<table>
<thead>
<tr>
<th>Programme</th>
<th>Integrating and Strengthening the European Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategic Objective</td>
<td>Networked businesses and governments</td>
</tr>
<tr>
<td>Integrated Project / Programme Title</td>
<td>Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Application</td>
</tr>
<tr>
<td>Acronym</td>
<td>ATHENA</td>
</tr>
<tr>
<td>Project No</td>
<td>507849</td>
</tr>
<tr>
<td>ATHENA – Project Name</td>
<td>Guidelines and Best Practices for applying the ATHENA Interoperability Framework to support SME participation in Digital Ecosystems</td>
</tr>
<tr>
<td>ATHENA A - Project No</td>
<td>A8</td>
</tr>
</tbody>
</table>

**Deliverable Number: D.A8.3**

*Updated ATHENA tools and infrastructures supporting B5 Sub Projects*

**Work package – A8.4**

Leading Partner: SAP

Security Classification: Restricted (RE)

February 2007

Version 1.0
Table of contents

1 Executive Summary ................................................................................................................................ 1

2 Introduction ............................................................................................................................................ 2

3 Interoperability Barriers Identified .................................................................................................. 4
  3.1 Business process configuration verification .................................................................................. 4
  3.2 Service Granularity and Behaviour ............................................................................................. 5
  3.3 Data Exchange ............................................................................................................................... 6
    3.3.1 Wrong Instantiation of Data Models .................................................................................... 6
    3.3.2 Different Data Restriction .................................................................................................. 7
    3.3.3 Incompatible Syntactic and Semantic Representation of Data .............................................. 8
  3.4 Process Model Exchange ............................................................................................................... 9
  3.5 Agent-based Process/Interface Adaptation .................................................................................. 10

4 Proposed Adaptation Requirements and Solutions .......................................................................... 11
  4.1 Business Process Configuration Verification ................................................................................ 12
    4.1.1 Conceptual Solution for Business Process Configuration Verification ............................... 12
    4.1.2 Technical Adaptation of Maestro to support Rule based business level verification of CBPs .... 13
    4.1.3 Verification of Business Processes ...................................................................................... 15
    4.1.4 Semantic representation of business processes .................................................................. 16
    4.1.5 Expressing business level requirements ............................................................................. 18
    4.1.6 Overall Architecture ........................................................................................................ 20
    4.1.7 Results ................................................................................................................................. 22
  4.2 Service Granularity and Behaviour ............................................................................................... 23
    4.2.1 Conceptual Solution ........................................................................................................... 23
    4.2.2 Technical Solution ............................................................................................................ 24
  4.3 Data Exchange .............................................................................................................................. 27
    4.3.1 Wrong Instantiation of Data Models .................................................................................... 27
    4.3.2 Different Data Restriction .................................................................................................. 29
    4.3.3 Incompatible Syntactic and Semantic Representation of Data .............................................. 30
  4.4 Process Model Exchange ............................................................................................................... 37
    4.4.1 Conceptual Solution ........................................................................................................... 37
    4.4.2 Technical Solution ............................................................................................................ 39
  4.5 Agent-based process/interface adaptation .................................................................................... 40
    4.5.1 Conceptual Solution ........................................................................................................... 40
    4.5.2 Technical Solution ............................................................................................................ 41

5 Conclusion ............................................................................................................................................ 47

6 References ........................................................................................................................................... 48
List of Figures and Tables

Figure 1 From scenario description to technical solution of a barrier ................................................... 2
Table 1 Barrier Description for Business Process Configuration .............................................................. 5
Table 2 Barrier Description for Service Behavior and Granularity .......................................................... 6
Table 3 Barrier Description for wrong instantiation of data models ...................................................... 6
Table 4 Barrier description for different Data Restriction ..................................................................... 7
Table 5 Barrier description for incompatible Syntactic and Semantic Representation of Data ............. 8
Table 6 Barrier description for Process Model Exchange .................................................................... 9
Table 7 Barrier description for Agent-based Process/Interface Adaptation ......................................... 10
Figure 2 Interoperability barriers mapped to the selected carrier-shipper-scenario ............................ 11
Figure 3 The simplified model of a business process, its entities and their relationships ..................... 12
Figure 4 Technical Solution for business process configuration verification ....................................... 15
Figure 5 Current process modeling sequence ...................................................................................... 15
Figure 6 Sequence of process modeling including a verification step .............................................. 16
Figure 7 Graphical representation of the verification result in Maestro tool ........................................ 16
Figure 8 Taxonomy of the model ontology.......................................................................................... 17
Figure 9 Main classes and properties of the model ontology .............................................................. 18
Figure 10 Business process modeling sequence including expressing business level requirements .. 19
Figure 11 Use case diagram showing the usage of extended process modeling sequence ............. 19
Figure 12 Add rule dialog .................................................................................................................... 20
Figure 13 Structure of the Add Rule Dialog ....................................................................................... 20
Figure 14 Overview Architecture ....................................................................................................... 21
Figure 15 Ontological Knowledge base ............................................................................................. 22
Figure 16 Integration in ATHENA Tool set ....................................................................................... 22
Figure 17 Conceptual Solution ........................................................................................................... 24
Figure 18 Basic Architecture of Johnson ............................................................................................ 25
Figure 19 Technical solution ................................................................................................................ 26
Figure 20 Knowledge base for Conformance Testing ........................................................................ 28
Figure 21 Conformance testing service ............................................................................................. 28
Figure 22 EXPRESS transformation .................................................................................................. 30
Figure 23 – Diagram of A3 conceptual solution for data reconciliation ............................................. 30
Figure 24 – Diagram of Technical solution from A3 ........................................................................... 31
Figure 25 WSDL Analyzer, Main Window ......................................................................................... 32
Figure 26 Change the parameters of the matching algorithm ............................................................ 33
Figure 27 Displaying the matching results .......................................................................................... 33
Figure 28 WSDL structure (a) vs. Web service structure (b) ............................................................. 34
Figure 29 Typecast by relabelling of nodes ......................................................................................... 35
Figure 30 Pruning edges ..................................................................................................................... 35
Figure 31 Removing intermediate nodes ............................................................................................ 36
Figure 32 Mapping SOAP messages ................................................................................................ 37
Figure 33 Using POP* as conceptual solution for process model exchange (illustration) ................. 37
Figure 34 POP* Process dimension metamodel ................................................................................ 38
Figure 35. Adaptation of POP* using reference ontology and semantic annotation ............................ 39
Figure 36. POP* implemented in MPCE platform .............................................................................. 39
Figure 37 Integration of external services into a PIM4SOA model .................................................... 41
Figure 38 The main transformation rules that map PIM4SOA models to executable JackMM models 42
Figure 39 Bi-directional mapping of service operations and agent plans ........................................... 44
Figure 40 From WSDL to JACK code .............................................................................................. 46
1 Executive Summary

SMEs are more and more involved in complex business scenarios requiring mature technical interoperability solutions. In previous work in A8 (D.A8.2) we identified certain barriers using a scenario (Carrier-Shipper-Scenario) representing various situations where SMEs can interplay in different roles. We found out that although the barriers identified on a business level are not significantly different to the one faced by larger enterprises, SMEs require a different technical approach for solving these barriers.

The most important aspect for SMEs is to keep the interoperability costs low and the entire IT Architecture as simple as possible. This aspect is poorly addressed by current solutions, especially when maintaining interoperability. SMEs are often involved in a conflicting situation, where they have to decide between two extreme alternatives: either investing in complex software providing a rich set of functionalities or renouncing on a certain level of interoperability. The first alternative comes with the risk that investments made to resolve existing interoperability barriers in products of major ICT-Suppliers may end up having the opposite effect of increasing costs and less flexibility. With the second alternative SMEs run the risk of not being able to enter in a business relationship with a partner due the lack of technical interoperability.

Based on this insight adaptation potential for ATHENA results and tools for their optimized application satisfying SME needs was identified and documented in D.A8.2. Based on the results documented in D.A8.2 we focus in this deliverable on finding a smooth intersection between the two different alternatives outlined above. Firstly we require that a tool covers the minimal functional requirements needed, in order to guarantee that an SME will be able to establish business relationships with potential partners. Secondly we look at how we can adapt the ATHENA results and tools in such a way that their application in an SME environment can bridge the gap between business and technical level. Based on the identified barriers and their analysis using the scenario we provide the following conceptual and technical adaptation for existing ATEHENA results to be used at SMEs:

- For the barrier of different Business Process Configuration possibilities and their alignment with business level needs of SMEs a rule based verification mechanism of Cross Organizational Business Processes based on semantic technologies is provided.

- For the Service Granularity and Behaviour barrier, the conceptual and technical solution depicted in this document looks at service infrastructure provided by ATEHNA A5 tools. Here a description of an adapted solution to the problem of the different granularity and interaction protocols of services is provided on top of Johnson tool.

- For the Data exchange barrier among others the conformance testing tool, WSDL Analyzer tool and a modified modeling language by means of adding mechanisms that permit one to restrict models in terms of expressed rules is provided.

- Regarding the Process Model exchange barrier the POP* language is adapted to cover SMEs needs for translating two different process models

- For Agent-based Process/Interface Adaptation an Integration mechanism of external Web service interfaces described in WSDL is provided.
2 Introduction

Small and medium enterprises have special requirements on a technical system supporting their interoperability. They experience barriers when using an ICT solution for interoperability or generally when they aim to interact with their partners. These barriers were identified based on a selected scenario described in WD.A8.1 [ATH06C]. Although these barriers are mostly very similar to those that can be observed for larger companies, SMEs require that a software solution for interoperability can be easily introduced and integrated in their IT and organizational structure (see D.A8.1 [ATH06D]). At the same time, SMEs go through the same product introduction and maintenance life cycle as other enterprises, which means that the cost of introduction of software represents only one segment of the necessary investment of an SME in an interoperability solution. One significant cost segment for SMEs having an IT based Interoperability solution is actually maintaining the solution itself. Especially this is the segment where most SMEs experience difficulties to remain technically interoperable with their partners. Thus one big challenge for SMEs is also how to easily (re)configure a solution, whenever a change of their business partners and consequently service interaction and data exchange formats etc forces them to customize their running solution in such a way that it continuously reflects its current technical interoperability requirements.

Given this situation and based on the identified barriers, the ATHENA results were analyzed in terms of how far they help the SMEs to overcome the identified barriers. One central question is how SMEs become and remain technically interoperable in an effective way. We achieve this objective by a two fold strategy: First the analysis showed that some ATHENA results have to be adapted or even sometimes extended to help SMEs to overcome the identified barrier according to the special situation that SMEs are facing. This is for instance the case when looking at Business process configuration barrier addressed. In such cases we decided to show beside a conceptual adaptation of the tool to address the barrier, also to implement the concrete adaptation proposal. On the other side, our work also showed that SMEs prefer to have an easy usable and maintainable technical tool with minimum required functionality, compared to a full-blown heavy weight solution with maximum possible features, which comes at the cost of its expensive maintenance. In such cases SMEs require to have a clearly described best practice solution available provided by simple easy to use software, when it comes to solving a particular interoperability barrier. This is the case for instance when looking at the Data exchange barrier. In such cases we concentrate rather on showing an implementation blue print for SMEs, based on the existing ATHENA tools/results, instead of the implementation of a possible technical tool extension.

