Welcome to the course “Practices of Interoperability in Small and Medium Enterprises”.

The current module is called Proposed Solutions.
Proposed Solutions

The approach adopted for developing interoperability is:

- First identify barriers to interoperability in SMEs
- Then identify solutions which allow removing those barriers

The approach adopted for developing interoperability in SMEs is: first identify barriers to interoperability in SMEs and then identify solutions which allow removing those barriers.

Barriers regarding carrier-shipper scenario have been explained in previous module.

In this module, for each barrier we will provide first a conceptual solution and then a technical solution.

Technical solutions are based on tools, languages, methodologies, proposed in ATHENA-IP project.
Business Process Configuration

Proposed Solutions:

1. Business Process Configuration
2. Service Granularity and Behaviour
3. Data Exchange
4. Process Model Exchange
5. Process/Interface Adaptation

Now we will analyse the proposed solutions.
Let's start with the solution for the first barrier: “Business Process Configuration”.

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

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:
1. a modelling language to describe the system;
2. a specification language to describe the correctness requirements; and
3. an analysis technique to verify that the system meets its specification.
The figure shows the overall conceptual steps required for the solution of this interoperability barrier:

1. Define a formal description for the business process model as an upper ontology.
2. 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.
3. Store the specific business process description as a semantic enriched model of the business process.
4. 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.

An inference engine is a computer program that tries to derive answers from a knowledge base.

It is the "brain" that expert systems use to reason about the information in the knowledge base, for the ultimate purpose of formulating new conclusions.
According to the conceptual solution, we propose following technical solution:

- **OWL-DL** is used as language for definition of the upper ontology.
- A modeling tool is used to create **SWRL** (Semantic Web Rule Language) Statements as constraints representing the business requirements.
- Business Process is modeled using **Maestro** and stored as an **OWL-DL instance** of the upper ontology.
- **KAON2** is the inference engine used to infer whether a selected business process violates the set of rules assigned to that business process or not.

**Maestro** is a tool which purpose is to solve the problem of connecting business processes from different enterprises. For this matter, Maestro allows the creation of process models. With this tool the user can view and change the models. After creating the process models, it also allows handling of cross-organizational business processes (CBP), its main goal.

- **KAON2** is the inference engine used to infer whether a selected business process violates the set of rules assigned to that business process or not.
Here you can find additional information for the mentioned technical solutions.

<table>
<thead>
<tr>
<th>About...</th>
<th>Athena courses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ontologies</td>
<td>ONT1- <em>Introduction to Semantics</em></td>
</tr>
<tr>
<td>Maestro Tool</td>
<td>EM4 - <em>Cross-Organizational Business Processes – Enabling Technologies and Tools</em></td>
</tr>
<tr>
<td>Other</td>
<td></td>
</tr>
<tr>
<td>OWL-DL</td>
<td><a href="http://www.w3.org/TR/owl-guide/">http://www.w3.org/TR/owl-guide/</a></td>
</tr>
<tr>
<td>KAON2</td>
<td><a href="http://kaon2.semanticweb.org/">http://kaon2.semanticweb.org/</a></td>
</tr>
</tbody>
</table>
Let’s continue with the solution for the second barrier: “Service Granularity and Behaviour”.

Proposed Solutions:

1. Business Process Configuration
2. **Service Granularity and Behaviour**
3. Data Exchange
4. Process Model Exchange
5. Process/Interface Adaptation
There are two main obstacles that must be pulled down:
- The absence of a common standard for defining service interfaces and exchanging messages.
- Different behavioural interface and granularity of services.

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.

Different shippers will require sending messages using different transportation protocols and messaging styles.

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 messages,
• and routing them to the appropriate endpoint through different transport protocols.
Before describing the technical solution we will introduce some technologies. Web Services Description Language (WSDL) is an XML formatted language used to describe a Web service's capabilities as collections of communication endpoints capable of exchanging messages.

The architecture builds on the Johnson and Lyndon tools. **Johnson** is a Web service execution infrastructure. The scope of the Johnson tool could be described as encompassing all the functionalities in a service-oriented architecture that have to do with handling messages at runtime. Johnson allows users to call Web Services in a way that hides the technical details away from them.

**Lyndon** is an application that represents the design-time counterpart of the Johnson tool. Lyndon 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.
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. These services are defined taking into account the online APIs provided by the carriers.

