Commius Project Summary

Community-Based Interoperability Utility for SMEs

FP7-ICT 213876
This page is intentionally left blank.
Commius Project Summary

Community-Based Interoperability Utility for SMEs

Project Context and Objectives

In today’s evolving industry, the ability of European SMEs to cooperate and interoperate plays a crucial role. There are 19 million SMEs within Europe, representing 99.8% of all registered businesses, and the economy depends upon their contribution to wealth creation and employment. Although SMEs have some strong advantages in flexibility and responsiveness, they face challenges in larger projects where efficient and effective collaboration with others is required.

Enterprise interoperability has been addressed by research for decades without real success and little impact on SMEs. The market still lacks an affordable, easy to integrate solution: SMEs instead face costly investment, extensive integration effort and significant revision of their working tools and systems. Solutions for enterprise interoperability are present only in huge enterprises and their SME suppliers who are forced to use their client’s solutions; besides, these systems are usually based on stable or even standardised processes. Nothing so far works for SMEs, especially for loosely-coupled, collaborating SMEs.

Following the suggestion of the IST Enterprise Interoperability Research Roadmap, the Commius project conceives interoperability as a utility-like capability (Interoperability Service Utility – ISU) for enterprises, a capability that is available at very low cost, accessible in principle to all enterprise, guaranteed to a certain extent in accordance with a set of common rules and not controlled or owned by a single private entity.

Commius approach focused on the creation of mechanisms and tools to offer such an easy-to-use interoperability solution for SMEs, allowing them to reuse existing and familiar applications for electronic communication, such as e-mail and the web. Commius research aimed to demonstrate that the application of Semantic analysis of message exchange and flexible and dynamic business process modelling, may turn these simple tools, e-mail clients and web browsers, in powerful services to support business automation and enterprise cooperation, achieving interoperability at system, semantic and business process levels.
• **System Interoperability**, addressing Message Analysis and Semantic Annotation mechanisms, adopting approaches based on Information Extraction and Semantic Annotation techniques.

Systems Interoperability has been addressed using e-mail for sending and receiving business documents, thus building interoperability solutions above existing simple ICT infrastructure. With this approach, existing useful properties of e-mail, such as action-oriented and asynchronous communication can be inherited. System Interoperability technology developed allows to analyse messages and extract attachments and relevant information. Messages are modified to include both related information and active information elements (links, buttons) linking each e-mail message with relevant enterprise resources needed to fulfil a set of interoperability tasks related to the message. Moreover, interface to legacy systems and existing infrastructures has been implemented using a SOA approach and standard access protocols.

• **Semantic Interoperability**, to provide alignment and integration of business data exchanged among different enterprises.

Semantic Interoperability assures quick and painless business document alignment using an extensible core of semantic descriptors which characterize relevant business document types. Semi-automatic semantic annotation of exchanged documents are performed, based on text analysis techniques. Moreover, protocols have been developed to support automatic negotiation of semantic alignment. The core ontology can be (semi)automatically extended to a particular (business) domain through enrichment with specific concepts and (document) instances.

• **Process Interoperability**, with the objective enterprise to allow synchronization, partial control and integration of business tasks governed and executed outside enterprise borders

Process Interoperability research has addressed the issue to allow seamless entry into a community of interoperable companies not only at the document but also at the process level. Here, appropriate coordination of activities within, and among, partners will be supported by the Commius interoperability software Components. Reference model templates and patterns of Business Processes has been identified and mapped onto executable protocols. Moreover, algorithms to detect Business Processes based on communication, and techniques for inter-enterprise process negotiation and adaptation has been implemented. To cope with the scientific, technological and business objectives and challenges, the Commius consortium included a group of European partners with advanced research capabilities, industrial expertise in IT, business process and innovation, and technology transfer capabilities.
The consortium has validated the project results by delivering a complete prototype and by testing and piloting the prototype in the context of three business cases provided by partners companies of the consortium and a network of other collaborating SMEs. These three business cases were respectively focused on collaboration within a technological district of SMEs, business contacts information search and classification, and quotation and price negotiation among retail shops and wholesalers.

Besides the mentioned piloting activities with Commius selected end users, a number of public dissemination events involving different groups of Commius potentially interested subjects have been carried out by the project partners during the project’s lifetime. The goal of these events was to spread knowledge about the projects’ objectives and results involving Commius end user audience, obtaining from them useful feedback that will help to ensure that the research and the development activities match the user needs and requirements and suggestions about future developments.

The Commius software framework developed has been published as an Open Source project, so that its source code can be accessed by software developer companies interested in interoperability issues, willing to contribute developing customised solutions on top of the framework, or enriching it with extensions addressing specific requirements. Commius Open Source framework is now hosted and available on sourceforge.net (http://commius.sourceforge.net/)

Commius outcomes will enable SMEs to react to the market requests as a net of collaborating companies, each providing specialised products, technologies and services. The almost zero-cost entry characteristic, the reusing of existing and familiar applications to provide interoperability solutions, and the seamless integration with existing legacy systems and ICT infrastructures, will prove essential to promote a wide adoption of our solution among SMEs throughout Europe. An SME user will not be forced to adopt new working tools nor change its organisational and business models. The Commius interoperability solution will evolve along with the organisation, adapting to the user needs and according to the existing infrastructure.
Commius Approach

Traditional approaches to enterprise interoperability rely on tightly controlling the details of the information exchanged between the two enterprises. In particular every message in such exchanges must follow a precise, pre specified syntax or even use specific software. While such approaches are unarguably effective they incur a considerable implementation cost both for the company receiving the messages, but also for the companies who must send messages to that company. They are therefore best suited for use either by large companies who have sufficient market presence to mandate that their suppliers should send them messages in a specific way or by companies in substantive, long term cooperation.

In contrast COMMIUS research aimed to provide interoperability support for an individual SME. Since such companies lack the market power to control how people contact them, any such system must be designed so that it can be deployed without placing any additional requirements on people contacting the SME.

