Capability Matching of Data Streams with Network Services

Han Gao[•], Ivan R. Judson[⋆], Thomas Uram[⋆], Terry L. Disz[⋆], Michael E. Papka[•⋆], Rick L. Stevens[•⋆]
[•]*Department of Computer Science*
University of Chicago
{hangao,papka, stevens}@cs.uchicago.edu
[⋆]*Mathematics and Computer Science Department*
Argonne National Laboratory
{judson, turam, disz}@mcs.anl.gov

Abstract

Distributed computing middleware needs to support a wide range of resources, such as diverse software components, various hardware devices, and heterogeneous operating systems and architectures. Current technologies are unable to implement a maintenance-free platform to be compatible with such different computing environments. This situation is presenting an increasing challenge as Grid computing becomes more widespread.

The infrastructure of network services (CENSA and CENSI) has been proposed to address this challenge. A seamless Grid computing environment, supported by network services, is composed of various streams such as data, video, audio, and text. We define a mathematical model of capability matching for three-party agreements: requests from users, resources, and network services. Based on the mathematical model, we provide a general approach for capability matching. We also present a new language schema for capability description. As an example, we embed the general matchmaker in the architecture of the Access Grid. Several tests of accuracy and performance are discussed.

1. Introduction

Many achievements [1–5] have been accomplished in distributed-computing resource management, such as resource description, resource discovery, resource registration, and resource selection. Most matching infrastructures are based on a two-party agreement model: resources and requests from users. According to the requisitions from users, the matcher allocates the corresponding resource to each client properly and precisely. However, today's work force is more mobile and distributed than ever before, with diverse software components and hardware devices used in heterogeneous computing environments. Current technologies are unable to implement a universal infrastructure that is compatible with emerging research results while maintaining existing features. For example, a collaborative bioinformatics project may involve several geophysically distributed research teams, including biology scientists, computer scientists, and statisticians. Some of these teams may be hampered from participating fully because of diverse formats and versions of data streams, various computing abilities, distinct scientific views, geophysical limitations, or social restrictions. Therefore, novel concepts of Web services [6,7], Grid services [8–10], and network services [11] are being explored, with the aim of enabling the seamless integration of a wide range of resources (in this paper, "resource" includes software, hardware, operating system, database, and all kinds of data streams).

We focus in this paper on network services. The speed of today's high-performance networks provides an opportunity to support distributed computing. Indeed, more and more applications are being developed and deployed for data streams with network services [12]. These services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Internet. They perform functions that range from simple requests to complicated research and business processes. For example, some network services transfer different data streams into one standard format accepted by a specific computing environment. Some services either provide users routes to previously unreachable distributed resources or minimize the cost of accessing those resources. Unfortunately, no suitable matchmaker model exists for selecting network services properly and precisely.

To address this issue, we propose a three-party (resources, requests from users, and network services) agreement matching infrastructure, based on a rigorous mathematical model and a set of strict mathematical definitions. A matchmaker not only selects the proper resources whose capabilities are matched for requests from users, but it also introduces the corresponding

services to allow users to work in heterogeneous computing environments.

We also define a lightweight language schema for capability description for users, resources, and network services. This schema describes both generic and detailed capability information. Moreover, it can expand its functionalities to a broader range over the RTP protocol [13].

This paper includes the following topics:

- We first present a mathematical model and definitions for a three-party agreement infrastructure.
- Based on the mathematical description, we use ClassAds [5] and XML [14] as a backbone to design a new lightweight language schema for describing streams and services, which enable matching with or without network services.
- We propose a general strategy and provide an algorithm for capability matching.
- We embed the matchmaker architecture in Access Grid (AG) [15] and implement the matchmaker as a network service via gSOAP [16].
- We discuss several test results regarding the accuracy and performance of our model.

2. Mathematical Model and Definitions

Our three-party matching model is based on the following definitions and properties.

Definition 1: A capability vector space over $\Re^n$ is a set of capability vectors for which the dimension of any capability vector is *n*.

Each type of stream capability has its own capability space.

Definition 2: A capability vector is an element of a capability vector space. In the commonly encountered capability vector space $\Re^n$, a capability vector is given by *n* coordinates and can be specified as $X = (x_1, x_2, ..., x_n)$.

