Péter Martinek

Schema matching methodologies and run-time solutions in SOA based enterprise application integration

PhD Thesis

ADVISOR
Dr. Szikora Béla

2009
Nyilatkozat

Alulírott Martinek Péter kijelentem, hogy ezt a doktori értekezést magam készítettem, és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltém.

Budapest, 2009-06-29

..........................................
Martinek Péter

A disszertáció bírálatai és a védésről készült jegyzőkönyv megtekinthető a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Karának Dékáni hivatalában.
Abstract

Enterprise Application Integration (EAI) is a permanent need since various information systems are employed at companies. Numerous standard systems of different vendors must be aligned to new or re-designed business processes, requiring for the communication between enterprise applications via given protocols and de facto standards. There are participant systems older than 10 years, and others developed only 1-2 years ago. This implicates a wide technological variance making the integration problem a real challenging issue.

The widespread of the Service Oriented Architecture seems to be one of the most promising approaches in EAI in the last few years. The participating applications provide their capabilities in the form of services acting as an interface to each other. Complex business processes are built upon services realizing a new generation of solutions all in software design, development and EAI.

Although the SOA based approach is already supported by solid technology and tools, the preparations for the integration process – creating service endpoints to legacy systems, aligning them to applied communication standards - are still open issues. Furthermore deploying executable processes, predicting and optimizing their non-functional performance is also often in the focus of current researches. In this thesis I provide a methodology and solution for supporting the preparation for EAI over SOA, and I design and create a solid runtime environment realizing the execution of composite processes between enterprise systems of different vendors.

Within the preparation for the task of integration the focus is set on the finding of semantically related entities between different data structures in the first part of my work. Entities representing the same real world concepts may be labeled and placed totally else how and elsewhere by specific software vendors, which makes this step unavoidable before designing alignment of communication interfaces to each other. This problem is covered by the so called schema matching approaches in current research and literature. In this thesis I introduce a newly developed method to find semantically related pairs of data structures. After presenting the detailed working of my approach, its efficiency is also evaluated against other existing approaches based both on theoretical calculations and experiments on a wide range of testing examples. Corresponding results are described in Chapter I.1 and summarized in Thesis I.1.

Although there are already well known methods and metrics in literature how to evaluate and compare schema matching solutions, most of them do not take into account the strong dependency of specific methods on given input schemas and on the given set of configuration parameters. Hence they probably fail to result in a correct analysis and comparison of schema matching approaches. Therefore I also examine the influence of those parameters and perform the evaluation and comparison in a correct way. My schema matching approach proved to be the same or even more accurate than other approaches. Detailed description can be found in Chapter I.2 while Thesis I.2 summarizes the results.

Besides the accuracy of schema matching approaches it is also important to know about their computational requirements. Because the estimation of computational complexity and requirements of schema matching algorithms is hardly discussed in current literature, I have also created a method based on mathematical techniques to perform that. After the correctness of the method was proved by specific implementations and run time experiments on several test inputs, different schema matching approaches were analyzed also from that point of view. The computational requirement and hence the expected run time cost of my schema matching approach seems to be significantly lower than it is of other approaches. Details are described in Chapter I.3 and Thesis I.3 concludes the corresponding results.
Another area of my work is the designing and creation of an open standard based run-time solution. The approach presented in the second Chapter provides a methodology and architecture for applying encapsulated services to align non-compatible service interfaces to each other. Corresponding results are described in Chapter II.1 and summarized in Thesis II.1.

After describing the architecture in details its performance is evaluated against other possible approaches designed and created also by me. The comparison is carried out both on a theoretical (analytical) and an experimental way. My approach seems to be stable and outperforms other candidates taking into account practical feasibility aspects and the amount of required run-time costs as well. It facilitates enhanced support for business collaboration and service adaptation to a standard based run-time environment while prediction of run-time performance still remains to be appropriate. Detailed results can be found in Chapter II.2 and summarized in Thesis II.2.

As a closing section I present the main ideas and results of the research project Fusion which I have actively worked on for 3 years. The main differences in methods and applied standards are introduced in details while possible connection points are also identified. My contribution to the project is described in a short list of overview as well.
Acknowledgements

First of all I would like to thank my advisor Prof. Béla Szikora for his help and useful remarks. He had already believed in me getting this far at the beginning of my PhD studies. His advices and long-long discussions played an important role in formulating the Thesis themselves.

I am also thankful to Prof. Gábor Harsányi, who always aimed to ensure the optimal research and working environment for me. His encouragement has many times helped me get through stalemate situations both in my work and research.

I would like to express my gratitude to my friends for their valuable help. Especially to Bence Kővári, Balázs Villányi, Bálint Horváth, Balázs Jilling and Csaba Budai. Bence was always open for detailed discussions about the current issues with my work and did an invaluable contribution to correcting some basic mistakes in the implementation. I am thankful for the collaboration in several publications to Balázs Villányi. Several loosely related ideas and experiments were implemented in the Project Laboratory and Diploma Thesis work of Balázs Jilling, which gave me further inspiration and provided corresponding background knowledge. I am very grateful to Bálint for proposing a lot of corrections in my English. And finally I can hardly forget the funny night that we had with Csaba at the lab performing neverending experiments till the sun-rise.

I would also like to say thanks for the support to every colleague at the Department of Electronics Technology.

Finally, and most naturally I am especially thankful to my family for having supported me since my childhood. They have always been there for me.
Table of Contents

1. INTRODUCTION...........................................................................................................1
  1.1 Preliminaries ........................................................................................................3
  1.2 Thesis overview ....................................................................................................4
  1.3 List of corresponding publications ....................................................................6

PART I
DETECTING SEMANTICALLY RELATED CONCEPTS IN A SOA INTEGRATION SCENARIO ..........................................................7
ABSTRACT ....................................................................................................................7
INTRODUCTION .............................................................................................................7
RELATED WORK .............................................................................................................8
  Information and information system integration .......................................................8
  Enterprise application integration .............................................................................9
  Schema matching ......................................................................................................9
  Schema matching approaches .................................................................................10
  Process composition and quality of services ...........................................................12
I.1 DETECTING SIMILARITIES OF ENTITIES IN DATA SCHEMAS .................13
  1.1 Definitions and background ..............................................................................14
  1.2 Characterizing of services .................................................................................15
  1.3 Relationship of complex types .........................................................................16
  1.4 Working on a short example ............................................................................19
  1.5 Algorithm customizing .....................................................................................22
I.2 EVALUATION OF APPROACHES ACCURACY .............................................25
  2.1 Schema matching approaches from the literature .............................................25
    2.1.1 Approach Similarity flooding ....................................................................25
    2.1.2 Approach WordNet ..................................................................................25
  2.2 Comparing algorithm accuracy .........................................................................26
    2.2.1 Introducing accuracy indicators .................................................................27
    2.2.2 Efficiency with pre-defined thresholds .........................................................30
    2.2.3 Efficiency with changing (optimized) thresholds .........................................31
  2.3 The human based experiment .........................................................................33
  2.4 Evaluation on large schemas ...........................................................................36
I.3 COMPUTATIONAL COMPLEXITY OF APPROACHES .....................................37
  3.1 Complexity of approach NTA ...........................................................................37
  3.2 Complexity of approach Similarity Flooding ....................................................38
  3.3 Complexity of approach WordNet ....................................................................40
  3.4 Comparison of the complexity of approaches ..................................................42
1. Introduction

Organizations today are hard to imagine without complex information systems. Software applications are applied for managing customer orders, timing the procurement and production, supervising shipping and billing, performing the financial booking, maintaining employee data etc. Adapting to the always changing needs of customers, newer products and services enforce the creation of new business processes and services. Unfortunately these do not conform to the boundaries of the different applications at the organization. Requested information and capabilities can come from many different information systems of the company or even from other organizations. This implicates the need for integrating our systems and realizing business processes over multiple enterprise applications. Based on the boundaries of integration we can talk about intra- and inter-organizational enterprise application integration.

Enterprise information system interoperability and integration has a history of almost 3 decades. Based on many factors e.g. level, type, scope and dominant architecture this is divided into three separable generations [40].

1. Generation
By the 1980s, corporations already had large amounts of data in different departments. The exchanging and sharing of the data between departments and within enterprises became a reasonable aim of developments. Much of the emphasis during this generation was put on achieving system interoperability, in particular by addressing the heterogeneity due to differences in DBMSs. Correspondingly, the emphasis was on data management and data (as opposed to information or knowledge for example).

The main contributions to the area in this generation were the understanding of the data models for structured databases, and the techniques to deal with data modeled with different representations. Furthermore, techniques for schema integration and understanding of and finding ways to deal with schematic or representational heterogeneity were the first remarkable results in enterprise application interoperability. The realization of the fact that schematic/representational issues and semantic issues are handled distinct by different sources (e.g. DBMSs) is one of the main findings of that period.

2. Generation
During the second generation two very important trends brought opportunities for interoperability and exploitation of data: firstly, the proliferation of a variety of data—from structured database, and semi-structured data, to digital media, and secondly the spread of the Internet and emergence of the Web. Applications such as digital libraries and electronic commerce provided the context of interoperability.

Some of the key trends and achievements of this generation were the technology for dealing with heterogeneity of systems, data, and representational levels and the support for a broader variety of data—not just structured databases, but also text, semi-structured, and unstructured (including image and video) data.

It is important to mention the role of metadata (data on data) and emerging methods and techniques based on it. To capture the semantic content (i.e., at a level of abstraction closer to that of humans), it is important to have some metadata modeling application and domain-specific information. Metadata enable the abstraction of representational details such as the format and organization of data, and capture the information content of the underlying data.
independent of representational details. Metadata enables representation of domain knowledge describing the information domain to which the underlying data belongs to. This knowledge may then be used to make inferences about the underlying data.

The increasing standardization or adoption of ad hoc standards during this generation resulted in significant progress towards achieving syntactic and structural interoperability of systems. Acceptance of the Internet as a standard for interconnections between the systems, and evolution of infrastructures and middleware that support distributed computing (RMI, CORBA, and DCOM), and database connectivity (e.g., ODBC, JDBC etc. for relational databases), have had significant positive impact on achieving system interoperability. Syntactic interoperability includes the ability to deal with formatting and data exchange as supported by standards such as HTML for most current Web-accessible documents, and MPEG-1 for pixel-level representation of image data. At the structural level, standardization for data modeling such as ANSI SQL, and object modeling standards and methodologies such as ODMG and UML have helped.

3. Generation
In the third generation the progress in global interconnectivity implicated the need to deal with more heterogeneous information consisting not only of a broader variety of digital data, but also operations and computations (such as simulations) that can create new data and information. The scale of the problem has changed from a few databases to millions of information resources. Increasing standardization is expected and interoperability at system, syntactic, and structural levels to address many issues. The key challenges are at the semantic level, where people would increasingly expect the information systems to help them not at the data level, but at the information, and at the knowledge levels. Increasing standardization at different levels of information systems architecture for corresponding type of interoperability played an important role so far. For example the XML means a de facto syntax for all forms of Web-accessible data, the RDF is for general purpose description of information sources and various object models for Web-based information exchange, the KIF for knowledge representation, etc.

The utopia of semantic interoperability stands on three legs:
- **Terminology (and language) transparency**, which allows a user to choose ontology of his or her choice, while allowing the information source to subscribe to a related but different ontology,
- **The Context-sensitive information processing** which means, that the information system recognize or understand the context of an information need and use it to limit information overload, both by formulating more precise queries used for searching information sources and by filtering and transforming the information before presenting it to the user, and
- **Semantic correlation**, which allows the representation of semantically-related information regardless of distribution and by the user or the third party.

The following main enablers are identified: The ontologies for terminology transparency, the economy of reasoning, managing inconsistent information and flexible semantics for the context. Information correlation may be realized by specific extension of standards (for example XML or RDF) and the application of an information-brokering architecture is also definitely a step forward.

However there is much work to do till the utopia of the latest generation fully realizes. With this Thesis I would like to cover a smaller set of the current issues of interoperability and I hope that some of my results may help the evolution at this field.
1.1 Preliminaries

The Service Oriented Architecture has appeared 4-5 years ago. It envisions an architecture where provided capability is offered by service providers and applied by service consumers. The services can be characterized by their short descriptions which are stored in service registries. Using the registry available services are easy to browse, search and find [48]. For example one can have a service offering booking at hotels. The characterizing description of this service should contain information about its capabilities e.g. the service is able to find an available hotel at a given place for a given time interval, and to perform or cancel booking. The complex capability of this service is divided into operations like check availability, perform booking and cancelling. The requested input and output data of each operation is also required in the description. Furthermore some technical details like the definition of service endpoints are needed in order to invoke the service. Having this description one is able to adapt the service and communicate with it properly.

SOA provides not only the possibility of simple one-to-one, request-answer communication. Whole business processes can be defined upon services. A combination that integrates the invocation of two or more services into a complex process is called orchestration [65]. An orchestration makes it possible to create new, more complex services. For example, if there is a service capable of booking flights, combining it with the previous hotel booking service a new complex service can be created. This will be already able to organize the whole trip – both journey and accommodation. The complex orchestration process should describe the right order of operation invokes while paying attention to given constraints for example the flight dates and the hotel booking dates should be the same, or if there is no available flight for a given time period the hotel cannot be booked (must be cancelled).

The vendors of enterprise application systems have created services to prepare the applications for participating in service oriented integration scenarios. The participating application system can be a service provider, a service requester or may be both in each integration scenario. In addition, the integration environment may also contain a service registry for storing the services available in different systems. This scenario creates new capabilities on existing services of applications.

The widespread of the Service Oriented Architecture seems to be one of the most promising approaches in EAI in the last few years. In a SOA based enterprise application integration scenario the applications provide their capabilities in the form of services acting as an interface to each other. Business processes are built upon them. Although the SOA based approach is already supported by solid technology and tools, the preparations for the integration process – creating service endpoints to legacy systems, aligning them to applied communication and entity standards - are still an open issue. Deploying executable processes, predicting and optimizing their non-functional performance is also often in the focus of current researches.

In the SOA architecture, there is no need to share the whole databases of each company. Background data and working logic are hidden from service requesters. The so called SOA private layer secures companies’ (secret) data, but makes possible to create collaborative business processes by extracting data from the enterprise application systems [26].

However different systems have different conventions for naming, grouping and applying their concepts. Because the offered services also rely on them, the data schema of each service reflects the conventions used by the application system designers, programmers and database experts at a given application vendor. However by the modeling of real world concepts in the databases of application systems, a lot of additional information is lost in the abstraction process by the software development. It merely remains in the minds of the aforementioned IT-experts at the application system vendor. This makes the identification of semantically related concepts in integration scenarios a hard task [70]. Nevertheless, building a business
process requires the common understanding of the data application concepts found in input and output schemas of the underlying services. Furthermore the processing of input and output data is needed at every service invoke and response in the business process. Thus the identification of semantically related concepts should be completed before undertaking the process.

This thesis provides an algorithm and methodology to find semantically related pairs of entities of different data structures in Part I. Upon the results of the presented schema matching method, interfaces of services can be aligned to a global taxonomy of concepts making possible the orchestration built on encapsulated services at the run-time architecture presented in Part II.

1.2 Thesis overview

This chapter briefly introduces the results of my research. There are two parts, containing 3 and 2 Thesis summarizing the outcome of the PhD Thesis. The corresponding articles are referred at the end of every Thesis, see list of publications in Chapter 1.3 below.

Part I. Schema matching approaches, analyzing the efficiency and complexity of schema matching approaches

I.1. I have developed a universal methodology and algorithm for comparing entities of different data schemas

I have developed a method and algorithm for determining the strength of semantic correspondences of different entities. Hence data transformations are built upon these entities, performing of this task is a certain precondition of creating transformations. To support the alignment of local and global schema entities a function returns the desired similarity values in the form of a weighted sum of three sub-functions. The sub-functions are analyzing the names of entities, the similarity of the set of connected terms and compare the set of connected attributes. This approach is called shortly NTA (Name, connected Terms, Attributes). The approach NTA is applicable in various enterprise application integration scenarios to perform the comparison of service interfaces and provides valid result under custom circumstances.

See details in Chapter I.1 of the PhD Thesis. Corresponding results were published in [L2] and [K3].

I.2. I have deeply examined the most popular schema matching approaches. I have showed that their accuracy strongly depends on their configuration settings and applied input data. Therefore a fair method was created to analyze and compare their performance.

I have showed that schema matching algorithms are sensible for the given input and the working parameter settings in their accuracy. Thus a novel method is needed to compare them in a fair way.

To measure the efficiency the widely used and accepted measurement methods and metrics of current literature are applied. The measurement units are the accuracy, the recall and the F-measure\(^1\). The main idea of the method is that the partial optimum point of parameters must be determined first and this must be applied by calculating the final value of the efficiency. Finally, the evaluation of the accuracy can be performed by comparing these calculated values.

\(^1\) Universal measurement unit of accuracy, calculated as the harmonic mean of accuracy and recall
Using the novel method I have also showed that the provided results of my approach NTA – presented in I.1 – are probably more accurate than other current schema matching approaches. The effectiveness and applicability of the approach NTA was also proved on the results of an experiment performed by the participation of 23 test subjects familiar with information sciences and enterprise computing in a cost model developed also by me.

See details in Chapter I.2 of the PhD. Thesis.
Corresponding results were published in [K1] and [K3].

I.3. I have created an analytic method to compare computational requirements of schema matching algorithms

I have created an analytic method to compare the complexity – i.e. the computational requirements – of schema matching algorithms. The complexity is estimated by the number of required computational steps. Calculating the number of computational steps it is possible to determine and compare the required computational costs of different approaches. The model was validated by sample implementations and run-time measurements run on various test data.

Using this method I have also proved, that approach NTA requires much lower computational costs to calculate results than other approaches found in current researches.

The calculated number of expected computational steps was verified by implementing more test schemas and the order of difference in the number of steps proved to be the same as the order of difference in the actual run-times measured in run-time experiments.

See details in Chapter I.3 of the PhD Thesis.
Corresponding results were published in [K2].

Part II. Service oriented architectures, analyzing the performance of given approaches based on service oriented architecture

II.1. I have developed a SOA based solution and framework for handling data incompatibilities during enterprise application integration. My proposal is built on open standards and corresponding tools exclusively

I have developed a methodology and framework for handling mismatches in data representation by applying transformations during composite process creation. It makes possible to adopt existing services of standard enterprise applications into SOA based integration scenarios. The proposed solution is modular, developed transformations and service interfaces are recyclable while interfaces of standards enterprise application services can remain intact. It uses open standards and corresponding free tools exclusively.

The practical applicability of the solution was proved by constructing a working framework and solving various test problems with it.

See details in Chapter II.1 of the PhD Thesis.
Corresponding results were published in [L3], [K4], [K5], [K6], [K7], [K8] and [K9].

II.2. I have developed a novel methodology for analyzing and compare the run-time performance of SOA based application integration solutions

I have developed an analytical and practical methodology for analyzing the run-time performance of SOA based application integration solutions also containing transformations to bridge incompatibilities in data representations. The novelty of my approach is the separated handling of resource requirements needed by the execution of transformations, by instantiating the processes and by performing the standard enterprise services. Furthermore I proposed also to separate the resource requirements between the enterprise application server
and the process run-time server. The reference throughput and response time of given configurations must be determined first. Hence the expected response time and expected number of served request during a given time period is already calculable.

The method was validated by various tests results and test scenarios in practical experiments.

See details in Chapter II.2 of the PhD Thesis.

Corresponding results were published in [L1], [K4] and [K5].

1.3 List of corresponding publications

The list of corresponding publications can be found below. The numbering of references is aligned to the numbering found in the short version of the PhD. Thesis.

Papers published journals in Hungary in English:


Papers published international journals in English:


Papers published in proceedings of international conferences in English:


PART I.

Detecting semantically related concepts in a SOA integration scenario

Abstract

In this part I present an approach to detect semantically related concepts in a service oriented environment. This method is essential when creating collaborative business processes. Standard enterprise application systems such as enterprise resource planning (ERP), customer relationship management (CRM), and supply chain management (SCM) etc. systems are certain participants of the integration and must be prepared for direct collaboration.

As a well defined first step system integrators assign a set of services from various application systems into the integration scenario during a task called service discovery. Nevertheless, building an operable business process out of these services also requires the aligning of them to the data schema used in the business process. This mapping must result in a global understanding of relevant business concepts in the integration scenario making possible the direct communication between services coming from different applications.

In this chapter I focus on the identification of semantically relevant concepts in different schemas in the participating services. The presented approach is deeply analyzed both from the point of view of accuracy and run-time performance. It is also compared with other approaches from current literature.

Introduction

The realization of enterprise application integration in a SOA-based environment is performed by defining and running composite (e.g. cross-application) processes. These processes consist of a set of services. The connection between these services is defined by dependencies, which determine for instance the right order by invoking these services. For example, in a process of purchasing a flight ticket, the flight reservation service should be invoked earlier than the service responsible for the on-line payment.

Business analysts are able to define composite process templates for specific business processes. These consist of virtual services and their dependencies. Virtual services are replaced by real service endpoint references later during the implementation phase of the SOA integration. (This is often called as binding.) However connected enterprise application services cannot yet participate in run-time scenarios, because determining the dependencies and constraints among them is not enough to get the process run. As explained earlier, the data representation of same real world concepts may differ in the applications developed by different vendors. So services applied in composed processes rely on their own schema in their input and output data. Hence, if we apply services from different software vendors we should create mappings to bridge existing semantic gaps among service interfaces within composite processes.

Knowing the semantic relationships between the concepts of participating services is a certain precondition of mapping. It is trivial, that only concepts representing corresponding
real world concepts can be mapped to each other. For example, if a service requires the name of a customer as an input, the address of the customer provided by another service cannot be connected to as customer name input.

The detection of the semantic relationships between the concepts coming from data-schemas of different applications is in the focus of this chapter. This is often called as schema matching in the literature.

The rest of this chapter is organized as follows: the next section summarizes other related researches of this area. Chapter I.1 provides the detailed description of my methodology and algorithm for detecting semantically related concepts of services. The evaluation of the efficiency of different approaches is presented in Chapter I.2, while Chapter I.3 focuses on required computational resources and run-time costs of different schema matching approaches. This chapter is closed by a chapter summarizing the main results and outlining some possible areas of future work.

Related work

Integration of different software systems and corresponding data structures is a certain need in information technology. The field of its applications is various starting from medical systems to enterprise application integration through E-commerce and E-government. Because system integration and interoperability is hardly separable from the mapping of different data structures, the schema matching problem i.e. comparing and aligning of entities of schemas to each other is mostly concerned in these works as well.

Information and information system integration

In [17] the authors are facing with a schema matching and integration problem in the field of radiology within medical enterprise systems. The aim of the work was to integrate the widely adopted standard DICOM (Digital Imaging and Communications in Medicine) with another RIS originated (Radiology Information Systems) CDA (Clinical Document Architecture) representation. As a solution, uniform DICOM to CDA translation scripts and business processes in a SOA architecture were specified.

Another popular field of application is E-government. For example in [4] an Ontology is built and used for managing distributed educational directorates in the prefecture of Achaia in Greece. Besides the implemented prototype for the given use-case scenario a general two-level semantic mediator model is also presented. This is responsible to both provide an integrated description of entities and map actual information to them.

In [47] we meet an application from the field E-commerce. To solve interoperation and communication problems a semantic specification and evaluation of bids in Web-based markets is presented. A specified Core Ontology of bids uses utility function policies for the compact representation of bids for configurable goods and services. Furthermore an implemented prototype makes possible dynamic selection of applied Web services (see process orchestration and services’ matchmaking later).

The authors of [44] introduce a solution based on semantic Web services and corresponding technologies in a flexible approach called ODSOI (Ontology-Driven Service-Oriented Integration) in microelectronics. Flexibility, data and function integration in EAI, services and semantic structuring, openness for widely adopted standards etc. are all aimed by the presented architecture, which was built over an enterprise ontology.

Digital Rights Management is in the focus of the work presented in [79]. XMLs and XML schemas are mapped to RDFs by using a special XPath processor also supporting
RDFS/OWL ontologies. The actual mapping is performed in the RDF representation by a model-mapping approach retaining the node order (see at structural matching approaches later).

The authors of [39] are dealing with the interesting problem of organizing and sharing distributed personal data from the Web. A common management interface is proposed for all kinds of Web accounts to achieve similarly high level of usability as we had in the case of desktop applications. Easy organizing, searching, manipulating, protection etc. of personal data are provided by the presented prototype framework called Menagerie.

Enterprise application integration

One of the main areas of application is enterprise computing and enterprise systems. There are numerous works dealing with integration and schema matching problem coming from this field.

For example in [6] the authors propose an Open Solution Business Process Integration and Management Implementation Framework, which is based on open standards and technology. The main idea behind is to have one wide core schema (similarly to the global schema conceptualization in my work, see later) in a SOA architecture and the corresponding standards are SOAP for communication, WS-BPEL for orchestration and WfMC XPDL for the choreography.

A classical part of the ERP systems, the manufacturing is in the focus of [89]. According to the demands of enterprise application integration (EAI) in manufacturing enterprises, an enterprise application integration oriented information classification code system (EAIO-ICCS) is presented in the paper.

In [43] we find a more theoretical part of the integration problem namely the knowledge representation and integration at enterprises. The construction, evaluation of an enterprise Ontology and workflow implementation is presented in details solving the tasks of describing domain-related knowledge, accumulating it in processes and describing the rules and constraints in real applications.

As we could see, integration of different information systems and data sources is a really live topic in current researches. However this area is rather huge to handle it as a monolithic unit. In the followings I will focus on some parts which are more relevant to my work.

Schema matching

One more or less separable task of the interoperability is the schema matching. The schema matching is often followed by the schema integration task. While schema matching only detects the differences and relationships between entities of whole schema structures, schema integration aims the creation of a common structure including concepts from all parties. This may be a reasonable aim in scenarios where a global conceptualization i.e. a taxonomy is required to realize the communication among numerous participant systems on a business bus.

For example a tree mining algorithm is presented in [69] to examine schemas and integrate them into a global one. The desired performance of the created method is a key aspect which is analyzed through experiments run on large schemas like OAGIS[63] and XCBL[91]. On one hand this algorithm seems to be really solid; on the other hand its efficiency may be queried hence it uses no enhanced evaluation of complex types with attributes (see at schema matching algorithms later).

In [77] the authors present a method to create so called schema dimensions over multiple schema graphs so that to be able to answer complex logical questions over an integrated structure. The required format of the data sources is XML. As the restriction of the possible
input to XML schema so is the graph representation of data structures a frequent approach in
the field of schema matching and integration.

Another work dealing with XML schemas and exploiting tree structured representation is
[31]. The authors have built a state-based synchronizer which recognizes schema conflicts
driven by the structure of the schemas. A case study on the database of simple address books
presents the working of the simple one-pass, recursive tree-walking algorithm.