The interoperability solution we developed consists of a system hooking into a selected email infrastructure. Each of the three main layers (system, semantic and process) is responsible of tackling a specific aspect of interoperability: from the identification of business objects referred implicitly or explicitly in the captured e-mail, passing to the tracking of business processes through the flow of them, then to enriching the e-mail with relevant additional information to interoperate. In concrete, COMMIUS aimed to pursue the following objectives:

- To investigate, design and develop methods and Components for system interoperability. COMMIUS uses the SMTP protocol and thereby interconnect existing ICT infrastructures to serve as an interoperability infrastructure for semantic and Process Interoperability Layers. Systems Interoperability is addressed using email for sending and receiving business documents, thus building interoperability solutions above existing simple ICT. Existing useful properties of email, such as action-oriented and asynchronous communication are inherited. Messages and attachments are analysed and information extracted. Messages are modified to include both relevant information and active information elements (links, buttons) linking email message with relevant enterprise resources needed to fulfil interoperability tasks related to the messages. Interface to legacy systems and existing infrastructures was designed using an SOA approach and other standard access protocols.

- To devise a framework for semantic interoperability. Pursuit of this included the investigation and development of reference business models, structures and semantic descriptors. Semantic Interoperability assures a quick and painless alignment using an extensible core of semantic descriptors which characterizes relevant business document types. Based on text analysis techniques semi-automatic semantic annotation of documents is performed. Protocols were developed to support automatic negotiated semantic alignment. The core ontology will be (semi-)automatically extended into a particular (business) domain through enrichment with specific concepts and (document) instances.

- To investigate, design and develop methods and Components for process interoperability. Thus, methods were developed enabling users to model recurring business processes. We examined novel ways to configure process model dependent, cross enterprise (e-Mail) communications and considered novel approaches to (semi-) automatic process detection. Process Interoperability addresses seamless entry into a community of interoperable
companies not only at the document but also on the process level. Here, appropriate coordination of activities within and among partners is supported by the COMMIUS interoperability modules. Reference model templates and patterns of Business Processes are identified and mapped onto executable protocols. Algorithms to detect Business Processes based on communication and techniques for inter-enterprise process negotiation and process adaptation were implemented.

- To deliver an open source framework for interoperability and collaboration suitable for seamless integration within a network enterprise environment and focused on SME’s needs. The COMMIUS solution will be modular and adaptable to different ICT infrastructure standards (which is often quite low in many SMEs). It will be able to evolve with enterprise interoperability needs. Open source is a key issue for the Commius interoperability framework and will support new business opportunities for SMEs in Enterprise Interoperability markets.

- To develop a pilot application allowing the validation of three industrial use cases:
  1. Support SMEs in the management of customs and export procedures in Spain
  2. Network of SMEs cooperating in a large software production application in Italy
  3. Support enterprise resource planning interoperability in Greece

In the following sections, we will describe the techniques and approaches investigated to reach these objectives, first describing the overall architecture and them describing the results obtained in the key innovation areas defined: system, semantic and process interoperability.
An Example

Let’s consider a concrete example of how Commius works. Commius hooks into a company’s email infrastructure, in a similar way as a common anti-virus or anti-spam software. Being designed to work both with POP3 and IMAP protocols, Commius comes into two versions; one version is designed to be installed as a mail server extension, while the other is installed on the mail client machine and operates locally. For the sake of simplicity, we will consider the latter case in this example, being the considerations valid for the server-side Commius version as well.

The scenario we are considering here is a very generic case where an SME employee receives an email containing, for instance, an order of purchase of some products the SME sells, sent by one of the company customers. This mail will include details of the order is a more or less structured way, for instance mentioning the name or product id of the required goods. Also the identity of the order sender is of course included in the incoming email.

Commius uses this information to infer which business process is currently going on, and to decide whether additional data or functionality could help the receiver to exploit the information contained in the incoming message. It then includes links to such data (or the data itself) or to the external applications in the email body. This is done by enriching the message body with HTML information, which is currently interpretable by every common email client.

The picture below illustrates this by showing a screenshot of a web client just after receiving a Commius-enriched email:
The center of the window shows the original mail text. This has been enriched by Commius in two ways: on the left of the screen a panel shows the business process steps as detected by Commius, while on the bottom of the email a set of links provides access to relevant additional information or useful applications supplied by the so-called Commius Business Models (More details on all this will be provided later in this document).

The picture in the previous page also shows partially a window of one of such Business Models user interface, which is provided by an external Web Browser window.

In order to understand the process by which these email enrichments are created, below the simplified version of Commius architecture is shown (a more detailed version of this diagram will be shown in later chapters).
The process begins when the email is received in the receiver’s mailbox. Here, it is intercepted by a specific Commius component called the Commius eMail Gateway (1). The task of this component is to capture incoming emails and pass them to Commius architecture layers.

The first layer which processes the eMail is the so-called, System Interoperability Layer (2). The main task of this layer consists in extracting a set of basic information tokens from the email text, using Information Extraction techniques explained in the following sections, in the form of key-value pairs. These information tokens consist, for instance, in product names, product ids, customer names, telephone numbers and so on. This layer’s responsibilities also include the extraction of text information from attachments such as pdfs, word documents etc.

In the next step, the email, together with the set of key-value pairs extracted by the System Layer is passed to the Semantic Interoperability Layer (3). Here, the information tokens are classified semantically using an extensible ontology based on the UN-CEFACT “Core Components” standard, then based on probabilistic evaluation on the set of detected semantic concepts, the whole email is classified as one of a set of possible “Document Types”, for instance a purchase order, an invoice, a request for information, etc. More on this in the section dedicated to the Semantic Layer.

The third of the main steps of analysis of the email is performed by the Process Interoperability Layer (4). Here, the system tries to identify from the sequence of document types it is receiving, which business process (and, inside this, which business process step) this email represents. For instance, after receiving a purchase order, an offer is made. The information of the current process step is then used in order to determine relevance of additional information or extended functionality to provide as email enrichment.