One capability vector represents one stream. As coordinates of the vector, several independent parameters are used to describe one specified capability of the stream. The following property should be obeyed.

Property 1: The parameters of a stream are efficient and independent.

For example, the description of an audio stream always includes four independent and nonredundant parameters: sample rate, bit rate (bandwidth), codec, and mode. An audio stream A, Linear16-18-mono, can be represented by $A = (16Khz, 8kbps, Linear-16K, 1)$, which means the audio stream uses Linear-16k compression codec, the sample rate is 16 KHz, the bandwidth is 8 Kbps, and it has only one channel.

Definition 3: The operation of a stream transformation is the operation of a transformation matrix in capability space that transformed the stream capabilities. A transformation matrix, $T \in \Re^{n \times n}$, is a concise and useful way of uniquely representing and working with stream transformations:

$$Y = TX \qquad X, Y \in \Re^n, T \in \Re^{n \times n}$$

In particular, for each stream capability transformation, there exists exactly one corresponding transformation matrix, and every matrix corresponds to a unique transformation.

Property 2: A network service that transforms streams by each direction is symmetric:

$$Y = TX \text{ and } X = TY \qquad X, Y \in \Re^n, T \in \Re^{n \times n}$$

A capability of a stream can be transformed to another type and also can be transformed back from the same service.

Property 3: A network service that transforms streams by only one direction is antisymmetric:

$$Y = TX \text{ and } X \neq TY \qquad X, Y \in \Re^n, T \in \Re^{n \times n}$$

In most of cases, audio-stream transformations are symmetric, since one obtains the transformation algorithm for one way and can obtain the other way easily. In our example, all audio transformation network services are assumed to be symmetric.

Definition 4: Exact match of two or more stream capabilities is isomorphism from two or more vectors projected into the identical vector in capability space.

Each capability of a stream can be projected into a vector in capability space. If all the vectors have the same coordinates, we have an exact match for all the members in this session. Hence, a common stream capability can be shared in this group.

Definition 5: Exact match with network services is isomorphic from two or more capability vectors

projected into the identical vector in capability space after transformation by transformation matrices.

With network services, some streams may be projected into a common capability vector in capability space by a transformation matrix. Hence, in a group session, if some users do not have the specified common capability, they can still interact with a colleague via network services.

3. General Method and Approach

In Section 2, we presented a mathematical model, definition, and description of capability matching of multimedia streams with or without network services. We now present a heuristic algorithm for capability matching.

Algorithm 1: Searching for a largest common capability vector or a set of common capability vectors from two or more capability vector sets.

In the simple case, without network service, we match all the possible capabilities and obtain

$$v^* = \arg\max\left\{v \middle| v \in S^*\right\} ,$$

where S^* is common capability vector set. Furthermore, according to the criterion or rule function *argmax*, we can obtain the largest common capability vector v^*.

Algorithm 2: Given transformation matrices, searching for a largest common capability vector or a set of common capability vectors and a set of transformation matrices that build the largest common vector or common vector set.

Given network services, we are able to build a larger common capability vector set

$$v^* = \arg\max\left\{v \middle| v \in S^*\right\}$$
$$T^* = \left\{T \middle| v^* = Tv\right\}$$

where $v \in \Re^n, T \in \Re^{n \times n}$. In this case, both the capability vector and transformation matrices are our searching target. This potentially increases the complexity of the approach.

4. Description Language for Stream Capabilities and Network Services

Describing the capabilities of multimedia stream and network services is a challenge for software engineers. Several languages have been developed for this purpose. Web Service Description Language (WSDL) [7] is a standard for service description; it specifies the location of service and the operations (or methods) the service exposes. Interface Description Language (IDL) [17] is a simple syntax for describing the interface of a software component. Both of these languages, however, lack important features such as the detailed description of service capability. On the other hand, the model-based specification language VDM-SL is able to capture by creating a system model and defining how a typical state of the model changes under the effect of operations; however, it has too many details for effective service matching. Gao and his colleagues [18] have defined a finite state model for Web service description, but it has a description of service only.

In this paper, we propose a language, called XML-ClassAd-Schema (XCS), for both stream capability and network service capability. We embed ClassAd [5] in XML [14] and present a schema for capability describing, advertising, requesting, and matching exactly and properly.