Because my work is strongly connected to the SOA methodology and standards I also work
on data represented in the format of XMLs conforming XML schemas. Nevertheless
supporting XMLs is not a strict restriction– all kinds of enterprise software have XML based
interfaces today. Furthermore backend relational databases can also be managed by tools and
methods of the XML world with the help of the techniques introduced in numerous current
researches like [22] and [61].

The focus of my work presented in Part I. is set on the schema matching. I disregard from the
solution of the schema integration problem (i.e. the creation of a common schema) because
various existing enterprise schemas can be used as global schemas. For example see [91] and
[63]. Moreover the actual creation of mapping information i.e. creation of data
transformations is also not aimed. Instead I restrict my research on the identification of
semantically relevant concepts in different schemas. In other worlds I intend to find the pairs
of data entities which probably must be connected by data transformations. This problem is
often called also as similarity measure in the literature.

There are 3 basic types of schema matching approaches today:

- Linguistic approaches examine the naming similarity of concepts using different string
  comparing functions for example searching for sub-strings or concatenations. Usually
  they are also extended by (domain specific) dictionaries and taxonomies to be able to
detect the similarities in the meaning of schema concepts as well.

- Structural methods are based on the comparison of the paths connected to given
  concepts and leading to the leaves, children or to the root element. The main idea
  behind this approach is, that two entities of two different schemas probably represent
  the same real world entity if their structural neighborhood is built similarly e.g. the
  two paths leading to the root element are similar. Path similarities are mainly
  measured by defining indicators for similar node correspondence, node order, etc.

- Combining the two approaches above and applying more specific algorithms within
  solid frameworks results in a solution called combined approach. Because of its
  robustness and effectiveness, most of the presented solutions are from this category in
  current literature. However algorithm complexity and computational costs are hardly
  taken into account (see Chapter 3 in Part I. in my Thesis later).

Schema matching approaches
An automated schema matching solution working on XML schemas is presented in [66]. The
described method combines linguistic and structural similarity extended with the evaluation
of data type compatibilities. Within the linguistic part abbreviations and acronyms are also
identified while prepositions and articles are disregarded due to domain-specific dictionaries
constructed for the examined schemas.

Similarly, the authors in [18] also present a combined approach. The evaluation starts with a
linguistic analysis based on the open dictionary called WordNet[25], but the main added value
is the comprehensive structural method performed on the schema trees. The structural
similarities are examined in three contexts: the children, leaves and root environment are
compared to all possible pairs of input schemas. An implementation of this approach is also judged in details against my method both from the point of view of accuracy and required computational cost. See later in Chapters I.2 and I.3.

The presented approach called Cupid in [49] also uses a wider set of techniques including discovering mappings based on their names, data types, constraints and schema structure. The authors constitute that most of the useful information can be found in the leaves of schemas. Thus the similarity of the leaf context is highly weighted in the calculations. The number of one to one comparisons is decreased by a separately clustering of concepts into categories at an early stage of the method. Otherwise the different implementations e.g. the structural similarity evaluation of this approach are quite similar to the previous one preventing me from the further description and comparison of them with my approach. Furthermore I can also judge my approach against two other proposals (DIKE[75] and MOMIS[14]) indirectly, because Cupid definitely outperforms both of them.

A generic schema matching tool called COMA++ is presented in [27]. It provides a library of individual matchers realizing a flexible platform for a combined matcher. The application of different matching strategies and the decomposition of large schemas with a fragment-based matcher[67] into smaller sets ensure high scalability for the presented schema matching solution. The approach is also evaluated on schemas from various sizes (containing a number of nodes between 27 and 843 and a number of possible paths between 34 and 26228) and is compared with other proposals extensively. Compared to my solution this framework is much more complicated. Thus it is harder to install, customize and maintain it while my method is also able to deal with schemas at the same size.

In [60] the authors present a schema matching method working on XML schemas. Similarly to the approach Cupid the evaluation starts with the clustering of schemas into various groups. The syntactic similarity measurement is performed in 3 steps namely preprocessing, data mining and postprocessing while a specific graph representation called dendogram facilitates the generalization and specialization processes of the clusters to develop an appropriate schema class hierarchy. Unfortunately the analysis of the results is restricted to the parameterization of the presented approach and is only presented in the unique metrics of the paper. This hinders the comparison with other approaches including mine. However taking into account the size of evaluated schemas and the values of applied efficiency indicators the performance should be at the same level as the methods in [18] and [49].

The advantages of online and offline schema matching are combined in the approach presented in [46]. The offline (semi supervised) matching is performed similarly to the linguistic methods presented in the approaches above but is only executed on a filtered sample. After finding this reliable dataset, a classifier (e.g. NNPLS classifier) can be trained upon it and is used for on-line analysis. The focus of the paper is set on the selection method of the reliable training data set. Although this approach may lead to much more accurate results, the supervision of a human expert is also required during the execution. Therefore selection of training data sample and application of on-line methods is out of the scope of my work.

The authors of [16] propose an algorithm based on tree similarity matching methods in a multi agent environment for a classical buyer-seller scenario. The similarity algorithm itself is implemented as a recursive functional program in a language called Relfun. Although this may save some computational costs in the run-time, an additional transformation step of classical XML based inputs i.e. the XML serialization in Object-Oriented RuleML is needed. Because there are no experiments on schemas of realistic sizes presented, the exact comparison of this approach with others would require enormous amount of work. Taking into account the applied structural algorithm and comparing it with other approaches the
performance of this approach can be similar (or maybe slightly better in run-time) to the methods in [18] and [49].

A special approach is presented in the work [57], which is hard to be identified upon the classification of schema matching approaches defined above. The main idea behind the similarity flooding is that the similarity of two given entities can be measured by the similarity of their neighborhoods. After transforming input data into an OIM format [15], an integrated graph called pairwise connectivity graph (PCG) is created. After that, the similarity values are flooded among the connected nodes during some iterative steps. The final values of similarities are contained by the nodes of the PCG hence its nodes were created by merging two nodes of the two input schemas. This algorithm is described in details and compared with my approach later in Chapters 2 and 3 in Part I.

The work in [19] compares some existing document similarity measurement methods from before 2004. This is the only paper of the presented set above, which also concerns some aspects of the algorithm complexity comparison besides the evaluation of algorithm accuracy. (An extensive complexity and computational requirements evaluation method and comparison is presented later in Chapter 3 in Part I.) Unfortunately the described method for comparing algorithm complexity is not applicable for other kind of algorithms for example structural or similarity flooding. The presented approach based on path singles outperforms other approaches significantly. However it is probably not competitive any more with the approaches existing today.

My proposed schema matching approach is of the combined type. It directly observes the linguistic similarities in the naming of entities in input schemas, but it also takes into account the similarities of additional information sets called terms. The evaluation of the attribute similarities and data types are performed recursively through the schema graphs realizing a special estimation of structural similarities. I prove, that my approach is approximately at the same accuracy level as the very bests of the above mentioned methods. On the other hand it requires much lower computational costs to solve the schema matching problem (see Chapters 2 and 3 in Part I).

Saving some cost of human work, in other words preventing the necessity of the presence of a human expert during the schema matching task is also a reasonable aim by developing different approaches and methodologies. There are several analytical models to define the amount of possibly saved human working effort, for example see [57] and [68]. I also present a simple model for constituting the applicability of the presented approaches for given test schemas in my Thesis. In other words I prove, that a significant amount of working cost of an integration expert can be saved by applying an automated schema matching approach instead of evaluating the schemas manually.

**Process composition and quality of services**

After the identification of relevant pairs of entities of schemas the actual mapping – i.e. the creation of data transformations – can be performed. And finally we can have the set of aligned services (interfaces) in our hand to start the actual process composition. Nevertheless, finding proper services to given positions of complex processes and maximizing non-functional run-time quality is another interesting area of current researches.

For example in [42] the authors aim to improve the overall quality by using service discovery method based on FQoS (Functional Quality of Service). The presented approach is strongly based on mathematical methods and increases the effectiveness of service discovery in process composition.
Automated checking of service compatibilities and service matchmaking is also often aimed by current researches. For example Fuzzy logic is used on OWL-S input standards for enhanced Web service matchmaking in [30].

Another approach for the detection of service consistency is presented in [26]. The method focuses on the prevention of deadlock situations in communication and support the realization of successful collaboration in B2B integration scenarios.

However, process composition i.e. the creation of complex Web services solving integration problems in a SOA scenario including service matchmaking and optimizing of non-functional service quality by dynamic service selection is already out of the scope of my work. Instead, I focus on analyzing and optimizing run-time performance in solutions containing transformations glued to standard enterprise services in SOA integration scenarios (see Part II. of my work).

**I.1 Detecting similarities of entities in data schemas**

The integration in a SOA based environment requires the bridging of possible data incompatibilities (i.e. differences in data representation). As already described before, this is actually performed by mappings between service interfaces. However creation of the mapping of every possible service input to every service output in an integration scenario may require enormous computational capacity. Furthermore, this point to point connection structure is also hard to maintain: if a new service is added to the integration scenario, the mapping to every other service has to be defined [55].

Defining a global schema can reduce the complexity of the system. The global schema covers all possible real world concepts in the integration scenario. The services are mapped only to the global schema concepts, and the communication between services in processes is done on the level of global schema concepts. Thus the mapping of (virtual) services on the process level is no longer a complex issue because corresponding real world concepts are represented by the same concepts in the global schema.

Mapping between services’ input and output concepts to the global schema is in the focus of this part, especially the important step of the identification of semantic relationships between the concepts of services and the concepts of the global schema.

The input and output of a service is typically a set of complex types. It contains attributes to describe different properties. Some attributes are simple elements, while others connect complex types to the ranges of attributes. For example, a `Person` complex type may have some simple attributes such as `name`, `age` or `sex` but it can have an attribute with a complex type `Address` which connects an `Address` complex type to the `Person` this way. The `Address` complex type may also have further simple and complex attributes so the structure of complex types can be described by a directed graph where nodes are the complex types and edges represent the connection of attributes with complex type in their range to their parent.

In my approach the global schema is structured as a directed graph too. Furthermore, let us assume that the graph of the global schema and the graphs of the services are also acyclic.

By mapping the service’s input and output to the global schema, we should be able to determine the semantic relationship between complex types of the service and complex types of the global schema. Semantic relationships can be divided into four categories [23]:

1. Equivalence: two complex types are equivalent if they represent the same real world concepts.
2. Inclusion: complex type A includes complex type B but only if A represents all real world concepts which are represented by B.
3. Overlap: complex type A and complex type B are overlapping if complex type A represents some real world concepts of B, but A does not include B, and B does not include A.
4. Disjoint: complex type A and complex type B are said to be disjoint if complex type A does not represent any real world concept represented by complex type B.

Complex types of the global schema that are disjointed from other complex types of service interfaces cannot be assigned to each other during the mapping. Complex types in relationships described in 1-3 above may be contained by the mapping rules between the global schema and the service.

In the next chapter I introduce an automatic method to detect semantic relationships between the complex types.

In this chapter I present a novel method to detect semantically related entities between two schemas. Although I focus on aligning services interfaces of enterprise applications to the concepts of a global schema the presented algorithm is applicable for solving every kind of schema matching tasks i.e. the identification of related entities between two schemas.

1.1 Definitions and background

Services provide a uniform interface for the integration. As already mentioned in the previous section, every service provides a set of complex types as an interface for the mapping. In my work these sets of complex types are handled as directed acyclic graphs (DAG). (Note that a schema containing a directed circle can be transformed to an equivalent one which does not contain any circles without harming the whole semantics of the data structure.) The global schema defined in this work is also described by a DAG.

To enable the services to participate in real processes they have to be mapped to the global schema first. This means, that the definition of data transformations must be presented. There are two transformations to each service:
1. The downcast transformation maps semantically relevant complex types of the global schema to the input of the service. The transformation allows us to invoke the service using concepts of our common reference (Ontology). At every service invocation, the input data described by ontological concepts are transformed into the input schema of the service.
2. The up cast transformation maps semantically relevant complex types of the global schema to the output of the service. The transformation allows us to provide the output of the service using concepts of our Ontology. At every service response, the output data described by the schema of the service output is transformed into the concepts of the Ontology.

The precondition of creating these transformations is the identification of semantically related complex types of the schemas. Transformations are well defined sets of rules among the elements of concepts. The elements of concepts are simple data types and complex data types. Based on the type of semantic relationship of complex types:
- transformation rules must be defined between the elements of the global schema complex type A and service’s complex type B if A and B are equivalent,
- transformation rules probably should also contain operands from service’s complex types C if global complex type A includes C, and
transformation rules may contain operands from service’s complex type D if global complex type A and D are overlapping.

Identifying the above mentioned types of semantic relationships between complex types, one possible set of operands can be offered for the creation of transformation. Because the exact type of the semantic relationship between complex types is not relevant from the point of view of the creation of mapping, my approach focuses only on the identification of somehow related complex types. So instead of determining the exact type of the semantic relationship I simply calculate a value for similarity called semantic distance (see later in section 2.3) between the complex types.

Please note, that from the point of view of the semantic relationship detection, it is unimportant whether the complex type of the service is from the input or from the output. Hence in the rest of my Thesis, I will simply regard to them as complex types.

Let \( S_1, S_2, S_3 \ldots S_n \) denote the services in the integration scenario, willing to participate in composed processes. My proposed methodology is as follows:

Identify the semantically related complex types pair wise between the services \( S_1, S_2 \ldots S_n \) and the global schema and create mappings between them.

This methodology ensures that corresponding services of the integration scenario are ready to participate in composite business processes. In the rest of this Part I focus on the identification problem of the above defined tasks.

### 1.2 Characterizing of services

Enterprise application services are typically described by Web Service Description Language (WSDL) files. This contains the input and output data schema in the form of XML schemas. (For the sake of simplicity I express my global schema also as an XML schema.) These schemas are not sufficient to capture real world semantics. Therefore, to compare the semantic relevance of complex types of these schemas, they are characterized by given identifiers. The characterization and possible extension by additional identifiers is processed on the level of complex types.

My characterizing identifiers are the following:

- The name of the complex type.
- The list of attached terms.
  - To describe the connection between a complex type and the modeled concepts of the real world, IT experts of enterprise application vendors may attach further terms to complex types. This can be applied as an extension of the name of the complex type. It can help service integrators to overcome semantic relevance detection problems like names in different languages or synonyms.
- The list of attributes. This is described by the name of the attribute and the data type connected to it. As already mentioned in the previous section, this data type can be a simple element (type) or another complex type.

Upon this, the complex types are modeled by a tuple: \(<N, T, A>\) where \(N\) is the name of the complex type, \(T\) is the list of associated terms and \(A\) is the list of attributes with their names with connected data types.

Now, consider a complex type represented in XML in figure 1.1.
In my model this is described as follows: `<person; human, man, woman, employee; surname(string), lastname(string), age(integer), idnumber(string), mothersname(string), placeofbirth(string)>`. It is obvious that complex types of data structures represented in XML schemas can be unambiguously transformed into my model. Thus building of my method on that given model of representation makes no limitation in the applicability of the approach.

In the next section I present the algorithm for detecting the semantic relationships between complex types described in this form.

### 1.3 Relationship of complex types

Let $C_1, C_2 \ldots C_n$ denote the complex types of the schema of the service and $G_1, G_2 \ldots G_m$ the complex types of the global schema where $n$ is the number of complex types contained by the service schema and $m$ is the number of complex types in the global schema. For each complex type $C_n$ we try to find a set of complex types $G_m$ which are semantically related to $C_n$. In other words the semantic distance between $C_n$ and the identified set of $G_m$ should be calculated. Consecutively, semantically related complex types of $G_m$ and $C_n$ must be connected with transformation. This means that the data transformations in the mapping will be defined between the concepts in $C_n$ and the concepts in the identified set of $G_m$.

Let $F$ be a function that returns a value of the semantic distance between two complex types. The semantic distance between complex types $C_i$ and $G_j$ is calculated as follows:

$$F(C_i, G_j) = w_1 \cdot N(C_i, G_j) + w_2 \cdot T(C_i, G_j) + w_3 \cdot A(C_i, G_j)$$  \hspace{1cm} \text{(Exp. 1.1)}$$

The functions $N$, $T$ and $A$ are derived from the characterizing identifiers introduced in the previous section. $N$, $T$ and $A$ are functions calculating the similarity between two classes from the point of view of a given identifier type. There is also a weight $w_i$ in $F$ before the functions $N$, $T$ and $A$, representing the degree of their contribution to the final result of $F$.

The proper selection of weights $w_i$ depends on a lot of circumstances (exact type of the application field, granularity of the schemas, complexity of the services, etc.). The specific value of the weights may be determined in every specific integration scenario regarding these. More about the influence of changing the values $w_1$, $w_2$ and $w_3$ and customizing aspects of my algorithm can be found later in Chapter 1.5.

The detailed calculation method of each function is presented as follows:
\( N(C_i, G_j) \) is a function comparing two complex types by their name. To determine the similarity of the names I use a syntactic method. The following definition for \( N(C_i, G_j) \) is used:

\[
N(C_i, G_j) = \begin{cases} 
1 & \text{if the name of } C_i \text{ is the same as the name of } G_j, \\
0.5 & \text{if the name of } C_i \text{ is a substring of the name of } G_j \text{ or vice versa,} \\
0 & \text{otherwise.} 
\end{cases} 
\]  

(Exp. 1.2)

Usually the name of a complex type has not much relevance to the represented real world concept. IT experts of different business application system vendors may use different names for the complex types representing the same real world concept, and different real world concepts may have the same (or similar) names in the complex types of services. In my experiments I set the value of \( w_i \) to 0.15.

\( T(C_i, G_j) \) is a function comparing the set of terms connected to the complex types \( C_i \) and \( G_j \). \( T(C_i, G_j) \) is computed as follows:

\[
T(C_i, G_j) = \frac{2 \cdot \| \text{Term}(C_i) \cap \text{Term}(G_j) \| + \| \text{Term}(C_i) \setminus \text{Term}(G_j) \|}{\| \text{Term}(C_i) \| + \| \text{Term}(G_j) \|} 
\]  

(Exp. 1.3)

where \( \| \text{Term}(C_i) \cap \text{Term}(G_j) \| \) is the number of common terms of \( C_i \) and \( G_j \), \( \| \text{Term}(C_i) \setminus \text{Term}(G_j) \| \) is the number of terms which are not the same but one is a substring of the other and \( \| \text{Term}(C_i) \| \) is the number of terms of \( C_i \).

The addition of terms to each \( C_i \) and \( G_j \) allows us to express a little bit more about the given complex type than a simple word (for example the name of the complex type) can do. The IT experts of application system vendors know the real world concept represented by the given complex type of a service so they can attach some effective synonyms or keywords to it. Creators of the global schema can do the same for every \( G_j \) so, if \( C_i \) represents related real world concepts to \( G_j \) they will probably have some terms that are the same. If there are no matching terms in \( C_i \) and \( G_j \) then \( C_i \) and \( G_j \) then the corresponding real worlds entities are probably also not related.

Although the degree of relevance of the function \( T \) is higher than the relevance of \( N \), it is not too relevant in the final result of the function \( F \). For example, granularity differences between the definitions of different data schemas influence the identification of terms: concepts representing the same real world concept but coming from schemas having different granularity may have no common terms or vice versa.

In my experiments I have set the value of \( w_2 \) to 0.35. Note, that the influence of selecting the weights properly is examined later in Chapter 1.5 detailed. Furthermore, optimal selection of weights will also be presented in a chapter introducing some possible customizations of my approach.
A(Cᵢ, Gⱼ) is a function comparing the attributes connected to the complex types Cᵢ and Gⱼ. A(Cᵢ, Gⱼ) is computed as follows:

\[
A(Cᵢ, Gⱼ) = \sum \frac{C(aᵢ, aⱼ)}{\|Attr(Cᵢ)\| \cdot \|Attr(Gⱼ)\|}
\]

(Exp 1.4)

where \(C(aᵢ, aⱼ)\) is a function returning the degree of correlation between attributes (\(aᵢ\) and \(aⱼ\)) of complex types \(Cᵢ\), \(Gⱼ\) and \(\|Attr(Cᵢ)\|\) is the number of attributes in \(Cᵢ\).

Let the function \(C(aᵢ, aⱼ)\) return the value 1 if attributes \(aᵢ\) and \(aⱼ\) probably correspond, 0.5 if they may correspond and 0 if they do not correspond. Function C is calculated as follows:

a.) If both \(aᵢ\) and \(aⱼ\) are from a simple (not complex) data type (for example attributes age or sex in the complex type Person) then:

\[C(aᵢ, aⱼ) = 1\text{ if the name of attribute } aᵢ \text{ is the same or is a substring of attribute } aⱼ, \text{ AND the connected data types of } aᵢ \text{ and } aⱼ \text{ are the same,}\]

\[= 0.5\text{ if the name of attribute } aᵢ \text{ is the same or is a substring of attribute } aⱼ, \text{ but the connected data types of } aᵢ \text{ and } aⱼ \text{ are not the same,}\]

\[= 0\text{ otherwise.}\]

b.) If the data type of \(aᵢ\) is a complex type and the data type of \(aⱼ\) is a simple type then:

\[C(aᵢ, aⱼ) = 0.5\text{ if } N(aᵢ, aⱼ) > 0 \text{ or the name of } aⱼ \text{ is the same or is a substring of one or more terms connected to } aᵢ. \]

\[= 0\text{ otherwise.}\]

(Exp. 1.5a)

(Exp. 1.5b)

c.) If the data type of \(aⱼ\) is a complex type and the data type of \(aᵢ\) is a simple type then:

\[C(aᵢ, aⱼ) = 0.5\text{ if } N(aᵢ, aⱼ) > 0 \text{ or the name of } aᵢ \text{ is a the same or is a substring of one or more terms connected to } aⱼ. \]

\[= 0\text{ otherwise.}\]

(Exp. 1.5c)

d.) If the data types of \(aᵢ\) and \(aⱼ\) are both complex types then:

\[C(aᵢ, aⱼ) = F(dᵢ, dⱼ)\text{ if } F(dᵢ, dⱼ)\text{ is already calculated, where } dᵢ\text{ is the complex type connected to data type } aᵢ, \text{ and } dⱼ\text{ is the complex type connected to data type } aⱼ.\]

\[= w₄ * N(dᵢ, dⱼ) + w₅ * T(dᵢ, dⱼ)\text{ otherwise, where } w₄ = 2 * w₂ \text{ and } w₅ = 2 * w₁\]

otherwise. (Please note, that depending on the given implementation of the algorithm it is possible to ensure a processing order of entities preventing the necessity of the application of the second formula. Based on my experiences this probably leads to a better result. Therefore the last version of the reference implementation is created this way).

The relevance of function A is significant within function F. The attributes of a complex type characterize it much more than its name or the connected terms. So that if there are 2 concepts, both having the same set of attributes, then they probably represent the same real
world concepts. Complex types with big differences in their attributes probably represent different real world concepts. I have set the value of $w_3$ to 0.5 in my experiment.

As the reader might recognize, functions $N$, $T$ and $A$ return values between 0 and 1. Because the sum of the weight coefficients $w_1$, $w_2$ and $w_3$ is 1, the return value of the semantic distance function $F$ is between 0 and 1.

To detect the semantic relations between the complex types of services and the complex types of the global schema, the above described functions should be calculated recursively in an integration scenario. After fixing a threshold value between 0 and 1, we can also diagnose complex types as related if they have returned an $F$ value greater than the threshold. On the other hand complex types with values under the threshold are probably semantically unrelated. Please note, that similarly to the weights $w_i$, the proper value for the threshold may depend on circumstances of the specific integration scenario. See details later in Chapter I.2.

The outcome of my method is the following: only semantically related complex types should be considered at the transformation of data when creating mappings between services and the global schema concepts.

In the next chapter I present the application of my method on a short example.

### 1.4 Working on a short example

To demonstrate the working of the methodology presented above, the detailed solution of a demo example is revealed in this section. Table 1.1 shows the schema of a service participating in the demo scenario. The names of the contained complex types are shown in column 1. The list of attributes in each complex type with connected data types can be found in column 2. The list of associated terms is placed in column 3, in the same row as the name of the complex type is given.

<table>
<thead>
<tr>
<th>Name of the complex type</th>
<th>Name of the attributes: connected data types</th>
<th>List of associated terms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CompanyData</strong></td>
<td>Name: string</td>
<td>EnterpriseData, OrganizationalData, Public Company, Limited Liability Company</td>
</tr>
<tr>
<td></td>
<td>ShortName: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Location: <strong>Address</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tax Office: Location string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tax Number: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V.A.T. Number: integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Company Type: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ILN Nr.: integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBAN Nr.: integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Weekly working hours: real</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal working days: string</td>
<td></td>
</tr>
<tr>
<td><strong>Address</strong></td>
<td>Street: string</td>
<td>Location, Permanent address, Temporary address</td>
</tr>
<tr>
<td></td>
<td>Country+PC: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Town: string</td>
<td></td>
</tr>
</tbody>
</table>
This service consists of 4 complex types as follows: CompanyData, Address, CustomerData, Contact. Contact and Address complex types are also connected to some attributes as data type of the attribute.

The global schema is shown in table 2. Please note, that in this demo example, only a relevant sample of the global schema is shown. Complex types of the global schema that are not presented in table 2 are semantically disjointed from the complex types in our service. Table 1.2 is structured the same way as table 1.1. The samples in my global schema consist of 4 complex types: Organization, Address, Customer and Days.

### Table 1.2. The applied sample of the global schema

<table>
<thead>
<tr>
<th>Name of the complex type</th>
<th>Name of the attributes: connected data types</th>
<th>List of associated terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>Name: string</td>
<td>Company, Enterprise, Partnership</td>
</tr>
<tr>
<td></td>
<td>Street: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Country+PC: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Town: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tax Office: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tax Number: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V.A.T. Number: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Company Type: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBAN: integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Weekly hours: real</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Working days: days</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>Street: string</td>
<td>Permanent, Temporary, ContactInfo, Location, Affiliation</td>
</tr>
<tr>
<td></td>
<td>Country+PC: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Town: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tel.: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax.: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E-mail: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Home Page: string</td>
<td></td>
</tr>
</tbody>
</table>
First the inner functions, \( N, T \) and \( A \) are calculated. Table 1.3. shows the returned values of the function \( N \) for every possible pair of the complex types of service and the global schema. Table 1.4 and Table 1.5 show the returned values of functions \( T \) and \( A \).