For this latter step, the system makes use of two kind of extensions provided by Commius architecture. The first extension mechanism, the so-called System Connectors (5), provides a way to connect to external data sources and legacy systems that might be part of the user’s IT infrastructure, like CRM systems, external quotation applications, or even a simple datasheet containing stock information.

The second extension mechanism is used to provide customized functionality to the Commius architecture. A set of extendible software components, called Business Modules (6), provide domain-specific or process-specific information.

System Connectors and Business Modules are used by the three main Commius layers to determine the contents of the enrichments to put in the modified email. A specific component, the Mail Post Processor (7), which is technically part of the System Layer, takes care of inserting such contents into the email body as HTML text.

This short explanation was meant to give a rough overview of the general process an email is subject to when received by a Commius-enabled system. In the following sections we will delve into more details about Commius general architecture and about the research and technical issues concerning the main interoperability layers and mechanism extensions.
Commius Overall Architecture

The following figure presents a more detailed view of Commius overall software architecture. For ease of representation, the three main layers are represented upside-down with respect to the previous simplified picture. The following sections will describe in a detailed way the different parts of this architecture.
In summary, Commius architecture includes the following main parts:

- **the Email Gateway Plugin**: is responsible for intercepting emails and passing them to the main architecture layer for information extraction and semantic analysis. It also takes care of post-processing emails (through the Message Post Processor, MPP, sub-component) after these are processed by the three layers. Email Gateway Plugin and Message Post Processor are technically part of the System Interoperability Layer, but are mentioned here separately for more clarity.

- **System Interoperability Components**: include Components providing functionalities for achieving system interoperability. See Commius System Interoperability section for a detailed description of this layer and the parts it’s composed of.

- **Semantic Interoperability Components**: include Components providing functionalities for achieving semantic interoperability. See Commius Semantic Interoperability section for a detailed description of this layer and the parts it’s composed of.

- **Process Interoperability Components**: include Components providing functionalities for supporting process interoperability. See Commius Process Interoperability section for a detailed description of this layer and the parts it’s composed of.

- **Data Management Components and Security and Privacy Components**: includes Components providing data management facilities which are needed by Commius, and Components providing security and privacy infrastructure for Commius.

- **Commius Business Modules** (or just Commius Modules) and Module Management: include system- and business-related Modules that handle interoperability tasks represented by email messages passed through the Commius. Modules in Commius will be built from Components which can be provided by different layers.

- **System Connectors**, providing a standardized interface to external or legacy systems.

The architecture described includes the Email Gateway Plugin which intercepts and post-processes the emails, and two main software categories, Modules and Components. Modules have well-defined interface; they can be exposed as software services, thus allow them being accessed through the Web. Components interact with each other via Component interface which defines data representations and operations to control the behaviour of a Component. In the Commius architecture, we did not define and manage common data repositories. Instead, data is exchanged via Component interface. For example, common profile information of users is managed by data management Components.

The Components of the Commius architecture can interact with external systems, which are not part of Commius. External systems include common legacy systems in SMEs as well as specific SME legacy systems and other services supporting the business of SMEs. While the main focus of Commius is to support legacy systems, Commius will also provide means to facilitate the interaction between core Commius elements and other external systems, e.g. Web services-based document repository, as well.
The architectural overview above describes a single Commius system which can be deployed for a single SME. Such a deployment can be done by the SME itself or by a Commius Service Provider. Individual Commius systems can be connected to establish a cooperative network of Commius to support a network of SMEs.

In the following sections a description of the main parts of the Commius Architecture, with the research issues addressed and the results achieved, is included. The lower layers of the architecture: Data Management and Security and Privacy Components, which, though important, are supported by standard, well assessed software methodologies and tools, are not included.
Commius System Interoperability

System or technical interoperability includes standards, protocols and also architectures which are built on top of protocols and interoperability standards. These necessarily overlap with basic technical interoperability to enable seamless communication. In short, system interoperability refers to the ability to connect systems by defining standard protocols (e.g. SOAP, HTTP, IP) and data formats.

Nowadays, the following crucial technologies are taken into account when dealing with system interoperability:

- Service Oriented Architectures (SOA)
- Peer-to-peer (P2P)
- Mash-up technologies based on Web 2.0

Service-Oriented Architecture (SOA) is a natural evolutionary step from Object and Component based approaches. The main value of SOA approaches is that it provides a framework for matching and combining needs, as well as capabilities to address those needs.

There are three major trends in Service Oriented Computing (SOC): Web Services, Grid Services and peer-to-peer (P2P) services. Web Services build on XML standards to provide a platform for building many distributed applications. New Web Services can be created on-the-fly using any existing Web Services (software components).

Grid Services came originally from the Grid Computing needs which are accessing distributed computational (grid) resources. Nowadays, this has moved slightly away from original Grid definition and includes software, data and knowledge capabilities. P2P has much success and has potential to be the most powerful trend; however, it still lacks any consensus on how applications should be built and what semantics should be supported.

Other interoperability approaches come under “Web 2.0”, which is characterised by the development of “lightweight” software (from standard technology building blocks) that can be released quickly over the Web using HTTP protocol. In addition, “mash-up” technologies use public APIs of (mostly big) firms with given infrastructures or databases to provide new services. These developments offer the means for an SME to significantly lower its “cost barrier” to entry into new businesses and markets.

Research and development in the area of SOA, P2P or Web 2.0 as well as standards in specific industry sectors brought valuable results for interoperability. However, SMEs are still excluded due technology and resource difficulties with such solutions. We believe solutions like COMMIUS, built around existing SME infrastructure such as email and web can successfully address SME’s needs and also bring or wrap-up existing solutions to the use of SMEs.