![Figure 1 From scenario description to technical solution of a barrier](image_url)
The conceptual basis for each ATHENA result adaptation/extension was set in WP A8.3 (see D.A8.2 [ATH06B]) by elaborating potential adaptation needs on existing ATHENA results. In WP A8.4 the solutions realizing the adaptation needs is accomplished first on a conceptual level and then on a technical level. This document is a report on these activities carried out in WP A8.4. Figure 1 illustrates the process of identifying an interoperability barrier based on the selected scenario until the realization of the technical solution. The figure should illustrate how the work reported in following is positioned in A8.

This document is structured as follows: In section 2 the identified interoperability barriers that were introduced detailed in D.A8.2 [ATH06B] are briefly recalled. In section 3 for each barrier the proposed conceptual and technical adaptation/extension of the affected ATHENA result is discussed. In section 4 we give the concluding remarks.
3 Interoperability Barriers Identified

Before we discuss in detail the conceptual and technical adaptations/extensions on existing ATEHNA results, we give in following a recall on the barriers identified, namely:

- Business Process Configuration
- Service Behaviour and Granularity
- different levels of Data Exchange barrier
- Process Model exchange barrier and finally
- Agent-based Process/Interface Adaptation

For each barrier, a brief introduction is given here and different dimensions of the barrier are illustrated using a template for each barrier.

3.1 Business process configuration verification

For SMEs the task of implementing interoperability and its maintenance according to the current situation of the company is of a complex and expensive task. Nowadays in particular SME focus on specific parts of a business process and depend on partners in a market to perform the additional parts of the process required to achieve a complete end-to-end business process. More and more SMEs prefer to buy interoperability solutions to support an easy and flexible implementation of their interoperability requirements. In most real life situations not fully configured ready to run cross-organizational Business Processes (CBPs) are delivered by software solution vendors to SMEs, since the vendors aim to cover the requirements of their different customers. These requirements are first known when introducing the process at SMEs. By configuration of CBPs we particularly mean that the SMEs are forced to integrate the set of available external business functionalities provided by partners and their internal systems in not fully configured CBPs in such a way, that its configuration fulfills the enterprise's situation.

The task of configuring a CBP according to the SMEs and its maintenance is very expensive and requires huge efforts from SMEs. They are forced to acquire not only business level knowledge to customize an interoperability solution, but also detailed technical know how about the purchased solution. At the end of the configuration of the CBP, it has to be verified to find out whether its current configuration in fact fulfills the business level requirements of that SME. This verification is today done mostly manually at the SME itself by a technical person. The configuration of a CBP is not a one time task, but a recurring task. This is due the fact that the business environment of enterprises has to change frequently according to frequently changing internal local business practices and capabilities of an enterprise, its business partners and the local legal regulations. Since the configuration of a SME is recurring task, an approach is required towards bringing a certain level of automation in verification of a CBP to help SMEs to lower the level of complexity to keep and maintain them interoperable.

As a solution to this barrier, we propose to semantically markup the CBPs. The semantic markup of business processes helps reducing communication efforts and increases the interoperability of business processes that cross company boundaries due to a consistent vocabulary and machine interpretable metadata for business processes. An upper ontology defining core business concepts related to CBPs is the basis requirement for this approach. Based on this upper ontology, the extension of a Business Process Modeling Tool supporting CBPs such as Maestro can semantically enrich business processes by representing them as ontology instances.

On top of these ontological representations of Maestro business processes, a user can define business rules on a process specific or on a cross-process level to define constraints on how a business process needs to be configured or how web services need to be integrated in processes. These semantic business rules represent business-level requirements defined by business analysts and build, together
with the business process ontology, an ontological knowledge base, which can then be processed by reasoning technologies. By means of reasoning over this knowledge base, business processes can be verified for compliance with the business rules defined and inconsistencies are reported to the user and not compliant parts of business processes are visually highlighted in Maestro.

For a broader definition of this barrier refer to deliverable DA8.2

[ATH06B]. A detailed conceptual solution as introduced briefly above will be given in 4.1 followed by a technical implementation applied on Business Process Design tool Maestro provided by A2.

<table>
<thead>
<tr>
<th>Template elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise levels concerned</td>
<td>Business, Process</td>
</tr>
<tr>
<td>Barriers to interoperability</td>
<td>Conceptual Barrier, Syntactical Barrier, Different Business Process Variants of the same Business Process, Gap between Business Level Interoperability requirements and technical configuration of a CBP</td>
</tr>
<tr>
<td>Interoperability problem</td>
<td>Business configuration problems in a CBP</td>
</tr>
<tr>
<td>ATHENA solution identified</td>
<td>A2 (Maestro, Nehemiah), Gabriel, A5 (Johnson)</td>
</tr>
<tr>
<td>Outcome of ATHENA results evaluation – Relevance to SMEs</td>
<td>Maestro: supports predefined processes but no verification of business process configuration during design-time</td>
</tr>
</tbody>
</table>

Table 1 Barrier Description for Business Process Configuration

3.2 Service Granularity and Behaviour

This barrier is closely related to how a partner exposes its business interface in a collaboration context. There are often several service providers on the market, which offer the same business functionality. If a SME wants to switch dynamically between them, it must support the behavioural interface provided by the service. In order to describe a service completely, it is not enough with the publication of its technical interface (e.g.: WSDL), the business protocol a service provides must be described as well. Another aspect to be taken into consideration when interacting with services is granularity. Granularity is defined as the amount of business function that gets performed in a single exchange of input/output messages [ROS06]. A high level of granularity (coarse grained), that is, a low number of big-sized operations, could cause vagueness and delay in the message exchanges. Conversely, a low level of granularity (fine grained), that is, a high number of small-sized operations, would result in complexity and increase in the number of exchanged messages.

The scenario introduced in WDA8.1 points out the interoperability barrier faced by a shipping SME trying to switch among different carriers. These carriers may provide typical services with similar functionality but with different interaction protocols and granularity, as expressed above. The two main companies offering services through the Web are UPS and FedEx. These two companies are the only ones who offer online tools with APIs for the integration with enterprise information systems. Both companies use REST-like architectural approach [FIE00] using HTTP/S as the communication protocol, using the POST method and embodying XML documents in HTTP packets.
Barriers to interoperability

Different service granularity and behavioural interfaces from different service providers

Interoperability problem

Services cannot be invoked in the same way by the consumer depending on the provider

ATHENA solutions identified

Conceptual solutions:
- PIM4SOA
- Technical solutions:
  - Lyndon, Johnson, WSDL Analyzer

Outcome of ATHENA results evaluation – Relevance to SMEs

The technical solutions don’t support dynamic execution of services and only work with WSDL files. SMEs may get locked-in to one provider if a solution is not provided.

Table 2 Barrier Description for Service Behavior and Granularity

For a broader definition of this barrier refer to deliverable DA8.2

[ATH06B].

### 3.3 Data Exchange

There are many possible causes to difficulty exchange of data, even more if the data is exchanged between different companies using different software. Data errors or data incoherence can be generated by different factors, including human distraction or faults, so, companies need ways to be sure if the data received is trustable.

#### 3.3.1 Wrong Instantiation of Data Models

The first barrier identified in data exchange was wrong instantiation. This barrier arises when a company uses or generates data not complaint to its data model, for example, not instantiating a mandatory attribute. In this case, even if different companies use same data model, errors can be found in the data exchanged.

<table>
<thead>
<tr>
<th>Template elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise levels concerned</td>
<td>Data</td>
</tr>
<tr>
<td>Barriers to interoperability</td>
<td>Conceptual barrier - Same model but wrong instantiation</td>
</tr>
<tr>
<td>Interoperability problem</td>
<td>Invalid data exchanged between companies</td>
</tr>
<tr>
<td>ATHENA solutions identified</td>
<td>Conceptual solutions: Validation of the data against the model</td>
</tr>
<tr>
<td></td>
<td>Technical solution: Conformance testing tool(A6)</td>
</tr>
<tr>
<td>Outcome of ATHENA results evaluation – Relevance to SMEs</td>
<td>Conformance testing tool available to SMEs</td>
</tr>
</tbody>
</table>

Table 3 Barrier Description for wrong instantiation of data models
3.3.2 Different Data Restriction

Another barrier in data exchange is when companies use the same conceptual data model but need to restrain parts of it to satisfy their business conditions. Sometimes restrictions are added to the original data model or only parts of it are used. This can work perfectly internally but creates a barrier when the company wants to exchange data with others. The restrictions in the model difficult the communication since the data sent may not fill attributes that are mandatory to the receiver implementation.

<table>
<thead>
<tr>
<th>Template elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise levels concerned</td>
<td>Data</td>
</tr>
<tr>
<td>Barriers to interoperability</td>
<td>Conceptual barrier - Different data restriction in two interacting companies</td>
</tr>
<tr>
<td>Interoperability problem</td>
<td>Data restricted by one company cannot be valid in the other even if they are using the same model</td>
</tr>
<tr>
<td>ATHENA Solutions identified</td>
<td>Conceptual solutions: Provide mechanisms to allow data restrictions required by the companies or adopt modeling languages that already has those mechanisms (e.g. STEP EXPRESS language)</td>
</tr>
<tr>
<td></td>
<td>Technical solutions: A6 tools (conformance testing tool, EXP2SCH, EXP2XSD)</td>
</tr>
<tr>
<td>Outcome of ATHENA results evaluation –</td>
<td>Conformance testing tool to validate message content and data restrictions. There is no mechanism in the Athena solutions to express rules restricting data models</td>
</tr>
<tr>
<td>Relevance to SMEs</td>
<td></td>
</tr>
<tr>
<td>Planned Adaptations</td>
<td>Modify ATHENA modeling languages, adding mechanisms that permit one to restrict models (express rules).</td>
</tr>
</tbody>
</table>

Table 4 Barrier description for different Data Restriction
3.3.3 Incompatible Syntactic and Semantic Representation of Data

The data model from one company can be very different from the one of other company. This diversity of models is not a problem when companies work alone, but when they want to exchange information with each-other, it presents an important barrier. The data exchanged based in different models normally appears in a different structure, but if the business are similar have the same meaning. The opposite can also happen, i.e. appear the same and have totally different meanings.