4.1 Specification of Stream Capability in CXS

We define a schema of stream capability in CXS as follows:

- ID: ID of group member who has a set of stream capabilities.
- TYPE: Multimedia type of the stream.
- DESCRIPTION: Explanation of terms used in the schema, including the description of stream and context. This entry is designed to be understood easily. It is not to be read, parsed, or compiled by the matchmaker.
- PREFERENCE: List of preferred stream capabilities. The preference depends on the conditions and constraints of the user's facilities or the user's personal choice.
- RESOLUTION: List of user-available stream capabilities. There is no special order for these. The items in the preference entry are a subset of those in the resolution entry.

Different multimedia streams should have different specifications. In an audio stream, each item should have the following structure:

- NAME: Specified capability name for each stream.
- CODEC: Name of the encoding or decoding method for each stream.
- SAMPLE RATE: Number of samples of a sound taken per second to represent the event digitally. It decides the precision of continuous wave reconstruction from samples: the more

samples taken per second, the more accurate the digital representation of the sound.

- BIT RATE: Number of bits in a bit stream occurring per unit time. It always indicates the bandwidth cost for certain stream.
- CHANNEL: Mode of the audio stream. It provides a means for delivering audio signals from one point to another.

4.2 Example of Stream Capability Description in CXS

We give an example of an audio-stream capability description file in CXS.

```
<?xml version="1.0"?>

<!-- EXAMPLE FOR AUDIO STREAM -->

<c>
   <a  n="ID"><s>audio.cs.uchicago.edu</s></a>
   <a n="type"><s>audio</s></a>
   <a n="description"><s>audio stream capability …</s></a>
   <a n="preference"><l>
      <c>
      <a n="name"><s>L16-8K-Mono</s></a>
      <a n="codec"><s>Linear-16</s></a>
      <a n="sample_rate"><n>8000</n></a>
      <a n="bit_rate"><n>128.0</n></a>
      <a n="channel"><n>1</n></a>
      </c></l>
   </a>
   <a n="resolution"><l>
      <c>
      <a n="name"><s>L16-8K-Stereo</s></a>
      <a n="codec"><s>Linear-16</s></a>
      <a n="sample_rate"><n>8000</n></a>
      <a n="bit_rate"><n>256.0</n></a>
      <a n="channel"><n>2</n></a>
      </c>
      … …
      <c>
      <a n="name"><s>LPC-8K-Mono</s></a>
      <a n="codec"><s>LPC</s></a>
      <a n="sample_rate"><n>8000</n></a>
      <a n="bit_rate"><n>5.6</n></a>
      <a n="channel"><n>1</n></a>
      </c></l>
   </a>
</c>
```

Given the ID of the user, the type of stream, the preference list, and the resolution list, this file can exactly represent the audio stream capability for this user.

4.3 Specification of Network Service Capability in CXS

We use audio network service as example. A specification in CXS is a frame with the following structure.

- ID: ID of the network service provider.
- TYPE: Type of network service.
- DESCRIPTION: Ontological description of network service, and explanation of terms used.
- OUTPUT/INPUT: Name of the specified output or input stream.
- OUTPUT/INPUT CODEC: Encoding or decoding algorithm for the output or input stream.
- OUTPUT/INPUT SAMPLE_RATE: Sampling frequency of the output or input stream.
- OUTPUT/INPUT BIT_RATE: Bandwidth cost of the output or input stream.
- OUTPUT/INPUT CHANNEL: Sound effect of the output or input stream. Usually, more channels produce a better sound effect.

4.4 Example of Network Service Capability Description in CXS

We next give an example of an audio network service in CXS.

```
<?xml version="1.0"?>

<!-- EXAMPLE FOR NETWORK SERVICE -->

<c>
   <a
n="ID"><s>networkservice1.gawaine.cs.uchicago.edu</s></a>
   <a n="type"><s>audioService</s></a>
   <a n="description"><s></s></a>
   <a n="output"><s>L16-8K-Mono</s></a>
   <a n="outputcodec"><s>Linear-16</s></a>
   <a n="outputsample_rate"><n>8000</n></a>
   <a n="outputbit_rate"><n>128.0</n></a>
   <a n="outputchannel"><n>1</n></a>
   <a n="input"><s>L16-16K-Mono</s></a>
   <a n="inputcodec"><s>Linear-16</s></a>
   <a n="inputsample_rate"><n>16000</n></a>
   <a n="inputbit_rate"><n>256.0</n></a>
   <a n="inputchannel"><n>1</n></a>
</c>
```

In this example, we assume that the audio network service is symmetric. Hence, we can exchange each item for an output and input stream. For convenience in this example, the higher resolution is put in the output side.