### Table 1.3. Return values of function \( N \)

<table>
<thead>
<tr>
<th>Name: Service/Global</th>
<th>Organization</th>
<th>Address</th>
<th>Customer</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompanyData</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Address</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CustomerData</td>
<td>0</td>
<td>0</td>
<td>0,5</td>
<td>0</td>
</tr>
<tr>
<td>Contact</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

After the inner functions \( N, T \) and \( A \) have been calculated, function \( F \) can also be calculated. The results are shown in table 1.5. The reader can see the returned value of \( F \) for each pair of complex types of the service and the global schema. The threshold in this experiment was set to 0.4. Complex types having a return value in the function \( F \) greater than this threshold should probably participate in the mappings. In other words, the transformations that define connections and operations between the attributes should be defined using the attributes of these complex types.

### Table 1.4. Return values of function \( T \)

<table>
<thead>
<tr>
<th>Terms: Service/Global</th>
<th>Organization</th>
<th>Address</th>
<th>Customer</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompanyData</td>
<td>0,357</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Address</td>
<td>0</td>
<td>0,5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CustomerData</td>
<td>0</td>
<td>0</td>
<td>0,3</td>
<td>0</td>
</tr>
<tr>
<td>Contact</td>
<td>0</td>
<td>0,182</td>
<td>0,444</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 1.5. Return values of function A.

<table>
<thead>
<tr>
<th>List of Attr.: Service/Global</th>
<th>Organization</th>
<th>Address</th>
<th>Customer</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompanyData</td>
<td>0.636</td>
<td>0</td>
<td>0.261</td>
<td>0</td>
</tr>
<tr>
<td>Address</td>
<td>0.429</td>
<td>0.6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CustomerData</td>
<td>0.316</td>
<td>0</td>
<td>0.64</td>
<td>0</td>
</tr>
<tr>
<td>Contact</td>
<td>0</td>
<td>0.727</td>
<td>0.25</td>
<td>0</td>
</tr>
</tbody>
</table>

The semantically related complex types in my example are as follows:
The *CompanyData* of the service schema and the *Organization* of the global schema are identified as semantically related. Although the name of the two concepts are not the same, and none of them is substring of the other, evaluating the similarities between the list of attributes with function A involves a higher value also in the function F. Examining these complex types manually, it is obvious that these concepts represent the same real world concepts. In spite of the different names of the concepts and differences in the list of associated terms, my method was able to identify the semantic relation between *CompanyData* and *Organization* complex types.

The *Contact* of the service schema and the *Address* of the global schema are also semantically related. The global schema does not contain any complex type describing the contact of a customer or a client. This information is contained by the address complex type of the global schema. Because the return value of function F is greater than the threshold, the complex type *Address* of the global schema should be attached to the complex type *Contact* by creating the mappings.

Other relationships have also been correctly identified. Obviously, *Address* complex type of the service should be mapped to the *Address* of the global schema and *CustomerData* should be mapped to the *Customer* complex type. The rest of the table shows that no other pairs of complex types have been identified as semantically related.

Table 1.6. Final results

<table>
<thead>
<tr>
<th>Final Results: Service/Global</th>
<th>Organization</th>
<th>Address</th>
<th>Customer</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompanyData</td>
<td>0.443</td>
<td>0</td>
<td>0.13</td>
<td>0</td>
</tr>
<tr>
<td>Address</td>
<td>0.214</td>
<td>0.625</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CustomerData</td>
<td>0.158</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Contact</td>
<td>0</td>
<td>0.427</td>
<td>0.281</td>
<td>0</td>
</tr>
</tbody>
</table>

1.5 Algorithm customizing

Although the presented approach is directly applicable for every kind of schemas, some customizing is reasonable before using it at the integration of productive systems. On one hand the weights $w_1$, $w_2$ and $w_3$ offer the possibility for fine tuning the algorithm which may lead to better results. On the other hand schema matching problems from different sizes requires totally different settings of the weights and threshold. So changing the weights and threshold for a given scenario leads to totally different results. In this chapter I analyze the influence of parameter settings to the performance of the approach.

Table 1.7 Ideal matching for the company-customer example

<table>
<thead>
<tr>
<th>Company-Customer</th>
<th>Address</th>
<th>Days</th>
<th>Customer</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Address</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CustomerData</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CompanyData</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
By the evaluation of the proposal I compare the results with an ideal solution. This ideal solution is provided by a human expert, who has the necessary knowledge to unambiguously define related pairs between the local and the global schema. These pairs get a value “1” in the solution matrix while other pairs are signed with value “0”. Table 1.7 shows this ideal matching for the example presented above.

One possible evaluation of the impact of parameter settings is the observation of the differences between the values of the returned similarity and the ideal matching. Changing of the weights \( w_1, \ w_2 \) and \( w_3 \) results changes in the difference. An indicator variable can be calculated as follows:
\[
I = \sum_{i,j} \text{abs} \left( \text{AlgRe} \ s_{i,j} - \text{IdealMatch}_{i,j} \right),
\]
(Exp 1.6)

where \( \text{AlgRe} \ s_{i,j} \) represents the matrix returned as similarity result by the proposed method, \( \text{IdealMatch}_{i,j} \) is the matrix containing the ideal matching and \( i, j \) are the row and column indexes of the matrices.

The value of this indicator variable shows the accuracy of the algorithm by the given settings of the parameters for the given scenario and for the given ideal matching definition of course. A lower I value shows a better configuration of the weight parameters and results a better result of the method.

Because the algorithm only contains linear equations in the calculations the theoretical minimum value of I is also analytically calculable for a given example. Disregarding the detailed description of calculations here, the algorithm parameters returning the lowest I value for the presented example are shown in table 1.8.

Table 1.8 Ideal parameter setting for company-customer scenario

<table>
<thead>
<tr>
<th>w1</th>
<th>w2</th>
<th>w3</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,05</td>
<td>0,1</td>
<td>0,15</td>
</tr>
<tr>
<td>0</td>
<td>10,94</td>
<td>10,84</td>
<td>10,75</td>
</tr>
<tr>
<td>0,05</td>
<td>10,8</td>
<td>10,71</td>
<td>10,64</td>
</tr>
<tr>
<td>0,1</td>
<td>10,69</td>
<td>10,62</td>
<td>10,56</td>
</tr>
<tr>
<td>0,2</td>
<td>10,61</td>
<td>10,55</td>
<td>10,51</td>
</tr>
<tr>
<td>0,3</td>
<td>10,57</td>
<td>10,52</td>
<td>10,49</td>
</tr>
<tr>
<td>0,4</td>
<td>10,55</td>
<td>10,52</td>
<td>10,5</td>
</tr>
<tr>
<td>0,5</td>
<td>10,57</td>
<td>10,55</td>
<td>10,54</td>
</tr>
<tr>
<td>0,6</td>
<td>10,57</td>
<td>10,57</td>
<td>10,54</td>
</tr>
<tr>
<td>0,7</td>
<td>10,62</td>
<td>10,61</td>
<td>10,62</td>
</tr>
<tr>
<td>0,8</td>
<td>10,69</td>
<td>10,7</td>
<td>10,72</td>
</tr>
</tbody>
</table>
| 0,9 | 0,8 | 10,82 | 10,86 | 10,91 | 10,98 | 11,07 | 0,8 | 1,3 | 4 | 9 | 6 | 5 | 6 | 8 | 2 | 7 | 5 | 3 | 4 | 6 | 3 | 23
A graphical representation of the effectiveness by different settings of weights is shown in figure 1.2. To achieve a positive indicator the presented results of the output scale are inverted with the following calculation:

\[ \text{ErrorRate} = 100 - \frac{\text{Effectiveness}}{100} \]

Furthermore it is normalized into the interval 0-100 by the following calculation:

\[ \text{Output} = (\text{Effectiveness} - 79.54) \times 10 \]

where the values of effectiveness are all between 79.54 and 89.54.

Please note, that for better visualization purposes the representation scale of figure 1.2 was reversed.

\[\begin{array}{cccccccccccccccc}
0.5 & 10.94 & 10.98 & 11.03 & 11.1 & 11.18 & 11.28 & 11.4 & 11.5 & 11.6 & 11.8 & 12.0 & 12.2 & 12.4 & 12.6 & 12.7 \\
0.65 & 11.55 & 11.63 & 11.72 & 11.83 & 11.95 & 12.1 & 12.7 & 13.2 & 13.4 & 13.7 & 14.2 & 14.9 & 15.2 & 15.6 & 16.3 & 16.7 \\
0.7 & 11.82 & 11.91 & 12.01 & 12.14 & 12.27 & 12.43 & 12.9 & 13.1 & 13.4 & 13.6 & 14.1 & 14.4 & 14.7 & 15.0 & 15.4 & 16.1 \\
0.75 & 12.11 & 12.22 & 12.34 & 12.47 & 12.63 & 12.8 & 13.3 & 13.6 & 13.8 & 14.3 & 14.6 & 14.9 & 15.2 & 15.6 & 16.3 & 17.1 \\
0.9 & 13.19 & 13.33 & 13.5 & 13.67 & 13.87 & 14.08 & 14.3 & 14.5 & 14.8 & 15.1 & 15.4 & 15.7 & 16.0 & 16.3 & 16.7 & 17.1 \\
0.95 & 13.61 & 13.77 & 13.94 & 14.14 & 14.34 & 14.57 & 14.8 & 15.0 & 15.3 & 15.6 & 15.9 & 16.2 & 16.6 & 16.9 & 17.3 & 17.7 \\
\end{array}\]

Figure 1.2 – Accuracy of my approach for the company-customer scenario by changing weights

Another analysis can be the comparison of the proposed pairs after using the threshold with the matrix of the ideal matching. This has a well documented, wide literature so far. Detailed analysis based on these evaluation methods is presented later in Chapter I.2. An extensive comparison of the accuracy of my proposal with other candidates from current researches is also presented there.
I.2 Evaluation of Approaches accuracy

To evaluate my approach (called approach NTA in the rest of my Thesis after its main parts examining the similarities in contexts Name, connected Terms and Attributes) the most current metrics of other researches and literature are applied [19, 49, 69]. The accuracy, the Recall and F-measure are widely accepted efficiency indicators for schema matching algorithms. However the applied methods for evaluation and comparison are not as accurate as they could be. To improve this, there is an in-depth analysis of the accuracy taking also into account the strong dependency of accuracy both on the specific input schemas and different selection of the configuration parameters e.g. the threshold level in my work.

First I shortly present two further approaches (the similarity flooding and a combined, structural matcher) from the literature in section 3.1. After that the results and analytics of the three different approaches on three different artificial schemas are introduced in section 3.2. In section 3.3 the results will be compared with the values extracted from an experiment involving more than 20 human test persons familiar with Information Sciences and Enterprise Application Systems. Finally, section 3.4 presents the evaluation of large schemas which are real candidates for using them as global schemas in integration scenarios.

2.1 Schema matching approaches from the literature

To be able to evaluate the efficiency and the run-time cost of my schema matching proposal (see previous chapter), two further schema matching approaches are presented and analyzed in my thesis. Sub-Chapter 2.1.1 summarizes the main idea behind the method called Similarity flooding and sub-Chapter 2.1.2 describes a combined matcher called here as approach WordNet shortly.

2.1.1 Approach Similarity flooding

The proposal similarity flooding is based on the following idea: if a graph is constructed where nodes represent entities like complex types or attributes and edges represent relationship among them like containing, inheritance or association, the relation between two given entity of the schemas is determined by the relation of their neighborhood. In other worlds, if the nodes close to entity A in the local schema are semantically related to corresponding nodes close to entity B in the global schema then entity A and entity B are probably semantically related and vice versa.

The key point of the approach is the algorithm for similarity flooding where current similarity values of the nodes in the close neighborhood is flooded among the connected nodes in a specially constructed pair wise connectivity graph (PCG). This is an iterative step, which is stopped by reaching a stable state or a given number of ran iterative steps. The results (similarity of entities creating common nodes in the PCG) are directly readable after the flooding is stopped [57].

In Chapter I.3 I briefly describe also every part of the approach.

2.1.2 Approach WordNet

Efficiency of schema matching can be improved by combining ideas of more approaches. The approach WordNet has a strong structural analyzer part but relies on the values of a dictionary called the WordNet [25]. After the initial values of similarity between entities is evaluated based on the WordNet, a complex structural matching algorithm determines the relations between the entities represented in directed acyclic graphs (DAGs) [18].

Every step of this approach is described shortly later in Chapter I.3 by analyzing the run-time performance of the algorithm.
2.2 Comparing algorithm accuracy

Three different matching problems were generated for testing purposes. Every problem contains two different representation of the same set of concepts of real world scenarios e.g. different data schemas originated from different enterprise application software vendors. The first one is the already presented small schema for managing basic company and customer data, see tables 1.2 and 1.3.

Because almost all presented approaches are strongly based on graph evaluations the graph representation of the schemas is also reasonable. The graphs are defined as follows:

Nodes are the complex types of schemas and simple types can be found at the leaves of the graph. Two complex types are connected with an edge if one (ancestor) has at least one attribute having the other complex type (child) as range. Children are always positioned below their ancestor. Finally, attributes of simple types (leaves) are connected to their parent node (to the complex type they belong to) and placed the same way below their parent as attributes having a complex type as their range. Figure 1.3 and 1.4 show the graph representations of the company-customer schemas for the local service and the global structure respectively.

Another test schema is of the university-publication scenario. See the two different schemas in figure 1.5. Detailed tables also containing list of attributes and associated terms can be found in appendix, see table 4.1 and 4.2.
The last test scenario is about car retailing. The overview of these schemas is shown in figure 1.6. For the details, see table 4.3 and 4.4 in the attachment.

### 2.2.1 Introducing accuracy indicators

Before the evaluation of the accuracy of matching algorithms some basic concepts and ideas should be clarified. One general schema matching problem of current researches and literature is defined as a solution consuming two different data schemas and providing an output containing matched pairs of entities. In other words most of the current approaches give a result in the form of matched pairs of entities. (Note that there are other definitions of the schema matching problem. For example [69] presents the creation of a global schema beyond several different schemas. Furthermore there are proposals which also aim at a full pairing of entities for example, see [57] and build on the stable marriage problem to provide a solution. See details within related work above.) To realize this, the calculated similarity values are filtered by a threshold: complex types returning a similarity value over the threshold are depicted as possible matches while others under the threshold as non-similar (semantically different) entities. After that, the efficiency of an algorithm is evaluated by comparing the set of matched pairs with the set of ideally matched pairs. For smaller schemas (at a size which can also be processed by a human) this set of ideally matched pairs can be defined by integration experts familiar with the given schemas.

Finally, the result itself (the accuracy) is evaluated using some indicators presenting the differences between the values produced by the given approach and the ideal matching.
The applied indicators are the following:
- The precision returns a value comparing the number of correctly found pairs with the number of all proposed pairs. (Correctly found pairs are both part of the ideal matching and evaluated algorithm result.) The precision is calculated as follows:
  \[
  \text{Precision} = \frac{\text{Correctly found pairs}}{\text{All proposed pairs}}
  \]
- The recall returns a value comparing the number of correctly found pairs with the number of all existing pairs in the ideal matching. The recall is calculated as follows:
  \[
  \text{Recall} = \frac{\text{Correctly found pairs}}{\text{All pairs of the ideal matching}}
  \]
- The F-measure is a combined indicator originated from the previous two:
  \[
  \text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
  \]

The ideal matching for every scenario was defined first. After that, the algorithms were run and results were filtered by different threshold values. There are two types of analysis in my work:
- Evaluation with pre-defined (default) threshold values, where the optimal thresholds were set for every approach before running the tests and these were applied to filter the results of all scenarios. See Chapter 3.1.2
- Evaluation with changing (optimized) threshold values, where the thresholds leading to the best results are accepted for each algorithm in each scenario. Contrary to many accuracy analyses and comparison found in current literature, this method may give a fair evaluation of different schema matching approaches because it also deals with the changing of efficiency indicators by changing threshold on every scenario one-by-one. See Chapter 3.1.3.

The returned similarity values of the different approaches for the company-customer scenario are the following:

<table>
<thead>
<tr>
<th>Scenario: Company-customer</th>
<th>Approach</th>
<th>NTA</th>
<th>Address</th>
<th>Days</th>
<th>Customer</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact</td>
<td>0.38</td>
<td>0</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>0.59</td>
<td>0</td>
<td>0</td>
<td>0.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CustomerData</td>
<td>0.03</td>
<td>0</td>
<td>0.34</td>
<td>0.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CompanyData</td>
<td>0</td>
<td>0</td>
<td>0.16</td>
<td>0.31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario: Company-customer</th>
<th>Approach</th>
<th>Similarity Flooding</th>
<th>Address</th>
<th>Days</th>
<th>Customer</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact</td>
<td>0.04</td>
<td>0.03</td>
<td>0.07</td>
<td>0.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>0.04</td>
<td>0.03</td>
<td>0.04</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CustomerData</td>
<td>0.06</td>
<td>0.06</td>
<td>0.2</td>
<td>0.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CompanyData</td>
<td>0.07</td>
<td>0.07</td>
<td>0.14</td>
<td>0.11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario: Company-customer</th>
<th>Approach</th>
<th>WordNet</th>
<th>Address</th>
<th>Days</th>
<th>Customer</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>0.11</td>
<td>0.27</td>
<td>0.19</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contact</td>
<td>0.08</td>
<td>0.16</td>
<td>0.15</td>
<td>0.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CompanyData</td>
<td>0.14</td>
<td>0.15</td>
<td>0.21</td>
<td>0.19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CustomerData</td>
<td>0.12</td>
<td>0.12</td>
<td>0.18</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The returned values for the University-publication and Car retailer scenarios are:

<table>
<thead>
<tr>
<th>Scenario: University-publication</th>
<th>Approach</th>
<th>NTA</th>
<th>Journal</th>
<th>Address</th>
<th>Proceed-</th>
<th>Article</th>
<th>Book</th>
<th>Publication</th>
<th>Researcher</th>
<th>University</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>0</td>
<td>0.88</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Author</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.51</td>
<td>0.51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Book</td>
<td>0.06</td>
<td>0</td>
<td>0.05</td>
<td>0.82</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>0.27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monograph</td>
<td>0.06</td>
<td>0</td>
<td>0</td>
<td>0.55</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0.27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Journal</td>
<td>0.51</td>
<td>0</td>
<td>0.08</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.33</td>
<td>0.38</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 1.12 Returned values of approaches for the University-publication scenario

<table>
<thead>
<tr>
<th>Scenario: Car-retailer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Approach NTA</strong></td>
</tr>
<tr>
<td>Car</td>
</tr>
<tr>
<td>Inventory</td>
</tr>
<tr>
<td>Available</td>
</tr>
<tr>
<td>Autotrader</td>
</tr>
</tbody>
</table>

| **Approach SF** | Item | Manager | Members | Options | Car | Stock | Showroom | Staff | Carseller |
| Car | 0.2 | 0.3 | 0.28 | 0.11 | 0.82 | 0.19 | 0.19 | 0.19 | 0.41 |
| Inventory | 0.25 | 0.26 | 0.26 | 0.13 | 0.39 | 0.23 | 0.19 | 0.2 | 0.25 |
| Available | 0.11 | 0.13 | 0.13 | 0.07 | 0.17 | 0.11 | 0.1 | 0.11 | 0.12 |
| Crew | 0.3 | 0.43 | 0.43 | 0.17 | 0.58 | 0.28 | 0.28 | 0.28 | 0.42 |
| Autotrader | 0.22 | 0.28 | 0.26 | 0.13 | 0.41 | 0.22 | 0.19 | 0.2 | 0.29 |

| **Approach WN** | Item | Options | Members | Manager | Car | Staff | Showroom | Stock | Carseller |
| Car | 0.29 | 0.26 | 0.26 | 0.28 | 0.48 | 0.25 | 0.38 | 0.37 | 0.3 |
| Crew | 0.32 | 0.29 | 0.36 | 0.33 | 0.27 | 0.35 | 0.26 | 0.28 | 0.28 |
| Available | 0.28 | 0.3 | 0.25 | 0.3 | 0.32 | 0.26 | 0.41 | 0.42 | 0.43 |
| Inventory | 0.31 | 0.3 | 0.27 | 0.31 | 0.36 | 0.28 | 0.47 | 0.57 | 0.52 |
| Autotrader | 0.24 | 0.24 | 0.25 | 0.26 | 0.22 | 0.29 | 0.21 | 0.29 | 0.41 |

The ideal matches are shown in table 1.13. Related pairs are signed with a value “1” in the given cells while other cells contain a value of “0”.

Table 1.13 Predefined ideal matches for the scenarios

<table>
<thead>
<tr>
<th><strong>Company-customer</strong></th>
<th>Address</th>
<th>Days</th>
<th>Customer</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Address</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CustomerData</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CompanyData</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>University-publication</strong></th>
<th>Journal</th>
<th>Article</th>
<th>Address</th>
<th>Proceeding</th>
<th>Article</th>
<th>University</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Author</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Book</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Monograph</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Journal</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Article</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Library</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>University</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
2.2.2 Efficiency with pre-defined thresholds

First, I have performed experiments using the same predefined threshold values for every scenario. Because the threshold is not customized for scenarios one-by-one, the optimal accuracy (i.e., maximal values of accuracy indicators) cannot be reached in every scenario. Moreover, by the determination of the pre-defined thresholds, I intended to find a value which results in similarly acceptable accuracy for all scenarios. The predefined values used in this chapter are the following:

NTA: 0.3; Similarity flooding: 0.053; WordNet: 0.2.

After applying the threshold value, the following entities of the Company-customer scenario were found semantically relative by the approach NTA. The relative entities are nominated by a “1” in the given cells of the matrix in Table 1.14.

Table 1.14: Semantically related entities based on the results of proposal NTA

<table>
<thead>
<tr>
<th>Scenario: Company-customer</th>
<th>Approach</th>
<th>Address</th>
<th>Days</th>
<th>Customer</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTA</td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The proposed semantically relative pairs of all approaches for all scenarios were also calculated but the size limit of this work prevents me to present all of them here. Nevertheless, interested readers can construct these tables easily and can check the rest of the calculations.

Table 1.15 shows the values of the efficiency indicators calculated with these thresholds for every scenario. The approaches are nominated with abbreviations, where SF nominates approach Similarity flooding and WN nominates approach WordNet.

Table 1.15: Efficiency indicators for the different approaches by pre-defined thresholds

<table>
<thead>
<tr>
<th>Scenarios:</th>
<th>Company-customer</th>
<th>University-publication</th>
<th>Car-retailer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NTA</td>
<td>SF</td>
<td>WN</td>
</tr>
<tr>
<td>Precision</td>
<td>0.8</td>
<td>0.25</td>
<td>1</td>
</tr>
<tr>
<td>Recall</td>
<td>1</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.89</td>
<td>0.4</td>
<td>0.86</td>
</tr>
</tbody>
</table>

As one can see, approach NTA performs well almost in every scenario in this experiment. However, it provides lower efficiency for the second (University-publication) scenario than the approach Similarity flooding. On the other hand, both approaches, Similarity flooding and WordNet, seem to be very sensitive to the selection of different threshold values for the different scenarios. Using the same threshold value for every scenario results rather low accuracy in 2 scenarios out of the total 3 by these approaches. Thus, different thresholds should be selected for every scenario – see sub-chapter 2.2.3. The main reason behind this lies in the different sizes (and different structural parameters like specific number of attributes per complex type, specific number of attributes having complex types as range per number of
attributes within a complex type, specific number of ancestors of a complex type, etc.) which results in similarity values arriving in a different range. Please note that the complex analysis of these reasons is out of scope of my work.

Although approach NTA performs quite well also with predefined threshold, further improvement is expected by using changing (optimized) threshold selection for the different scenarios.

2.2.3 Efficiency with changing (optimized) thresholds
Changing the threshold values changes the values of the performance indicators precision, recall and F-measure. On one hand, the precision is lower at low threshold values because a lot of false matches are proposed as related concepts. On the other hand the recall is high then, because the algorithm probably finds all of the related pairs of the ideal matching. At higher threshold values the precision and recall indicators behave just vice versa.

Because the F-measure is a combined efficiency indicator of the previous two, it is expected to have a maximum somewhere in the middle of the examined interval. Therefore the F-measure is not only applicable for overall ratings of the results of different approaches, but it can also be used to find the optimal threshold value. I make the assumption in my work that the optimal threshold value can be found at the maximum point of the F-measure function.

Table 1.16 shows the changes in the indicators at various threshold values for the approach NTA at the Car-retailer scenario. The optimal selection for the threshold is highlighted in the table. For more spectacular presentation figure 1.7 shows the values in graphs.

<table>
<thead>
<tr>
<th>Scenario: Car-retailer by changing thresholds</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTA, Thresholds:</td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.15</td>
</tr>
<tr>
<td>0.2</td>
</tr>
<tr>
<td>0.25</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td>0.35</td>
</tr>
<tr>
<td>0.4</td>
</tr>
<tr>
<td>0.45</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.55</td>
</tr>
<tr>
<td>0.6</td>
</tr>
</tbody>
</table>

Figure 1.7 Efficiency indicators for the approach NTA in scenario Car-retailer by changing thresholds

Due to the existing limits in size of this work I skip the detailed presentation of precision and recall values. Only the changes of F-measure are introduced below. Table 1.17 shows the F-
measure values for changing thresholds for all approaches and for all scenarios. Because the F-measure function is relevant (not constant) not at the same range for the different approaches, the examined interval and step for the threshold values is different for every approach. These are shown in columns Tre-NTA, Tre-Sim and Tre-WN in table 1.17 corresponding to approaches NTA, Similarity flooding and WordNet respectively. Optimal threshold values and corresponding maximal F-measure values are highlighted.

Table 1.17 Changes of F-measure values by changing thresholds

<table>
<thead>
<tr>
<th>Scenarios:</th>
<th>Company-customer</th>
<th>University-publication</th>
<th>Car-retailer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Treshold</td>
<td>F-measure</td>
<td>Treshold</td>
</tr>
<tr>
<td>Tre-NTA</td>
<td>Tre-Sim</td>
<td>Tre-WN</td>
<td>Tre-NTA</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.05</td>
<td>0.01</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.1</td>
<td>0.02</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.15</td>
<td>0.03</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.2</td>
<td>0.04</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.25</td>
<td>0.05</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.3</td>
<td>0.06</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.35</td>
<td>0.07</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.4</td>
<td>0.08</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.45</td>
<td>0.09</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.5</td>
<td>0.1</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.55</td>
<td>0.11</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.6</td>
<td>0.12</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.65</td>
<td>0.13</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.7</td>
<td>0.14</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.75</td>
<td>0.15</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.8</td>
<td>0.16</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
<tr>
<td>0.85</td>
<td>0.17</td>
<td>0.61 0.61 0.61</td>
<td>0.61 0.61 0.61</td>
</tr>
</tbody>
</table>

As the reader can see there may exist more maximum points for the F-measure function. On one hand, the technical reason behind this is that the results are rounded values and the range is discrete. On the other hand the precision and recall values can change exactly the same extent with changing the threshold, so it is possible that departing from the “maximum point” the recall decreases while the precision increases to the same extent or vice versa. For example see the results of the approach WordNet for the Car-retailer scenario at threshold values 0.275 and 0.325. The precision and recall values are 0.47-0.57 and 0.6-0.43 where the precision increased the same extent (from 0.47 to 0.6) as the recall decreased (from 0.57 to 0.43). In these cases I highlighted the more balanced threshold (where the average deviation of precision and recall is smaller) as optimum point.