<table>
<thead>
<tr>
<th>Template elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise levels concerned</td>
<td>Data, Service</td>
</tr>
<tr>
<td>Barriers to interoperability</td>
<td>Conceptual barrier - Incompatible syntactic and semantic representation of data at each interacting partner</td>
</tr>
<tr>
<td>Interoperability problem</td>
<td>Different models adopted by the companies makes data exchange difficult as enterprises cannot exchange their data automatically</td>
</tr>
<tr>
<td>ATHENA solutions identified</td>
<td>- Conceptual solutions: Annotation of proprietary models according to common ontology to allow data reconciliation</td>
</tr>
<tr>
<td></td>
<td>- Technical solutions: A3 tools, WSDL Analyzer</td>
</tr>
<tr>
<td>Outcome of ATHENA results evaluation – Relevance to SMEs</td>
<td>- Adoption of the common generic ontology reflecting the business domain</td>
</tr>
<tr>
<td></td>
<td>- The WSDL Analyzer detects mismatches between data a service expects and provides mediation functionalities between services</td>
</tr>
<tr>
<td></td>
<td>- Relevant for SME which receive required interfaces of big companies which expect that their smaller business partners adapt to their interfaces</td>
</tr>
<tr>
<td>Planned Adaptations</td>
<td>Possibility to manipulate the generated mappings between heterogeneous interfaces.</td>
</tr>
<tr>
<td>Remarks</td>
<td>There exist other solutions for data mapping. However, none of them is directly concerned with Web service interface compatibility</td>
</tr>
</tbody>
</table>

Table 5 Barrier description for incompatible Syntactic and Semantic Representation of Data
3.4 Process Model Exchange

Interoperability that takes place at the process level is concerned with the two main aspects: (i) exchange process models between two companies for mutual understanding, (ii) connect process models of two companies for collaboration work. Main interoperability problem in process model exchange is concerned with translating one process model elaborated using a particular process modeling language to another language. Main barriers to interoperability are both syntactic and semantic incompatibilities.

<table>
<thead>
<tr>
<th>Template elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise levels concerned</td>
<td>Process</td>
</tr>
</tbody>
</table>
| Barriers to interoperability | Conceptual barrier  
Enterprises use different process languages to build process models  
There are syntactic mismatches of different process models |
| Interoperability problem | Enterprises cannot exchange their enterprise models automatically                                                                                             |
| ATHENA solution identified | Conceptual solution: POP* language / metamodel  
Technical solution: POP*/MPCE tool |
| Outcome of ATHENA results evaluation – Relevance to SMEs | POP* language (process dimension) seems adapted to cover SMEs need for translating two different process models  
The commercial availability of POP*/MPCE tool is not known, also the price and condition to use this tool is not known  
The relevance of this issue is more related to the case of SME-Big company process interoperability rather than in the case of SME-SME |
| Planned Adaptations | Investigate if adaptation needed at the POP* language level  
Investigate the availability and usability and price of POP*/MPCE tool |
| Remarks | Besides of ATHENA POP* solution, other solutions also exist in the State-of-the Art. For example, PSL (Process Specification Language) and UEML approach developed by UEML project  
Most of the techniques developed to day use Unified Approach. It consists in mapping different process models via a metamodel elaborated beforehand. |

Table 6 Barrier description for Process Model Exchange
### 3.5 Agent-based Process/Interface Adaptation

When integrating different partners and their services/processes, several adaptations have to be provided in order to enable seamless communication between the partners. The adaptations comprise harmonizing the involved processes and integrating the different data structures used by the collaborating partners. For SMEs, the problem and proper solutions for it are especially important, since resources for integration efforts are scarce.

<table>
<thead>
<tr>
<th>Template elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise levels concerned</td>
<td>Service, Process</td>
</tr>
<tr>
<td>Barriers to interoperability</td>
<td>Conceptual barrier and technological barrier</td>
</tr>
<tr>
<td></td>
<td>Enterprises use different interfaces for interaction</td>
</tr>
<tr>
<td>Interoperability problem</td>
<td>(Business) services cannot seamlessly interact</td>
</tr>
<tr>
<td>ATHENA Solution identified</td>
<td>PIM4SOA transformation to agent platform and integration of external Web services</td>
</tr>
<tr>
<td>Outcome of ATHENA results evaluation – Relevance to SMEs</td>
<td>Relevant for SME which have to adapt to collaborative processes described on a PIM level, but have already implemented service interfaces</td>
</tr>
<tr>
<td>Solution categories</td>
<td>PIM4SOA to agent transformation is a technical solution</td>
</tr>
<tr>
<td>Planned Adaptations</td>
<td>Integration of external Web service interfaces described in WSDL</td>
</tr>
</tbody>
</table>

Table 7 Barrier description for Agent-based Process/Interface Adaptation
4 Proposed Adaptation Requirements and Solutions

The carrier-shipper-scenario is used as a vehicle to show the occurrence of the interoperability barriers by identifying concrete instances of them in that scenario. But these barriers have a generic nature and are not restricted to a specific business domain. They occur in nearly every situation, where players in the market, larger companies or SMEs, with heterogeneous technical and business environments aim an (semi-) automated integration and usage of their partner’s business functionality. Thus the scenario does not intentionally represent a specific complete end-to-end-scenario for a single specific SME, it is rather the demonstrator of different situations in a business domain, where different SMEs may experience some of the barriers introduced. A summary of the different instances of the barriers occurring in the scenario is shown in Figure 2. While larger companies may have same or similar interoperability barriers, SMEs require special solutions suitable to be used in their specific technical and organizational environments.

![Figure 2 Interoperability barriers mapped to the selected carrier-shipper-scenario](image)

For each barrier we analyze the ATHENA results provided, which are in the scope of the barriers. These are for:

- Business Process Configuration: Maestro
- Service Behaviour and Granularity: Lyndon, Johnson, WSDL Analyzer
- Data Exchange: A6 Tools (Conformance Testing Tool, EXP2SCH, EXP2XSD), A3 Tools and WSDL Analyzer
- Process Model exchange: POP* language and MPCE tool
- Agent-based Process/Interface Adaptation: PIM4SOA
4.1 Business Process Configuration Verification

In this section we present a general approach for the formal verification of a process configuration that can be used in the verification of the CBPs. The approach enables automatic discovery of all parts of a business process that do not satisfy a predefined business requirement. This process we will call inconsistency detection.

We first introduce a general conceptual solution for the verification, which is independent from underlying technology (e.g. the Business Process Modelling tool). In a second step we show a concrete technical implementation of the proposed conceptual solution. In this phase we apply the proposed conceptual solution to the Maestro Business Process Modelling tool. Our analysis in [ATH06B] yielded to the result that a Business Process Modelling approach has to be extended with a rule based business level verification mechanism.

4.1.1 Conceptual Solution for Business Process Configuration Verification

A process is a sequence of logically related activities connected through several types of connectors (join, split, switch). For each activity a set of properties can be defined, like input, output, assigned resources. Finally, for each activity a set of (web) services can be bound. This means that according to Figure 3, an activity may be fulfilled as a service (or even a web service invocation), but it can be although accomplished by a manual human interaction.

Verification of the business process configuration is realized using formal methods. These methods seek to establish a logical proof that a system works correctly, i.e. that it is correctly configured. A formal approach provides:

- a modelling language to describe the system;
- a specification language to describe the correctness requirements; and
- an analysis technique to verify that the system meets its specification.

The model describes the possible behaviours of the system, and the specification describes the desired behaviours of the system. The statement the model P satisfies the specification is now a logical statement, to be proved or disproved using the analysis technique.

Since the goal of the inconsistency detection is to check whether a process description satisfy the required specification, it can be treated as a verification problem in which a modelling language to describe a system is defined through the above mentioned process model, a specification language...
corresponds to the consistency constraints that must be preserved and an analysis technique can be treated as inference process.

To formally prove the correctness of a model, the first decision is about what claims to prove. In our case, the claim is that there is no violation of the requirements regarding binding of services. It means that the system for business process verification has to return an error value in the case that an activity does not comply with a predefined binding. That can be formally described as follows [NAM06]:

\[
\text{isCompliant}(X) \leftrightarrow \neg \text{ErrorBinding}(X)
\]

\[
\text{ErrorBinding}(X) \leftrightarrow \text{Activity}(X) \land \text{WebService}(Y) \land \text{Binding}(X, Y) \land \neg M(\text{Property}(X), \text{Property}(Y))
\]

where

A is the set of activities from a business process

WS is a set of web services

Property(x) is a function that retrieves characteristics of an entity x. A characteristic is defined according to the underlying process model.

From the conceptual point of view, these constraints can be formally represented as declarative rules and a Rule Engine (i.a. Reasoner or Inference Engine) is used to evaluate these rules in the process of model verification.

Following outline summarizes the overall conceptual steps required for the solution of this interoperability barrier:

- Define a formal description for the business process model as an upper ontology.
- Express the business requirements of the SME regarding the interoperability situation of that enterprise according to the terms and concepts defined in that formal upper ontology
- Store the specific business process description (i.e. in our scenario the shipping process as described in WDA8.1) as a semantic enriched model of the business process
- Use an Inference Engine, which takes as input the declarative rules and the semantic instance of the business process model and infers whether the current business process configuration variant violates the existing set of the given rules.

4.1.2 Technical Adaptation of Maestro to support Rule based business level verification of CBPs

We adapt the Maestro Tool with a rule based verification mechanism to address the interoperability barrier of business process configuration at SMEs. Since our analysis yielded to the result that a rule based mechanism is not provided in Maestro, this is rather an extension than an adaptation of the tool.

One main task of the verification solution is capturing knowledge, i.e. business knowledge of business analysts. This knowledge is to be stored in some kind of Knowledge base (KB). After having been captured and stored, the knowledge should be accessible for verification purposes during business process modelling (design time). Ontologies allow for the formal and explicit specification of knowledge. They provide a powerful solution for building and analyzing knowledge bases. [STU98] state that "Ontologies mainly play a role in analyzing, modelling and implementing the domain knowledge". Additionally, today's ontological technologies are mature enough to provide sophisticated reasoning techniques that can be used for verification of ontological knowledge bases. In this context, [SUR03] say that "inference mechanisms for large ontologies have been developed and implemented". Examples of
ontology reasoners that are widely-used are Pellet\(^1\), RacerPro\(^2\), and KAON2 \(^3\). Due to these reasons, we decided to base the verification solution on ontology technologies.

Consequently, the knowledge is represented by an ontology-based knowledge base. The first step for building the knowledge base is a semantic representation of Maestro business processes.

More precisely, an Upper ontology acting as the model ontology containing all relevant concepts and relationships regarding business processes is needed. At this point, it has to be decided whether an existing ontology is used or, due to the lack of a suitable existing ontology, a new ontology is modeled. One example of an Upper ontology available for use is OWL-S\(^4\), which provides an Upper ontology for Web services and business processes. Another one would be Web Service Modeling Ontology (WSMO)\(^5\), which is ontology for describing various aspects related to semantic Web services. The motivation behind the development of these Upper ontologies was rather dynamic Web service discovery, selection and their composition into business processes, where our objective is the verification of already existing business processes built from a Web service repository. Furthermore, these Upper ontologies do not support the process abstraction concept developed in context of ATHENA. Due to these reasons, we decided to build a new Upper ontology for Maestro business processes.