Email repositories and email activity are valuable assets in any modern, internet based business organisation. Even small companies can generate large e-mail traffic and fill e-mail repositories
with high volumes of data needed to accomplish their daily tasks. Email is the second most-used internet service after Web. The following features are common in the use of emails in enterprises of all sizes, including SMEs:

- Every organisation, without exception, will have an e-mail infrastructure before it reaches the stage of developing or adopting any interoperability solution.

- E-mail communication in a modern organisation is over 78% action-oriented, according to a study. Communication is perhaps the foundation for most organisational action.

- Managers, and knowledge workers of all kinds, interact with their e-mail systems on a daily basis.

When building a solution on top of email communication, an organisation does not have to change its way of doing business when such a solution is installed and set up in organisation. Users simply receive emails as before, but additional information or knowledge relevant to the interoperability or collaborative aspects is Attached, as appropriate to the email. Work to connect knowledge or context-sensitive information with emails has been done in several projects such as the kMail system: this integrates e-mail communication with organisational memories, however, it also forces users to use a special email client and lacks a closed knowledge cycle loop. Another related tool is Zimbra, which offers web based client with functionality to detect objects such as phone numbers or addresses and allows some actions on these objects. Similar to kMail, Zimbra requires a particular email client and server application and thus changes existing ICT infrastructure in organization on both client and server side. Gmail, a webmail developed by Google, supports content-sensitive advertising and some actions as “add event to calendar” with the email.

Extensive work on email processing and active context sensitive information and knowledge provision has been undertaken by Commius partner IISAS, where ACoMA and EMBET frameworks were developed.

Commiius will innovate by building an interoperability service utility above existing ICT infrastructure. It exploits existing useful properties of email, such as action oriented communication, available in all enterprises and the asynchronous nature of email communication. Commiis can be used to complement to existing interoperability architectures through an automatic process based on SOA, P2P and Web 2.0 mash-up approaches. Moreover, standards as ebXML can be started upon receipt of an email by invoking an appropriate interoperability module to process information and data extracted from the email. On the other hand, if no Commius is installed, when received an email will still be human readable (e.g. by using XSL template within attached XML message) and thus can be processed manually if needed. In addition we can support interoperability tasks requiring human interaction or intervention.

The System Interoperability Layer

COMMIUS System Interoperability Layer groups the Components focused on designing and developing a basic interoperability infrastructure over SMTP. It supplies methods and Components to provide data, information and functionality to be used by interoperability Modules and other
interoperability layers inside COMMIUS architecture, namely the Semantic and Process Interoperability Layers, with their software Components.

The COMMIUS platform is hooked through the System Layer into the mail server or mail client, similarly as email antivirus programs are used at the server or desktop side. This way the system can be used within any email client or even mobile device, without requiring changes to working practices or the adoption of new tools. By combination of server or desktop use, users can ensure that security and privacy issues are taken into account. Users and providers can be aware of what data is shared, passed via the communication.

Email communication is passed through the COMMIUS system, processed, and additional information is added to email messages in form of links in HTML or text attachments. This additional information contains relevant information and knowledge, hints or links to business resources such as document repositories, databases or information systems needed in detected business context. The business context is detected from email using semiautomatic pattern based semantic annotation using predefined regular expressions patterns.

When checking new emails, a user will receive a modified email message, with embodied information. Please note that inline text attachments are directly displayed in most of email clients and they appear as part of an email message, however they do not change the email message itself. We also allow possibility to change email into the HTML message with additional info and add original message as attachment. Thus, the users can configure COMMIUS according to their needs depending on what devices they’re using, the email client with which they send and receive emails, or depending on user preferences.

In summary, the functionality provided by this layer includes:

- **Connection to Email Infrastructure.** Technical tasks, such as connecting the COMMIUS architecture with widely-used email servers such as Exchange, Postfix, Qmail or Sendmail, are addressed by the COMMIUS System Layer.

- **Message processing:** text analysis and information extraction techniques. Information analysis and extraction is based on advanced pattern matching techniques (regular expression, XPath, XML data), information retrieval methods, POS tagging, NLP methods, and statistical analysis, identifying and developing the most suitable methods, based on analysis of content involved in (interoperability) communication. Information analysis and extraction is also used by semantic annotation and other interoperability Components from the Semantic and Process Interoperability Layers.

- **Connection to existing SME infrastructure.** An important task consists in connecting to existing SME infrastructure(s), document repositories and legacy systems using system connectors built on standard interfaces, such as web services, SQL-based database access, XML-RPC or file access. The COMMIUS System Layer develops connectors to access and feed data to intranets and internet applications using Web 2.0 technology, such as URI based access or wrapping of HTTP requests. Connectors are used by relevant interoperability Modules, and in semantic and process interoperability layers to provide required connections to enterprise information resources. Security and policy concerning information access are considered and employed where applicable.

- **Message post-processing.** It covers the inclusion of active information elements (URLs of resources, action buttons, etc.) relevant to fulfilling the interoperability task(s) relayed by the message.
In the following sections, each of these is described in some details.

**Connection to Email infrastructure and the Email Gateway Plugin (EGP)**

There are two basic approaches to connect COMMIUS to the e-mail infrastructure: Using either client-side or server-side connection. In the case of client-side connection the software responsible for e-mail processing is installed on a client’s computer, while the case of server-side connection, the e-mail processing and e-mail enrichment is enacted on the server upon message receiving or sending. There are further possible categorizations. One such important categorization for us is that the server-side connection might be either hosted on a dedicated server or plugged-in into an existing e-mail server system.

In the latter case the EGP (Email Gateway Plugin), the COMMIUS Component in charge of connection with email infrastructure, is integrated at server level, the communication between EGP and the server is enacted in SMTP (Simple-Mail Transfer Protocol). In the case of client-side version of EGP connection, the EGP is deployed at the user’s desktops. POP3 protocol is supported for the client-side connection.

The main difference between the server and client-side connection is the place where the email is enriched. Client-side connection can be viewed as distributed while server-side connection is more centralized. Consequently, client-side EGP enables e-mail processing only for a single user while the server-side connection enables e-mail message processing available for more e-mail recipients while naturally providing higher support for collaboration and knowledge/information sharing.