Figures 1.8-1.10 shows the values for every approach in graphs.
Summarizing the above shown trends in F-measure values we can say that my approach (NTA) is at least as accurate as other approaches and most of the cases it also outperforms them. By searching for the optimum points of every approach in every test scenarios I have ensured the same optimal environment for all examined approaches. Based on this we can definitely trust the results of the accuracy evaluation.

### 2.3 The human based experiment

As already mentioned the test schemas were also analyzed by human test users familiar with IT and enterprise computing. The 23 people had to evaluate the whole table of data structures of all the 3 scenarios and propose semantically related pairs. The received results were summarized and average values were calculated for the similarity values of every possible pairs of entities between the global and the local schemas. More precisely, the values of proposed related pairs – signed with a value “1” by the human users – and values of not proposed pairs – signed with a value “0”– were averaged. The ideal matching was defined upon these calculations and predefined threshold values which were selected for every scenario. Table 1.18 shows the ideal matching calculated this way.

**Table 1.18: Ideal matching defined upon the average results of human tests**

<table>
<thead>
<tr>
<th>Company-customer</th>
<th>Days</th>
<th>Address</th>
<th>Organization</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Contact</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CompanyData</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CustomerData</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Similarly to the overall accuracy table at the fixed threshold analysis versus the ideal matches, table 1.19 shows the overall performance of the approaches for all scenarios.

<table>
<thead>
<tr>
<th>Scenarios:</th>
<th>Company-customer</th>
<th>University-publication</th>
<th>Car-retailer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicators</td>
<td>NTA, SF, WN</td>
<td>NTA, SF, WN</td>
<td>NTA, SF, WN</td>
</tr>
<tr>
<td>Precision</td>
<td>1, 0.67, 0.55, 0.63</td>
<td>0.8, 0.57, 0.57</td>
<td>0.88, 0.55, 0.57</td>
</tr>
<tr>
<td>Recall</td>
<td>1, 0.4, 0.57, 0.48</td>
<td>0.88, 0.5, 0.5</td>
<td>0.88, 0.53, 0.53</td>
</tr>
<tr>
<td>F-Measure</td>
<td>1, 0.5, 0.75, 0.54</td>
<td>0.88, 0.53, 0.53</td>
<td>0.88, 0.53, 0.53</td>
</tr>
</tbody>
</table>

As already mentioned in chapters above, every integration task (e.g. data migration between different enterprise systems or process orchestration above services of several applications) begins with finding the relevant (semantically related) entities of different enterprise systems. Hence the same schema matching task should be performed by human users (IT and/or integration experts) as in my experiment. The question is that how efficient could this task be supported by the schema matching approaches?

Because automatic schema matching solutions never (or just occasionally) can provide a perfect solution, it is quite obvious that this task can not be fully performed without human intervention. The algorithms provide only proposals with given error rate and results should be supervised by human users. The supervision consists of two tasks:

- inspecting the proposed related pairs if they are really semantically related entities in the given schemas of given systems in the given integration scenario and
- checking for possible further related pairs which were not detected by the algorithm.

To decide the applicability of the different approaches the following simplified model of costs is introduced:

- the cost of finding one false proposal and deleting it from the list of related pairs is 1 unit,
- the cost of finding one additional related pair of entities and adding it to the list of related pairs is 1 unit as well and
- the run-time cost of the algorithms is disregarded, because now I focus only on the human work effort and algorithms can be run in the background without any human intervention.
It is trivial, that the total cost of finding all related pairs by a human user without applying any of the approaches as a support is the number of all related pairs:

\[ \text{Cost}_{\text{onlyhuman}} = \text{RelatedPairs} \]  

(Exp. 1.7)

On the other hand, using a schema matching algorithm as a preparation step and finalizing the results manually after that costs as follows:

\[ \text{Cost}_{\text{supported}} = \text{Falsehits} + \text{Missedhits} = \text{RelatedPairs} \times \left[ (1 - \text{Precision}) + (1 - \text{Recall}) \right] \]  

(Exp. 1.8)

By comparing the cost of these the following relations hold:

\[ \text{Cost}_{\text{onlyhuman}} \leq \text{Cost}_{\text{supported}} \iff \text{RelatedPairs} \leq \text{RelatedPairs} \times \left[ (1 - \text{Precision}) + (1 - \text{Recall}) \right] \nonumber \]

\[ \nonumber \iff 1 > \left[ (1 - \text{Precision}) + (1 - \text{Recall}) \right] = 2 - \text{Precision} - \text{Recall} \]  

(Exp. 1.9)

, where \( > \leq \) is a relation bigger or smaller or equal, and can be substituted later.

Upon this it is reasonable to use a similarity matching algorithm if

\[ \text{Cost}_{\text{onlyhuman}} > \text{Cost}_{\text{supported}} \iff 1 > 2 - \text{Precision} - \text{Recall} \iff 1 < \text{Precision} + \text{Recall} \]  

(Exp. 1.10)

Evaluating the efficiency indicators Precision and Recall in table 1.19 we can see, that based on the simplified cost calculation model presented above, it is reasonable to also run a matching algorithm instead of creating the results fully manually in integration problems. Note, that more sophisticated cost calculation models and approaches can also be created and may lead to another conclusion about the applicability of the schema matching approaches based on these results. Evaluating several cost models are out of the scope of my work so far. Nevertheless, the measured performance of my approach – approach NTA – is convincing by finding definitely more than the half of the related pairs by an error rate lower than 20% for the proposed pairs in the worst case scenario as well.

Before applying a schema matching algorithm the detailed aims of its usage should be clarified as well. As the reader could see, there is not an exact optimum value for the parameterization (e.g. selecting the threshold value) so it must be aligned to the given aim of schema matching preparation. Basically, there are two different ways for the application:

- Maximizing the value of precision by keeping the recall still in an acceptable interval results in a proposed matching, which probably not contains false proposals but actually misses to find a lot of related pairs. This method can be used if we desire that proposed pairs are correct and can disclaim about the high rate of found pairs compared with the ideal matching. Therefore, users, who want to have an automatic solution providing mostly only correct results and are not willing to find all of the pairs of the ideal matching by all means are suggested to use this method. Generally precision can be increased by selecting a higher value for the threshold.

- On the other hand maximizing the value of recall by keeping the precision still in an acceptable interval results a proposed matching containing a high rate of the pairs of the ideal matching but having also a high number of false proposed pairs. Therefore users, who prefer detecting and deleting of false proposed pairs in contrast to searching for missing ones should use this method. The recall can be increased mostly by lowering the value of threshold.

Note, that finding missing related pairs in large schemas (e.g. containing more than 100 complex types) is hardly performable for a human user. Hence selecting parameters resulting higher recall values is reasonable for large schemas. (Evaluation of algorithm efficiency on large schemas will also be described in Chapters 2.4 and 1.3 later.)
2.4 Evaluation on large schemas

The evaluation of the accuracy of approach NTA was performed also on real business schemas. However checking the results can be done only manually because no ideal matrix the size of this schemas prevent to create an ideal matrix or simply examine it is available bye a human user.

As input schemas the ordermanagement domain of the xCBL[91] and OAGIS[63] were used. The previous uses more then 100 complex type while the last contains only 24. This shows already the significant difference in the granularity of these schemas. Nevertheless our experiment proved that the NTA algorithm is able to find related pairs also between test schemas from different granularity like this.

It took the NTA algorithm 4.11 seconds to perform the given task. (Note that before running the algorithm a parsing and basic modification of the input schemas was necessary to convert them into a compatible form with our reference implementation.)

Here are some detected related types for further evaluation:
- The main entities (the root elements) of the schemas called \textit{PurchaseOrderType} and \textit{OrderType} were found related. So the algorithm was able to find this trivial pair of entities.
- Similarly other important (“container”) entities like \textit{PurchaseOrderHeaderType}, \textit{OrderHeaderType} and \textit{PurchaseOrderLineType}, \textit{OrderLineType} were also successfully detected as related pairs.
- The \textit{PartyType} entity of the Oagis schema was identified as related to the following entities of the xCBL: \textit{OrderConfirmationResponsePartyType}, \textit{OrderConfirmationPartyType}, and \textit{OrderRequestPartyType}. Taking into account the differences in the granularity of the schemas these are also valid results, because these entities of the xCBL schemas should be connected with the \textit{PartyType} of the Oagis depending on the actual messaging type i.e. request, response or confirmation.
- The \textit{PurchaseOrderLineType} and \textit{OrderHeaderType} related. It is obvious that the corresponding real world concepts really have connection and may be connected with transformations in given mapping of given integration scenarios. Nevertheless the algorithm gave them lower similarity values than for the pair \textit{PurchaseOrderLineType}, \textit{OrderLineType} which seem to be also a right working.
- The \textit{CodesType} entity of the Oagis schema was paired with a lot of possibly related concepts of the xCBL. Similarly to the \textit{PartyType}, \textit{CodesType} really have more possible right pairs in xCBL because the granularity differences implicates the possible selection of more entities in the other schema based on the given messaging type in the given integration scenario.

As we could see the algorithm was able to successfully detect right pairs also between real business schemas. However the evaluation of not found similarities is not possible because it simply can not be checked by a human user at the given order of schema sizes.

Other approaches are expected to give results of similar accuracy level. Hence they are not tested for these real schemas. On the other hand only required computational cost of the different algorithm could only be compared which is presented later in the next chapter.

The next section describes the complexity of the algorithm from the point of view of computational costs.
I.3 Computational complexity of approaches

Besides the expected accuracy of the results the run time cost of an algorithm is also a key aspect when looking for the solution to a given problem. More accurate solutions may require much more resources (e.g. computational performance) to provide results. This can lead to enormously long run-time in the given hardware configurations which can not be accepted by some systems e.g. online, real time systems or even by a solid development environment. Thus algorithm accuracy should always be evaluated with respect to the run-time costs of the solution.

The costs of an algorithm are strictly connected with the complexity of the given method. In most cases complexity is predicted by the expected number of the steps at the execution. Because there is a (single) computer operation in the background of each performed step, this approach can follow the real computational requirement (indicating the actual value of the run-time in a given computer configuration) close enough. Hence the number of the expected steps for every approach is determined in the next paragraph.

3.1 Complexity of approach NTA

This section presents the computational complexity of my above presented approach. By evaluating the correspondence between the naming of different complex types of schemas I use function $N(C_i,G_j)$ which is a single one-step operation – function substring. So the number of steps of function $N(C_i,G_j)$ is 1 for a given evaluation of two different complex types.

Calculating the similarity based on attached terms uses the function $T(C_i,G_j)$. This compares all attached concepts of evaluated couples which results $(\|\text{Term}(C_i)\|+1)\times(\|\text{Term}(G_j)\|+1)$ number of steps, where $\|\text{Terms}(C_i)\|$ is the number of terms connected to complex type $C_i$ and +1 represents the name of the complex type, which was added to the set of terms before the comparison.

The correspondence between the attributes of complex types is calculated by the function $A(C_i,G_j)$. Within this every attribute of the local schema is compared with every attribute of the global schema resulting $\|\text{Attr}(C_i)\|\times\|\text{Attr}(G_j)\|$ number of steps where $\|\text{Attr}(C_i)\|$ represents the number of attributes connected to complex type $C_i$.

By the comparisons of whole local and global schemas all functions depicted above should be calculated to all pairs of the complex types of the schemas. This means $\text{Steps}_{NTA\_Perform}(C,G) = \sum_{i,j}1 + (\|\text{Term}(C_i)\|+1)\times(\|\text{Term}(G_j)\|+1) + \|\text{Attr}(C_i)\|\times\|\text{Attr}(G_j)\|$ (Exp 1.11) steps for schemas C and G. Note that in my understanding the symbol $\sum_{i,j}$ means a sum to all possible pairs of complex types $C_i$ and $G_j$. So instead of the expression $\sum_{i} \sum_{j} (...)$ I will use the previous one in the rest of my Thesis.

Remember, that the execution of the algorithm requires some preparation e.g. constructing trees from complex types and ranking possible paths by their depth. This requires the processing of all complex types with their all attributes which costs
Steps\textsubscript{NTA\_Prepar} (C, G) = \sum \|\text{Attr}(C)\| + \sum \|\text{Attr}(G)\| + \|C\| + \|G\| + \|\text{Term}(C)\| + \|\text{Term}(G)\|, \quad (\text{Exp} \ 1.12)

where \(\|C\|\) is the number of the complex types in the local schema and \(\|G\|\) denotes the number of complex types in the global schema. (Note that the ranking of paths by their depth can be implemented in the same time by the whole run of the complex types. Thus it requires no additional costs.)

Concluding, the NTA algorithm requires
\[
\text{Steps}_{\text{NTA}} (C, G) = \text{Steps}_{\text{NTA\_Prepar}} (C, G) + \text{Steps}_{\text{NTA\_Perform}} (C, G)
\]
(Exp 1.13) steps for total evaluation of two given schemas.

Note that the method can be optimized for performance significantly by eliminating the comparison of complex types coming from very different granularity levels (in other words from very different distances from the root of trees of global and local schemas). To my experiences these complex types hardly have semantic relation to each other. However, I calculate with the worst case in the rest of my thesis as theoretically all complex types of the local schema can be relevant to all complex types of the global schema. Note, that this is also often the case by smaller (extremely horizontal structured) schemas.

To the later evaluation of the effectiveness of the NTA algorithm other candidates should also be examined. The next two sections present the main idea and complexity of two other proposals. The expected steps of the methods similarity flooding and Wordnet are determined in Chapters 3.2 and 3.3 respectively.

### 3.2 Complexity of approach Similarity Flooding

As a first step an open information model (OIM) based graph [15] is constructed both from the local and global schema. (The OIM is a general representation for every kind of data sources e.g. database tables or XML schemas. The latter makes it also possible to represent the entities of my research in OIM.) Similarly to the initial graph construction in the NTA approach this step requires the processing of all complex types with all their attributes in both schemas and constructing a special structure of data for latter use. The computational cost of this step is
\[
\text{Steps}_{\text{OIM}} (C, G) \approx 2 \left( \sum \|\text{Attr}(C)\| + \sum \|\text{Attr}(G)\| + \|C\| + \|G\| + Y_c + Y_G + 3 \right), \quad (\text{Exp} \ 1.14)
\]
where \(Y_c\) (\(Y_G\)) is the number of existing data types in the local schema (global schema respectively) and the constant value of 3 at the end of the expression is the insertion cost of the three special category nodes (complex type, attribute, and attribute type).

An initial mapping between the nodes of the two OIM graphs is calculated after that. This will create a rough starting value for the similarity of nodes (used later by the initializing of the flooding algorithm at the PCG). The initial mapping is calculated as the rate of the longest conformable prefix and the length of the name of the nodes in the OIM. Because the generated number of nodes to each complex type depends on the existence of its attributes and whether the range of some attributes are also complex types, the OIM contains 2 or 4 nodes for each attribute of each complex type. However, nodes representing given data types are “reused” to define the data type of further attributes from the same data type. Hence the
number of generated nodes of all attributes is 2 times the number of attributes plus 2 times the number of data types existing in the given schema.

Besides, there are exactly 2 nodes generated to every complex type and 3 initial nodes for signing complex type, attribute and data type category nodes. During the creation of initial mapping every node of the OIM generated to the local schema must be compared to every node of the OIM generated to the global schema. Thus the number of the steps (comparisons) performing in this section is the following:

\[ \text{Steps}_{\text{Init - map}}(C, G) = \left( 3 + 2||C|| + 2\sum_i ||\text{Attr}(C_i)|| + 2Y_c \right) \ast \left( 3 + 2||G|| + 2\sum_j ||\text{Attr}(G_j)|| + 2Y_G \right) \quad \text{(Exp 1.15)} \]

The **pair wise connectivity graph** (PCG) is created along the same type of edges of the two schemas. (The type of edges can be type, name, and other relation like attribute and data type). To do so, every edge of the same type from the two schemas is evaluated. Before counting with the whole number of steps, the number of edges from each type in the OIM graph should be calculated. The number of edges from each type (type, name, attribute and data type respectively) in the OIM of service schema is the following:

\[ \text{EdgeNum}_{\text{Type}}(C) = ||C|| + \sum_i ||\text{Attr}(C_i)|| + Y_C, \quad \text{(Exp 1.16)} \]

because all nodes of the OIM (nodes for the complex types, attributes and data types) are categorized with one type edge connecting them to one of the 3 category nodes.

\[ \text{EdgeNum}_{\text{Attribute}}(C) = \sum_i ||\text{Attr}(C_i)||, \quad \text{(Exp 1.17)} \]

\[ \text{EdgeNum}_{\text{Data - type}}(C) = \sum_i ||\text{Simple - Attr}(C_i)||, \quad \text{(Exp 1.18)} \]

where \( \text{Simple - Attr}(C_i) \) represents the set attributes of \( C_i \) having a simple type as their range,

\[ \text{EdgeNum}_{\text{Name}}(C) = ||C|| + \sum_i ||\text{Attr}(C_i)|| + Y_C, \quad \text{(Exp 1.19)} \]

The number of the edges of each type can be described exactly the same way for the OIM graph of global schema (e.g. \( \text{EdgeNum}_{\text{Attribute}}(G) = \sum_j ||\text{Attr}(G_j)|| \)). \quad \text{(Exp 1.20)}

The overall number of steps of this section is the sum of comparison of edges from the same type coming from the OIM graphs of the local and the global schema:

\[ \text{Steps}_{\text{PCG}}(C, G) = \sum_{\text{TYPE}} \text{EdgeNum}_{\text{TYPE}}(C) \ast \text{EdgeNum}_{\text{TYPE}}(G), \quad \text{(Exp 1.21)} \]

where TYPE stands for type, attribute, data type and name.

Substituting expressions 1.17-120 into expression 1.21 and performing some contractions the overall steps of the PCG creation is the following:
The expected number of computational steps during the flooding itself is the number of the nodes in the PCG. This can be calculated by observing the creation method of the PCG graph and identifying the situations when new nodes are created into the PCG. Hence the number of created new nodes is in correspondence with the number of evaluated edge-pairs from types Type and Name in the two OIM graphs is as follows:

\[
\text{Steps}_{\text{PCG}}(C, G) = 2 \left( |C| + \sum_i |\text{Attr}(C_i)| + Y_C \right) \times \left( |G| + \sum_j |\text{Attr}(G_j)| + Y_G \right) + \sum_i |\text{Simple \_Attr}(C_i)| \times \sum_j |\text{Simple \_Attr}(G_j)|
\]

(Exp. 1.22)

The flooding itself must be repeated for a number of iterative steps. This requires

\[
\text{Steps}_{\text{Flooding}}(C, G) = \sum_{\text{steps}} \text{Nodes}_{\text{PCG}}(C, G) = \sum_{\text{steps}} \left( 2 \times \left( |C| + \sum_i |\text{Attr}(C_i)| + Y_C \right) \times \left( |G| + \sum_j |\text{Attr}(G_j)| + Y_G \right) \right)
\]

(Exp. 1.24)

where \(\sum (...)\) signs that this must be counted for all iterative steps.

The overall number of steps of the similarity flooding approach is calculated by the sum of the previous four sections:

\[
\text{Steps}_{\text{Sim \_Flooding}}(C, G) = \text{Steps}_{\text{OIM}}(C, G) + \text{Steps}_{\text{Sim \_map}}(C, G) + \text{Steps}_{\text{PCG}}(C, G) + \text{Steps}_{\text{Flooding}}(C, G)
\]

(Exp. 1.25)

The next chapter presents a combined matcher based on the WordNet taxonomy system [25] and predicts its computational costs.

### 3.3 Complexity of approach WordNet

There is a preparation cost of constructing the DAG representation of the two schemas by this approach as well. The number of expected steps of this is:

\[
\text{Steps}_{\text{TreeConstr}}(C, G) = \sum_i |\text{Attr}(C_i)| + \sum_j |\text{Attr}(G_j)| + |C| + |G|
\]

(Exp. 1.26)

Because all complex types and their all attributes should be processed and inserted into the right place of the trees.

Furthermore, I have implemented a solution, which retrieves the similarity values to all corresponding entities between the global and the local schema from the WordNet dictionary at the same time. Because there are multiple calculations later during the method requiring this information the one time evaluation and caching of results may decrease the overall steps of WordNet accesses. On the other hand for schemas of realistic size (50-100 complex types maximum) it is a treatable overhead of memory usage and may indicate significant improvement in algorithm run-time later. Because every entity of the local schema must be related by the WordNet to every entity of the global schema, this step costs
The structural similarity of graph nodes is calculated from 3 aspects. The ancestor, child and leaf context all contribute to the final value of the similarity.

The **ancestor** similarity is calculated as follows:

\[
\text{Sim}_{\text{ancestor}}(C_i, G_j) = \text{PS}(P_1, P_2) \times \text{LS}(C_i, G_j),
\]

where \( \text{PS}(P_1, P_2) \) is the path similarity function for the path starting from entities \( C_i \) (\( G_j \) respectively) to the root element of the given trees and function \( \text{LS}(C_i, G_j) \) represents the result of WordNet request for \( C_i \) and \( G_j \). Because the values of \( \text{LS} \) are already available only the two paths should be found. Because the computational complexity of the functions contained by the path similarity is in \( \text{O}(n^2) \), the number of steps in this task is estimated as follows:

\[
\text{Steps}_{\text{ancestor}}(C_i, G_j) = \text{length}(P_1) \times \text{length}(P_2)
\]

The **child** context is also based on the function \( \text{LS} \). It is calculated by the average similarity values of the highest 25% (or 50%) of the \( \text{LS} \) function between all child of the given entity of the global schema (\( G_j \)) and the given entity of the local schema (\( C_i \)). This requires the following number of steps:

\[
\text{Steps}_{\text{child}}(C_i, G_j) = \left( \text{child}(C_i) \times \text{child}(G_j) \right) + \left( \text{child}(C_i) \times \text{child}(G_j) \right) \times 1.39 \times \log_2 \left( \text{child}(C_i) \times \text{child}(G_j) \right)
\]

where \( \text{child}(C_i) \) (\( \text{child}(G_j) \) respectively) represents the number of the children of the node \( C_i \) (\( G_j \) respectively) and \( \left( \text{child}(C_i) \times \text{child}(G_j) \right) \times 1.39 \times \log_2 \left( \text{child}(C_i) \times \text{child}(G_j) \right) \) is the sorting cost of the results. (In the implementation I used the sort function which is based on a quick-sort algorithm having an average computational requirement of \( 1.39 \times n \times \log_2 n \) steps.)

The **leaf** similarity is calculated similarly. The path similarity (\( \text{PS} \)) must be calculated to all leaves of the global and local schema and multiplied by the return value of function \( \text{LS} \) (WordNet) for the given leaf elements. Finally, the average value of the best 25% (or 50%) is calculated and designated as leaf context structural similarity.

Finding the paths from the leaves to the two entities requires a number of steps as follows:

\[
\text{Steps}_{\text{leaves..paths}}(C_i, G_j) = \sum_{\text{leaf}(C_i)} \sum_{\text{leaf}(G_j)} \text{length}(P_1) \times \text{length}(P_2)
\]

where \( \text{length}(P_1) \) (\( \text{length}(P_2) \) respectively) is the length of the path from the given local schema entity (global schema entity respectively) till a given leaf and \( \text{leaf}(C_i) \) (\( \text{leaf}(G_j) \) respectively) represents the number of all paths to leaves from a given local (global) schema entity \( C_i \) (\( G_j \) respectively).

To determine the best 25% (or 50%) of the results of the function \( \text{LS} \) and \( \text{PS} \) the results must also be ordered. This means an additional inserting cost which can be approximated by the log function. The overall cost of leaf similarity calculation is the following:
Steps_{ancestors}(C_i, G_j) = \sum_{i=1}^{G} \sum_{j=1}^{C} \text{length}(P_i) \times \text{length}(P_j) + \text{leaf}(C_i) \times \text{leaf}(G_j) + 1.39 \times \log_2 \text{leaf}(C_i) \times \text{leaf}(G_j) \quad (\text{Exp. 1.32})

where similarly to the expression used at the child context  
\text{leaf}(C_i) \times \text{leaf}(G_j) + 1.39 \times \log_2 \text{leaf}(C_i) \times \text{leaf}(G_j) \quad \text{denotes the sorting cost of the evaluated leaf-pairs into an ordered set preparing for the selection of the best 25% (50%).}

For the overall method the calculations of ancestor, leaf and child context must be performed for all possible pairs of complex types of the global and the local schema. Extending it with the preparation steps (tree construction and retrieving of WordNet) the number of expected steps is the following:

\sum_{i,j} \text{Steps}_{ancestors}(C_i, G_j) + \text{Steps}_{child}(C_i, G_j) + \text{Steps}_{leaves}(C_i, G_j) \quad (\text{Exp. 1.33})

In the next section I compare the calculation complexity of the different approaches.

### 3.4 Comparison of the complexity of approaches

The first presented method, the approach NTA requires the following number of steps for preparation:

\sum_{i,j} \text{Steps}_{NTA, prepare}(C_i, G_j) = \sum_{i=1}^{G} \sum_{j=1}^{C} \text{Attrel}(C_i) + \text{Attrel}(G_j) + 1 + \text{Attrel}(C_i) + \text{Attrel}(G_j) + 1 \quad (\text{Exp. 1.34})

and the following number of steps for the actual evaluation of similarity values:

\sum_{i,j} 1 + (\text{Attrel}(C_i) + 1) \times (\text{Attrel}(G_j) + 1) + \text{Attrel}(C_i) + \text{Attrel}(G_j) + 1 \quad (\text{Exp. 1.35})

One can see that besides the number of the complex types in the global and the local schema (G and C respectively) the complexity of the algorithm also strongly relies on the number of terms attached to complex types and on the number of attributes connected to the complex types. Thus before further evaluation the expected number of related terms and attributes should be estimated. Based on my experiences on evaluating schemas from normal complexity (schemas accruing in everyday integration problems among standard enterprise systems) the average number of attached terms is set to 4 and the average number of connected attributes is set to 5. The tree representation of the presented example is shown in figure 1.11.

![Figure 1.11 Tree representation of the sample schema for algorithm complexity demonstration](image-url)
For the sake of simplicity I use the same schema as a local and as a global schema in the demonstration of calculation complexity in this chapter.