Please recall that the solution should enable the user, i.e. the business analyst, to reflect business level requirements for business processes. For this purpose, he should be able to define conditions and constraints that should hold in certain ontologies of the knowledge base. This makes the use of rules necessary. These rules can be expressed using a rule language. Therefore, the used ontology markup language should support the definition of rules in some way.

Since the language OWL (Web Ontology language) standardized by W3C is best supported by ontology management tools at the moment, we decided to use OWL for building the ontologies and for basing the KB on. More precisely, we are going to use OWL-DL, which is one of the three sublanguages of OWL. As the name implies, it is based on description logics. OWL-DL guarantees decidability while still offering great expressiveness. Concerning the definition of rules on top of ontologies, OWL-DL itself does not include rule support, but there exist extensions to OWL-DL that offer rule support, the most well-known one being SWRL [HOR03]. Having decided on using OWL-DL with a rule extension as technology for building the ontological KB, a solution for reasoning this KB and for smoothly integrating this solution into the existing ATHENA Maestro tool is to be found. One challenge to overcome in this context is the reasoning of the OWL-DL KB extended by rules.

Early research on the integration of DL based knowledge bases and rules showed that reasoning over such a KB is not a trivial task due to undecidability when reasoning over DL based knowledge bases with rules [LEV98]. Moreover, when starting with a OWL-DL KB, which is decidable during reasoning, and combining it with a KB containing rules, which is decidable during reasoning as well, it is not given that the resulting KB is still decidable when reasoning occurs [ROS05].

Most state of the art DL reasoners implement tableau algorithms and achieve good performance when reasoning over knowledge bases with large TBoxes and lower performance on ABox reasoning. By contrast, the DL reasoner KAON2 obtains relatively low performance on TBox reasoning and very good performance when reasoning knowledge bases with large ABoxes (Motik and Sattler, 2005). Because of the good ABox reasoning performance KAON2 is the tool that supports our requirements best. Therefore, we will utilize KAON2 in our approach.

---

\(^1\) Pellet is an open-source Java based OWL DL reasoner. It is based on the tableaux algorithms developed for expressive Description Logics. Please see [http://www.mindswap.org/2003/pellet/](http://www.mindswap.org/2003/pellet/)

\(^2\) RacerPro is an OWL Reasoner, which supports the full OWL standard. It was developed at Hamburg University of Technology. Please see [http://www.sts.tu-harburg.de/r.f.moeller/racer/](http://www.sts.tu-harburg.de/r.f.moeller/racer/)

\(^3\) KAON2 is an infrastructure for managing, reasoning, and inferencing OWL-DL, SWRL, and F-Logic ontologies. Please see [http://kaon2.semanticweb.org/](http://kaon2.semanticweb.org/)

\(^4\) [http://www.w3.org/Submission/OWL-S/](http://www.w3.org/Submission/OWL-S/)

\(^5\) [http://www.wsmo.org](http://www.wsmo.org)
According to discussion above and the conceptual solution introduced before, we propose following high level technical solution as shown in Figure 4:

![Figure 4 Technical Solution for business process configuration verification]

The following summarizes the steps taken for realization of the technical solution:

1. We use OWL-DL6 as the language for the definition of the upper ontology for business processes modelled in Maestro.

2. Use the Semantic Web Rule Language (SWRL) [HOR03] as the rule language to express the business requirements as constraints over the terms and concepts provided in the upper ontology. Since SWRL is not decidable, we use the DL-Safe subset [MOT04] of SWRL to ensure the decidability of the approach. From an implementation point of view, all the constraints shown in the scenario can be expressed as DL-Safe-rules.

3. Extend the functionality of the Maestro in such a way, that the business process models will additionally be stored as OWL-DL-Instances of the upper ontology defined in step 1.

4. Use KAON27 as the Inference Engine.

5. In following we discuss the realization steps of the technical solution shown in Figure 4 more detailed.

### 4.1.3 Verification of Business Processes

People with business knowledge, who we call business analysts, should be able to define business process specific constraints to capture business level requirements. The more technical experienced user, whom we call process modeler, could then, during modeling of business processes, assure that the processes are consistent with the companies' regulations and requirements on the business level by verifying them. This kind of verification mechanism is needed in terms of business process analysis to avoid system failures resulting from wrongly modeled or wrongly configured business processes. The current sequence of business process modeling with the ATHENA tool set is depicted in Figure 5.

![Figure 5 Current process modeling sequence]

---

6 [http://www.w3.org/TR/owl-guide/](http://www.w3.org/TR/owl-guide/)
7 [http://kaon2.semanticweb.org/](http://kaon2.semanticweb.org/)
At design time, business processes are modeled in Maestro and then saved to the business process repository. At runtime, the business processes can be enacted by the Nehemiah runtime engine.

The verification mechanism proposed is supposed to be integrated in this sequence right after the business process modeling step, which is illustrated in Figure 6. The figure outlines that after the business process has been modeled or modified, it is first being verified by a reasoner in a business process verification step, before it is saved to the business process repository.

![Figure 6 Sequence of process modeling including a verification step](image)

Figure 7 shows a snapshot from the Maestro tool illustrating how the verification process is integrated visually into the Maestro. The snapshot shows that the red highlighted process step in the selected CBP violates the rule in the KB described in the Text area in the bottom part of the snapshot.

![Figure 7 Graphical representation of the verification result in Maestro tool](image)

4.1.4 Semantic representation of business processes

CBPs are modeled and enacted across several businesses. By means of the semantic information, software agents can access and merge business processes (semi-) automatically. Furthermore, the semantic representation of business processes is necessary for the verification mechanism because defining the structure, the concepts, and the relationships of Maestro business processes semantically builds the basis for being able to express and to capture the business level requirements, i.e. the constraints. The semantic representation in a way provides the vocabulary for expressing these
constraints. Practically, the process modeler needs to be able to save a semantic process instance, i.e. instance ontology, of the business process he is currently modeling in Maestro. This, in a first step, requires the creation of an Upper ontology, which can be called a semantic process model or model ontology.

It contains all relevant classes, concepts, and relationships regarding business processes and acts as a blueprint for each instance ontology that is created. In the following we further detail the Model Ontology developed:

The model ontology, which was modeled with the ontology editor Protégé⁸, defines all relevant classes and properties that can be used for the description of CBPs. An OWL ontology consists of classes, properties, and individuals. In order to avoid redundancy, it is common to model the classes and properties in a model ontology and to store the individuals, i.e. the actual business process instances, in a separate OWL file, which we call instance ontology. The model ontology described in this section is regarded to be an Upper ontology (also known as Top-level ontology) since it describes very general concepts that give general notions on terms in the field of business processes. The taxonomy of the model ontology is shown in Figure 8.

The root class of the model ontology, which is the class owl:Thing for all OWL ontologies has the following subclasses (Figure 9):

- Graph
- Business Process
- Partner
- Node
- Edge
- Task

Graph represents a Maestro graph containing one or several business processes. The BusinessProcess class is related to nodes and edges by the object properties hasNode and hasEdge, respectively. The class Partner stands for one business partner, who is linked to the BusinessProcess class by its object property hasPartner. A Node represents any kind of node contained in a business process. By means of the object properties isPredecessorOf, isDirectPredecessorOf, isSuccessorOf, and isDirectSuccessorOf each individual of it is in some way linked to all other nodes within the according business process. Its subclasses are ActivityNode (representing an activity in a business process),

⁸ Protégé is an open source ontology editor. See http://protege.stanford.edu/
SenderNode (standing for a node that enables outgoing communication in a CBP), ReceiverNode (standing for a node that enables incoming communication in a CBP), and CoordinatorNode (representing a node that aligns and manages the execution of a business process). The class Edge links two individuals of the Node class within one graph with each other. An individual of the Task class refers to a task profile that can be attached to a node of a business process. The subclasses of this class are UserTask, Private-ServiceTask, SenderServiceTask, and ReceiverServiceTask. The main classes of the model ontology and its properties are visualized.

Each class in the following diagram is shown as a rectangle. Inside the rectangle, all object properties and datatype properties of the according class are shown. Object properties are indicated by a small blue rectangle, datatype properties by a small green rectangle next to their names. Thick black arrows depict subclass relationships. Thinner black arrows show object property relationships.

Figure 9 Main classes and properties of the model ontology

4.1.5 Expressing business level requirements

A mechanism is needed that provides business analysts with the possibility to express business level requirements. These business level requirements are best expressed by constraints on business processes. Such a business level requirement that can be captured could be, e.g., a constraint on which Web services have to be called in a certain activity of a business process, or after which activity of a business process a certain user task has to be performed. Although these constraints are related to a certain business process, they should be decoupled from the actual technical representation of the business process, so that they can persist, in case the belonging business process is redesigned or even deleted. Thus, the constraints are to be stored separately from the business process itself.
We should extend the business process modeling sequence shown in Figure 6 by the extended modeling sequence depicted in Figure 10 and its usage further illustrated in a use case diagram in Figure 11.

![Figure 10 Business process modeling sequence including expressing business level requirements](image1)

Figure 10 Business process modeling sequence including expressing business level requirements

![Figure 11 Use case diagram showing the usage of extended process modeling sequence](image2)

Figure 11 Use case diagram showing the usage of extended process modeling sequence

Based on the carrier shipper scenario, the following example rule will be expressed: “After the sales order was created, the rate and the routing code need to be calculated.”

Expressed in terms of literals and the vocabulary given by the model ontology described earlier, this rule could be defined as follows in Figure 12:
The "Add rule" - dialog constrains the vocabulary that can be used for defining business rules by offering only the available vocabulary through combo boxes. For each literal, no matter if it is a body or a head literal, the same vocabulary exists. Each literal of a rule consists of a predicate and a set of terms. The predicates that may be used for modeling the business rules are confined to binary predicates, also called properties. Binary predicates, as the name suggests, have exactly two terms, which is why the literals of the "Add rule" - dialog always consist of one property and two terms. Two types of properties exist, namely object properties and datatype properties. The properties available in the "Add rule" - dialog reflect the object properties and datatype properties defined in the model ontology. In order to make the "Add rule" - dialog more user friendly, it does not show the names of the properties as modeled in the ontology, but it displays a description of them that is easier to read for the user. The two terms that belong to every property are also called domain and range. Figure 13 identifies the domain, the property, and the range of an example literal displayed in the "Add rule" - dialog. The domain of a literal, with regard to the "Add Rule" - editor, is always some kind of variable. The property may be an object property or a datatype property. In case it is an object property, the range is a variable, whereas in case the property is a datatype property, the range is a string value.