**Information Extraction**

In order to provide useful recommendation, we need to identify objects and object properties in the text so as to formalize email message content and context. To this goal, in COMMIUS we use Information Extraction and semantic annotation techniques. In our solution we use the Ontea annotation tool, developed in the course of a previous research project, and now available as an Open Source framework (http://ontea.sourceforge.net/). Ontea is based on regular expressions, but other advanced NLP (Natural Language Processing) and Information Extraction tools and approaches can be integrated easily by producing key-value pairs.

The annotation tool identifies objects, their properties or their position in the text by applying patterns (regular expressions) on a text. The input of the method is the mail text and the set of defined patterns, and the output is the set of extracted key-value pairs, which can be chained by various transformations such as lemmatization, relevance identification, system connectors’ transformation or transformation into RDF/OWL ontology individuals if needed. The power of this approach is in its simplicity compared to more advanced but heavy solutions such as GATE (General Architecture for Text Engineering, see http://gate.ac.uk/) as well as ability of transformation chaining and connection with information system environment (databases, documents, intranets, and internet).
The aim of information extraction is thus to create and share patterns for objects and their properties. We believe the best approach to be used for information extraction will be pattern-based extraction for several reasons:

- Pattern can be adopted for enterprise business needs. For example, product codes can differ for different companies.
- Patterns can be defined, improved, evaluated, and shared for groups of SMEs or communities around similar business models or industry types.
- Patterns are applicable for different languages without the need for advanced NLP tools.

Email content contains many objects extractable by patterns, such as email addresses, phone numbers, people names, company names, dates, websites, addresses, or other contact details. In addition, social networks, interaction, and message passing can be extracted from email headers using patterns.

Interoperability content in general usually contains many objects and properties which can be extracted via patterns such as amounts, money amounts, product codes, bank account numbers, or customers. One of the possible evolutions of this work is toward intelligent pattern creation with the user involved in the loop.

Meanwhile, patterns (e.g., regular expression) need to be defined manually, and success of extraction depends much on how well the patterns are defined and tested. On the other hand, such solutions are very powerful in an enterprise environment where business-specific patterns need to be defined in order to identify business objects. Patterns can also be shared within a community when defined and tested for common types of objects such as people names, companies, geographical locations, addresses, or contact details. In this case, one can use them directly and know its quality and expected success rates. When having such tools for detecting needed objects and their properties, they provide us the context for recommendation systems.

Social Network Information Extraction

In terms of Information Extraction, particular interest is covered by the extraction of Social Network information: Email communication analysis allows extraction of social networks with further connection to people, organizations, locations, topics, or time. This information can be extended into a graph of relations among elements detected in the messages, segments of message parts, or relations in the conversation threads.

We believe we can use such a graph to identify the meaning of a message in terms of the most relevant elements or business objects related to the message itself, even if the object does not have to be directly extracted from text. For graph inference, we would like to use spread activation in a similar way as IBM Galaxy in the Nepomuk project (http://nepomuk.semanticdesktop.org/). We believe such an approach for inference is especially good in the way we use COMMIUS as a kind of a recommendation system.

The use of social network within the email for better email management was already proved to be valid in Xobni System (http://www.xobni.com/), which uses social network to suggest relevant contacts, attachments, or email messages for the user. While Xobni can provide general suggestions
based on generic social network, in COMMIUS we can provide suggestions based on general, interoperability specific and application specific context, in case COMMIUS is customized and connected with enterprise a legacy systems.

Connection to Existing SME infrastructure

An important task within the System Interoperability Layer is to connect the COMMIUS system to existing SME IT infrastructure. COMMIUS should connect and provide interoperability with legacy systems such as document repositories, relational or XML databases, spreadsheet documents, intranet applications and ERP systems, used by SMEs to store their data.

COMMIUS Components providing access to those legacy systems are called System Connectors. Other COMMIUS Components can utilize System Connectors via unified interfaces, which provide necessary transparency hiding the complexities of interactions with legacy systems.

COMMIUS exploits the information extracted from the SME legacy systems to annotate and enrich incoming e-mails with relevant data. In addition, System Connectors invoke operations on legacy systems, as required by business processes, managed by the process layer of COMMIUS.

In order to support interaction with legacy systems beyond the boundaries of COMMIUS pilot scenarios, System Connectors will be easily extensible and configurable to provide the needed flexibility. Also, two security issues have been considered for systems connectors: usage of security tokens required to access legacy systems and authentication and authorization to invoke system connectors.

Common legacy systems required in COMMIUS user stories are:

- SQL databases
- Spreadsheet Applications
- Web applications
- Document Repositories
- Legacy enterprise systems (ERPS, order management systems, etc.)

In terms of architecture, a System Connector Logic sub-component is responsible for pre-processing of input data, retrieval of data/information from legacy systems and post-processing of extracted data to the required form, while Module Specific Configuration defines parameters required by Legacy System Client as well as System Connector configuration (e.g. RDMS System Connector configuration may contain RDMS client configuration - server, port, database - and definition of a query to be issued from the System Connector). Finally, a Legacy System Client provides mechanisms for connecting and retrieving data from legacy system.
**Message Post-Processing**

As its name suggests, Message Post Processing (MPP) Component is responsible for post processing of email messages. This includes message parts composition as well as composition and integration of information and results from other interoperability layers and interoperability Modules. The message is modified to include relevant information and active information elements (links, buttons) needed to fulfil interoperability task related to this message.

The MPP Component represents the last Component in the chain passing through system, semantic and processing layers visualizing its outputs. Such outputs can require different level of functionality and complexity. The integration of outputs from different Components to a single easily understandable user interface is the main goal of the MPP Component. MPP comprises of the two visualization sub-Components according to the resulting visualization application:

Email output is usually built as static HTML or text content that can be easily incorporated into any email client. The original email message can be enhanced by additional information or the email is reprocessed and the information is incorporated directly into the email content.