The calculation of similarity values of complex types for these schemas (consisting of 5-5 complex types) means

\[ \text{Steps}_{\text{NTA \_Prepar}} (C, G) = 5 \times 5 + 5 \times 5 + 5 + 5 \times 4 + 5 \times 4 = 100 \] steps of preparation, and

\[ \text{Steps}_{\text{NTA \_Perform}} (C, G) = 25 + 25 \times 5 + 25 \times 5 \times 5 = 1275 \] steps of similarity calculation for this estimated size of schemas. Hence the number of overall steps of the NTA approach for these schemas is:

\[ \text{Steps}_{\text{NTA \_Sample}} (C, G) = 100 + 1275 = 1375 . \]

The next presented approach was the similarity flooding. It requires

\[ \text{Steps}_{\text{Sim \_Flooding}} (C, G) = \text{Steps}_{\text{OIM}} (C, G) + \text{Steps}_{\text{Init \_map}} (C, G) + \text{Steps}_{\text{PCG \_Create}} (C, G) + \text{Steps}_{\text{Flooding}} (C, G) \]

steps to evaluate overall similarity of complex types.

The steps to create the OIM graph representation of both schemas is as follows:

\[ \text{Steps}_{\text{OIM}} (C, G) = 2 \times \left( \sum_i |\text{Attr}(C_i)| + \sum_j |\text{Attr}(G_j)| + |C| + |G| + Y_c + Y_G + 3 \right) =, \quad (\text{Exp. 1.36}) \]

\[ 2 \times (5 \times 5 + 5 \times 5 + 5 + 5 \times 6 + 6 + 3) = 150 \]

where values \( Y_c \) and \( Y_G \) should have been estimated. These are data types e.g. string, boolean, integer, float, date and time, so the number of existing data types both in the global and the local schema and is estimated now with 6 for this sample.

The steps for the initial mapping is calculated as follows:

\[ \text{Steps}_{\text{Init \_map}} (C, G) = \left( 3 + 2 |C| + \sum_i |\text{Attr}(C_i)| + 2 Y_c \right) \times \left( 3 + 2 |G| + \sum_j |\text{Attr}(G_j)| + 2 Y_G \right) =, \quad (\text{Exp. 1.37}) \]

\[ = (3 + 2 \times 5 + 2 \times 5 \times 5 + 2 \times 6)^2 = 5625 \]

The PCG graph creation costs

\[ \text{Steps}_{\text{PCG \_Create}} (C, G) = 2 \left( |C| + \sum |\text{Attr}(C)| + Y_c \right) + \left( |G| + \sum |\text{Attr}(G)| + Y_G \right) + \sum |\text{Attr}(C)| + \sum |\text{Attr}(G)| + \sum |\text{Simple \_Attr}(C)| + \sum |\text{Simple \_Attr}(G)| = 2(5 + 5 \times 5 + 6)^2 + (5 \times 5) + 21 \times 21 = 3658 \]

where the overall number of simple type attributes was simply counted for the given test schema. (Note, that this counting was supported by appropriate implementation used mainly by the estimation of computational requirements of the WordNet approach, see later)

Based on the statements in the corresponding paper of the similarity flooding [57] 6-8 iterative steps are needed to evaluate final results at the flooding (this number is dependent on the selected convergence formula and input and output schema complexity) at schemas at these sizes. Hence I set the number of iterative steps to equal 7. Based on these estimated values and substituting the average number of the attributes of complex types and the number
of complex types from my example, the computational cost of the flooding can be calculated as follows:

\[\text{Steps}_{\text{flooding}}(C, G) = \sum_{\text{steps}} \text{Nodes}_{\text{PCG}}(C, G) = \sum_{\text{steps}} \left( 2 \left( \| C \| + \sum_{i} \| \text{Attr}(C_i) \| + Y_c \right) \right) = \left( \| G \| + \sum_{i} \| \text{Attr}(G_j) \| + Y_G \right) \]

\[= 7 \left( 2 \left( 5 + 5 + 5 + 6 \right)^2 \right) = 18144 \quad \text{(Exp. 1.39)}\]

The total number of steps can be calculated as follows:

\[\text{Steps}_{\text{sim...flooding}}(C, G) = \text{Steps}_{\text{OMI}}(C, G) + \text{Steps}_{\text{Init..map}}(C, G) + \text{Steps}_{\text{PCG}}(C, G) + \text{Steps}_{\text{Flooding}}(C, G) =
\]

\[= 150 + 5625 + 3658 + 18144 = 27577 \quad \text{(Exp. 1.40)}\]

Hence the overall number of steps of the similarity approach applied to my example schemas is significantly greater than the number of steps required by the NTA algorithm.

\[\text{Steps}_{\text{sim..flooding..Sample}}(C, G) = 27577 \]

The last presented approach was the so called WordNet approach. Substituting the values of my example into the equations for the steps of the initial tree construction and retrieving WordNet we get the following results:

\[\text{Steps}_{\text{TreeConstr}}(C, G) = \sum_i \| \text{Attr}(C_i) \| + \sum_j \| \text{Attr}(G_j) \| + \| C \| + \| G \| = 5 \cdot 5 + 5 \cdot 5 + 5 + 5 + 5 = 60 \quad \text{, and}\]

\[\text{Steps}_{\text{WordNET}}(C, G) = \left( \| C \| + \sum_i \| \text{Simple\_Attr}(C_i) \| \right) \left( \| G \| + \sum_j \| \text{Simple\_Attr}(G_j) \| \right) =
\]

\[= (5 + 21)^2 = 676 \quad \text{(Exp. 1.41)}\]

The rest of the calculations are the evaluation of similarities in the ancestor, child and leaf context. Thus \(\text{Steps}_{\text{ancest}}(C_i, G_j) = \text{length}(P_1) \times \text{length}(P_2)\) the average length of the path from the given node till the root should be estimated. Unfortunately this depends strongly on the given schemas and can not be estimated based on the number of complex types in the schema and average number of connected attributes per complex type. However some values can be assigned to my example. Suppose that the maximal depth in my schema forest is 4 and there is 1 complex type at distance 3, 2 complex types at distance 2 from the root element and 1 complex type is the root element itself. So the average length of path is estimated by 2, 4 and the average for the expected number of steps for evaluating the ancestor similarity of two given complex types is:

\[\text{Steps}_{\text{ancest}}(C_i, G_j) \approx 2.4 \times 2.4 = 5.76 \quad \text{(Exp. 1.42)}\]

Please note, that special implementations were developed to easily calculate the average length of different paths in the graph structure. With these I was able to easily estimate the values for this sample schema and for all of the test schemas used at the performance analysis in Chapter 3.5 later.

To be able to estimate the computational complexity of the calculation of child context similarity the average number of children of complex types should be set. For both the local schema tree and the global schema tree this is exactly 5, because every complex type has 5 attributes in my example.

Hence the number of steps for calculating child context similarity is:

\[\text{Steps}_{\text{child}}(C_i, G_j) = \| \text{child}(C_i) \| + \| \text{child}(G_j) \| + \| \text{child}(C_i) \| \times \| \text{child}(G_j) \| \times 1.39 \log_2 \| \text{child}(C_i) \| \times \| \text{child}(G_j) \| =
\]

\[= (5 \times 5) + (5 \times 5) \times 1.39 \log_2 (5 \times 5) = 186,374 \quad \text{(Exp. 1.43)}\]
The leaf similarity can be calculated in a number of steps as follows:

\[
\text{Steps}_{\text{leaves}} \left( C_i, G_j \right) = \sum_{\text{leaf}(C_i)} \frac{\text{length}(P_i) \ast \text{length}(P_j) + \text{leaf}(C_i) \ast \text{leaf}(G_j) \ast \text{log}_2 \left( \frac{189}{10} \right)}{\text{leaf}(C_i) \ast \text{leaf}(G_j)} = 10.6^2 \ast 2.96^2 + 10.6^1 \ast 1.39 \ast \text{log}_2 \left( 10.6^1 \right) = 2049.858
\]

(Exp. 1.44)

where \( \text{leaf}(C_i) \) and \( \text{leaf}(G_j) \) were estimated with the average number of leaves under the nodes in my sample tree, which is:

\[
\text{leaf}(C_i) = \text{leaf}(G_j) = \frac{21 + 18 + 9 + 5}{5} = 10.6,
\]

(Exp. 1.45)

and the average length of the paths to the leaves is estimated with:

\[
\text{length}(P_i) = \text{length}(P_j) = \frac{2 \ast (5 + 5 + 4 + 4 + 3) + 3 \ast (5 + 4 + 9) + 4 \ast (5 + 4) + 5 \ast 5}{21 + 18 + 9 + 5} = 2.96.
\]

(Exp. 1.46)

These calculations were also facilitated by appropriate implementations also mentioned above of course.

Substituting the estimated number of steps of preparations and calculating ancestor, child and leaf context to each node into expression 1.47 the overall performance is approximated as follows:

\[
\text{Steps}_{\text{Combined}} \left( C, G \right) = \text{Steps}_{\text{treeContext}} \left( C, G \right) + \text{Steps}_{\text{WordNet}} \left( C, G \right) + \sum_{t_p} \text{Steps}_{\text{ancestor}} \left( C_i, G_j \right) + \text{Steps}_{\text{child}} \left( C_i, G_j \right) + \text{Steps}_{\text{leaves}} \left( C_i, G_j \right) = 60 + 676 + 5 \ast 5 + (5.76 + 186.374 + 2049.858) = 56786
\]

(Exp. 1.47)

So comparing the results for the 3 different approaches we get the following order:

\[
(\text{Steps}_{\text{NTA}} \left( C, G \right) = 1375) < (\text{Steps}_{\text{SimFloodin}} \left( C, G \right) = 27577) < (\text{Steps}_{\text{Combined}} \left( C, G \right) = 56786)
\]

(Exp. 1.48)

, where one can see, that the estimated cost of my proposal is significantly smaller than the cost of other approaches.

However algorithm complexity is only one major component by analyzing the performance of an approach. The accuracy and error-rate of a method is even more important. Hence the importance of this chapter can not be interpreted without taking into account the efficiency of the proposals described in Chapter 2.3 above.

### 3.5 Complexity of large schemas

To estimate and analyze working cost i.e. number of run-time steps for larger schemas more samples were implemented. As the reader could see in the previous chapter, calculation of the expected number of run-time steps is not an easy method especially for custom (e.g. real-life) scenarios. On the other hand computational costs of real schemas can also be estimated with that of some artificial schemas of the right size. Thus specific samples were created so that the required parameters e.g. average path length to the root, or to the leaves can be easily calculated. These schemas conform to strict rules as follows:

- every intern node (complex type having at least one attribute with a range complex type as well) has the same number of children (i.e. the same number of attributes with a complex range),
- every node has the same number of attributes (in other worlds, the summary of complex and simple attributes for every nodes is constant),
- all leaves of the tree can be found at the same level (in other words, the distance of complex types having only simple attributes from the root element is always the same) and
- every complex type has the same number of attached terms.
Upon these the complexity of the sample schemas can be described by the following parameters:
- the number of branches represents the number of children nodes for intern complex types,
- the number of deepness shows the distance of leaf nodes from the root,
- concept attribute is the number of attributes connected to complex types and
- terms depicts the number of associated terms of complex types.
The implemented samples have the following parameters, see table 1.20.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Branches</th>
<th>Deepness</th>
<th>Attributes</th>
<th>Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example1</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Example2</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>Example3</td>
<td>5</td>
<td>5</td>
<td>12</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 1.20 Parameters of implemented sample schemas

Please note, that besides the newly implemented schemas called Example1-3, the presented sample of the previous chapter – nominated as Sample in the table - is also used by these calculations. Because that sample does not follow the rules defined above, the parameters describing complexities like branches and deepness had to be approximated by calculated, not integer numbers.

Graphical representation of Example1 can be viewed in figure 1.12. Further schemas can be found in appendix in figures 4.1 and 4.2.

![Graph representation of Example1](image)

Figure 1.12 Graph representation of Example1

To be able to calculate the number of expected steps the following values were also approximated:
- number of different simple types and flooding iterations for approach similarity flooding and
- average number of children and leaves, and average length of path to the root and to the leaves for calculating the steps of the combined matcher (approach WordNet).
The estimated number of run-time steps for these scenarios for every approach is presented in table 1.21 and figure 1.13 below.

<table>
<thead>
<tr>
<th>Test scenario</th>
<th>NTA</th>
<th>Sim. Flood</th>
<th>WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample vs Sample</td>
<td>1375</td>
<td>27577</td>
<td>56786</td>
</tr>
<tr>
<td>Example1 vs Sample</td>
<td>2451</td>
<td>33663</td>
<td>95619</td>
</tr>
<tr>
<td>Example1 vs Example1</td>
<td>4396</td>
<td>57335</td>
<td>252207</td>
</tr>
<tr>
<td>Example2 vs Sample</td>
<td>59471</td>
<td>495365</td>
<td>5865024</td>
</tr>
<tr>
<td>Example1 vs Example2</td>
<td>91809</td>
<td>847019</td>
<td>12086636</td>
</tr>
<tr>
<td>Example2 vs Example2</td>
<td>1921843</td>
<td>16930913</td>
<td>61037758</td>
</tr>
<tr>
<td>Example3 vs Sample</td>
<td>2152256</td>
<td>20763658</td>
<td>361249806</td>
</tr>
<tr>
<td>Example1 vs Example3</td>
<td>3882655</td>
<td>35530560</td>
<td>609118736</td>
</tr>
<tr>
<td>Example2 vs Example3</td>
<td>81375634</td>
<td>1204698630</td>
<td>3762299563</td>
</tr>
<tr>
<td>Example3 vs Example3</td>
<td>3448208988</td>
<td>50469880392</td>
<td>2,31949E+12</td>
</tr>
</tbody>
</table>

Table 1.21 – Estimated number of run-time steps for different scenarios

To create some scenarios of different complexity all combinations of the test samples were tested. This means 10 different schema matching problems: 4 problems comparing entities of schemas from the same type and 6 comparing schemas having different parameters and structures.

Figure 1.13 – Estimated number of run-time steps for different scenarios

The estimated values were also validated with experiments. As already mentioned above, these samples were implemented and the 3 different algorithms were executed on them. Running the applications made possible to count the exact number of required steps and compare it with the estimated values. Furthermore actual run-time costs i.e. the required time for performing the calculations, was also measured and compared with the expected complexity of different scenarios using the 3 special approaches. Table 1.22 shows experimental result which are slightly different from the estimated values. The sign “N/A” stands for experiments which could not be executed because of the large size of the given test input and the complexity of the given approaches. Specific to my implementations the operational memory of the test computer was not sufficient to execute the given tasks. However, even if the memory had fit out (for example by a modified version of the implementation) the actual run-time would have been too much to run the test in an
acceptable time. Nevertheless, optimization of given implementations on the code level on a specific compiler and under a given environment is not in the scope of my Thesis.

<table>
<thead>
<tr>
<th>Test scenario/Approach</th>
<th>NTA</th>
<th>Sim. Flood</th>
<th>WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample vs Sample</td>
<td>1375</td>
<td>18827</td>
<td>54256</td>
</tr>
<tr>
<td>Example1 vs Sample</td>
<td>2451</td>
<td>32837</td>
<td>113933</td>
</tr>
<tr>
<td>Example1 vs Example1</td>
<td>4369</td>
<td>57371</td>
<td>240700</td>
</tr>
<tr>
<td>Example2 vs Sample</td>
<td>59471</td>
<td>483215</td>
<td>5821611</td>
</tr>
<tr>
<td>Example1 vs Example2</td>
<td>91809</td>
<td>847037</td>
<td>12024888</td>
</tr>
<tr>
<td>Example2 vs Example2</td>
<td>1921843</td>
<td>16930931</td>
<td>622136464</td>
</tr>
<tr>
<td>Example3 vs Sample</td>
<td>2152256</td>
<td>20255808</td>
<td>380147160</td>
</tr>
<tr>
<td>Example1 vs Example3</td>
<td>3882655</td>
<td>35530578</td>
<td>790026787</td>
</tr>
<tr>
<td>Example2 vs Example3</td>
<td>81375634</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Example3 vs Example3</td>
<td>344820888</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 1.22 – Measured number of run-time steps for different scenarios

Corresponding with the required number of steps actual execution times were also measured to validate the correctness of my method. Because the measure times were in the expected order of magnitude comparing with the expected number of computational steps, the performed experiment has validated my proposal for estimating and comparing computational requirements of schema matching approaches. (Note that more accurate estimation of run-time costs is hardly possible without the detailed examination of given hardware and software environments, which is out of the scope of my work.) See results in table 1.23.

<table>
<thead>
<tr>
<th>Test scenario/Approach</th>
<th>TTA</th>
<th>Sim. Flood</th>
<th>WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample vs Sample</td>
<td>0,03</td>
<td>0,06</td>
<td>0,08</td>
</tr>
<tr>
<td>Example1 vs Sample</td>
<td>0,03</td>
<td>0,06</td>
<td>0,08</td>
</tr>
<tr>
<td>Example1 vs Example1</td>
<td>0,03</td>
<td>0,08</td>
<td>0,11</td>
</tr>
<tr>
<td>Example2 vs Sample</td>
<td>0,08</td>
<td>0,63</td>
<td>1,00</td>
</tr>
<tr>
<td>Example1 vs Example2</td>
<td>0,08</td>
<td>1,11</td>
<td>1,95</td>
</tr>
<tr>
<td>Example2 vs Example2</td>
<td>0,92</td>
<td>25,94</td>
<td>98,23</td>
</tr>
<tr>
<td>Example3 vs Sample</td>
<td>3,56</td>
<td>41,91</td>
<td>101,88</td>
</tr>
<tr>
<td>Example1 vs Example3</td>
<td>2,52</td>
<td>406,48</td>
<td>159,06</td>
</tr>
<tr>
<td>Example2 vs Example3</td>
<td>40,52</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Example3 vs Example3</td>
<td>2264,84</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 1.23 – Measured run times of different scenarios (seconds)

As the reader can see the number of run-time steps is significantly lower for my approach than it is in the case of other approaches. Remember, that together with the conclusion of the previous chapter (namely that the efficiency measured with performance indicators i.e. precision and recall of my approach is equal or greater than it is by other approaches) my approach seems to be a solid method for solving the schema matching problem in SOA integration scenarios.

Conclusion and future work

This part has presented an approach for identifying semantically related concepts in a SOA integration scenario. Detecting semantically related complex types of services and the global schema is crucial before designing collaborative business processes. The mapping of services in the integration scenario should be processed on semantically related complex types exclusively.

The complex types of the service are enriched with some terms before they are analyzed. My method relies on characterizing identifiers names, terms and attributes which help
calculate the semantic distance between complex types of service and the global schema. My method automatically detects semantically related complex types.

The proposed approach is rather sensitive for the selection of the right values for weights $w_i$. In my experiments I have always checked the results manually as well and made more iteration steps to find the proper values for the weights. This can hardly be done for larger schemas. The circumstances of a specific integrations scenario could be evaluated to determine the right values for the weights. Another solution could be the partitioning of schemas into smaller sets of complex types but this raises a lot of questions as well e.g. how to identify the smaller sets?

Although my method identifies the complex types participating in transformations, it gives no promotion for the creation of transformations. Semantic relationships between the elements of complex types may also be evaluated in future work. This can result some proposals for creating specific transformation rules. Furthermore creation of proper transformations may also be reliable under some specific circumstances and hence could be supported by methods and tools in the future work.

There is also a business ontology in most of the presented semantic frameworks in the literature. This usually also serves as a common thesauri for service taxonomies and also describes service capabilities and behavior. Attaching such references to the description of services may also support the better understanding of its input and output data concepts making the recognition of semantically relevant pairs easier. On the other hand ontologies of different business domains and corresponding standards and techniques for semantically enriching the service descriptions are all already available. Hence, a future extension of this work should consider identified capabilities, pre- and post- conditions of services with an ontological background.

**Applicability of results**

The methodology and presented algorithms in Chapter I.1 can be used to compare entities of data schemas and to define the strength of connections among them also in productive systems. Because standard enterprise applications provide services with interfaces having data structures of about 5-100 entities max and described mostly by XML Schema standard, my prototype implementation provides result in a few seconds – see the run-time estimations and experiments in Chapter I.3 – for two input schemas of services. Furthermore complex schemas, like a global schema are also comparable with simpler (10-20 entities) service interfaces using my prototype in a reasonable time. Nevertheless my reference implementation can be extended and exchanged by an implementation in specific enterprise application environment based on the detailed description of the methodology and algorithms in Chapter I.1. After that it can also be applicable in the comparison of entities in a custom industrial environment on a solid framework of standard enterprise applications.

The solution presented in Chapter I.2 can be used to determine and compare the accuracy of specific schema matching approaches. The expected accuracy level can be estimated and algorithms can also be customized (optimized) for given productive environment and given input schemas of enterprise applications. This can be a great help, by deciding to use one specific schema matching approach during a given enterprise application integration scenario or even during a specific analysis of given service interfaces and global schemas of a domain.

Based on the Thesis I.3 we can estimate the computational requirements of different schema matching approaches for given input complexity. This facilitates the sizing of hardware components for schema matching solutions, which is an important task by putting software solutions in practice in a productive environment. Furthermore using my approach the approximate run-time of specific algorithms for given input data in a given hardware environment can also be estimated.
Part II

SOA based Web service adaptation in Enterprise Application Integration

Abstract

In this part I propose a technological solution and architecture for the adaptation of standard enterprise services into SOA integration scenarios providing support for applying data transformation to bridge data incompatibilities. To be able to evaluate my approach three other possible solutions are designed and implemented. An in detailed analytic and experimental comparison of the approaches is also presented. Both theoretical analysis and experiments proved that my proposal offers a solid run-time realization with predictable resource need while there is no significant increasing in response time or decreasing in throughput during the execution.

Introduction

Organizations have early recognized the applicability of SOA methodology and technologies for solving integration issues [40, 72, 89]. Orchestrated processes can realize complex business processes – the aim of the orchestration is to create the requested new business functionality – and participating services are already developed interfaces of enterprise application systems. However there are several difficulties and preconditions in designing and applying such an integration solution in a business environment.

Different systems and their services provide heterogeneous data semantics. The provided data can differ in simple naming conventions e.g. name of a person entity can be depicted as “guest” by the hotel booking service and as “customer” by the flight booking service. It is obvious that both services cannot be invoked by the same entity naming convention. This means, that we cannot choose a “standard” representation for this entity for a whole trip organizing service summoning the previous two – instead of using the same entity (name of a customer) the request data invoking the two services must be different. There can also be other differences between provided data semantics e.g. differences in data structure or messaging protocol. These must be handled by several transformations, see [52, 54]. However applying transformations can slow down the designing issues and strongly influences the non-functional capabilities e.g. response time or throughput of the new complex processes.

In this Part I present a methodology and technical solutions for creating the necessary transformations and present an approach which conserves the applicability of solid and standard SOA technologies and tools while the run-time performance of the system remains acceptable. The focus of this part is on the proposed run-time framework and on the prediction of non-functional capabilities.
Related work

The theoretical approach of the data heterogeneity problem called schema matching was introduced in details in the chapters of Part I. Most of these works concentrates on the design time tasks of schema matching, mapping and the creation of transformations and do not cover the run-time aspects completely. For example see [1, 30, 41, 92], presented already in the related work section within Part I. Designing and implementing run-time tools and instances is also a key aspect of EAI in a SOA environment. Furthermore detailed analysis of run-time performance should be carried out to be able to use the created mappings in real life scenarios e.g. in the productive enterprise information systems of companies.

Data heterogeneities in enterprise application integration

Numerous researches are dealing with data heterogeneity during enterprise application integration. To overcome the problem of data incompatibilities the meaning of entities is examined and compared instead of the name of the concepts. These approaches add semantic descriptions to the operations of services based on specific semantic taxonomies. For example see [10, 19, 29, 40, 59]. However most of these works do not focus on the execution of semantic (or semantically enriched) service or run-time performance prediction and optimization.

A solid framework is presented in [29] for solving interoperability problems among SMEs. The paper describes the model for performance evaluation used in the project called Abilities [42]. The paper also provides some guidance to data transformation. A reconciliation engine is responsible for the proper execution of the translations on the UBL documents and an ontology is used for storing required knowledge about document classification, content, structure, etc. However detailed analysis on more test scenarios is not included. An important difference between this work and mine is that mine does not require a complicated domain ontology. A simple complex schema (for example an XML schema) is enough to define and execute transformations in my run-time solution.

The authors of [10] present a method creating mediators over service wrappers. Besides we get no information about the capabilities of the wrappers (for example, are there any transformation applied to align the interface of wrappers to a global schema?) the usage of an ontology is required in this approach as well. The framework is able to answer complex queries and was successfully applied in two contexts: bioinformatics [13] and digital libraries [12]. However the node representation is in a multi-level structure where each node of the P2P network at the second level should be updated if a new endpoint is added to the system. Contrary to my business bus based approach which requires the alignment of all participant service only to one common global schema this method requires much more maintaining costs. This hinders its application in real life integration scenarios where high run-time performance is one of the main concerns.

An information and workflow infrastructure for common tasks of marketing and public relations is presented in [40]. The platform called MIKSI is a SOA based integration framework built on widely adopted open standards like BPEL for the process composition, RDF for message representation and SOAP for the transportation. However there is no detailed analysis and prediction of the process run-time. On the other hand the translation between plain text, RDF and HTML probably results in a solution with lower throughput than my proposal.
The important problem of managing communication of participants in the Supply Chain Management (SCM) area is addressed by the work presented in [19]. The authors also deal with the bridging of data representation mismatches. Therefore an ontology called Onto-SCM is implemented for the SCM domain, which is the cornerstone for the semantic information integration. The focus of this work is set to the creation of this ontology. Contrary to this I suppose the existence of proper candidates for all kinds of required schemas in the business domain (for example see [91], [63]). Hence constructing a new taxonomy or ontology is out of scope form the point of view of my work.

**Predicting and optimizing of non-functional QoS parameters in SOA based integration**

There are also numerous researches in the literature about predicting non-functional QoS parameters (e.g. system throughput and response time) of composite services [17, 26, 42, 44, 80]. Some of them are based on the usage of intelligent agents to evaluate and increase the reliability and performance of orchestrated processes, for example see [6, 18]. Most of these approaches do not deal with the existence of interoperability problems like input-output data incompatibilities.

In [80] the authors present a solution for increasing QoS for composite services. There is a complex monitoring process for the running instances which is also able to detect violated behavior from services’ contract. Services with any kind of malfunctions (e.g. services with increased response time or decreased availability ratio) can be dynamically replaced by compatible services performing better within the given time. The main idea behind the implementation is the extending of the BPEL code with specific comment elements carrying extra information for enhanced monitoring capabilities. Although this approach definitely leads to better performance it is hardly able to predict and improve the run-time performance of composite processes in the integration scenarios at enterprise application integration. Enterprise application integration scenarios typically do not have replaceable services in their different systems, and a given service of a given application system have to be used at a given point of a collaborative business process.