In order to identify contradictions between the rules and the belonging business process, the individuals that are not rule compliant somehow need to be identified by the rule.

For this purpose, the rule is, after having been expressed by the user, restructured. The following expression representing the rule

\[ \text{Condition}1 \land \ldots \land \text{Condition}n \Rightarrow \text{Consequence} \]

will be restructured internally in following way, in order to make it identify objects that are fault objects according to the basis rule above:

\[ \text{Condition}1 \land \ldots \land \text{Condition}n \land \neg \text{Consequence} \Rightarrow \text{Fault} \]

### 4.1.6 Overall Architecture

The architecture of the prototypical verification extension consists of the following three main parts: The Maestro application provides the UI, the KAON2 framework, which is responsible for ontology and rules management and additionally for reasoning of the ontological Knowledge base, and the ontological KB containing OWL ontologies and rules. The functionality offered by the KAON2 framework can be separated in three parts, namely Reasoning Engine, Ontology Management, and Rules Management, as shown in Figure 14. It gives an overview on the architecture of the verification extension. On the one hand, Maestro enables the process modeler to save instance ontologies of business processes, which
are then created through KAON2 Ontology Management and saved as OWL ontology files. On the other hand, it allows for the creation of rules by the business analyst, which are processed in KAON2 Rules Management and saved to rule files. These rule files together with the model ontology and the instance ontology build the so called ontological knowledge base. The model ontology provides all relevant concepts and the relationships between them regarding business processes, whereas the instance ontology, which can be seen as an instance of the model ontology, represents an actual business process modeled by the process modeler. On the basis of this ontological KB, reasoning can be conducted by the KAON2 reasoning engine and the results can be passed through to the Maestro UI and thus to the user, i.e. the process modeler.

The ontological knowledge base is threefold. It consists of the model ontology for business processes, the instance ontology representing the business process modeled by the process modeler, and the rules file containing the rules expressed by the business analyst.

The model ontology is an OWL-DL ontology that contains all concepts and properties of Maestro business processes in general, so it is regarded to be TBox knowledge. It is business process independent, thus it is static, and it builds the basis for each process-specific KB. Accordingly, only one global OWL ontology file for the model ontology exists. The instance ontology is an OWL-DL ontology containing business process specific information. It consists of individuals of the concepts that are described in the model ontology. Therefore, the knowledge contained is ABox knowledge. Consistent to the idea of ontologies, the instance ontology has to import the model ontology, since it uses knowledge from the model ontology. The creation of instance ontology is to be started by the process modeler through the Maestro UI after he modeled a business process. The instance ontology is stored in an ontology file that is named according to the name of the business process it belongs to.

On the basis of the existing KB consisting of the model ontology and the instance ontology, rules can be expressed by the business analyst. The rules are saved in a separate OWL ontology file, only containing information on the rules. These rules are business process-specific as well. The rules ontology is also imported by the instance ontology, so that the knowledge from the rules ontology is also available in the instance ontology. To clarify the structure of the ontological KB, it is visualized in Figure 15.

![Figure 14 Overview Architecture](image-url)
4.1.7 Results

Figure shows the extension of the Maestro Tool and how it is integrated into the available ATHENA results in context of business process and service management provided from A2 and A5.

Summarizing following functionality has been provided by the rule based extension of the BPM functionality in ATHENA:

- Provide a Modelling Tool to create SWRL Statements as constraints representing the business requirements regarding the activity-service-assignments
- Add the rules to the Knowledge Base
- Store a Business Process model not only in tool specific format (in this case Maestro), but also as an OWL-DL-Instance according to the developed upper ontology in the Knowledge Base.
- Provide a reasoning mechanism to infer whether a selected OWL-DL-Instance of a selected business process violates the set of rules assigned to that business process or not.
- In case of detecting a violation, provide a mechanism to visualize which rule at which process step is violated.

Performance and fast reasoning response time was not considered as a critical requirement for the technical solution, since the verification mechanism is done during design time of a business process on the business process models and does not affect the runtime behaviour of a business process.

4.2 Service Granularity and Behaviour

This section depicts a solution for the interaction of SMEs with different providers in the context of the scenario defined in A8. In this case, carriers are providing services through different interaction protocols and defining different levels of granularity. SMEs find it difficult to adapt to the different constraints so they normally get locked-in with one carrier. This barrier is extensively described in [ATH06B] and a summary is provided in section 2.2 of this document. The first sub-section will introduce the conceptual solution without the underlying technologies’ details, whilst the next sub-section will depict the proposed technical solution.

4.2.1 Conceptual Solution

As mentioned in deliverable D.A8.2 [ATH06B], there are two main obstacles that must be pulled down in order to allow shippers’ to transparently and automatically invoke services from different carriers:

- The absence of a common standard for defining service interfaces and exchanging messages.
- Different behavioural interface and granularity of services.

Within this scenario, client application developers are forced to manually construct different requests for each carrier’s online API. A mechanism that allows a shipper to consume services from different carriers transparently is needed. This mechanism should provide a sort of middleware infrastructure between shippers and carriers that will allow the former to integrate carrier’s services within their internal information systems and to switch between carriers transparently.

The list of requirements for this infrastructure includes:

- Provision of a common interface definition for the carriers’ services.
- Connectivity with the carrier’s online APIs
- Transformation of messages.
- Message buffering.
- Routing and addressing of messages
- Support for more than one transport protocol

Figure 17 shows the role of the middleware architecture within the A8 scenario. Different shippers will require sending messages using different transportation protocols and messaging styles (RPC or document-style). The middleware architecture, broadly, will be in charge of receiving those messages, transforming them from the shipper’s schema to the carrier’s required schema and vice versa, buffering shipper’s request messages and carrier’s reply messages, (document-style) and routing them to the appropriate endpoint through different transport protocols.
4.2.2 Technical Solution

The middleware architecture proposed in the previous section is similar to what nowadays is known as an Enterprise Service Bus (ESB). Indeed, the requirements depicted look very similar to the capabilities required for an ESB or at least form an important part of what an ESB should provide [ROB04]. The term was introduced by the Gartner Group to define a new type of application integration middleware: “An Enterprise Service Bus (ESB) is a new architecture that exploits Web services, messaging middleware, intelligent routing, and transformation. ESBs act as a lightweight, ubiquitous integration backbone through which software services and application components flow.” (Source: Roy Schulte, Gartner)

The ATHENA Service Bus represented mainly by the Johnson and Lyndon tools developed within A5 project provides a degree of functionality close to the A8 scenario’s requirements. However, in order to fully comply with these requirements some functionality should be added.

Figure 18 shows the basic architecture of the Johnson tool. The core features of Johnson include the creation, sending and reception of SOAP messages. Besides, as Johnson has been designed with modularity in mind it is quite easy to extend it with additional functionalities. A list of the functionalities that have been added to Johnson using the extension mechanism so far would include:

- Support for advanced Web Services specifications, such as WS-Addressing or WSReliableMessaging.
- On-the-fly generation and storage of semantic annotations related to messages.
- Message reconciliation between service provider and service consumer.
- Content-based routing of messages in a service-oriented architecture.
Messages, endpoints, processing chains and processing modules are the main concepts Johnson employs. A message can be any piece of data that can be transmitted as the payload of an HTTP message. However, in the context of ATHENA only SOAP messages have been taken into consideration. The concept of an endpoint has been introduced to abstract from the URLs on which a message is received or to which a message is sent. Each endpoint is associated to a processing chain composed of a number of processing modules. Each processing module implements an aspect of the processing of a message, whether inbound or outbound. Any kind of processing can be applied to a message through the implementation of processing modules (e.g.: WS-Addressing, WSReliableMessaging).

On the other hand, Lyndon can be seen as the design-time counterpart of the Johnson tool. It analyses WSDL files and automatically configures Johnson for playing either the role of a consumer or provider of the described service. Lyndon parses a WSDL file and determines which endpoints need to be created and which processing chains need to be assigned to them. Determining which processing modules have to be included in the processing chain takes into account information extracted from the WSDL files and from the user’s choices.

Up till now, Johnson and Lyndon (currently they form an integrated solution called Johnson) have been presented as a pair of tools bound to specific standards such as SOAP and WSDL. Under these premises, ATHENA solutions seem to fall short of the desired capabilities. The main hindrance would be the message formats employed by the two main players in the package carrier niche (i.e.: REST architecture). Currently, Johnson is only able to handle SOAP messages and no compatibility with document-style messaging exists. However, the architecture of Johnson has been designed in such a way that it is possible to provide specialised processing modules and processing chains that can handle any type of payload, as long as it is transported over HTTP.

The main abstraction that Johnson deals with is messages. In particular, the notion of service was deliberately kept outside of Johnson. One reason for that is that the notion of service is particularly vague and it would not be wise to tie Johnson to a particular representation of services (e.g.: WSDL). Another reason is that a lot can already be achieved by simply working with messages, as is already hinted at with approaches to services such as REST.

Having said that, an architecture for the technical solution is proposed in Figure 19. The architecture builds on the Johnson and Lyndon tools but suggests some modifications on the use and functionality of these tools. A set of steps have been defined to show the operation of the proposed solution:

1. A set of WSDL interfaces are created and uploaded to Lyndon. There exists a document-style and an rpc-style WSDL for each service. These services are defined taking into account the online APIs provided by the carriers.
2. These WSDL files are used to configure Johnson as explained in [ATH06A]. An instance of Johnson is configured as a service consumer and another one as a service provider. The service consumer could be placed in a third party broker together with the service provider or could be installed at the shipper.

3. The shipper SMEs will send the request message corresponding to the selected service using rpc-style or document-style messaging as desired. Besides, the shippers will have the choice to run the solution using the Johnson user interface or access Johnson's web service interface integrating it in their shipping solutions.

4. The Johnson provider instance will receive the messages, process them according some SOAP to REST transformation rules and route them to the designated carrier. The processing modules will implement the two-way transformation from messages based on SOAP to REST-like messages and vice versa.

5. Transformed messages will be sent to the different carriers that will process the XML documents and send the reply. Processing chains in the Johnson’s service provider instance will transform the XML document to SOAP messages and send the reply back to the customer with the necessary data. The Johnson instance will buffer all the unnecessary information sent by the carriers for subsequent requests from the shippers.

![Figure 19 Technical solution](image)

In order to make this architecture work the Johnson tool should be modified and improved in the following way:

- Different WSDL files should be defined in order to provide a uniform interface to the shippers.
- Several processing modules should be implemented in order to configure Johnson with the provided WSDL files, process the message exchange between shippers and carriers and to store the information sent by the latter in a buffer.
- A schema should be defined for the messages that are sent using document-style messaging. Message bodies should be built according to this schema that should be stored in both instances of Johnson to be used by the shippers and the processing chains at the provider instance.
- A user-friendly interface allowing the easy definition of requests should be provided by Johnson.
The proposed technical solution is considered relevant for SMEs because it will allow them to interact with carrier services in a homogeneous way without incurring into further efforts and costs. The mediator solution could be run by a third party provider of interoperability services as mentioned in [ATH06B]. This way, SMEs could easily switch from one carrier to another through an invocation to different services offered through a web portal.