Web browser output providing information content is built using mash-up technologies. The cornerstone of the visualization sub-Component is Google Web Toolkit (GWT) library that provides basic application interface for building dynamic user interface (including layout, styles, application logic, etc.). GWT framework provides a novel approach of web application development according to AJAX methodology.

According to the usage scenarios of COMMIUS project, the message post-processing Component should be deployable as server application as well as client application. Such different scenarios bring many pros-n-cons for particular deployment, but the main reason is to allow user(s) to control scalability and security. The first application design of the MPP Component counts on the Jetty application server that is suitable mainly for client-side deployment, but is also usable on server-side. Later MPP should be run from appropriate application server residing in an OSGi container. The MPP Component lays on the modular design, where the core of MPP application is designed and implemented as main layout and visualization Component which allows loading Modules from separate java archive files. This allows separating the development of general GUI design and specific module visual widgets. This kind of modularity is solved within the scope of MPP module and should not be misinterpreted that this feature is a part of OSGi framework.
In general terms Semantic interoperability signifies meaningful communication among two or more parties. Generally, it refers to a shared understanding of data, information, process models, etc., within and among organisations. As such, it underpins other kinds of interoperability between collaborating enterprises and thereby constitutes an essential prerequisite for enterprises to be truly interoperable. It is typically achieved by fixing a unique meaning (semantics) through a commitment to a common ontology.

Currently, semantic interoperability is typically achieved “off-line”, chiefly through adherence to standards governing both the structure of the information being exchanged and the manner in which this information should be processed (for example RosettaNet and ebXML); or in particular, through commitment to a shared ontology, which is then used as metadata.

Indeed, many proponents of the Semantic Web seek a universal medium for information exchange based upon XML syntax. This has given rise to such standards as the Resource Description Framework (RDF) and its elaboration in RDF Schema or the Web Ontology Language (OWL); and is also reflected in the vision of Semantic Web Services and related standards, namely, the Web Service Modelling Ontology (WSMO), the Web Service Modelling Language (WSML) or OWL-S. The predominant use of ontology to foster semantic interoperability is reflected by the numerous research efforts, and software tool development and support in this area.

In response to approaches to ontological modelling, such as those cited above, e.g. RDF, a number of tools for ontology editing, storage, querying and reasoning are now available. These include several semantic frameworks for accessing and manipulating documents in OWL, RDF and RDFS. The most prominent semantic framework is Jena. It provides a wide range of functionality through its APIs.

Ontologies can be created using ontology editors, such as Protégé. This is an open source development environment for ontologies and knowledge-based systems which was developed at Stanford University (USA). OWL plug-ins for Protégé supports the editing of OWL ontologies and the ongoing CO-ODE project provides valuable support for user modelling in OWL.

Automated annotation of the Web documents is a key challenge for the realisation of the Semantic Web. Web documents are structured, but this structure is typically understandable only for humans, while the semantic web requires them to be machine understandable. This is typically achieved by annotating them with concepts from a formal ontology. Several annotation protocols exist, including Annotea, Rubby and RDF annotation. Annotation solutions can be manual, such as CREAM and Magpie, or semi-automatic based on natural language processing (NLP), a document structure analysis or a learning approach (which requires training sets or supervision). Moreover, there exist pattern-based semi-automatic solutions, such as PANKOW and C-PANKOW, SemTag or pattern based approach Ontea.

In COMMIUS we will exploit the pattern-based solutions supplied by the COMMIUS System Layer, which uses metadata to semantically enrich messages, communications and also documents exchanged (See the previous issue of this newsletter for a more detailed description of the COMMIUS System Interoperability Layer).
The creation of domain specific ontologies and in particular as extensions of agreed, “upper” ontologies achieves semantic interoperability through semantic agreement. Developing domain specific ontologies is clearly not “zero-cost”. Moreover, this agreement on a common semantic model means that existing approaches to semantic interoperability are typically centralised. Naturally, many have recognised that the use of a single ontology is untenable in a distributed environment and this has led to research in “ontology alignment”, for example, by mapping from one to another or by creating some “meta-structure” to relate the ontologies. In either case, one must “learn” the local structures and create appropriate mapping rules or “meta-structures”. Again, this is clearly not “zero cost”.

An alternative approach to this problem has been investigated as part of the MaBE and Crosswork European research projects. It advocates abstracting from matters of syntax to focus on the information passed within a concept. When a concept is passed to a recipient with a local ontology, the approach allows a range of options, starting from only retaining the subset of the concepts overall information which that recipient will understand; to a decision by the recipient to learn the complete new concept because of the future utility of this concept. This permits actors/trading partners to meaningfully communicate without the need for potentially expensive ontology agreement or alignment.

The Semantic Interoperability Layer

Traditional approaches to interoperability have adopted the upper ontology approaches discussed in the previous section - the set of documents which the companies will use to communicate and their detailed structure are both agreed upon and fixed. Such approaches are very accurate and work very well for large companies in stable partnerships but also incur considerable costs in their initial configuration and do not work well for more dynamic groupings.

The aim of the semantic interoperability layer, which includes the semantic interoperability software components, within COMMIUS is specifically addressed to provide interoperability for SMEs. Such companies are typically engaged in small projects with dynamically changing partners, and indeed cannot rely on the documents they receive being in any kind of formal feedback. Indeed a crucial overall goal of the COMMIUS project is to produce a piece of software which a single company can easily install and use, and so the system can make no assumptions regarding the structure of incoming documents.

The principal challenges faced by the layer are therefore to:

- Allow users of the system to specify a set of internal documents which they find significant enough to wish to detect and treat automatically,
- Interpret incoming, free text, documents and identify which, if any, of these document types they most closely correspond to.