4.5 Specification of Network Service and Stream in ClassAd and XML

ClassAd [6] is a semi-structured language. As discussed earlier, we use ClassAd embedded in XML

[5]. We also specify the behaviors of programs using both ClassAd and XML. See the audio stream example and audio network service example.

5. Architecture of General Matchmaker

The philosophy of the general matchmaker is originally from CENSI and CENSA [11] (Figure 1). The input set includes all kinds of capability description files—stream capability, network service capability, and so forth. The rule set provides some parameters and criteria so that the specified types of capability description files can be understood by the matcher.

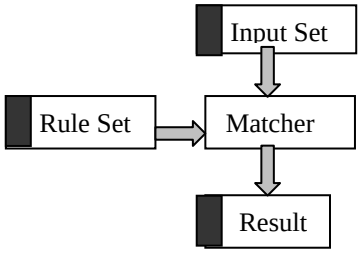

Figure 1: General matchmaker: each dark block represents a schema for each capability description file and rule set file.

The general matchmaker includes the following properties:

- Reusability – It can match whatever the resources are: stream capability (audio, video), network services, and resources.
- Portability – It can be embedded in any resource matching or selecting architecture.
- Reconfigurability – Its schemas and core matching algorithms can be rewritten without affecting other components.

We use the Access Grid (AG) as a reference model to implement this architecture (Figure 2).

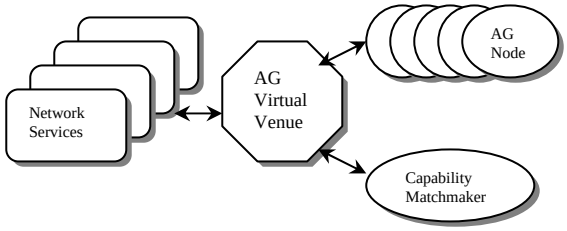

Figure 2: Matchmaker is embedded in the AG for capability matching with network services.

The AG virtual venue collects all the capability files, which represent the current facilities, resources, services, and environment from users whenever they log into the AG.

6. Implementation

As discussed earlier, we use C++ to initialize a generic model for matching. Via gSOAP [16], we obtain a Web service implementation of this matchmaker model. The advantage of this tool is that we can develop the C++ codes without thinking about the restrictions, type definitions, and syntax matters of SOAP [19].

In our model, for simplicity and efficiency, we have integrated all the details into one function for matcher server and client. This function is defined in the *matcher.h* file:

String matcher_match(list **inputs, int rules)*

The function comprises two inputs, data input and rule set, and one outcome, result. The data input is a list of strings (the capability description files for users, resource and network services). The rule set is encoded into integers, each representing a distinct rule set. The structure of program also follows the philosophy of our general matchmaker in Figure 1.

7. Results

We conducted several experiments to test the accuracy and performance of our new tool.

7.1 Capability Matching without Network Services

Suppose we have four users in one group session. The audio capability files for each user are described as follows:

User1: {(L16-8K-Mono, L16, 8K, 1)(preferred),
(L16-16K-Mono, L16, 16K, 1)};
User2: {(L16-8K-Mono, L16, 8K, 1)(preferred),
(L16-32K-Mono, L16, 32K, 1)};
User3: {(L16-8K-Mono, L16, 8K, 1)(preferred),
(L8-8K-Mono, L8, 8K, 1),
(L8-16K-Mono, L8, 16K, 1)};
User3: {(L16-8K-Mono, L16, 8K, 1)(preferred),
(G276-32-16K-Mono),
(LPC-8K-Mono)}

The result is as follows:

The following preferred streams are available:

The stream L16-8K-Mono is available for all 4 users
Codec : Linear-16
Sample Rate : 8000Hz

Bit Rate : 128kbps
Channel : 1

All of the available streams as follows:

The stream L16-8K-Mono is available for all 4 users
Codec : Linear-16
Sample Rate : 8000Hz
Bit Rate : 128kbps
Channel : 1

That is, we obtain one standard audio stream: *(L16-8K-Mono, L16, 8K, 1)*. It is also the preference of all four users.