4.3 Data Exchange

Based on the data exchange barriers identified and its conceptual solutions, technical solutions for each one were identified within ATHENA tools.

4.3.1 Wrong Instantiation of Data Models

4.3.1.1 Conceptual Solution

The first barrier identified was the wrong instantiation of data models. This barrier can be caused by many factors, like transmission errors, human errors and data manipulation errors by the computer system. So, the conceptual solution needs also to consider this kind of issues when it's conceiving.

To solve the wrong instantiation of data models the conformance testing techniques can be applied. The conformance testing provides, based on the model, a way to detect and identify the errors. Based on these identified errors reports will be generated which an SME can use to correct the occurred errors.

This conformance testing is based on two different stages:

- Model Structure validation,
- Model Semantic validation.

The first stage will validate the structure of the data using the company model stored in a knowledge base. The second stage is the semantic validation, which will validate rules and constraints defined on the data, also using company’s model knowledge base.

To ensure the correct exchange of data between different companies, it must be defined also where the data tests will be applied. In this case, two different solutions can be adopted:

- Middle point testing
- Start and end point testing

The middle point testing is when the conformance testing of data file is done in a central system that manages the exchange of data. With this test the companies must trust in the transmission, since when the company receive the file, it will be not tested again.

In start and end point testing the company system has the conformance testing embedded, so before sending the file can validate it. When receive one file can be also validation tests. In this case transmission errors it will be detect when the file received is tested, nevertheless each file is validated two times and this spend more time.

4.3.1.2 Technical Solution

For this barrier, and based on its conceptual solution, the technical solution for the wrong instantiation of data models identified was the ATHENA Conformance testing tool.
This tool allows performing content level conformance tests in XML data. It is assumed that the data model is represented in the XSD (XML schema) for the basic structure and constraints and complemented by SCHEMATRON (language used based on assertions) for the more complex rules.

In the following figure an example of an entity “DeclaredValue” is represented in XSD and its rules in SCHEMATRON. So, these two schemas will compose the knowledge base used by the Conformance test Framework when the XML file, which needs to be validated, contains an entity “DeclaredValue”.

```xml
<xsd:element name="DeclaredValue" type="DeclaredValue" substitutionGroup="ex:entity" nillable="true"/>
<xsd:complexType name="DeclaredValue">
  <xsd:complexContent>
    <xsd:extension base="ex:entity">
      <xsd:sequence>
        <xsd:element name="Value_" type="xs:normalizedString" nillable="true" minOccurs="0"/>
        <xsd:element name="CurrencyCode" type="xs:decimal" nillable="true" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