Similarly to the previous work in [17] we meet a wider set of applied services. All Web services from UBRs, to service portals and search engines are targeted for gathering information about their QoS parameters. However as a result we get only an extended statistic about the current state of all kind of services on World Wide Web. The information is neither used in process composition nor in prediction and analysis of the QoS parameters of composed services.

One of the most interesting aspects of the approach presented in [44] is definitely the adaption of a genetic algorithm for increasing the performance of composite processes. Proxy services are used to enable dynamic service selection and QoS monitoring. Although these proxy services may seem very similar to my encapsulated (proxy) service, here are some important differences: in my architecture I use only one proxy service providing a common interface for all instances of applied services. Furthermore my proxy also enables the execution of transformations glued to the standards services. However the presented service discovery and run-time monitoring capabilities could be a reasonable extension of my work in the future.

A QoS-aware middleware for Web service composition is presented in [18]. The so called AgFlow system architecture relies on ontologies for improving QoS of composite processes and both local and global optimization policies are considered. The whole QoS problem, prediction and optimization are fully covered by deep formal analysis in this work as well. However the run-time execution of data transformation and mappings is not solved.
The Web Services Agent Framework is applied in [6] for extending service publication and selection with appropriate QoS parameters. The approach makes possible for interoperating participants to determine each other’s service quality and trustworthiness which is reasonable if dynamic service selection and binding is available during the composite process implementation and execution. Unfortunately the latter is not the case with enterprise application integration scenarios. Nevertheless extending of the rather syntactic description of enterprise application services with semantic information as presented in the paper is an area of possible future work.

Resource management and performance analysis in run-time environment

Another current topic is the dynamic resource management to optimize resource allocation in service oriented architectures over grid systems [49, 64]. The work of the run-time engine is optimized in [64] to increase throughput of business process execution for processes composed of Web services. The method is monitoring the state of the run-time engine continuously and it is also able to dynamically reconfigure the system by changing the resource allocation in the grid system. This may improve the performance independently from the language of the workflow but requires a complex grid system of computers which is hardly feasible in case of enterprises. Moreover companies usually rely on solid tools of commercial technologies (like standard BPEL run-time engine) and do not intend to modify or replace them because of unique solutions. Furthermore this proposal can only react on the changes of system load and is not able to predict run-time performance for given load of given composite business processes.

The authors of [49] focus on improving the overall QoS of composite service workflow after all the individual component services are selected into a composite process. To achieve this, dynamic resource allocation and service replication is performed by the presented platform. The novel part of the work is the extensive tracing of service request traversal through the workflow instance tree and monitoring of service time, transition probability and replication overheads etc. Nevertheless, such complex analysis is probably redundant in the case of enterprise application integration because run-time performance can be predicted by a simpler model as well, and services of composite business processes cannot be replaced at all.

The authors of [71] use a stochastic model to analyze the performance and reliability bottleneck of composite processes. A continuous-time Markov chain formulation including failure states is developed to compute both performance and bottlenecks. However, this approach is based on the run-time monitoring of participating services and consumes a lot of resources for the dynamic prediction of overall performance and for the modification of the composed process at run-time. Because my integration approach focuses on the integration of enterprise applications, unreliable and instable services from the internet are not involved and resource requirements of services and processes can be predicted directly upon the specifications provided by reliable software vendors. Furthermore if a given function of a given enterprise application is required at a given point of the orchestrated process it simply cannot be replaced since there is probably no other available service performing the same functionality in the same system.

Cardoso [21] states that highly complex composite services tend to be less flexible and may cause decreased performance compared to optimized and carefully designed processes. The paper analyzes the control flow complexity of processes and proposes several methods for simplification. Thus this approach can extend my work by the structural analysis of composite services and can help us to predict the resource consumptions of complex processes On the other hand, data heterogeneity should also be considered and resolved as proposed in this Thesis.
The vieDAME framework presented in [58] provides a complete solution for performance optimization and for overcoming interoperability problems. The presented architecture contains several components for example for intercepting service invokes, for monitoring operation performance, for storing and executing transformations, for communicating with the run-time engine etc. However overall performance of a configuration is not predicted and bottlenecks remains hidden from system operators. Hence the application of my performance analysis methods can be a useful extension of this work.

II.1 SOA integration methodology and architecture

Vendors of enterprise applications are already prepared for SOA. Information systems offer their capabilities in the form of services. The service interfaces are mostly matched to worldwide standards which results in easy to use services in a technical sense.

Unfortunately this is not enough for creating real, working processes because services of different application systems offer their capabilities in heterogeneous data semantics [54]. This means that the output data provided by a participating service cannot be fed directly into another service as an input. Data transformation must be applied within the chain of services to adjust input and output data of different services to each other.

There are many ways to adopt the required transformations into a composite process. The simplest solution is the following:

1. Create the process ignoring data heterogeneity problems. Participating services are composed by defining their relations e.g. invoke order, logical expressions for branches etc.
2. Search for incompatibilities caused by data heterogeneity. Design and apply transformation at every service invoke where entity mismatches appear.

Although this approach is easy to understand and can work at every kind of data heterogeneity, it offers a non-optimal solution. The unique transformations applied before service invokes are totally customized for the given environment. Hence they cannot be adopted to another process or not even for another invoke of the same service within the same process. This implies that by updating the process or by creating a new one, the possibility of transformation re-using is excluded.

Instead of creating custom transformations by process orchestration, applying a global data schema can ensure data compatibility on the process level. The global data schema can define every related entity of a given business domain regularizing the proper usage of given business concepts. For example in the global schema we can define the usage of term guest for the tourism domain excluding the usage of synonyms like client, customer, etc. in the domain. Thus applying regularizations defined by global schemas can prevent data mismatches on the process level. One application methodology for this scenario can be the following:

1. Define the global schema for the domain – determine the structural and linguistic terms for every possible entity of a given domain.
2. Define transformations to every participating service. The transformation(s) are strictly connected to the service, which results in an encapsulated service behaving the same way as the original but communicates using the right concepts and terms of the global schema.
3. Design the composite service (process) defining the relationships among participating services.
This solution already makes possible to re-use the defined transformations. The encapsulated services containing the necessary transformations are applied in new processes or in newer version of the same process. The next chapter presents the above steps detailed.

Note that the main advantage of this approach is that the transformations must be created to each participating service to each domain only once. The encapsulated service containing the transformations and the original service can be created on demand (right before we need it at a given point of the orchestration for the first time) or all participating services of given enterprise applications can be encapsulated before starting actual SOA based integration. If large enterprise software vendors agree on given global schemas, the creation of encapsulated services can already be handled globally by them preventing any kind of data heterogeneity by service orchestration in real integration scenarios. Moreover software vendors have the necessary experience and developer teams to easily overcome the task of transformation creation and fitting service schemas to widely used global schemas. For example the most applied global schemas in the field of on-line business are [3, 91].

1.1 Preparing for orchestration

The aim of this step is to align the service input and output to the global schema. In other words we would like to create a service that works in the same way as the original service but communicates with the concepts of the global schema towards the outside world. More precisely, the applied interfaces should conform to the global schema but the original service of the given application should be invoked in the background.

For example let our flight reservation service require the name of the client in a complex data structure containing two fields: firstname and lastname. In contrast to this let the hotel booking service require one field containing the full name (firstname and lastname separated with a space) called guestname. The process (composite service) communicates with the global schema concepts and the corresponding domain of the global schema called tourism standardizes the person entity in the separated firstname, lastname form. In the case of the flight service the process invoke data can simply be forwarded to the service as a service call. However, invoking the hotel booking service require a transformation to be glued to the service input which concatenates the firstname and lastname fields. After the transformation, the hotel booking service receives the required input in the right form and is able to serve the request.

Another direction of transformation may also be necessary. For example consider that after a successful booking the flight reservation service responds with a confirmation message containing a field called smoking restrictions. However our global schema contains the same entity called smoking regulations. Hence the given concept of the service output should be renamed to ensure later reusability and common understanding at the process level. To perform this, a transformation is glued to the output of the flight reservation service.

Upon these there are 2 types of transformations:
- Down-cast transformations transform the concepts of the global schema into the required form of a given service input before invoking the service and
- Up-cast transformations transform the concepts of the service output into the corresponding form of the global schema right after the service response.

As already described above, there are numerous proposals to detect data incompatibilities, design and attach transformations in current researches, for example [17, 59, 65, 72, 78]. However a lot of them do not focus on the composite service deployment and run-time architecture. To account for these shortcomings I provide a possible solution (methodology and architecture) presented in the next few chapters.
1.2 Executing encapsulated services

Designing and attaching transformation rules to services does not mean that our composed service is ready to run right after the orchestration. The complex service should be deployed into a run-time environment. The run-time engine is responsible for accepting service invocations, create a process instance and perform the tasks of the given instance. These tasks are invoking participating services, evaluating logical expressions and performing any kind of business logic defined on the process level.

There are lots of solid process run-time environments available. However they are based on standards and are not prepared for executing transformations glued to participating services. To overcome this problem my proposed architecture ensures a standard service interface for the encapsulated services. This means, that the run-time engine can interact with the services in the standard way without knowing anything about its inner architecture and glued transformations.

The standard service interface is provided by a special proxy service. The proxy service intercepts the service invoke, determines which service is willing to be invoked, relays the service request and response to the given service and performs required transformations on the service input and output. So the proxy service acts as a single service towards the outside world. It represents the original services of enterprise applications, behaves the same way as them but communicates with the concepts of the global schema towards the process instance (e.g. towards the process run-time engine). The strict matching of transformations and service inputs and outputs are managed also by the proxy architecture.

This approach also implies that there is no need to modify the standard services and endpoints of enterprise applications: all data incompatibilities can be handled and prevented with aligning services to the global schema by designing and attaching up- and down-cast transformations. Furthermore this architecture is able to hide real services from the business partners, which makes it possible to create an additional private layer in the collaborative business process. Figure 2.1 shows one encapsulated service answering a service call.

The next chapter briefly presents some technical details of my solution.

1.3 Applied standards and technology

To be compatible with standard process run-time engines my solution follows current open standards. The de facto standard to design executable business processes is the Business
Process Execution Language (BPEL) [4]. Thus the focus was set to design processes in BPEL and apply solid BPEL run-time engines.

The main building elements of a SOA are usually Web services communicating mostly with Simple Object Access Protocol (SOAP) [81]. Created services of enterprise applications are also equipped with Web service interfaces. Thus my proxy implementation provides standard Web service interfaces towards the process run-time engine and is prepared for invoking standard Web services of enterprise information systems. The standard Web service invocation is for sure an available and widely used option for interacting with collaborative business partners in BPEL processes.

Standard Web services are described by the Web Service Description Language (WSDL) standard [83]. This defines one or more operations to the service containing requested input and output data structure per each operation. Hence my transformations must be connected to the operations. The number of attached transformations of a service is equal to the number of contained operations multiplied by 2 at maximum (1 up-cast transformation to the output and 1 down-cast transformation to the input can be defined to each operation). Every data incompatibility between the global schema and a given service input or output is covered with one complex transformation. In other words, simple transformation rules eliminating one given data mismatch of one given data concept are collected and performed by one complex transformation for each operation input or output.

To align to widely used standards the XSL Transformation standard was selected to realize transformations [87]. The expression power of the XSL transformation language is NP full, which implicates that it is able to cover every kind of incompatibilities. Furthermore there are lots of solid XSLT designer applications and run-time engines available which can be applied in my architecture.

The creation of the proxy service interface to each service makes it also possible to extend the WSDL description of standard enterprise application services with additional semantic information. This makes service discovery and selection easier which can be a great help for the service orchestration. See [3, 91] for further details. However this is out of the scope of my Thesis.

It is obvious that my solution provides some additional functionality by solving data incompatibility problems. But what cost should we pay for that? Apart from the additional design cost of transformations, proxy services must be created and transformations must be performed during the process run-time. Because the creation of proxy services is done in design time it does not really influence the performance of the running system. But what additional cost do we have for handling Web service invokes, relaying them to and from the Web services of enterprise applications and executing transformations? The next chapter provides some analysis about the run-time overhead of my architecture. Moreover some configuration and installation issues are also presented.

II.2. Performance Analysis in SOA-based integration

Current researches often deal with QoS parameters of complex services. There are two types of QoS parameters:

- Functional parameters describe service capabilities. In other words, they present what and how the service does. For example for our flight reservation service these parameters contain the fact that there are 3 operations search for flight, book flight, cancel booking and the search for flight operation requests departure, destination, and
- Non-functional parameters describe all kind of non-technical information about the service. These are availability, average response time, error-rate etc.

In this part of my Thesis the focus is set on the prediction and optimization of non-functional parameters. Description of the functional parameters of my encapsulated services can be found in [54].

There are working methods and algorithms to predict system parameters – like response time and throughput – of composite processes [56, 94]. Most of them are based on the idea of dividing the process into components whose performances can be directly evaluated and the parameters of the whole system can be calculated based on the defined connections (e.g. sequences, branches with logical conditions of predictable likelihoods, parallel run-time threads, etc.) among these components. Because the basic building blocks of processes are services they are the smallest undividable parts which actually determine the system performance. Thus I mainly focus on the performance analysis of the encapsulated service.

The questions to be answered are the following: How does the usage of transformations influence the response time of composite processes? What is the maximum throughput of a composite process in my architecture compared to other approaches? Is this the optimal solution for handling interoperability problems at process composition from the point of view of overall performance?

Response time and throughput are determined by resources of system hardware components. More precisely, services and processes require a given amount of resources and performance is depending on the relation of resource requirements and available resources. Extension of available hardware resources can result in faster response times and higher throughput while application of non-optimal solutions in process compositions may lead to significant decrease in non-functional (quality) parameters. Thus prediction of expected resource consumption is critical when designing composite processes.

There are numerous resources needed by a process at run-time. Central processing unit (CPU), memory, input/output operations (I/O) and network bandwidth are some of them. Nevertheless there is always one type of resource which is critical in a given environment determining the actual values of response time and throughput. This is called the bottleneck. Strengthening the bottleneck resource or increasing the amount of available resource yields better performance while expanding of other resources may be ineffective. After consuming all of the bottleneck resource the response time of the system decreases with higher number of requests or this can even lead to malfunctions like refusing of requests. This means that an optimal efficiency (minimal response time) can be achieved only if the load of the bottleneck resource is at a balanced level. On the other hand system throughput reaches its maximal value (maximal number of performed transactions per a given time unit) if the consumption of the bottleneck resource is maximized. Upon these in the rest of my Thesis performance is analyzed through these two different (conflicting) performance parameters. Hence the actual usage rate of the bottleneck resource will be one of the main issues in the analysis.

The next chapter analyzes my proposal comparing its efficiency with other possible approaches.
2.1 Analysis and comparison of my approach

My approach is based on the encapsulation of the standard services of enterprise applications. One up-cast and one-downcast transformation may be glued to every operation of every service. In the rest of my Thesis I assume that there are always both up- and downcast transformations required to be attached. Because this is the worst case from the point of view of added overhead in resource requirement, the efficiency of my solution will surely not be overestimated.

The resource requirement of a given enterprise application service (operation) is determined by the application itself. Since it cannot be changed during the process composition and it will determine the future performance of the composite process, its resource requirement and non-functional parameters in the given environment can serve as reference values for my tests.

Consider that a given operation (e.g. the book flight operation of our service) requires $R_{service}$ unit of the bottleneck resource. The optimal (minimum) response time of the system for that is $T_{service}$ unit of time. There is a certain amount of available bottleneck resource denoted by $R_{avail}$.

Service request may come from more clients (or from more process instances) at the same time. Thus total resource requirement comes to $\sum_{n_{conc}} R_{service}$ where $n_{conc}$ is the number of concurrent service requests. The response time can be calculated as follows:

\[
T_{response} \sim T_{service} \quad \text{if} \quad \sum_{n_{conc}} R_{service} \ll R_{avail} \quad \text{(Exp. 2.1.a)}
\]

which means that response time does not increases significantly when the load is much lower than the available amount of the bottleneck result.

\[
T_{service} < T_{response} < T_{service} * \frac{\sum_{n_{conc}} R_{service}}{R_{avail}} \quad \text{if} \quad R_{service} \ll \sum_{n_{conc}} R_{service} \leq R_{avail} \quad \text{(Exp. 2.1.b)}
\]

which means, that response time increases by the load, because the resource consumption is not distributed steadily in time.

\[
T_{response} = T_{balanced} * \frac{\sum_{n_{conc}} R_{service}}{R_{avail}} \quad \text{if} \quad \sum_{n_{conc}} R_{service} > R_{avail} \quad \text{where} \quad T_{balanced} = T_{response} \quad \text{by a load}
\]

where $\sum_{n_{conc}} R_{service} = R_{avail}$

\[
\text{(Exp. 2.1.c)}
\]

which means that the response time increases proportionately with the resource requirement above the available amount of the bottleneck resource.

After the resources and response times are determined for all participating operations the performance of the whole process can be predicted. The requested resource of the process ($R_{process}$) consists of two separable components:
the calculated cost of the orchestrated service \( R_{orchest} \) built of encapsulated services which depends on the given orchestration e.g. structure and business logic of the process [56, 94] and

- the constant cost of process management \( R_{constructs} \), which includes process instance creation, variable creation and process maintenance.

My experiments show that at reasonable process complexity and available resources (<50 participating service, 5-10 branches and logical expressions, etc.) the calculated orchestrated service cost \( R_{orchest} \) is almost equal to the sum of the cost of invoked participating services. Thus the overhead of applying a process is almost equal to the cost of process management \( R_{constructs} \). Moreover \( R_{constructs} \) seems to be independent from the given instance of processes and remains to be a constant value in a given hardware environment even for different type of processes.

Expression 2.2 shows the calculation of the amount of requested resource of the process.

\[
R_{process} = R_{constructs} + R_{orchest} = R_{constructs} + \sum_{path} R_{service} \quad (Exp. 2.2)
\]

where \( \sum_{path} R_{service} \) symbolizes the resource requirement of invoked participating services of the composite process.

By substituting expression 2.2 into expression 2.1 the response time of a process instance can be calculated as follows:

\[
T_{response} = \sum_{path} T_{service} + T_{constructs} \quad \text{if } \sum R_{process} << R_{avail}
\]

\[
\sum_{path} T_{service} + T_{constructs} < T_{response} < \left( \sum_{path} T_{service} + T_{constructs} \right) \frac{\sum_{n_{conc}} R_{process}}{R_{avail}} \quad \text{if } R_{process} << \sum R_{process} \leq R_{avail}
\]

\[
T_{response} = T_{balanced} \frac{\sum_{n_{conc}} R_{process}}{R_{avail}} \quad \text{if } \sum R_{process} > R_{avail}
\]

where \( T_{balanced} = T_{response} \) by a load where \( \sum R_{process} = R_{avail} \) and where \( \sum R_{process} \) is the resource requirement of all concurrently invoked processes. The response times of the services participating in the given execution path of the process must be summarized \( \sum_{path} T_{service} \). Because \( R_{constructs} \) is constant, the response time overhead \( T_{constructs} \) effected by it is a constant value by lower loads and influences the response times only as a constant multiplier in higher intervals.

In my interpretation the **throughput** is the number of served operation invoked in a given time unit. This can be an invoked operation of a simple or a complex service \( (T_{service}, T_{process} \) respectively). After having used up the entire bottleneck resource (in other words the bottleneck is fully loaded) the throughput reaches its maximal value, denoted by \( T_{service, max} \). So the maximal throughput for a given service invoke describes the pure capacity of a given
hardware-software environment. Such hardware-software environments consisting of several computers connected into a network are called configurations in the rest of my Thesis.

The throughput of a configuration for a given service (process) can be predicted as follows:

\[ T_{p_{\text{service}}} = T_{p_{\text{service}\_max}} \frac{\sum R_{\text{service}}}{R_{\text{avail}}} \]  

if \( \sum R_{\text{service}} \leq R_{\text{avail}} \), which means that the throughput increases proportionally with the load below the total consumption level of the bottleneck resource,

\[ T_{p_{\text{service}}} = T_{p_{\text{service}\_max}} \]  

if \( \sum R_{\text{service}} > R_{\text{avail}} \), which means that the throughput has a well defined maximum level which holds for a longer period after total consumption level has reached and

\[ T_{p_{\text{service}}} < T_{p_{\text{service}\_max}} \]  

if \( \sum R_{\text{service}} >> R_{\text{avail}} \), which means that the throughput starts to fall back after a while because of the extreme overloading of the configuration.

The same types of equitations hold for processes as well:

\[ T_{p_{\text{process}}} = T_{p_{\text{process}\_max}} \frac{\sum R_{\text{process}}}{R_{\text{avail}}} \]  

if \( \sum R_{\text{process}} \leq R_{\text{avail}} \),

(Exp. 2.4.)

\[ T_{p_{\text{process}}} = T_{p_{\text{process}\_max}} \]  

if \( \sum R_{\text{process}} > R_{\text{avail}} \),

\[ T_{p_{\text{process}}} < T_{p_{\text{process}\_max}} \]  

if \( \sum R_{\text{process}} >> R_{\text{avail}} \).

Furthermore there is a trivial connection between the maximal throughput and response time. If we assume that the concurrent requests all came from the same type of service (or process) the maximal throughput can be predicted as follows:

\[ T_{p_{\text{service}\_max}} \sim \frac{n}{T_{\text{response}}} \]  

where \( \sum R_{\text{service}} = R_{\text{avail}} \) and

(Exp. 2.5)

\[ T_{p_{\text{process}\_max}} \sim \frac{m}{T_{\text{response}}} \]  

where \( \sum R_{\text{process}} = R_{\text{avail}} \).

Note, that expression 2.5 does not give an exact value because of the distribution of the current response times. Hence the accuracy of this prediction is higher at lower loads because the deviation is lower at that period.

Now I have the necessary formalism to analyze the performance of complete configurations of given approaches.

### 2.2 Evaluation of proposals

Four different approaches are compared on the same configuration in this part. Because the configurations are the same, the efficiency of the approaches can be directly compared. Furthermore other parameters of these proposals (like component re-usability, modularity, transparency etc.) are also evaluated. The following sub-chapters contain the detailed analysis of each proposal.
2.2.1 Proposal XSLT

My already presented proposal is evaluated first. To implement my approach on the same configuration (see later) the transformations are stored and executed on the same computer (or grid of computers) as the enterprise services are executed on. The proxy service is also installed here, so invocation of an encapsulated service can be served without adding new resources (computers) to the existing configuration of the enterprise application system. This proposal is called as proposal\_XSLT (because of the applied XSL transformations) in the rest of my Thesis. The configuration and typical process execution can be viewed in figure 2.2.

The two transformations (up- and downcast) require \( R_{transf} \) resource from the bottleneck resource (the worst case scenario i.e. the type of the bottleneck resource is the same both for the transformations and service invokes, is assumed and calculated again) and the work of the standard Web service interface requires \( R_{ws\_interf} \). So the additional resource consumption of an encapsulated service is:

\[
R_{add} = R_{ws\_interf} + R_{transf}
\]  
(Exp. 2.6)

Assuming the worst case scenario again, the bottleneck resource of these operations is the same as the bottleneck of the original operation. In this case the invocation of the encapsulated service costs:

\[
R_{encaps} = R_{service} + R_{add}
\]  
(Exp. 2.7)

By substituting expression 2.7 into expression 2.1 the response time can be calculated as follows:

\[
T_{response} \sim T_{service} + T_{add} \quad \text{if} \sum_{n_{conc}} R_{encaps} << R_{avail}
\]  
(Exp. 2.8a)

where \( T_{add} \) is the additional response time compared to the stand-alone enterprise service. This also means that response time does not increases significantly when the load is much lower than the available amount of the bottleneck result.
which means, that response time increases by the load because the resource consumption is not distributed steadily in time.

\[
T_{response} = T_{balanced} + \sum_{n_{encaps}} \frac{R_{encaps}}{R_{avail}} \quad \text{if} \quad \sum_{n_{encaps}} R_{encaps} > R_{avail} \quad \text{where} \quad T_{balanced} = T_{response} \quad \text{by a load where} \quad \sum_{n_{encaps}} R_{encaps} = R_{avail}
\]

which means that after the fully consumption of the bottleneck resource response time increases proportionally with the load because service request should wait for each other to be served.

From expression 2.5 the maximum throughput can be calculated as follows:

\[
T_{p_{encaps,\text{max}}} = \frac{n}{T_{response}}, \quad \text{where} \quad \sum_{n_{encaps}} R_{encaps} = R_{avail}
\]

During the orchestration my encapsulated services are built in into a complex service (process). However the process run-time engine is placed separated from the enterprise application server(s). This determines the distribution of process run-time costs i.e. resource requirements of process management (\(R_{\text{constructs}}\)) and orchestrated process (\(R_{\text{orchest}}\)) between the enterprise application and the process management server. As already mentioned \(R_{\text{orchest}}\) is almost equal to the resource requirement of invoked participating services in a certain environment. Thus \(R_{\text{orchest}}\) is mainly loaded on the application server(s). On the other hand process management requirements (\(R_{\text{constructs}}\)) are served by the process management server(s).

The two run-time components (the enterprise application and the process server) are connected by a network which is responsible for the communication during the execution of the composite process. While the network connection may influence the response time, it barely modifies the throughput of the system because it is unlikely to be the bottleneck resource in any systems applied for process execution today. Hence dedicating separated resources for enterprise services and process run-time is reasonable and it results greater performance than by applying the enterprise service environment also for process management.