In order to do this the semantic interoperability layer uses the following features:

- The software comes with a set of ‘core Components’ - these represent the basic information tokens that are expected to be found within business documents.
• When a user first installs COMMIUS they define both the set of internal document types they consider of interest and the set of core Components they would expect to find in a ‘typical’ example of each document type.
• When an e-mail is received text mining techniques are used in order to detect the set of core Components present within that e-mail.
  o The set of core Components detected is compared to the definitions of the internal document types and a set of scores produced, together with a recommendation concerning which document type the e-mail most closely matches. The user is free to override this recommendation.
• Classification algorithms which, over time, learn from the sets of core Components in the e-mails assigned to each class are also under development. In addition to providing a basic idea of how to interpret each e-mail this data is crucial for the operation of the process layer. Finally the detection of the core Components present within each e-mail enables the COMMIUS system to automatically fetch additional information of use to people interpreting e-mails.

The primary aim of COMMIUS Semantic Interoperability Layer is to establish a framework to allow collaborating SMEs to quickly and painlessly achieve semantic alignment. As a first step, we compared the UN CEFACT “Core Components” standard with the document types used by our case study partners, and also with important business ontology standards. The findings were used to identify a core of semantic descriptors which will be used as “building blocks” to characterise relevant business document types.

A first characterisation of documents from our case study partners provided the initial core ontology of business documents. Together, these two aspects constitute an extensible “Semantic Core”.

The second step has been the design of mechanisms to support each new partner in applying this Semantic Core to its own documents, extending and customising it where appropriate. This mechanism includes a “feedback loop” to enable an automatic enrichment of the core ontology with those specific concepts and instances which are found to be common among a number of partners. This is linked with (semi-) automatic semantic annotation and appropriate results from the System Interoperability Layer, for example, the automatic analysis of email messages and existing partner documents. Work done also include the development of a visualisation tool to support users in mapping the structure of their existing documents and databases to elements of the core ontology. This is meant to support both the classification of existing documents onto an existing core and inform appropriate re-categorisations (of the core) based on specific domain concepts and instances.

The third step consisted in developing semantic alignment protocols between trading partners. Upon receiving a message containing an unknown document type, the ontology alignment mechanism first attempts to automatically find a match for this in the ontology of the receiving partner. “Close enough” matches are handled by adding a “note” containing the additional information. If this fails, the mechanism initiates a “hidden” discussion with the sender about either using a more abstract document (one from the core ontology, for example), provided the two parties are happy to tolerate a certain loss of information for this particular transaction. However, if this partnership is of a particular importance for one of the parties, they may be prepared to “learn” or internalise a new document type, automatically adjusting their document repositories for the new document.

COMMIUS architecture’s Semantic Interoperability Layer contains several basic software Components, briefly listed below. A detailed discussion of these Components and their operation
can be found within the project public architecture deliverable D3.1.2, available on the project web site (http://www.commius.eu/).

- The **Semantic Manager** functions as the Semantic Interoperability Layer interface. Its basic functionality consists of managing all the semantic interoperability, from receiving a document for initial semantic annotation to initiating any needed semantic negotiation. It provides the layer output as well consisting of Mapped Documents.

- The **Semantic Annotator** parses Documents to identify valuable information in terms of CC for further mapping to internal Document Types. It uses a combined approach of machine-learning and pattern matching.

- The **Document Interpreter** reads the set of Core Components found in an Annotated Document and finds an appropriate correspondent to a Document Type. This is done with the aid of the Semantic Core and the Document Type Repository which are managed by the DocumentTypeManager.

- The **Semantic Negotiator** is in charge of reaching common agreements with any other Semantic Negotiator in another COMMIUS system when the Document Interpreter cannot find a close mapping to a Document Type. Such agreements consist on finding a new common set of Core Components to represent the same information.

- The **Document Type Manager** functions as an interface to manipulate both the Semantic Core and the Document Type Repository. It provides operations for the creation, addition, updating, and removal of Document Types. It updates the Semantic Core by allowing the addition of new Document Instances.

- The **Semantic Core** consists of an FCA lattice representing the Core Components relationships existing among Document Instances.

- The **Document Types Repository** is a collection of Document Types available to a COMMIUS system.

The following figure shows the Semantic Interoperability Layer Components along with their dependencies and interactions. As can be appreciated, the Semantic Manager functions as a layer interface receiving Incoming Documents from the System Interoperability Layer and providing Mapped Documents (and possibly additional information yet to be defined) to the Process Interoperability Layer for supporting business processes (which will be described in the next issue of this newsletter). Moreover, it controls the execution order of the subordinate Components. First the Semantic Annotator, then the Document Interpreter and finally, if needed, the Semantic Negotiator. The first two Components modify the initial input which is an Incoming Document transforming it into an Annotated Document and a Mapped Document, respectively, which is the final output of the Semantic Interoperability Layer. The Semantic Negotiator works on a partial Mapped Document only when a concept negotiation is required.

The Document Type Manager provides an interface to manipulate both the Semantic Core and the Document Type Repository. It supports the Document Interpreter for mapping Annotated documents into Document Types as well as the Semantic Negotiator for updating both the Semantic Core and existing Document Types according to the negotiation results. Finally, the Visual Mapping Tool can also operate on the Semantic Annotator, Document Interpreter, and Document Type Manager for assisting the user in manual Local Document classification.
Semantic Annotation and the Extensible Semantic Core

The purpose of the Semantic Interoperability Layer within COMMIUS is to attempt to correctly interpret the information contained within documents received by an SME. Semantic annotation is a way of assigning a “meaning” to the set of terms extracted from the emails. Semantic annotations are essential in the process of understanding email content. Thus it is important to correctly annotate as many entities as possible in the email text. With semantic annotation we understand metadata assigned to a text entity. Such metadata semantically describe entities and thus semantic annotation provides meaning to them.
Manual annotation is a kind of semantic annotation created by a human. However the use of manual annotation is sensitive to errors due to factors such as familiarity with the domain, amount of training, motivation and complexity. Manual annotation is also quite expensive process and rarely considers different perspectives of the same text by means of different taxonomies. Another problem with manual annotation is the volume of information, which for emails is quite extensive. Therefore the manual semantic annotation has led to a knowledge acquisition bottleneck.