2. These WSDL files are used to configure Johnson. An instance of Johnson is configured as a service consumer and another one as a service provider.

3. The shippers will send the request message corresponding to the selected service.

4. The Johnson provider instance will receive the messages, process them according to some SOAP to REST transformation rules and route them to the designated carrier.

5. Transformed messages will be sent to the different carriers that will process the XML documents and send the reply.

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.
Let’s continue with the solution for the next barrier: “Data Exchange”.
### Data Exchange

Data Exchange can have some barriers like:

- Wrong Instantiation of Data Models
- Different Data Restriction
- Incompatible Syntactic/Semantic Representation of Data

---

Date exchange can have some barriers to be improved, like:

- Wrong Instantiation of Data Models,
- Different Data Restriction and
- Incompatible Syntactic and Semantic Representation of Data.
To solve the wrong instantiation of data models, the conformance testing can be applied. The conformance testing can detect the errors, and based on its reports these errors can be corrected.

The conformance testing is based in two different stages: Model validation, and Semantic validation.

To allow the verification of errors in all the data exchanged, the conformance test uses the data model like base to both validations.
The technical solution to solve the wrong instantiation of data models identified was the 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 XML schema (XSD) for the basic structure and constraints are complemented by a language used for the more complex rules: SCHEMATRON.

On this way, when one company sends or receives data information to another company, calls the conformance testing web-services to check the file validity.

The conformance testing tool will send back a report with errors found in the data file.
To identify this problem, the conformance testing (described in previous solution) can be applied, because the conformance testing can also check restriction.

The solution for a company that wants to exchange data with others companies with different data restrictions is alert the other companies that its restriction should be satisfied.

To identify this problem, the conformance testing (described in previous solution) can be applied, because the conformance testing can also check restriction.
The first solution for incompatible syntactic and semantic representation is to create a mapping of schemas of the companies that want to exchange data, and then build a reference ontology.
The second solution is generate a mapping between a message to Service A and a message to Service B.

Based on this mapping, the values sent to A can be extracted and inserted into the corresponding tags for B.

The translation can be done automatically, if there is a one-to-one correspondence between elements.

However, if exists several possible corresponding elements translation, this method requires intervention from a user in order to transform parameters unambiguously.
Now we will analyse the solution for the fourth barrier: “Process Model Exchange”.

Proposed Solutions:
1. Business Process Configuration
2. Service Granularity and Behaviour
3. Data Exchange
4. Process Model Exchange
5. Process/Interface Adaptation
There is a great number of enterprise modelling languages in use. Collaboration between enterprises often involves sharing or exchanging each others models. The approach to supporting this is to offer a model exchange mechanism based on a common format (a meta-model) that contains a set of basic modelling constructs.
The exchange mechanism mentioned is the POP* and consists of a meta-model together with guidelines and scenarios for its management and use.

The process dimension includes constructs related to activities, process roles, flows, decision points, etc.

Simplified version of POP* for Small and Medium Enterprises may be necessary.

**Metamodel** is literally a model for (applied to) a model. In our case it is a model which defines the language for constructing other models. Metamodels are called “reflective” if they are capable of defining themselves.
Here you can find additional information for the mentioned technical solutions.
Last but not least, Let’s see the proposed solution for the fifth barrier: “Process-Interface Adaptation”.
Process/Interface Adaptation: Conceptual Solution

- This barrier arises when a partner has to be integrated into a collaborative process
- The solution should facilitate the transformation from a process engineering level to an execution level

This barrier arises when a partner has to be integrated into a collaborative process for which an obligatory description of the interaction process has already been established.

The solution should facilitate the transformation from a process engineering level to an execution level.
Pimforsoa can be seen as a metamodel which facilitates the transformation from a process engineering level to an execution level.

Pimforsoa metamodel defines a language that could be used to describe Service Oriented Architectures at a platform independent level.

Pimforsoa defines:
• The collaborations that need to be implemented to perform the interoperability objectives stated at the business level.
• The sequence of messages exchanged to perform the collaboration.
• The exchanged information.
• The non-functional requirements that the interaction has to meet.

The resulting Pimforsoa should be able to support a formal transition between enterprise models and software implementations.
Learning Game Placeholder

Learning Game: Sequence
Title: Sequence
Learning Game Placeholder

Learning Game: Word Quiz

Title: Word Quiz
This course has been developed under the funding of the European Commission with the support of the ATHENA-IP Project.