Expressions 2.6-2.9 described the resource requirements, expected response times and throughput at the enterprise application server(s). Expression 2.2 must be separated due to the separated resource allocation for enterprise services and process management. Furthermore expression 2.3 should also be modified because of the two possible bottleneck points i.e. the bottleneck of the whole system can be either the enterprise or the process server. Based on expression 2.2 and 2.3 the expected response time of the configuration loaded by a given process is the following:

\[
T_{response} = \sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}}
\]

\[
\text{if} \quad \sum_{n_{\text{orchest}}} R_{\text{orchest}} << R_{\text{avail, enterprise}} \quad \text{and} \quad \sum_{n_{\text{constructs}}} R_{\text{constructs}} << R_{\text{avail, process}},
\]

(Exp. 2.10.)
\[
\sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} < T_{\text{response}} < \sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} \leq \sum_{\text{path}} T_{\text{response}} + T_{\text{constructs}} \frac{R_{\text{constructs}}}{R_{\text{avail_process}}}
\]

if \( \sum R_{\text{orchest}} \ll R_{\text{avail_enterprise}} \) and \( R_{\text{constructs}} \ll \sum R_{\text{constructs}} \leq R_{\text{avail_process}} \)

\[
\sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} < T_{\text{response}} < \left( \sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} \right) \times \frac{R_{\text{orchest}}}{R_{\text{avail_enterprise}}} + T_{\text{constructs}} \times \frac{R_{\text{constructs}}}{R_{\text{avail_process}}}
\]

if \( R_{\text{orchest}} \ll \sum R_{\text{orchest}} \leq R_{\text{avail_enterprise}} \) and \( R_{\text{constructs}} \ll \sum R_{\text{constructs}} \leq R_{\text{avail_process}} \).

\[
T_{\text{bal_constructs}} = T_{\text{proc_response}} \quad \text{(the response time of the process server) by a load where}
\]

\[
\sum_{\text{n_conc}} R_{\text{constructs}} = R_{\text{avail_process}} \quad \text{and}
\]

\[
T_{\text{response}} = \left( \sum_{\text{path}} T_{\text{ent_balanced}} \right) \times \frac{R_{\text{orchest}}}{R_{\text{avail_enterprise}}} + T_{\text{bal_constructs}} \times \frac{R_{\text{constructs}}}{R_{\text{avail_process}}}
\]

if \( \sum R_{\text{orchest}} > R_{\text{avail_enterprise}} \) and \( \sum R_{\text{constructs}} > R_{\text{avail_process}} \).

\[
\sum_{\text{path}} T_{\text{ent_balanced}} = T_{\text{ent_response}} \quad \text{(the response time of the process server) by a load where}
\]

\[
\sum_{\text{n_conc}} R_{\text{orchest}} = R_{\text{avail_enterprise}} \quad .
\]

According my experiences during the pilot evaluations the load of the bottleneck resource at the process server is never less then that at the enterprise application server. This implicates that cases where

\[
\sum R_{\text{orchest}} \ll R_{\text{avail_enterprise}} \quad \text{and} \quad \sum R_{\text{constructs}} \gg R_{\text{avail_process}} \quad \text{are not worth for analysis.}
\]

The proper calculation of actual and maximal throughput (based on expressions 2.4 and 2.5) are left for the reader.

To sum it up, this proposal loads the process server by \( R_{\text{constructs}} \) at each process invoke. Moreover the enterprise server is loaded with an extra \( R_{\text{add}} \) by each participating service in a process execution path by each process invoke. Note that based on my experiments this additional cost is negligible compared to the cost of the original enterprise services. Thus one
may focus on calculating the resource requirements $R_{constructs}$ and predicted performance of the process server.

Remember that the encapsulated services are re-useable in subsequent process orchestration scenarios. Furthermore applying encapsulated services the modularity can be achieved and it hides possible business interests within the non-transparent encapsulated service.

### 2.2.2 Proposal Direct

Another proposal for dealing with data heterogeneity in integration consists of the following steps:

1. Create the composite process ignoring data heterogeneities,
2. Design and add necessary transformation at the process level.

As already mentioned besides defining relationships among participating services and creating run-time branches by logical expressions it is also possible to define some business logic at the process level. (For example the BPEL standard offers the usage of XPath and XQuery standards for defining copy rules in data assignments [4, 84, 85].) Thus both up- and downcast transformations can be designed and added at the process level. One upcast transformation is added before each operation (service) request and one downcast transformation is added after each operation response in worst case. This proposal is called *proposal_Direct* (because data heterogeneities are bridged *DIRECTLY* at the process level) in the rest of my Thesis. The configuration and typical process execution can be viewed in figure 1.3.

![Figure 1.3 Execution of a process request at proposal Direct](image)

The resource consumption is increased by the needs of transformation at the process server but requested resource remains intact at the enterprise server. The extra cost added to each operation (service) invoke is:

$$R_{add\_Direct} = R_{transf}$$  \hspace{1cm} (Exp. 2.12)

My experiences concluded that pure transformation costs are almost the same at the process level as they are in the case of the XSLT execution within my encapsulated services. However, with this proposal it is possible to avoid the creation and employment of an
additional service interface (the service interface of the proxy service). Comparing expression 2.12 and 2.7:

\[ R_{\text{add}_\text{XSLT}_\text{transf}} = R_{\text{add}_\text{Direct}_\text{transf}} \quad \text{and} \quad R_{\text{add}_\text{XSLT}_\text{add}} = R_{\text{add}_\text{Direct}_\text{add}} + R_{\text{ws}_\text{interface}} \]  

(Exp. 2.13.)

However \( R_{\text{add}_\text{Direct}_\text{transf}} \) loads the process server and not the enterprise application server by the execution of transformation. (Note that this may be an advantage or disadvantage of this proposal depending on the availability of the bottleneck resource on the process server or the enterprise server.) By using this along with expressions 2.10 and 2.12 the response time can be calculated as follows:

\[ T_{\text{response}} = \sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} \]

if \( \sum_{\text{n_conc}} R_{\text{orchest}} \ll R_{\text{avail}_\text{enterprise}} \) and \( \sum_{\text{n_conc}} R_{\text{constructs}} \ll R_{\text{avail}_\text{process}} \)  

(Exp. 2.14)

\[ \sum_{\text{path}} T_{\text{service}} + \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} < T_{\text{response}} < \sum_{\text{path}} T_{\text{service}} + \left( \sum_{\text{path}} T_{\text{add}} + T_{\text{constructs}} \right) \sum_{\text{n_conc}} R_{\text{constructs}} \]

\[ \sum_{\text{path}} R_{\text{orchest}} \ll R_{\text{avail}_\text{enterprise}} \quad \text{and} \quad R_{\text{constructs}} \ll \sum_{\text{n_conc}} R_{\text{constructs}} \leq R_{\text{avail}_\text{process}} \]

if \( R_{\text{orchest}} \ll \sum_{\text{n_conc}} R_{\text{orchest}} \leq R_{\text{avail}_\text{enterprise}} \) and \( \sum_{\text{n_conc}} R_{\text{constructs}} \ll R_{\text{avail}_\text{process}} \), where

\[ \sum_{\text{n_conc}} R_{\text{constructs}} = R_{\text{avail}_\text{process}} \]

\[ T_{\text{response}} = \sum_{\text{path}} T_{\text{ent balanced}} \sum_{\text{n_conc}} R_{\text{orchest}} \]

\[ \sum_{\text{path}} T_{\text{ent balanced}} = T_{\text{ent response}} \]  

(Exp. 2.14)

where instead of \( R_{\text{orchest}} R_{\text{constructs}} \) contains the additional resource requirement of transformations compared to proposal_XSLT.

Similarly to the analysis of the previous proposal some unrealistic combinations of load periods of the two servers are dismissed here as well.
To sum it up, some increase in the performance (lowering the response time) is possible compared to the proposal_XSLT in specific cases. However additional cost of maintaining a service interface are not significant compared to the service ($R_{service}$) and process running ($R_{constructs}$) costs, so possible performance increase is not significant.

This little improvement probably is not worth the price we have paid for it. Because my transformations are defined at the process level they are process dependent. Hence probably these transformations cannot be reused in other process compositions. Furthermore by adding data transformations to the business logic we have messed up the clear structure and view of the orchestration. (The transformations were added to given copy rules of assign elements in the BPEL process where the structural assignments of services – connection of global BPEL and local service variables – were also defined. These cannot be separated and differentiated easily later by a process upgrade or modification.) So the modularity of the orchestration was ruined and the possibility of creating an additional private business layer upon the enterprise service was largely reduced.

### 2.2.3 Proposal BPEL

The next presented proposal also defines the necessary transformations at the process level, but it eliminates upcoming low modularity and non-reusability problems of proposal_Direct. The methodology of creating working composite processes is the following:

1. Define a collection of possible concepts for the given domain (global schema).
2. Adopt each participating service to the process level, each of them into a new process.
3. Design and implement transformations in these processes to align complex service (process) interface to the global schema.
4. Orchestrate business process using complex services created in the previous step as building blocks.

Similarly to proposal_Direct the transformations are defined at the process level (Copy rules are defined in assign elements in the BPEL process). On the other hand complex services containing only one operation (service) invoke and glued up- and downcast transformations are equivalent with the encapsulated services of proposal_XSLT.

This proposal is called *proposal_BPEL* (because additional BPEL processes are also applied to bridge data heterogeneities at each service invoke) in the rest of my Thesis. The configuration and typical process execution can be viewed in figure 2.4.

![Figure 2.4. Execution of a process request at proposal BPEL](image-url)
The additional resource requirement is significant. Besides the additional cost of transformations ($R_{transf}$) there is a process management overhead ($R_{construct}$ and $R_{orchest}$) at every participating service invoke. Although the orchestration costs are negligible (each “encapsulated” services contains only 2 assign elements and 1 service invoke) the creation of process instance and managing process interface adds significant overhead ($R_{construct}$ to each participating operation) in the run-time.

Similarly to proposal_Direct additional performance requirements are loaded to the process server. Based on expression 2.14 the response time can be predicted as follows:

$$T_{response} = \sum_{path} T_{service} + \sum_{path} T_{add} + T_{constructs}$$

if $\sum_{n_{conc}} R_{orchest} << R_{avail\_enterprise}$ and $\sum_{n_{conc}} R_{constructs} \leq R_{avail\_process}$, (Exp. 2.15.)

$$= \sum_{path} T_{service} + \sum_{path} T_{add} + T_{constructs} < T_{response} < \sum_{path} T_{service} + \sum_{path} T_{add} + T_{constructs} \frac{\sum_{n_{conc}} R_{constructs}}{R_{avail\_process}}$$

if $\sum_{n_{conc}} R_{orchest} << R_{avail\_enterprise}$ and $R_{constructs} \ll \sum_{n_{conc}} R_{constructs} \leq R_{avail\_process}$,

$$= \sum_{path} T_{service} + \sum_{path} T_{add} + T_{constructs} < T_{response} < \sum_{path} T_{service} \frac{\sum_{n_{conc}} R_{orchest}}{R_{avail\_enterprise}} + \sum_{path} T_{add} + T_{constructs} \frac{\sum_{n_{conc}} R_{constructs}}{R_{avail\_process}}$$

if $R_{orchest} \ll \sum_{n_{conc}} R_{orchest} \leq R_{avail\_enterprise}$ and $R_{constructs} \ll \sum_{n_{conc}} R_{constructs} \leq R_{avail\_process}$, where

$$T_{bal\_constructs} = T_{proc\_response}$$ (the response time of the process server) by a load where

$\sum_{n_{conc}} R_{constructs} = R_{avail\_process}$ and

$$T_{response} = \left(\sum_{path} T_{ent\_balanced}\right) \frac{\sum_{n_{conc}} R_{orchest}}{R_{avail\_enterprise}} + T_{bal\_constructs} \frac{\sum_{n_{conc}} R_{constructs}}{R_{avail\_process}}$$

if $\sum_{n_{conc}} R_{orchest} > R_{avail\_enterprise}$ and $\sum_{n_{conc}} R_{constructs} > R_{avail\_process}$, where

$\sum_{path} T_{ent\_balanced} = T_{ent\_response}$ (the response time of the process server) by a load, where

$\sum_{n_{conc}} R_{orchest} = R_{avail\_enterprise}$,

where $\sum_{path} T_{add}$ contains also $T_{constructs}$ which is the cost of each built in complex service invoke.

The extra cost added to each operation (service) invoke is:
which is significantly greater than the extra costs of proposal XSLT and proposal Direct.
(Based on my experiences, \( R_{constructs} \gg R_{ws\_interface} \)). Thus the performance of this proposal is expected to be lower (higher response times, and lower throughput) than that of the previous proposals.

On the other hand applying transformations at the process level has some advantages too. The solid tools for BPEL process design and run-time can be reused and no further equipment is required to create and run the transformations. Furthermore the same process run-time environment can be applied for deploying and running both stand-alone “encapsulated” processes and orchestrated composite services. Hence the enterprise application and its server environment remain totally intact.

Similarly to proposal XSLT the “encapsulated” processes (services) are reusable in later orchestration scenarios. Moreover the proposal involves modularity and makes it possible to create an additional private layer beyond the enterprise services at the process level of stand-alone processes.

The reader may also note, that extending my configuration with an additional process server environment the “encapsulated” processes can be separated from the orchestrated processes which probably results in significant improvement in the throughput at higher loads.

After the experiments showed much more load on the process server than on the enterprise server I have decided to install an additional process run-time engine (BPEL engine) at the enterprise server so that the load caused by process invokes (\( R_{constructs} \)) could be separated between the two servers of my configuration. Modification of expression 2.10 according to this (evaluate \( \sum_p R_{add\_BPEL\_service} = \sum_p R_{transf} + R_{constructs} \)) is left for the reader.

2.2.4 Proposal Native

The realization of the last presented proposal consists of the steps as follows:

1. Define a collection of possible concepts for the given domain (global schema),
2. Modify the standard enterprise application services and/or its interface directly at the application level to assign participating operations to the global schema,
3. Orchestrate business processes using the revised services of enterprise applications created in the previous step as building blocks.

To modify an enterprise application a special development environment of the application is required. This is mostly unavailable at the customer of the system. Furthermore employees of the organization are usually not allowed (or do not have the proper knowledge) to modify standard functions of enterprise applications they use. Thus alignment of services and their interfaces to a global schema remains to the software vendor. Because integration (the creation of collaborative business process) can affect several application systems at more companies with various vendors, the process orchestration requires the contribution of numerous developers and can slow down the whole integration progress. Contrary to this, the creation of transformations and processes can be performed by integration experts responsible directly for process orchestration.

This proposal is called proposal Native (because bridging of data heterogeneities is solved at the NATIVE, enterprise application implementation level) in the rest of my Thesis. The configuration and typical process execution can be viewed in figure 5.
After all participating services are aligned to the global schema the performance of this proposal is convincing. The additional resource requirements of the modified enterprise service are not significant and no more additional resource consumption is required nor by the preparation nor by the orchestration at all. The enterprise server is loaded by the resource costs of invoked operations and the process server is loaded by the costs of the management of orchestrated composite service ($R_{constructs}$). Due to the additional cost of all proposals the following relations hold:

$$R_{add\_Native\_service} < R_{add\_Direct\_service} < R_{add\_XSLT\_service} < R_{add\_BPEL\_service}$$  \hspace{1cm} (Exp. 2.17) AND

$$R_{add\_Native\_service} = R_{add\_Direct\_service} - R_{transf} = R_{add\_XSLT\_service} - R_{transf} - R_{ws\_interface}$$  \hspace{1cm} (Exp. 2.18)

The response time of the configuration can be predicted by the following expression:

$$T_{response} \sim \sum_{path} T_{service} + T_{constructs}$$

if

$$\sum_{n_{conce}} R_{orchest} \ll R_{avail\_enterprise}$$ \hspace{1cm} and \hspace{1cm} $$\sum_{n_{conce}} R_{constructs} \ll R_{avail\_process}$$ \hspace{1cm} (Exp. 2.19)

$$\sum_{path} T_{service} + T_{constructs} < T_{response} < \sum_{path} T_{service} + T_{constructs} \times \sum_{n_{conce}} R_{constructs} \ \ \ \ R_{avail\_process}$$

if

$$\sum_{n_{conce}} R_{orchest} \ll R_{avail\_enterprise}$$ \hspace{1cm} and \hspace{1cm} $$R_{constructs} \ll \sum_{n_{conce}} R_{constructs} \leq R_{avail\_process}$$

and

$$\sum_{path} T_{service} + T_{constructs} < T_{response} < \sum_{path} T_{service} \times \sum_{n_{conce}} R_{orchest} \ \ \ \ R_{avail\_enterprise}$$

if

$$R_{orchest} \ll \sum_{n_{conce}} R_{orchest} \leq R_{avail\_enterprise}$$ \hspace{1cm} and \hspace{1cm} $$R_{constructs} \ll \sum_{n_{conce}} R_{constructs} \leq R_{avail\_process}$$

if

$$R_{orchest} \ll \sum_{n_{conce}} R_{orchest} \leq R_{avail\_enterprise}$$ \hspace{1cm} and \hspace{1cm} $$\sum_{n_{conce}} R_{constructs} > R_{avail\_process}$$ \hspace{1cm} where
\[ T_{\text{bal}_\text{constructs}} = T_{\text{proc}_\text{response}} \] (the response time of the process server) by a load where

\[ \sum_{n_{\text{conc}}} R_{\text{constructs}} = R_{\text{avail}_\text{process}} \] and

\[ T_{\text{response}} = \left( \sum_{\text{path}} T_{\text{ent}_\text{balanced}} \right)^* \frac{\sum_{n_{\text{conc}}} R_{\text{orchest}}} {R_{\text{avail}_\text{enterprise}}} + T_{\text{bal}_\text{constructs}} \sum_{n_{\text{conc}}} R_{\text{constructs}} \right) ^* \frac{R_{\text{avail}_\text{enterprise}}} {R_{\text{avail}_\text{process}}} \]

if \[ \sum_{n_{\text{conc}}} R_{\text{orchest}} > R_{\text{avail}_\text{enterprise}} \] and \[ \sum_{n_{\text{conc}}} R_{\text{constructs}} > R_{\text{avail}_\text{process}} \], where

\[ \sum_{\text{path}} T_{\text{ent}_\text{balanced}} = T_{\text{ent}_\text{response}} \] (the response time of the process server) by a load, where

\[ \sum_{n_{\text{conc}}} R_{\text{orchest}} = R_{\text{avail}_\text{enterprise}} \]

where \( R_{\text{constructs}} \) is the process management cost of the orchestrated service and \( R_{\text{orchest}} \) is the run-time cost of the invoked participating enterprise services (operations).

Based on resource requirements this proposal may yield an optimal (maximal) throughput for a given composite service (process). On the other hand it is hard to be carried out because upcoming modification requests (alignment of services to the global schema) can only be treated at the enterprise application development level.

Concerning modularity and transparency this proposal does not differ from the orchestration of standard enterprise services. Updated services remain black-boxes, and are reusable in subsequent orchestration scenarios.

Table 2.1 summarizes the main aspects (advantages and disadvantages) of the 4 presented proposals.

**Table 2.1 – Comprehensive table of proposals**

<table>
<thead>
<tr>
<th>Proposal</th>
<th>Resource overhead per enterprise operation invoke</th>
<th>Modularity of the orchestrated process</th>
<th>Reusability of transformations &amp; encapsulated services</th>
<th>Applied technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposal_XSLT</td>
<td>R_transf + R_wsinterface</td>
<td>Modular encapsulated services</td>
<td>Encapsulated service are reusable</td>
<td>Process (BPEL)* and transformation (XSLT)</td>
</tr>
<tr>
<td>Proposal_Direct</td>
<td>R_transf</td>
<td>Orchestration logic is messed up by transformations</td>
<td>Transformations are NOT reusable</td>
<td>Process (BPEL)*</td>
</tr>
<tr>
<td>Proposal_BPEL</td>
<td>R_transf + R_constructs</td>
<td>Modular processes containing 1 operation and 2 transformations</td>
<td>Single processes containing 1 operation and 2 transformations are reusable</td>
<td>Process (BPEL)*</td>
</tr>
<tr>
<td>Proposal_Native</td>
<td>None</td>
<td>Same as in standard SOA architecture</td>
<td>Same as in standard SOA architecture</td>
<td>Process (BPEL)</td>
</tr>
</tbody>
</table>

* Process (BPEL) means that included technologies of the BPEL standard (e.g. XPath, XQuery) are used for transformation creation and execution as well.
The next chapter presents my experimental results. All proposals were implemented for the same integration scenario (same data heterogeneity problem) on the same configuration. The results have confirmed the analytic methods and predicted values described in this section.

2.3 Experiments

The applied configuration contains two computers in my experiments. One was used as the enterprise server and the other one as the process server. Both work with a Pentium Core2Duo processor at 2.1Ghz and is equipped with 4 and 2GigaBytes of memory, uses high speed disks and internet connection. As the tests will show later, the bottleneck resource of the configuration is the computational requirement (CPU usage). Thus further technical parameters of the configuration are not described in details.

To run the test (simulate concurrent service requests) an additional computer was also necessary. Because the test machine only sends requests and receives responses to and from the orchestrated service its resource consumption is significantly lower than the load on the servers. None of the resources was consumed totally, so the bottleneck of the system was not the test computer. Moreover in real usage scenarios client requests are mostly distributed on several computers. Thus performance on the side of the client is generally not evaluated in details in my experiments.

2.3.1 Measurement method and metric

There are many possible testing metrics and strategy for (complex) service invoke. After functional tests (giving different input parameters and comparing the response with the desired output) have proved (complex) service and system availability and proper working, the examination of the non-functional parameters (performance) can be accomplished.

To simulate real life situations the variance of virtual client requests must be set. It is proven, that the arrival of client requests has an exponential distribution in a live environment. Changing the parameter of the testing distribution makes also possible to set different levels of load (different section of a day provides different load of the real systems – e.g. typical services related with business activities like enterprise application requests, database request, etc. – are used significantly more for example between 9-11 a.m. and 13-15 p.m. than in other time period). On the other hand system engineers are interested only at performance parameters at a typical (normal) and at some high load level.

Although maximal faithful simulation should seem to be the most reasonable, it can be substituted by more “artificial” testing loads to achieve exact knowledge of system performance in shorter time within higher confidence level. Testing with a variance probably results in high positive and negative amplitude in momentary load which may lead to unexpected test values. Moreover the values will highly depend on the actual state of the system (e.g. cache saturation) during the whole test and averaged results may also differ even at longer test runs. Therefore instead of applying exponential distribution as request simulation method, constant loading strategies are preferred in my work. These are able to exclude almost all undesired random effects like cache overflow, or even high cache hit which cannot serve as a base for further calculations and predictions of system performance.

There are two ways of testing with constant load:

- The simple load strategy simulates concurrent clients sending their request following a defined time interval after having received the answer to the previous request. This time interval (often called as delay) and the number of concurrent clients can be set for this test. For example one realistic load test can be the testing of a service with 10 concurrent clients, sending requests with a delay of 100ms. Although this type of test
can provide finely set load, the feedback from the server application is not considered. This means, that higher loads cannot be achieved only by changing the value of delay if the number of applied concurrent clients is too low. Moreover this kind of test requires complex calculations because of the simultaneously changing number of the two factors, the number of concurrent clients and the specified delay. Hence determination and comparison of the resource requirements of different test scenarios is not trivial. On the other hand the test can be configured by custom granularity to create comprehensive test results for every configuration. Consequently this test can be used for testing every kind of configurations in a normal (not to high) loading period because the resource requirement of the test can be more balanced.

- The burst load strategy simulates concurrent clients sending their requests right after they have got the response for its previous request. Because there are no further configuration parameters besides the number of concurrent clients, current resource requirement of a given test can be identified unambiguously. So only the number of concurrent client can be set in this test. Compared with the simple load strategy this test is more applicable for finding performance bottleneck, determine maximum configuration capacity and system overall performance at higher loads. The disadvantage of this strategy is that it is hardly customizable for every configuration for measuring performance parameters at lower (normal) loads. My experiments showed that only with 1 or 2 (or with any lower number of) concurrent clients this testing can already reach the maximum bottleneck consumption on configurations with lower amount of resources. This may hinder the analysis of system performance at different lower loads.

Fortunately after having troubles at configurations built on weaker hardware components I was able to install a configuration, which was strong enough to run tests based on the burst strategy also at the lower load period. Thus test results created only by burst strategy tests are presented in this work. Moreover tests results of lower, normal, high and overloaded intervals can be presented in the same diagrams this way.

There are other “mixed” testing strategies as well. For example it is possible to change the number of simulated concurrent clients dynamically at the test run-time or different random factors for the simple load time interval can be set. Although these options make it possible to highly customize the tests, they are usually excluded because tests results are hard to be analyzed because of the numerous varying factors during the tests.

Another important occurrence is the lead in period of a test. The values measured in the lead in period are not valid for further analysis because the system startup can result significantly different and unexpected values. The background of this may be the overhead caused by resuming the listening server applications from standby or higher momentary availability of cache or buffer resources. After a given time the performance indicators stabilize and the test results converge to a stable value. This stable value (limit value) is the right performance indicator and should be used for further calculations. Thus lead in period of the tests is removed from the results before different configurations and approaches are evaluated in my work. Note, that performing longer tests can increase the accuracy of the stable value calculated as an average of all measured momentary values. Figure 2.6 shows the evaluation of throughput and response time before removing the results of the lead in period. The lead in period can be viewed at the left side of the diagram.
Another meaningful method for testing is running various tests with the same parameters and calculating the average of the results. This can help increase confidence by eliminating unexpected momentary amplitudes during the tests. All tests presented in my work were performed at least three times.

Although the configuration and software environment remains the same during all performed tests there were some slight differences among tests results of experiments executed on different days (or even a few hours later). The explanation is the applied Microsoft Windows operation systems (Windows Xp and Windows 2003 Server). It is almost impossible to ensure the totally same memory and other resource usage conditions because some applications (e.g. anti-virus, live-update) needs to be run and updated continuously which leads to small changes for example in the memory allocation. Furthermore the test computers were simple clients at a laboratory so they had to be turned off sometimes (e.g. every evening) and restarting the computer and building up necessary software environment did not result exactly in the same resource allocation. Fortunately these changes affects only the level of the performance indicators while well determined tendencies and major differences between test scenarios remains the same. Moreover there are typical simple test scenarios which make it possible to calculate the accurate value of the small differences between the performance of configurations at different times. Using this, the results coming from different testing days could be revised and compared.

Functional tests strongly depend on the given value of the input data. More precisely, the aim of functional tests is to evaluate the response data of the system concerning the expected output for the given input. For example if a simple adder operation is tested the desired output for inputs 2 and 5 is 7. The tests are typically run for multiple different input data because it is possible that there are some input data proving the right working of the operation while for other input data the operation fails to respond the desired response value. On the other hand, testing applications always with same input data can result, that the return values are responded from some kind of cache preventing the actual execution of the operation which was not intended to be tested. Besides this may lead to a false evaluation of the functional test results, the performance indicators are also strongly influenced hence reading cached values instead of calculating actual values by executing program code of real operations is probably much faster.

My experience is that the tested Web services provide their response always by run-time re-executing of requested operations so return values cannot be originated from a cache. This makes it possible to use the same data for performance tests which is actually a great help since creating several types of test input data is unnecessary. On the other hand applying various test input may slightly slow down the testing client influencing the performance result.
values of the configuration. Consequently, the same test input data was used during all performance tests by all simulated client request presented in my work.

Because the main performance indicators of a composite process environment are response time and throughput, the experiments mainly intend to evaluate these. The implemented proposals of the previous section are all triggered by the same burst loading strategy so that measured results should be directly comparable.