7.2 Capability Matching with Network Services

We again have four users in one group session, but we have added in this test two network services.

Network service 1:
(L16-8K-Mono, L16, 8K, 1)↔(L16-16K-Mono, L16, 16K, 1)
Network service 2:
(L8-8K-Mono, L8, 8K, 1) ↔(L16-16K-Mono, L16, 16K, 1

Matching Result:

The following prefered streams are available:

The stream L16-8K-Mono is available for all 4 users
……
All of the available streams as following:

audio2.gawaine.cs.uchicago.edu:networkservice1.gawaine.cs.uchicago.edu:L16-16K-Mono <--> L16-8K-Mono

audio3.gawaine.cs.uchicago.edu:networkservice1.gawaine.cs.uchicago.edu:L16-16K-Mono <--> L16-8K-Mono

audio3.gawaine.cs.uchicago.edu:networkservice2.gawaine.cs.uchicago.edu:L16-16K-Mono <--> L8-8K-Mono

audio4.gawaine.cs.uchicago.edu:networkservice1.gawaine.cs.uchicago.edu:L16-16K-Mono <--> L16-8K-Mono

The stream L16-16K-Mono is available for all 4 users
……
The stream L16-8K-Mono is available for all 4 users
……

We observe that *networkservice1* is able to transfer between codecs *L16-16K-Mono* and *L16-8K-Mono*, whereas *networkservice2* encodes between *L16-16K-Mono* and *L8-8K-Mono*. These two network services make this group session have one acceptable audio stream: *(L16-8K-Mono, L16, 8K, 1)*.

We have only one-hop network service in this example. After adding two more network services, we implement multihop transcoding.

Network service 3:
(L16-24K-Mono, L16, 24K, 1)↔(L16-32K-Mono, L16, 32K, 1)
Network service 4:
(L16-16K-Mono, L16, 16K, 1)↔(L16-24K-Mono, L16, 24K, 1)
Matching Result:
……
audio3.gawaine.cs.uchicago.edu:
networkservice1.gawaine.cs.uchicago.edu: L16-16K-Mono <--> L16-8K-Mono
networkservice4.gawaine.cs.uchicago.edu: L16-24K-Mono <--> L16-16K-Mono
networkservice3.gawaine.cs.uchicago.edu: L16-32K-Mono <--> L16-24K-Mono
……

7.3 Performance Analysis

We use Pentium 4 1.66 Hz, 512 MB memory, RedHat 7.3 as our testbed.

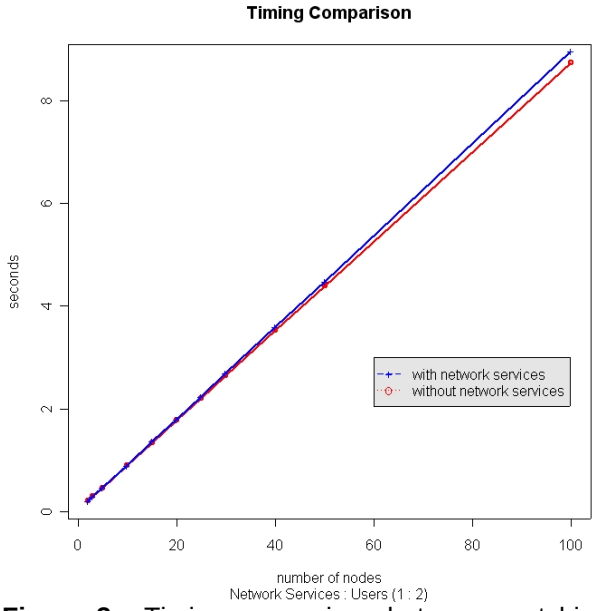

Figure 3: Timing comparison between matching with network service and without network service. The number of nodes ranges from 2 to 100.

In Section 3, we presented a general algorithm for the three-party agreement model. The complexity is *O(n)*. The test result follows the complexity analysis exactly (see Figure 3, which shows that the time cost has a linear relationship with the number of user nodes). Compared with matching without network service, the overhead of network services with matching is relatively small.