```xml
<pattern name="DeclaredValue - WR1">
  <rule context="DeclaredValue">
    <assert test="not((Value_)) or ((CurrencyCode) and (CurrencyCode = 'USD')) or @ref">ERROR</assert>
  </rule>
</pattern>
```

Figure 20 Knowledge base for Conformance Testing

With this tool it is possible to accomplish the goals described in the conceptual solution, since it allows the structure and semantic validation. The tool is available as web-service, so it is possible to apply the two methods mentioned in the conceptual solution. Thus, when one company wants to check one XML file, it calls the conformance testing web-services to check it. The conformance testing tool will send back a report with errors found in the data file. In the figure below, the whole conformance testing process described is shown in detail.

Figure 21 Conformance testing service
4.3.2 Different Data Restriction

4.3.2.1 Conceptual Solution

The second barrier identified was the different data restriction. In this case the solution identified to a company that wants to exchange data with others companies, and these companies don't have same restrictions (like the example described in Different Data Restriction interoperability barrier), is basically the same as in previous case.

Basically the companies want to be alert when their required restriction on data are violated. So, this can be considered as data error. Applying the conformance testing described in the previous conception solution, and if the restrictions are defined in the model, the data errors in terms of restrictions will be detected.

A difficulty can arise when the restrictions are not defined in the data model, for example because the data model language doesn't support restrictions. When this occurs the conformance testing doesn't have direct support to its tests. So, it must be ways provided to improve this situation, like alternative description languages for the model or mapping restrictions directly to Conformance testing knowledge base.

4.3.2.2 Technical Solution

For the different data restriction barrier, the technical solution was found in A6 tools. The tools identified were the Conformance testing tool, EXP2SCH and EXP2XSD.

As in the previous solution, the use of the conformance testing tool can allow the identification of errors. In this particular case, the conformance testing will check the data conformance with the restrictions defined by the company.

In the conceptual solution, the problems were mentioned that one company has to face with when defining restrictions in the data model. To help the companies to describe their restrictions, the EXPRESS language was identified as a possible solution to describe the data model. This language is a good solution when the company wants to define restrictions in its models, once it supports all kind of restrictions. Another advantage of this language is is used by ISO to describe the Application Protocols. Thus, this is suitable if the company wants to apply standards or interconnect different models.

If the company adopts EXPRESS to describe its data model, the EXP2SCH and EXP2XSD (from A6 project) can also be used, which allow the transformation of the model defined in EXPRESS to XSD and SCHEMATRON, used in the conformance testing. An example of these transformations is shown in the figure below, where the entity, “DeclaredValue” defined in EXPRESS is transformed into XSD and SCHEMATRON.
4.3.3 Incompatible Syntactic and Semantic Representation of Data

4.3.3.1 Conceptual Solution 1

The barrier identified with the incompatible syntactic and semantic representation of data between the companies can be minimized by using an improved method to reconcile the exchanged data.

In A3 project a method was defined to solve this issue, by mapping the schemas of the companies exchanging data. A reference ontology is build based on these schemas which serves the data reconciliation. In following figure the conceptual solution is depicted in more detail.

Figure 23 – Diagram of A3 conceptual solution for data reconciliation
4.3.3.2 Technical Solution 1

To allow data reconciliation between companies, the ATHENA A3 sub-project developed some tools to solve this issue. The main purpose of these tools is to create a reference ontology which allows the data exchange when this data is based upon different models.

So, the first step to apply this solution for the incompatible syntactic and semantic representation of data, using ATHOS, is to define the reference ontology. This ontology will be used by A* to create the annotation repository that ARGOS uses to generate the reconciliation rules. These rules will be used by ARES to the reconciliation data, allowing companies to exchange data based on different models. Below the usage of the A3 tools to overcome this barrier is shown.

![Diagram of Technical solution from A3](image)

4.3.3.3 Conceptual Solution 2

Data exchange per se within an SOA environment is unproblematic, if partners comply with the same data structure. However, basic data interoperability problems occur when one of the partners uses a slightly different or even heterogeneous data structure in its interface. In this case, parameter passing requires an additional mapping or mediation step which transforms data from one schema to the other. This step can be prepared manually at design time by formulating reconciliation rules or by applying schema matching algorithms which help to create mappings between schemas automatically.

The WSDL Analyzer is first and foremost a tool for detecting similarities and differences between WSDL files. The tool can be used to find a list of similar services (brokering). Since the similarity algorithm produces a mapping between WSDL files, the tool can also be used for supporting mediation between services.

A WSDL specification is the description of a Web service including a description of its interface and a description of where the actual implementation exists and how it can be used. The basic idea and assumption underlying the WSDL Analyzer is that, if two services are conceptually similar they are more
likely to also be structurally similar than otherwise. In a WSDL description of a service, data is encoded in XML, i.e. structured trees. The algorithm assumes that the two trees being compared are WSDL specifications and relies on the structure of the WSDL schema to simplify the tree comparison.

WSDL specifications are hierarchical. At the lowest level of the hierarchy, the data types are defined, which themselves are defined in XML and are hierarchical; one layer above the messages are defined, whose structures depend on the defined data types; the next layer specifies the service operations, which are composed of messages. Finally, the whole service is defined as the composition of its data types and operations.

4.3.3.4 Technical Solution 2

The idea for using the WSDL Analyzer is that the user already knows a service which provides the correct format. The WSDL of this service can be used as requirement for a similarity search. Candidates for search and mapping can be obtained e.g. from a registry. The WSDL Analyzer allows browsing the original WSDL and the candidate files (see Figure 25).

![Figure 25 WSDL Analyzer, Main Window](image)

For selected candidate WSDL files, the similarity between requirement WSDL and candidates can be checked. The result is a ranking of the candidates according to their matching score. The mapping between two WSDL descriptions is indicated by coloured nodes, where an equal colour indicates the mapping of an element in the requirement WSDL to an element in the candidate. Elements for which a mapping could not be found are left without colour. By clicking directly on an element in the requirement, the correlated element on the candidate side is highlighted (see Figure 27).

In the settings, the details of the algorithm can be selected (Figure 26). It is possible to match only the structure, using only data types as label, and/or match element names, and to use WordNet\(^9\) to find semantic relations between labels, in order to improve the accuracy of the matching.

---

Two papers by [WAN03,WAN03b] build the foundation for the WSDL Analyzer. There, the structure matching approach is combined with methods stemming from Information Retrieval (IR) research. For the WSDL Analyzer, we concentrated on finding structure similarities and omitted the IR part. The reason is that while IR methods help to find similar services, they do not help in generating a mapping from one WSDL file to another, since the vector-space approach commonly used in IR neglects information about the structure of documents. For the WSDL Analyzer, the algorithm proposed by [WAN03], for finding structural similarities is improved by taking into account additional features of the XML-based WSDL structure.
The structure-matching algorithm is inspired by traditional signature-matching methods for component retrieval [ZAR95]. Another related research area is schema matching [Rah01] which aims at identifying semantic correspondences between two schemas, e.g. database schemas, ontologies, XML message formats, etc. The advent of XML as common data representation format lead to a revival of matching algorithms already investigated in the database and pattern matching research (e.g. [SHA97]). Since WSDL is based on XML, the results from this research can be applied in a modified way to finding similarities in WSDL descriptions of services.

The WSDL Analyzer approach, following [WAN03], exploits various types of schema information (e.g. element names, data types and structural properties), characteristics of data instances, as well as background knowledge from dictionaries and thesauri (e.g. WordNet10). The algorithm calculates the similarity between the structure of a required service and the structures of a set of candidate services. The algorithm respects the structural information of data types and is flexible enough to allow relaxed matching and matching between parameters that come in different orders in parameter lists.

The comparison of two WSDL files (see Figure 28 for the WSDL structure) is a multi-step process: it involves the comparison of the operations set offered by the services, which is based on the comparison of the structures of the operations’ input and output messages, which, in turn, is based on the comparison of the data types of the objects communicated by these messages.

The tree-edit distance measure [APO97] and the concept of a weak subsumption relation [NAG04] can be seen as a theoretical foundation of this approach. The more recent concept of weak subsumption is not considered by [WAN03] and the tree-edit distance is only mentioned, but not related to their algorithm, hence the rather arbitrary character of their algorithm.

The idea of the tree-edit distance is that a similarity between two XML structures S and $S'$ can be measured by stepwise transforming a tree representation of S into the tree representing $S'$. The steps

---

10 WordNet: http://wordnet.princeton.edu/
necessary for that transformation provide the measure for their similarity, and, at the same time, induce
the mapping between the schemas. Possible steps are basic edit operations such as node inserts,
deletes and relabels. The algorithm proposed by [WAN03] considers only node matching without editing,
or simple renaming operations such as changing a data type from `string` to `int`. [NAG04] introduce the
concept of a weak subsumption. They give three different types of weak subsumption, replacing labels
(Figure 29), pruning edges (Figure 30) and removing intermediate nodes (Figure 31). These operations
can be correlated to specific tree-edit operations, namely relabeling and deleting nodes. Weak
subsumption restricts the allowed editing steps.

![Figure 29 Typecast by relabelling of nodes](image)

![Figure 30 Pruning edges](image)

Weak subsumption is of importance for two reasons:

- Structural similarity: If the relation holds between two Web services, the services are structurally
  similar.
- Transformation: If a Web service D1 weakly subsumes D2, we can transform a message
  destined for D1 to a message compliant with D2 by applying the subsumption mapping and
  transform the reply message.
The overall process for comparing two WSDL descriptions starts by constructing a logical tree representation of the WSDL files which are compared. The root node is the service itself; the bindings, the port types, operations and messages are added. The data type section is inserted as node below the message level. It is interesting to note here that the data types of web services specified in WSDL are XML elements (to be precise, XML schema definitions, XSDs); as such, they can potentially be highly complex structures. Leaf nodes of the logical tree are the primitive XSD data types.

Comparing two WSDL trees comes up to finding corresponding paths in the trees. The overall score of how well the two services match is computed by identifying the pair-wise correspondence of their corresponding nodes (service, bindings, port types, operations, messages, data types) that maximizes the total sum of the matching scores of the individual pairs. The result of this comparison is a matrix assessing the matching scores, i.e., the degree of similarity, of pair-wise combinations of source and target nodes.

If a mapping from a WSDL describing service A to a WSDL describing service B is generated, a translation of a SOAP message instance destined for A in a message for service B can be supported. Figure 32 shows a SOAP message destined for the FDX carrier which is transformed into a message for a new carrier. Based on the mapping, the values sent to FDX can be extracted and inserted into the corresponding tags in the message for the new carrier. The translation can be done automatically, if there is a one-to-one correspondence between elements. However, if there are several possible corresponding elements, translation requires intervention from a user in order to unambiguously transform parameters.

The latter case shows the limitation of the structural approach. There are possible mismatches which can be detected with the help of the WSDL Analyzer, but not automatically corrected.
4.4 Process Model Exchange

4.4.1 Conceptual Solution

The POP* process metamodel is an ATHENA solution designed for process model exchange for both large companies and SMEs. The POP* process dimension metamodel is an conceptual interoperability solution to remove mainly syntactic barriers; it aims at translating process model elaborated using a particular language, for example GRAI tool of company 1 to another process model for example ARIS tool of company 2, as shown in Figure 39. This issue is relevant in the case where SMEs work with large companies. In particular large companies tend to interoperate with SMEs (sub contractors, parts providers) processes and consider the last ones as their “own processes”.

Figure 32 Mapping SOAP messages

Figure 33 Using POP* as conceptual solution for process model exchange (illustration)
The POP* process dimension includes constructs related to activities as well as whole processes and can represent internal and or cross-organizational processes.

In order to meet the requirements of transforming heterogeneous modelling languages, the concepts of the process dimension are constructed for capturing a broad range of possible process descriptions. The representation of POP* processes is based on connection points that are connected by flows. Flows can be either material (e.g. a product flow) or immaterial (e.g. control flow). Further on, flows can represent the state a process has at a certain point of time, e.g. the state “Decision was taken”. The timely and logical order in which connection points are executed is also specified via the attributes in-flow logic and out-flow logic included in all connection points. An overview of the Process metamodel is shown in Figure 40.

**Figure 34 POP* Process dimension metamodel**

Adaptation for SMEs may be necessary in the following points:

- Simplified version for SMEs (We consider that the current version can be used directly by SMEs without simplification, but a simplified version would allow SMEs to easily better understand the metamodel, to gain time in mapping their own models to the POP* metamodel)

- The current version of POP* focuses on syntactic barrier, semantic mismatch barrier is not sufficiently addressed. To overcome this, the use of semantic annotation and semantic enrichment techniques developed in A3 project of ATHENA may be necessary if the process models of two companies have a semantic barrier. The solution can be conceptually described by the figure below.
4.4.2 Technical Solution

The current POP* metamodel is implemented in MPCE (Modelling Platform for Collaborative Enterprises) (see Figure 42). The MPCE is a generic platform developed by COMPUTAS/TROUX for distributed concurrent design, services engineering, model-generated workplaces, etc. POP* will be integrated inside the MPCE, which will enable enterprises to exchange enterprise models among different Enterprise Modelling Tools. However, other platforms and solutions may be developed as well in the future.

![Figure 36. POP* implemented in MPCE platform](image)

The following requirements are considered important for SMEs. First it is necessary to have an open POP* portal for SMEs. This portal must provide: (1) free access to the actual POP* release; (2) free access to standard software components; (3) references to software providers; (4) service offers; (5) (restricted) POP* interest group (Vendors, Industry, Researchers).

Concerning the implementation adaptation for SMEs, according to A1.3.1 deliverable, there are two possible technical implementations to use POP* metamodel.
In the first solution the common POP* repository contains all the modelled data covered by the POP* meta-model. Each of the different tools can check-out required parts of the whole enterprise model in its own POP* repository to work on it. Afterwards, it can check-in the information back to the common POP* repository. This mechanism will reduce the risk of information loss between the different modelling tools, because all tools will work in a POP* modelling environment and on a complete model at all times, irrespective of the individual tools being able to manipulate all the information covered by POP*.

In the second solution there is no central common model repository. The information about the enterprise models are distributed in different partial models in different tools. Each partial model can be a complete enterprise model by themselves or a part of it. The model exchange is based on the local POP* repositories of each tool involved in the study. They are completely distributed without a common server or repository. Partners have to provide models, partial models or interfaces to their models to other partners of an enterprise network.

It has been considered that both solutions are adapted to SMEs according to the situation:

If the SME is working with a large company, the first solution may be adopted as the central common repository and server can be managed by the large company working with different SMEs.

If the SME is involved in a networked SMEs (for example a virtual enterprise composed of SMEs), then the second solution may be adapted. It requires loosely coupled models and technical supports to SMEs.

4.5 Agent-based process/interface adaptation

4.5.1 Conceptual Solution

An important interoperability barrier arises when a partner – especially an SME which has only limited resources for changing its IT systems – has to be integrated into a collaborative process for which an obligatory description of the interaction process has already been established. In ATHENA, the PIM4SOA was proposed as a metamodel for a description of service interactions.