The next chapter presents a simple toy example which is implemented to be compared by my different proposals during the tests.

2.3.2 Toy example
To simulate real integration scenario and data heterogeneities a toy example was used in my tests. My virtual organization offers all kinds of healthcare and wellness services to the clients. To help estimate customers’ demands and to give advice on healthy living the company decided to implement a portal based on SOA architecture and methodology. For example, a composite process is able to ask (retrieve) a client’s parameters and propose optimal sport activity and personalized dietary. Since there are numerous enterprise systems at the virtual organization, composite processes often communicate with several systems. To overcome data heterogeneities the leadership decided to align all services (and service interfaces) to a global schema of the domain sports and healthcare. In the tests only the alignment of one service is presented. The standard Body-Mass-Index calculator (shortly BMIcalculator) service of our enterprise system is equipped with necessary up- and downcast transformations and the performance of its application in composite services is measured during several tests. Based on the proposals described in the previous section, the performance of the four different implementations is compared via various tests described below. This will also validate the basic analytic equations of the previous sections.

The standard BMI calculator service requests full name, height and weight as input parameters and responds the full name, the BMI and a category of a person as an output. (See the input and output data structure of the standard BMI service in figure 4.3 of the appendix) Because the application was made in the UK, the service requests height given in feet, weight in pounds and full name in one field – first name and last name separated with a space. However our applied global schema (and most of other enterprise applications of ours) calculates the height in centimeters and the weight in kilograms. Moreover the name of the clients is stored in a structure containing two fields: one for the first name and one for the last name. The responded category of the BMI status should also be translated for Hungarian customers and for other applications: categories underweight, normal, overweight and obesity are translated into sovány, normál, túlsúlyos and kórosan elhízott. (See requested input and output data structure in the description of the proxy service interface in figure 4.4 of the appendix.) Thus up-cast transformation is required to change between different metrics and concatenate separated last name and first name and down-cast transformation is required to translate the value of category and split full name string value into two separated fields.

The standard BMI service and all necessary transformations and constructs of all proposals were implemented. (See the code of the up- and downcast transformations in figure 4.6 and in figure 4.7 of the appendix.) For the implementation .Net framework, Visual C# and Active BPEL [2] designer were used. (See orchestrated processes of the proposals in figure 4.8 in the appendix.) The running was performed on Internet Information Services (IIS) and on Active BPEL run-time engine. Native level enterprise transformations (in proposal_Native) were also implemented in C# and XSLT transformation and the proxy service were executed on the .Net framework as well (in proposal_XSLT). Process level transformations were defined using the
Xpath and XQuery standards in the Active BPEL designer. (See the down-cast and up-cast transformations of Proposal BPEL defined in BPEL copy rules in figures 4.9 and 4.10 in the appendix.)

2.3.3 Tests and results analysis

To determine basic efficiency of the given experimental configuration the standard BMI service and the enterprise server was evaluated first. The method to find the bottleneck resource was the following:

The BMI service was tested by different burst load. Starting from 1 simulated client the number of concurrent clients was increased continuously in the tests. Meantime the changes of performance indicators (response time and throughput) and current resource consumption of resources were monitored. Processor usage in percentage, memory usage in Megabytes, disk usage in Kilobytes/sec and Network usage in Kilobytes/sec are key resources of hardware configurations. The consumption value of each resource type was converted into percentage value to see which one is reaching the total (100%) first. (See a screenshot of the working performance monitor tool in figure 4.11 in the appendix)

With 9 concurrent clients sending requests simultaneously the CPU usage has reached 95% usage and stayed at this level with any higher number of concurrent clients. On the other hand the usage of other resources remained significantly below the available maximum. The possible maximum value of the resources was estimated as follows: for memory it was the capacity of the physical memory of the test computer that is 4096 Megabytes, for disk usage the estimated maximum of the transfer speed was 7000 Kilobytes/sec and it was 10000 Kilobytes/sec for the network interface. Note, that these values are a little underestimated so that to exclude the false detection of the bottleneck resource. The maximal consumption of memory during the tests was 1556 Megabytes and it was 622 Kilobytes/sec and 4212 Kilobytes/sec for the disk and network transfer respectively. The usage level of the bottleneck resource (CPU) has never reached higher values then 95-96%. Accordingly the bottleneck resource was the CPU usage during the tests. Figure 2.7 shows the usage level of all resources at this load.

Changing the number of concurrent clients the response time and throughput moved closely enough to the expected values which confirm my expressions 2.1-2.5. In the period before reaching maximal consumption of the bottleneck resource the response time stayed close to a constant (5 millisecond) and throughput increased due to the number of simulated concurrent

![Figure 2.7 Resource usage reaching maximum bottleneck resource usage by standard service invoke](image-url)
clients (from 1 to 9 clients it increased from 700 to 4200 transactions/sec). All tests were performed for 3 minutes long until the average results have stabilized while values of the lead-in (setup) period were removed from the results. Figure 2.8 shows the changing of response time and throughput according to the changes in the number of concurrent clients.

![Figure 2.8 Evaluation of response time and throughput invoking standard service](image)

In the same way as the enterprise server environment was tested, the process server was tested as well. A stand-alone process containing no external operation invokes was implemented for this purpose. Figure 2.9 shows the usage level of different resources when bottleneck resource load reached its maximum 96% and figure 2.10 shows response time and throughput reflecting the changes in the number of concurrent process requests.

![Figure 2.9 Resource consumption reaching maximum bottleneck resource usage at the process server](image)

![Figure 2.10 Evaluation of response time and throughput by invoking an empty process](image)
The implementation based on proposal XSLT was evaluated first. Similarly to previous test results the bottleneck resource was the computational efficiency of the system, more precisely the process server has reached its maximum consumption of the CPU usage at a level of about 96%. Because the CPU usage of the enterprise server was about 10%, I conclude that $R_{\text{constructs}}$ is significantly greater than $R_{\text{orchest}}$. Figure 2.11 shows evaluation of response time and throughput reflecting to the changes in the number of concurrent requests. The experiments provide evidence on my expressions (2.1-2.5 and 2.10) founded in the analytical section.

![Figure 2.11 Evaluation of response time and throughput by invoking processes of proposals XSLT](image)

The evaluation of proposal Direct showed the expected differences compared to proposal XSLT. However, these differences in response time and throughput are small between the two proposals. Response time and throughput are showed in figure 2.12. Based on the test results the following relations can be stated: $R_{\text{ws\_interface}}$ influences the overall performance not much and $R_{\text{add\_Direct}}$ is not significant compared to $R_{\text{constructs}}$. Furthermore, measured values of response time confirmed the expression 2.14.

![Figure 2.12 Evaluation of response time and throughput by invoking processes proposals XSLT](image)

Proposal BPEL was observed to be the slowest approach according to the test results. Adding $R_{\text{constructs}}$ resource overhead to every participating operation-involve caused a significant increasing in response time and a decrease in throughput compared to other proposals. Figure 2.13 shows test results.

![Figure 2.13 Test results for BPEL](image)
Proposal Native has slightly outperformed the other 3. On one hand the reduction of costs by eliminating transformations was yet not significant compared to $R_{\text{constructs}}$. The resource need caused by $R_{\text{constructs}}$ determines the overall load again, which exhausts the bottleneck resource (CPU) on the process server. Figure 2.14 shows response time and throughput according to changes in the number of concurrent requests (clients).

The performed tests have showed the applicability of each proposal and proved that the analytic results presented in the previous section were right.

Conclusions and areas of possible future work are presented in the next section.

**Conclusion and future work**

This part has presented an approach to bridging semantic heterogeneities during the enterprise application integration. The framework is based on the concepts, standards and technologies of the SOA methodology. To evaluate the efficiency of my solution analytical tools were developed and presented in this part. Besides my approach, three further proposals were implemented and also compared. The analytical results were reinforced by extensive experiments and the test results have confirmed the performance values expected from formal calculations. Hence, this part has presented a complete methodology and detailed technological proposal to implement integration configurations and predict non-functional QoS parameters of orchestrated services.

The approach has presented a solution for attaching and executing transformations before (and after) standard service invoke (and after service response). However this part of my
Thesis gives no promotion to detect data inconsistencies and to create up- and downcast transformations. The integration environment and the number of participating enterprise services can be huge at large organizations making the task of transformation creation costly. Thus human integration experts and developers could also be supported by (semi)-automated methods and tools to detect data mismatches and create transformations between the concepts of the global schema and the service interface data. Since semantically related concepts can be identified by applying the approach presented in Part I, attaching a proper solution for transformation creation support among identified related concepts can be a reasonable extension of my work in the future.

The experiments were only performed on one given configuration and implemented on one given task (integration of the BMI calculator service). More complex tests and more refined configurations would probably lead to more detailed results. Real services of standard enterprise application could also be involved into the analysis and experiments in the future.

Applicability of results

The presented SOA based integration methodology and created framework is able to solve enterprise application integration problems in small sized organizations. Furthermore after implementing more effective tools (e.g. proxy service and service interface implementation, engine for executing data transformations etc.) for the framework it also provides an applicable solution for performing collaborative processes of larger applications. The methodology and the framework presented in Chapter II.1 of the PhD Thesis can also be adopted into productive environment and realized by solid, industrial tools. This probably provides already a solution for data incompatibilities and SOA based integration problems of any size.

The solution presented in Chaper II.2 makes possible to analyze the run-time performance of specific SOA based integration implementations in enterprise application integration. The productive applicability and hardware sizing is supported by the analytical estimation of resource requirements and by the calculation of non-functional QoS (Quality of Service) parameters based on reference measurements. The necessary methods and tools for a possible productive applicability are all presented in the PhD Thesis.
PART III
Project FUSION

Introduction
The aim of Project FUSION (FP6) was to design and develop a semantic based SOA integration platform and methodology to support the collaboration of small and middle sized enterprises (SMEs) in the enlarged Europe [32]. The project started in 2006 and finished 2008. As a core member of the participating BUTE team I was involved almost in every project phases. Our team was mostly responsible for designing and implementing a solution for aligning services to a global semantic schema (Ontology) and for creating a run-time solution to them. Personally I supervised the specification of the semantic profiler (a tool for attaching additional semantic and run-time information to encapsulated services), the mediator service creation (since the proxy services also implements communication between semantic global and non-semantic local layers they are called mediator services in FUSION) and deployment to semantic service repository. This chapter summarizes the main components and results of FUSION focusing on a comparison and relation with my Thesis.

1. Problem formulation in FUSION
One of the main tasks of FUSION was to develop a solution which offers advanced possibilities for service discovery at the process orchestration. The orchestration itself can be performed in two different ways [34, 36]:
- Predefined process templates can be customized to given integration scenarios by the manual composition. Since typical business processes like scheduling of the production, management of the procurement or billing and shipping are mostly standardized and follows the same idea at any organizations, process templates can be predefined by business experts. The templates are abstract processes which contains the business logic (the relation of involved services and operations called choreography) and the description of required services. The compatible services can be found in the service registry from this description. Appropriate services of given enterprise applications are substituted then into the given location of the process template resulting a working, executable composite process.
- Simpler processes can also be created semi-automatically. After defining an orchestration goal the semi-automatic composer intends to create a business process. Attached service capabilities (behavioral aspects like pre- and post-conditions of a given service invoke), input and output schema of operations are taken into account during the process composition. The main idea behind the semi-automatic composition is the theory of abstract state machines simulating the work of participating services with complex behavior. If semi-automatic process composition has failed to create a process fulfilling the orchestration goal, a process template should be chosen for continuing with manual composition.

The finding of the right service (operation) for the right place of a composite process is the key point of both approaches. This can be supported by additional attached semantic information stored in the service registry of the FUSION system [38].

FUSION also builds on the existence of standard enterprise service interfaces, which were assumed to be already created by application vendors. However these offered services are usually described only by technical (syntactic) information. This information is enough to
invoke the service with some test data, but not enough to understand its behavior and participate in FUSION. Business process orchestration also requires attached pre- and post-conditions of service operations, which should be taken into consideration by the process composition. Because of the installed state machines it is also necessary to label service operations by their indentified capabilities. Thus the integration architecture should also offer tools for enterprise application developers to attach such (semantic) information to service descriptions [34, 35].

Although service discovery and process composition are fully covered by FUSION, the runtime aspects like process performance and non-functional QoS parameters of complex services are not detailed. From this point of view my thesis is a good complementary to project FUSION helping the live, on-line application of it. On the other hand semantic services, service discovery and orchestration are well supported by solid tools and complex methodology were developed in FUSION, and thus were not intended to be further improved within this thesis.

The next chapter shortly introduces the tools and the architecture of the FUSION framework.

2. The FUSION framework

Based on the specific time of the usage the components of FUSION can be divided into design-time and run-time tools. Tools responsible for the preparation for service orchestration, process development and composition are design-time components, while process administration, deployment, performing and monitoring are tasks done by the run-time components. Figure 3.1 shows the system architecture. Design-time components are on the left and run-time components are on the right hand side of the figure.

Using the semantic profiler tool one can attach additional information about service behavior to the service descriptions. Semantically enriched descriptions are stored in the semantic registry. Using the discovery interface of the registry one can already identify existing enterprise application services and compose processes from them. Process templates are designed and maintained by the process designer tool which is actually a graphical process designer application.

![Figure 3.1 Overview of FUSION system architecture – source D2.1 Fusion Approach[32]](image-url)
Besides that additional semantic information facilitates the discovery and process composition it also makes possible to create a new semantic interface for services. So data heterogeneity problems emerging during the process orchestration are bridged by aligning services to a semantic data layer acting as a global schema for the communication. The local data layer of service operations and the semantic layer are bridged by up- and downcast data transformations similarly to my approach presented in the previous part. However the underlying technology of the mediator (proxy) services differs a lot—see later. Similarly to the global schema concepts the FUSION ontology contains every possible upcoming business concept divided into several domains. The global concepts of the Ontology [33] are managed by the concept designer tool and stored in the Ontology repository.

There is an administration component which acts as an interface between the run-time repository and the design time components. Composed processes are stored at the run-time server and there is also a run-time container for semantically enriched (encapsulated) FUSION mediator (proxy) services. Semantically enriched services and encapsulated standard services of enterprise applications are shown in the lower corner on the left of figure 3.1.

The next section shortly presents the technologies and standards applied in FUSION comparing them with my proposals.

3. Technical details and differences

Because BPEL became also the de facto standard of process description and execution it was selected both in my work and FUSION. A minor difference in the technological level is, that new constructs from the BPEL 2.0 standard [4] were also used in my experiments while FUSION strictly relies on the former (version 1.1) BPEL standard. Nevertheless, the usage of BPEL version 2.0 does not influence significantly nor design-time methods neither run-time performance. The manual process designer in FUSION applies a customized abstract BPEL process proposal, while the executable BPEL standard is adopted into the run-time solution both in FUSION and in my work.

The global schema of FUSION relies on semantic description of business entities. This means that beyond the unambiguous identification, classification and syntactical specification of concepts semantic relationships are also assigned among the concepts of a domain. The Ontology Web Language (OWL) [82] standard was selected for this purpose. However my thesis builds only on syntactical considerations of service adaptation covered by the XML-Schema [86] standard.

The semantic services of FUSION were described by OWL-S [50] in the first prototype. Because standard BPEL engines are not able to run OWL-S services they were encapsulated into Web services described by WSDL [82]. Later with the emerging of the WSDL-S standard [5], the project consortium decided to use WSDL-S instead of OWL-S and WSDL. This resulted, that the OWL-S semantic layer of the services could be eliminated and semantic concepts of the ontology could be referenced directly from the mediator service description described in WSDL-S. Furthermore WSDL-S services remain compatible with standard BPEL run-time engines so mediator services can be directly adopted into composite processes in FUSION. Because my work does not pay attention to semantic annotation of service description, (only) Web services with standard WSDL descriptions are used in my proposals and experiments.

The implementation and installation environment of FUSION and of my work differ as well. While encapsulated (mediator) and sample standard enterprise services are developed in JAVA in FUSION, I have chosen .NET platform for that purpose. My personal experience is that the .NET environment is more solid and better supported than JAVA based approaches.
and reference implementations. On the other hand, easy to use, stable tools and run-time engines respecting the BPEL standard are also implemented in JAVA. Both FUSION and I decided to prefer Active-BPEL process orchestration and run-time solution from Active Endpoints [2]. Similarly to differences in the adopted version of the BPEL standard there is also a minor difference in the applied tools and its versions. FUSION builds on the plug-in version of the Eclipse framework for the process template creation and service binding while I apply the stand-alone ActiveVOS designer version 6.x [2]. The differences in run-time environment are only implicated by the different versions of supported BPEL executable process standards.

There is also a theoretical difference between the approaches of encapsulated (mediator) service implementation. The mediator services of FUSION are fully customized and deployed one by one which means that there is a unique mediator service instance created and deployed for every annotated enterprise service. On the other hand I use a common proxy service interface and implementation which forwards the requests and responses to and from the right services according to the attached invoke parameters (e.g. binding information of the standard enterprise service). My opinion is, that the latter solution is more general, is easier to maintain and saves lot of storage resources while does not influence the run-time performance significantly.

The next chapter summarizes my contribution to the results of Project FUSION.

4. My contribution to Project FUSION

As a BUTE team member I participated at almost every project meeting. Technical conversations and discussions resulting important methodological and technical decisions happened with my involvement. Besides personally I was responsible for supervising the creation of deliverables as follows:

- D2.1b – FUSION Semantic Profiler Specification [35],
- D2.2 – Development of FUSION semantic services analyzer [36]

I have created the following materials:

- Chapter 1, Semantic Profiler in the FUSION architecture (with sub-chapters 1.1 Components overview, 1.2 Semantic Profiling Walkthrough and 1.3 Versioning capabilities in D2.1b,
- Chapter 1.1 Components overview (within Chapter 1 Semantic Profiler), in D2.2,
- Chapter 1.2 Components tutorial (within Chapter 1 Semantic Profiler), in D2.2,
- Chapter 1.2.3 Creating transformations, in D2.2,
- Chapter 1.2.4 Publish, in D2.2,
- Chapter 1.3 Development summary, in D2.2,
- Chapter 1.4 Component’s prototype, in D2.2,
- Chapter 2.2 Transformation Editor, in D2.1b,
- Chapter 2.3 Enterprise Service Wrapper, in D2.1b,
- Chapter 2.4 Transformation Repository, in D2.1b,
- Chapter 3.2 Appendix of Profiling Lifecycle, in D2.1b.

Furthermore I have contributed to

- D1.3 FUSION Methodology (Chapter 7.2 The Semantic Uplifting Procedure with sub-Chapters 7.2.3 Definition of XSLT Transformations and 7.2.4 Generation and Deployment of the Mediator Service) and
- **D2.1 FUSION Approach** (Chapters 3.2.1.2 Semantic Profiler and 4.3.2.1 Transformation in Semantic Profiler).

Although some basic ideas and my motivation definitely originate from FUSION, the main results of my thesis’s and my own work can be separated from the results and work of project FUSION.
References


[32] D1.2 The FUSION Approach (Report, Month 6, PU), 2006.

[33] D1.4 FUSION Ontology (Prototype, Month 12, RE), 2007.

[34] D2.1 Specifications of the Semantic Services Analyzer (Report, Month 15, PU), 2007.


[38] D3.2 FUSION Integration Mechanism (Prototype, Month 21, CO), 2008.


Appendix

The aim of the appendix is to provide in-depth understanding of applied technology, tools and presenting further examples which were cut from the normal sections because of size limitation. This appendix contains several code samples (service data structure and interface descriptions) and screenshots (images of composite processes and process monitoring screen captures) to provide detailed technical information for experts as well.

<p>| Table 4.1 University-publication local schema |  |</p>
<table>
<thead>
<tr>
<th>Name of the complex type</th>
<th>Name of the attributes: connected data types</th>
<th>List of associated terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>City:string, State:string, Zip:integer</td>
<td>Address, Location, Permanent Address, Temporary Address</td>
</tr>
<tr>
<td>Author</td>
<td>Name:string, Address:Address</td>
<td>Publisher, Writer, Researcher, Creator</td>
</tr>
<tr>
<td>Book</td>
<td>Price:string, Title:string, Publisher:string, Author:Author</td>
<td>Tome, Collection, Paper</td>
</tr>
<tr>
<td>Monograph</td>
<td>Price:string, Title:string, Publisher:string, Author:Author</td>
<td>Hand-out, Article, Paper</td>
</tr>
<tr>
<td>Journal</td>
<td>Name:string, Editor:string</td>
<td>Article, Magazine, Essay</td>
</tr>
<tr>
<td>University</td>
<td>Name:string, Location:string</td>
<td>Highschool, Science Lab, Research Center</td>
</tr>
</tbody>
</table>

<p>| Table 4.2 University-publication global schema |  |</p>
<table>
<thead>
<tr>
<th>Name of the complex type</th>
<th>Name of the attributes: connected data types</th>
<th>List of associated terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>City:string, State:string, Zip:integer</td>
<td>Location, Permanent Address, Temporary Address</td>
</tr>
<tr>
<td>Book</td>
<td>Price:string, Title:string, Publisher:string</td>
<td>Tome, Collection, Paper, Work</td>
</tr>
<tr>
<td>Publication</td>
<td>Hand-out, Paper, Newspaper, Journal, Magazine</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Researcher</td>
<td>Scientist, Examiner, Publisher</td>
<td></td>
</tr>
<tr>
<td>University</td>
<td>Highschool, Science Center, Research Facility</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.3 Car retailer local schema**

<table>
<thead>
<tr>
<th>Name of the complex type</th>
<th>Name of the attributes: connected data types</th>
<th>List of associated terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>Make:string, Type:string, Version:integer</td>
<td>Vehicle, Four-wheeler, Saloon, Estate, Coupe</td>
</tr>
<tr>
<td>Inventory</td>
<td>New:Car, Secondhand:Car, OnDisplay:Car</td>
<td>Stock</td>
</tr>
<tr>
<td>Available</td>
<td>Cars:Car</td>
<td>Supply, ModelPalette</td>
</tr>
<tr>
<td>Crew</td>
<td>Name:string, Address:string, Phone:integer</td>
<td>Colleague, Junior, Senior</td>
</tr>
<tr>
<td>Autotrader</td>
<td>Stock:Inventory, Models:Available, Staff:Crew</td>
<td>Merchant, Agent</td>
</tr>
</tbody>
</table>

**Table 4.4 Car retailer global schema**

<table>
<thead>
<tr>
<th>Name of the complex type</th>
<th>Name of the attributes: connected data types</th>
<th>List of associated terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Code:integer, Name:string</td>
<td>Option, Equipment, Extra</td>
</tr>
<tr>
<td>Manager</td>
<td>Name:string, Address:string, Phone:integer</td>
<td>Boss, Head, Leader, Director</td>
</tr>
<tr>
<td>Members</td>
<td>Name:string, Address:string, Phone:integer</td>
<td>Colleague, Junior, Senior</td>
</tr>
<tr>
<td>Options</td>
<td>Item:Item</td>
<td>Equipment, Extra, Supplementary</td>
</tr>
<tr>
<td>Car</td>
<td>Make:string, Type:string, Version:integer</td>
<td>Vehicle, Four-wheeler, Saloon, Estate, Coupe, Cabrio, Coupe</td>
</tr>
<tr>
<td>Stock</td>
<td>New:Car, Used:Car</td>
<td>Inventory, Repository</td>
</tr>
<tr>
<td>Showroom</td>
<td>Size:string</td>
<td>DisplayHall, NewCar, Saloon</td>
</tr>
<tr>
<td>Staff</td>
<td>Manager:Manager, Members:Members</td>
<td>Team, Crew, Colleague</td>
</tr>
<tr>
<td>Carseller</td>
<td></td>
<td>Trader, Merchant, Agent</td>
</tr>
</tbody>
</table>
Figure 4.1 – Sample schema called Example2 for run-time steps prediction and measurement

Figure 4.2 – Sample schema called Example3 for run-time steps prediction and measurement

```xml
<wsdl:definitions targetNamespace="http://tempuri.org/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      <s:element name="Calculate">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="fullName" type="s:string"/>
            <s:element minOccurs="1" maxOccurs="1" name="weightPounds" type="s:double"/>
            <s:element minOccurs="1" maxOccurs="1" name="heightFeet" type="s:double"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
</wsdl:definitions>
```
Figure 4.3 Standard BMI calculator service description – requested input and output data structure

Figure 4.4 Proxy service calculator interface description – requested input and output data structure of the global schema
Figure 4.5 Upcast transformation code for global BMI service

Figure 4.6 Downcast transformation code for global BMI service
Figure 4.7 Orchestrated processes of the proposals in ActiveVOS designer

```
<bpel:assign name="Assign_of_Variables_Including_Down-cast_Transformations">
  <bpel:targets>
    <bpel:target linkName="L1"/>
  </bpel:targets>
  <bpel:sources>
    <bpel:source linkName="L2"/>
  </bpel:sources>
  <bpel:copy>
    <bpel:from>
      concat( $BmiBpelRequestMessage.CustomerFirstName ," ", $BmiBpelRequestMessage.CustomerLastName )
    </bpel:from>
    <bpel:to variable="CalculateLocalRequest">
      <bpel:query>ns1:fullName</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>$BmiBpelRequestMessage.CustomerWeightMetric * 2.20462262</bpel:from>
    <bpel:to variable="CalculateLocalRequest">
      <bpel:query>ns1:weightPounds</bpel:query>
    </bpel:to>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>$BmiBpelRequestMessage.CustomerHeightMetric * 0.032808399</bpel:from>
    <bpel:to variable="CalculateLocalRequest">
      <bpel:query>ns1:heightFeet</bpel:query>
    </bpel:to>
  </bpel:copy>
</bpel:assign>
```

Figure 4.8 Down-cast transformation in assign element of Proposal BPEL

```
<bpel:assign name="Assign_of_Variables_Including_Up-cast_Transformations">
  <bpel:targets>
    <bpel:target linkName="L3"/>
  </bpel:targets>
  <bpel:sources>
    <bpel:source linkName="L4"/>
  </bpel:sources>
  <bpel:copy>
    <bpel:from>substring-before( $CalculateLocalResponse/ns1:CalculateLocalResult/ns1:FullName ," ")</bpel:from>
    <bpel:to part="CustomerFirstName" variable="BmiBpelResponseMessage"/>
  </bpel:copy>
  <bpel:copy>
    <bpel:from>substring-after( $CalculateLocalResponse/ns1:CalculateLocalResult/ns1:FullName ," ")</bpel:from>
    <bpel:to part="CustomerLastName" variable="BmiBpelResponseMessage"/>
  </bpel:copy>
</bpel:assign>
```
Figure 4.9 Up-cast transformation in assign element of Proposal Direct

Figure 4.10 Performance monitor tool – 3 minutes long test for Proposal XSLT with 10 concurrent simulated clients