In a usual group session over the Access Grid, we have at most 30 user nodes. Thus, the corresponding time cost of capability matching for this session is reasonable.

8. Conclusions and Future Work

Application and data decentralization are becoming a major trend in the industry. We have designed and developed a matchmaker for the three-party agreement model: requests from users, resources, and network services in a distributed computing environment. We also have proposed a lightweight language based on ClassAds and XML for capability description.

Implementing the matchmaker over the RTP (a protocol for real-time data streams) gives us a blueprint for future development of real-time data streams matching. We plan to explore ways in which to improve the generic resource-matching model.

Acknowledgments

We thank the entire Futures Laboratory and Access Grid team. Funding for this work has been provided by the National Science Foundation Middleware Initiative, with additional funding for the Access Grid research and development provided by the U.S. Department of Energy under Contract W-31-109-Eng-38 and from Microsoft Research.

Reference

[1] Chapin, S. J., Katramatos, D., Karpovish, J., and Grimshaw, A., Resource Management in Legion. In *Proc. 5th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP '99)*, San Juan, Puerto Rico, 1999.

[2] Czajkowski, K., Fitzgerald, S., Foster, I., and Kesselman, C., Grid Information Services for Distributed Resource Sharing. In *10th IEEE International Symposium on High Performance Distributed Computing*, IEEE Press, 2001, 181–184.

[3] Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., and Tuecke, S., A Resource Management Architecture for Metacomputing Systems. In *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, 1998, 62–82.

[4] Wolski, R., Spring, N., and Hayes, J. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. *Journal of Future Generation Computing Systems*, 15 (5–6), 757–768. 1999.

[5] Raman, R., Livny, M., and Solomon, M. H. , Matchmaking: Distributed Resource Management for High Throughput Computing. In *High Performance Distributed Computing*. Chicago, IL: IEEE Computer Society, 1998, pp. 140–146.

[6] Universal Description Discovery and Integration (UDDI)." 2001: **http://www.uddi.org/**.

[7] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., Web Services Description Language (WSDL) 1.1, 2001: **http://www.w3.org/TR/wsdl**.

[8] Foster, I., Kesselman, C., Nick, J., and Tuecke, S., The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure Working Group, Global Grid Forum, 2002: **http://www.globus.org/research/papers/ogsa.pdf.**

[9] Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., and Kesselman, C., Grid Service Specification, Open Grid Service Infrastructure Working Group, Global Grid Forum, 2002: **http://www.ggf.org/ogsi-wg/drafts/GS_Spec_draft03_2002-07-17.pdf**.

[10] Foster, I., Kesselman, C., Nick, J., and Tuecke, S., Grid Services for Distributed System Integration, *IEEE Computer*, 35, 37–46, 2002.

[11] Gao, H., Papka, M. E., and Stevens, R., The Design of Network services for Advanced Collaborative Environment. In *Proc. 3rd Workshop of Advanced Collaborative Environment*, Seattle, Washington, June 22, 2003.

[12] WebServices Web page: **http://www.webservices.org/**.

[13] RFC 1889 – RTP: A Transport Protocol for Real-Time Applications: http://www.faqs.org/rfcs/rfc1889.html.

[14] Brown, A., Fuchs, M., Robie, J., and Wadler, P., XML Schema: Formal Description, 2001: **http://www.w3.org/TR/2001/WD-xmlschema-formal/20010925/.**

[15] Access Grid Tutorials Web page: **http://webct.ncsa.uiuc.edu:8900/public/AGIB/**.

[16] van Engelen, R. A., and Gallivan, K. A., The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks. In *Proc IEEE/ACM International Symposium on Cluster Computing and the Grid*, Berlin, Germany, May 21–24, 2002.

[17] Snodgrass, R., *The Interface Description Language: Definition and Use*, Computer Science Press, 1989. ISBN 0-7167-8198-0.

[18] Gao, X., Yang, J., and. Papzoglou, M., The Capability Matching of Web Services. *IEEE Fourth International Symposium on Multimedia Software Engineering (MSE'02)*, IEEE Computer Society, Newport Beach, California, December 11–13, 2002, pp. 56–63.

[19] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, Simple Object Access Protocol (SOAP) 1.1, 2000: **http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.**