrand Larsen, editors, *CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification*, pages 166–179, London, UK, 2002. Springer-Verlag.

[138] RealNetworks, Inc., PO Box 91123, Seattle, WA 98111-9223. *Helix Server Administration Guide, Helix(tm) Server Version 12.0*, 28 March 2008. `http://docs.real.com/docs/server12/wireline/HelixServerAdmin_v12.pdf`.

[139] **Dumitru Roman, Holger Lausen, and Uwe Keller**. D2v1.3. Web Service Modeling Ontology (WSMO), October 2006. `http://www.wsmo.org/TR/d2/v1.3/`.

[140] **Atul Sajjanhar, Jingyu Hou, and Yanchun Zhang**. Algorithm for Web Services Matching. In Jeffrey Xu Yu, Xuemin Lin, Hongjun Lu, and Yanchun Zhang, editors, *APWeb*, volume 3007 of *Lecture Notes in Computer Science*, pages 665–670. Springer, 2004.

[141] **Davide Sangiorgi and David Walker**. *PI-Calculus: A Theory of Mobile Processes*. Cambridge University Press, New York, NY, USA, 2001.

[142] **Claudio Schifanella**. *Reasoning on web services with choreographies and capabilities*. PhD thesis, Computer Science Department, University of Torino, 2008.

[143] **H. Schulzrinne, A. Rao, and R. Lanphier**. RFC 2326 - Real Time Streaming Protocol (RTSP). Request For Comment, Internet Engineering Task Force (IETF), April 1998.

[144] **Andy Seaborne**. RDQL - A Query Language for RDF. Technical report, Hewlett-Packard, HP Labs Bristol, January 2004.

[145] **John Rogers Searle**. *Speech Acts*. Cambridge University Press, 1969.

[146] **Munindar P. Singh**. A Semantics for Speech Acts. *Ann. Math. Artif. Intell.*, 8(1-2):47–71, 1993.

[147] **Peter Soetens and Matthias De Geyter**. Multi-step media adaptation: implementation of a knowledge-based engine. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 986–987, New York, NY, USA, 2005. ACM.

[148] **N. Srinivasan, M. Paolucci, and K. Sycara**. Adding OWL-S to UDDI, implementation and throughput. In *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, San Diego, California, USA, July 2004.

[149] **Ralf Steinmetz and Klara Nahrstedt**. *Multimedia: computing, communications and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[150] **Clemént Stenac**. VideoLAN. In *Free and Open Source Software Developers' European Meeting (FOSDEM)*, Brussels, Belgium, February 2008.
`http://www.videolan.org/`.

[151] **Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan**. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, 2001.

[152] **Sun Microsystems**. RFC 1057 - RPC: Remote Procedure Call, Protocol Specification, Version 2. Request For Comment, Internet Engineering Task Force (IETF), June 1988.

[153] **Frederick Winslow Taylor**. *The Principles of Scientific Management*. Gutenberg, 1911.

[154] **Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn**. XML Schema Part 1: Structures. Technical Report Second Edition, World Wide Web Consortium (W3C), October 2004.

[155] **Vassileios Tsetsos, Christos Anagnostopoulos, and Stathes Hadjiefthymiades**. On the Evaluation of Semantic Web Service Matchmaking Systems. In *ECOWS '06: Proceedings of the European Conference on Web Services*, pages 255–264, Washington, DC, USA, 2006. IEEE Computer Society.

[156] **Sascha Tönnies, Benhamin Köhncke, Ingo Brunkhorst, and Wolf-Tilo Balke**. A Service Oriented Architecture for Personalized Universal Media Access. Technical report, in submission, 2008.

[157] **W. M. P. van der Aalst**. Three Good reasons for Using a Petri-net-based Workflow Management System. In S. Navathe and T. Wakayama, editors, *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96),*, pages 179–201, 1996.

[158] **Wil van der Aalst and Maja Pesic**. DecSerFlow: Towards a Truly Declarative Service Flow Language. In Frank Leymann, Wolfgang Reisig, Satish R. Thatte, and Wil van der Aalst, editors, *The Role of Business Processes in Service Oriented Architectures*, number 06291 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[159] **Wil M. P. van der Aalst, Lachlan Aldred, Marlon Dumas, and Arthur H. M. ter Hofstede**. Design and Implementation of the YAWL System. In Anne Persson and Janis Stirna, editors, *CAiSE*, volume 3084 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2004.

[160] **Wil M. P. van der Aalst and Maja Pesic**. DecSerFlow: Towards a Truly Declarative Service Flow Language. In Frank Leymann, Wolfgang Reisig, Satish R. Thatte, and Wil M. P. van der Aalst, editors, *The Role of Business Processes in Service Oriented Architectures*, volume 06291 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), S chloss Dagstuhl, Germany, 2006.

[161] **Wil M. P. van der Aalst and Arthur H. M. ter Hofstede**. YAWL: yet another workflow language. *Inf. Syst.*, 30(4):245–275, 2005.

[162] **Martin Vasko and Schahram Dustdar**. A View Based Analysis of Workflow Modeling Languages. In *PDP '06: Proceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 293–300, Washington, DC, USA, 2006. IEEE Computer Society.

[163] **A. Vetro and C. Timmerer**. Digital item adaptation: overview of standardization and research activities. *Multimedia, IEEE Transactions on*, 7(3):418–426, June 2005.

[164] **Anthony Vetro**. MPEG-21 Digital Item Adaptation: Enabling Universal Multimedia Access. *IEEE MultiMedia*, 11(1):84–87, 2004.

[165] **B. Weidemann**. *Lernen mit Medien*, pages 415–466. Beltz, 2001.

[166] **WfMC**. Workflow Management Coalition: Terminology & Glossary, 1994-1999.

[167] **Kevin Wilkinson, Craig Sayers, Harumi A. Kuno, and Dave Reynolds**. Efficient RDF Storage and Retrieval in Jena2. In Isabel F. Cruz, Vipul Kashyap, Stefan Decker, and Rainer Eckstein, editors, *SWDB*, pages 131–150, 2003.

[168] **Martin Wolpers, Ingo Brunkhorst, and Wolfgang Nejdl**. An O-Telos Provider Peer for the RDF-Based Edutella P2P-Network. In Gianluca Moro and Manolis Koubarakis, editors, *AP2PC*, volume 2530 of *Lecture Notes in Computer Science*, pages 150–157. Springer, 2002.

[169] **Andrew Woolf, Keith Haines, and Chunlei Liu**. A Web Service Model for Climate Data Access on the Grid. *Int. J. High Perform. Comput. Appl.*, 17(3):281–295, 2003.

[170] **Yonglei Yao, Sen Su, and Fangchun Yang**. Precise Matching of Semantic Web Services. In Vassil N. Alexandrov, G. Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, *International Conference on Computational Science (4)*, volume 3994 of *Lecture Notes in Computer Science*, pages 164–167. Springer, 2006.

[171] **LLC YouTube**. YouTube.
`http://www.youtube.com`.