PIM4SOA can be seen as a metamodel which facilitates the transformation from a process engineering level to an execution level. In ATHENA A2, an EPC to PIM4SOA transformation was developed [ATH05] and evaluated on the basis of a scenario from the furniture industry. The next step contains a transformation from PIM4SOA to agent models which is described in more detail in [HAH06]. There, it is demonstrated how PIM4SOA models can be transformed into agent models that can be directly executed by specific agent execution platforms. In our case, the Jack Intelligent agent framework is used for the execution of BDI-style agents. A brief summary of the mapping between PIM4SOA models and agent models can be given as follows. In PIM4SOA the interaction described in business processes are represented by collaborations. For mapping these concepts to agent models the team mode supported in Jack is especially helpful. Team members correspond to the partners involved in the collaboration while the processes can be directly mapped to team plans where a team plan is executed by an individual partner.
Figure 37 Integration of external services into a PIM4SOA model

Figure 43 shows the overall approach for facilitating user-composable services. The interaction described in a PIM4SOA model is transformed into executable agent models. If an already existing service has to be integrated – e.g. because it is too costly for an SME to change the service according to the PIM4SOA model – a WSDL description of the external service is taken to generate the components of the Jack model which are necessary to invoke the service. Thus, it is possible to support the integration of a new partner although the partner’s interface does not fully comply with the given interaction specification.

4.5.2 Technical Solution

In case of a mismatch between process models, an adaption or mediation step of the different processes is necessary. If e.g. an SME wants to include a new service which expects a different order of messages than the services which are already used, a collaborative process description can be used for adapting the new services. In the context of ATHENA, the PIM4SOA metamodel [BEN06] was developed for specifying collaborative processes on a platform-independent (PIM) level.

Using WSDL2JACK, it is possible to support the integration of a new partner although the partner’s interface does not fully comply with the given interaction specification.

4.5.2.1 From PIM4SOA to Jack metamodel

In order to execute collaborative processes specified on the PIM level, the first step consists of a transformation from PIM4SOA to agent models that can be directly executed by specific agent execution platforms. In our case, the Jack Intelligent agent framework [AOS] is used for the execution of BDI-style agents. The constructs of the PIM4SOA metamodel are mapped to BDI agent systems represented by the Jack metamodel (JackMM).
In the following, we concentrate on the five transformation rules that are illustrated in Figure:

At first glance an agent seems to be the best match for a service provider, but since ServiceProvider references Roles, it is recommended to assign it to a Team. The name of the service provider coincides with the name of the team, its roles are the roles the team performs and the team makes use of the roles specified in the CollaborationUse, in which it participates, as bound roles.

A Collaboration is mapped onto a team that may again consist of any number of agents. Although the metamodel for PIM4SOA allows to specify constraints on the behaviour of the participating collaborations and their roles, up to now it is unclear how these constraints might look like.

Beside introducing a role in JackMM for each role a service provider and collaboration performs, we define a team and two TeamPlans for every role service providers and collaborations make use of. The atomic team is only represented by the corresponding role in the PIM4SOA. The team plans specify how the requested service is invoked and how the corresponding team reacts on a service request.

The Process of a PIM4SOA can easily be transformed into team plans. The roles a team plan uses are extracted from the bound roles in the collaboration use the corresponding team interacts. As a first approach, we transformed the sequential process structure of a PIM4SOA model into a sequential team plan.

Finally, Messages that are sent by the roles we already have transformed are mapped to Events in JackMM.
4.5.2.2 Integrating Web services

The easiest way to enable the integration of Web services into BDI agent implementations derived from JackMM is to encapsulate existing services using service descriptions that are directly derived from the original PIM4SOA. The PIM4SOA model also includes the definition of the documents involved in the modelled business process and guarantees a common data model. If the involved services directly implement this data model, as do the agent implementations based on JackMM, data integration should not be an issue. Still, it may not be an easy task to adapt the underlying legacy systems and applications that are encapsulated by the Web services based on the descriptions derived from PIM4SOA.

It becomes obvious that complete integration can be very expensive to achieve and in general tends to be very fragile. Slight changes of the implementations may already break the putative harmony. In the same way, in a model-driven architecture, changes of the model that was used to bring about the integrated architecture are likely to break possible additions to automatically generated parts. To avoid this, it pays off to clearly separate the part that was automatically generated through model serialisation from the part that was handcrafted to achieve the integration of given components.

One solution to this problem would be to reverse engineer all functionality that was added on the implementation level and lift it to model representations. But the hierarchy of ascending abstraction and decreasing detail from implementation to platform independent model representations usually prevents this course of action. The second solution is to encapsulate the environment dependent parts on the lower level in a versatile way that helps to prevent incompatibility on regeneration from higher level models.

The concept of integration agents in the context of PIM4SOA and JackMM provides a suitable starting point to encapsulate functionality that helps to deal with a situation when changes are made to the underlying PIM4SOA or JackMM models, or if the legacy components that are to be composed to implement the SOA are modified or exchanged. Prototype integration teams can easily be generated to access services that are in accordance with the PIM4SOA description. But even if the services change or do not directly correspond to predefined mapping schemes the integration teams can easily be modified to accommodate for the changes without any impact on the model-derived implementation parts. In the same way model changes can be mitigated and even data model translation can be put in place to guarantee ongoing interoperability.
An obvious starting point for a mapping are messages. The message concept on both sides is very similar, as they represent the encapsulation for communicating data among entities. Messages have a name and carry some sort of payload. In WSDL, payload data is typed by XML schema definitions. Agent messages can generally have arbitrary content, but for communication with other agents it is necessary to agree on some content language.

For interaction with Web services, agent message content needs to be restricted to XML that conforms to the schema that is expected by the service.

A port constitutes the external entry point (endpoint) of a Web service. It is essentially a binding of abstract functionality specified by a port type to a transportation protocol and address. An agent is the addressable endpoint concept on the agent side and provides an encapsulation of functionality that can be mapped to a Web service port.

The functionality of a port type is defined by a set of operations. Operations are typed depending on the combination of incoming and outgoing messages. Those operation types that start with an incoming message can directly be mapped to corresponding plans.

Request-response and one-way operations are mapped to plans that are triggered by the incoming message and in case of request-response include sending of the outgoing message before the plan terminates. Although solicit-response and notification operations are not in practical use, they can be mapped to plans that are triggered by internal events. Figure sketches the mapping that realises direct mapping of messages from agents to Web services.

When a dynamic SOA is to be built from existing components (possibly Web services) the JackMM implementation provides a flexible platform for execution of the dynamic runtime aspects and integration teams loosely couple the existing components into that process. In this way, and especially when different partners are involved, agents and agent teams provide suitable representations for the autonomous participants of a process.
A major advantage that can be achieved by introducing agent representations as placeholders for services into SOAs, is the possibility of late binding of service calls. Integration teams can easily encapsulate and aggregate multiple services and utilise their BDI reasoning capabilities to choose at run time, which operations and which services to actually call.

The agent platform JACK Intelligent Agents serves as BDI MAS implementation, Axis2\(^\text{11}\) is chosen for Web service implementation and deployment, with the Java programming language serving as a common denominator between the two. A comprehensive selection of MDA tools is available for the Java-based Eclipse framework and some of them can be employed to support the proposed model driven development approach.

In order to implement the mappings between Web services and agents as described above, JACK, Axis2 and Eclipse MDA can be applied. The provided implementation follows the proxy agent approach, but a bi-directional gateway could also be realised using the same technology.

An option would be to extend either JACK or Axis2 to natively support integration mechanisms. But in prospect to an integration with the PIM4SOA framework, a flexibly deployable solution that does not depend on any registration mechanisms, nor relies on any proprietary constructs, seems to be more appropriate.

The provided WSDL2JACK implementation allows the generation of proxy agents that can communicate with Web services and the generation of agents as an implementation for a given Web service description that can serve as a proxy for Web services to communicate with agents in the multiagent system. In both cases WSDL2JACK expects a WDSL service description as input and by default the first port of the service is considered for generating an agent implementation. In the current implementation only the mapping from a single service endpoint to an agent is supported, but WSDL2JACK can easily be extended to generate agent teams for services with multiple endpoints.

4.5.2.3 Making Web Services Available in JACK

One usage of WSDL2JACK is to make a Web Service accessible by JACK agents. The input WSDL file describes a Web service that should be available to the agent platform. WSDL2JACK takes the WSDL file and generates a JACK proxy agent that can be used to access the service. The structure of such a proxy agent can easily be derived by directly applying the mapping as shown in Figure.

The biggest obstacle for direct Web service – Agent interaction is the usage of different message formats on both sides. JACK agents communicate by passing of message events, which are essentially Java objects. Web services normally communicate through SOAP XML messages. The challenge is to translate SOAP messages into message event java objects and vice versa. The first step in an approach to this challenge is the parsing and interpretation of SOAP messages. Rather than building a SOAP parser from scratch it is advisable to use available technology for this purpose. Axis2 is the system of choice, as it provides a powerful open-source Java-based SOAP engine. All handling and parsing of SOAP data can automatically be done by Axis2 and even message content can automatically be converted into Java objects.

WSDL2JACK utilises Axis2 by directly calling the wsdl2java tool to generate a Java stub class, which in turn can be used to transparently call the Web service as specified by the input WSDL file. Wsdl2java only considers one port of a service for code generation (the first port by default). Consequently, WSDL2JACK also considers the same port for proxy agent implementation.

The actual agent is generated by further inspecting the WSDL file using the WSDL4J API. Rather than generating agent code directly, WSDL2JACK outputs JACK Gcode that can be loaded into the GUI of the JACK development environment to be adapted to a specific application context. Although the Gcode format resembles XML it unfortunately does not conform to the XML standard. Moreover, no formal specification for the Gcode syntax is available and the format may be subject to undocumented change among different JACK versions.

To address this problem, WSDL2JACK does not generate Gcode output directly but defines the proxy agent concepts using the JackMM metamodel. As a first generation step, an XMI file is generated representing the JackMM model of the proxy agent.

The XMI representation of the proxy agent model contains all the necessary concepts for the generation of a proxy agent, such as the agent itself, plans, events, etc. Such an approach has many advantages. Not only is the volatile Gcode generation decoupled from the building of the agent concepts and may easily be adapted if needed, but more importantly, the model of the agent represents an asset on its own. It provides an abstraction of the Web service proxy agent that can among other things easily be adopted to other BDI agent platforms via model transformations.

General MOFScript serialisations can be implemented for serialising JackMM models into JACK Gcode. By encapsulating the necessary plans and events into a capability, the functionality for interacting with a service is packaged into a reusable module. The capability can now also be plugged into other agents that need to access the service. Additionally, it is straightforward to enable a service to access different services by adding multiple capabilities.

Conflicts may arise, if the different services have operations with the same name and the same input message. In such a case, incoming messages will trigger plans of both capabilities. If this behaviour is not desired, it is up to the user to resolve the conflict manually by providing additional code specifying how the event is to be handled.

It is necessary for plans to check if the incoming message event has the operation field set to the correct operation name that is handled by the plan. The payload of a message event is contained in a special document attribute. When invoking the Web service, the content of the document attribute is passed as a parameter to the call of the service’s Axis2 stub. Plans can use the getWebServiceStub() method to retrieve an instance of the stub via which they can invoke the actual Web service. If the stub method has a return value, meaning the Web service sends a response, this response is returned by the plan in a response message to the incoming message event.

![Figure 40 From WSDL to JACK code](image)

The overall generation process is summarised in Figure 40. When the generation process is complete, the proxy agent can be deployed into an agent platform. Other agents can use its exposed message events to trigger Web service invocations and also receive responses in the form of message events.
5 Conclusion

The different barriers for SMEs becoming and remaining interoperable are mostly based on different facts such as organizational structure of such enterprises and their limited technical and financial resources. We formulated that the core requirement of most SMEs for the interoperability software is its easy integration and configurability, optimally by the SME company itself. Recall that SMEs still lack of ICT skills to deal with interoperability and integration issues.

We first addressed the relevant ATHENA results and tools in order to see how far they support the SMEs on a technical level to become and remain interoperable. We came up with the following results:

First we saw that some ATHENA results and tools fulfil a certain level of low complexity and easy usability to be used by SMEs. The main requirement on the ATHENA results was that they must provide the minimum functionality level to implement a certain technical interoperability situation as formulated by the scenario. In this case we showed how an SME can use identified ATHENA results and tools to address a barrier by implementing that special part of the defined scenario. This implementation represents a best practice for that result/tool to be used by an SME.

Second the work showed that for addressing some barriers identified for SMEs by an ATHENA result or a combination of them, the results need a certain level of adaptation to be used optimally in a typical SME environment. In such cases we did the adaptation work in two phases: First we worked out the necessary tool adaptation on a tool independent conceptual level and in a second phase on a tool specific technical level. This separation is due to the fact that each ATHENA result/tool represents a concept for enabling and supporting the interoperability and the tool itself is a technically grounded implementation of that interoperability concept. The same applies for an adaptation intent on a technical tool/result. We keep the conceptual work mostly independent from a certain implementation decision. Thus we see the work done on the conceptual level not only as a basis for the technical implementation shown for adaptation of certain ATHENA results, but also as a reusable concept that can be used to adapt other technical tools addressing the same interoperability level as proposed by AIF. The adapted ATHENA solutions have now following features suited to the needs of SMEs on business, process, service and data interoperability level:

On the business and process level we looked at the business process modelling approach provided in ATHENA. Here we have provided a rule based verification mechanism in such a way that non technical persons in SMEs can express the way they require a business process to work in terms of rules. The current technical configuration of the business process model is verified against these rules and the possible constraint violations on the current configuration of the business process model are reported. Further on the process level the POP* language (process dimension) is adapted to cover SMEs need for translating two different process models.

On the service interoperability level we describe a solution to the barrier of different behavioural interface of services on top of existing service infrastructure in ATHENA. Here we use the concept of processing module provided in Johnson tool to overcome this barrier. On this level we also provide an Agent-based Process/Interface Adaptation as an Integration mechanism of external Web service interfaces described in WSDL.

On different levels of Data exchange barrier we have shown different approaches how existing ATHENA results can be used by SMEs. This should serve them as a best practice solution to overcome the data exchange barrier. Here particularly the conformance testing tool, the WSDL Analyzer tool and a modified modeling language by means of adding mechanisms that permit one to restrict models (in terms of express rules) are used.
6 References


[SUR03] Sure, York ; Angele, Jürgen ; Staab, Steffen: OntoEdit: Multifaceted Inferencing for Ontology Engineering. In: J. Data Semantics 1 (2003), cf. 128_152