To overcome such a bottleneck, semi-automatic annotation of emails is implemented in COMMIUS. Fully automated annotation methods are not yet possible with the current technologies, and thus all existing annotation systems rely on human intervention. In contrast, semi-automatic annotation provides the scalability needed to address large volume of emails and reduces the burden of manual annotation. The currently available semantic annotation platforms can be classified based on the type of annotation methods used. The main two categories are pattern-based annotation and machine learning-based annotation.

**Pattern-based annotation** platforms perform pattern discovery or have patterns manually defined. An initial set of entities is defined and the corpus is scanned to find the patterns with existing entities. New entities are discovered, along with new patterns. This process continues recursively until no more entities are discovered, or the user stops the process. Annotations can also be generated by manual rules to find entities in the text. In the domain of email annotation, the regularity of the patterns is as obvious as it is to frequently find email addresses, dates and time patterns, etc., and therefore the initial set of annotations can be easily accomplished by such approaches. However, pattern-based methods as such are not sufficient as there is significant amount of further information to be extracted and used.

**Machine learning-based** platforms use two methods: probability and induction. Probabilistic platforms use statistical models to predict the locations of entities within text. Induction methods use the machine learning induction algorithms to induce the initial set of models and then refine them based on further samples. Machine-learning based platforms usually require learning and testing corpus in order to train the model. In the email domain this has shown to be quite complicated due to the existence of only a few corpora suitable for annotation.

Semantic annotation in COMMIUS is based on a multi-strategy platform, i.e. a combination of pattern and machine learning methods. The COMMIUS approach focuses on a simple approach based on regular expression patterns with possibility to integrate existing advanced semantic annotation approaches based on NLP or machine learning techniques.

Within this context, the purpose of COMMIUS Semantic Core is to provide a language which is suitable for describing the basic units of information contained within both the documents used internally by SMEs and those documents that they typically receive by email. Such basic units are represented by a set of identified semantic building blocks which are called Core Components.

The COMMIUS Semantic Core is built upon the UN/CEFACT CC (Core Components) standard. This standard describes and specifies a new approach to the well-understood problem of the lack of information interoperability between applications in the e-business arena. A Core Component (CC) is defined as “a building block for the creation of a semantically correct and meaningful information exchange package. It contains only the information pieces necessary to describe a specific concept”. The CC standard presents a methodology for developing a common set of semantic building blocks that represent the general types of business data in use today and provides for the creation of new business vocabularies as well as restructuring the existing ones. This
standard also provides a way to identify, capture and maximise the re-use of business information to support and enhance information interoperability across multiple business situations. For example, the CC technical specification can be employed whenever business information is being shared or exchanged among and between enterprises, governmental agencies, and/or other organisations in an open and worldwide environment. A key aspect of the approach to semantic interoperability involves the removal of document structure and the translation of meta-data (e.g. key-value pairs) into descriptors deriving from the UN/CEFACT CC, essentially standardised semantic building blocks to promote information interoperability. The removal of structure and translation into standardised semantic building blocks produces an abstract version of the original incoming document. With it, it is possible to align or map the document to a structure specific to the company, which in COMMIUS terminology is called a Document Type. In the end, such an alignment is the output of the Semantic Interoperability Layer which provides support to the Process Interoperability Layer or any other module that requires semantic matching.

From an implementation point of view, the Semantic Core can be roughly defined as a lattice structure representing how the semantic building blocks included in documents are interrelated one another providing an idea of similarity among documents. This functions as a repository where the internal semantic building blocks represent documents are interrelated emphasising their similarities between one document to another. The extension mechanisms on the other hand aims to find document categories common to a set of collaborating partners using COMMIUS. The current status of this mechanism is the definition of the approach, i.e. to use emergent semantics over a network.

A running COMMIUS system may receive different types of Documents e.g. special requests, purchase order, invoices, quotations, etc., and the Semantic Interoperability Layer should be capable of correctly interpreting those Incoming Documents. Starting from the CC standard, the COMMIUS Semantic Core is built from real documents from our case study partners. Such documents were used to construct the lattice of concepts by means of Formal Concept Analysis techniques.

In the following figure, an example of a Document interpreted using the Core Components is shown: The left part of the figure below shows an example of an Incoming Document which is annotated upon reception and some Core Component related information is highlighted. The figure shows a yellow text highlighter only for illustration purposes, the annotation output is not expected to be delivered in this way between the Semantic Interoperability Layer Components. However, the Annotated Document is expected to be represented in ebXML using the Core Components tags. The right part of the figure shows the Core Components extracted from the email in ebXML syntax.
The ebXML document is then used in the document alignment process by the Document Interpreter aided by the Document Type Manager. After the completion of the document alignment process the system stores a record of the set of core Components found within this instance together with a note of which internal document type it was recognised as. Some of the classification algorithms under consideration within COMMIUS use this information to improve their classifications over time.

The Visual Mapping Tool

One of the goals of COMMIUS in relation to Semantic Interoperability was the development a Visual Data Mapping Tool (VDMT), or just Visual Mapping Tool for short, with which target users can easily work with documents of different Document Types, manipulating Document Types and instances, and mapping information to properties of documents. This tool is of particular importance in allowing users to easily define the set of internal document types which they wish to treat as significant internally.

The provided visual mapping tool comprises of four discrete areas (also called panes); the left pane, the central pane, the right pane and finally the bottom pane. Each pane visualises different kind of information. The following figure depicts all panes together. The left pane is the Core Component visualisation pane. When the visual mapping tool is started an XML file that contains a document instance is visualised. This document instance contains an email that has already been processed by the Semantic Annotation. The tool “consumes” the Annotation output and automatically selects the proper Core Component that this document instance refers to. The selected Core Components are represented in a hierarchical tree structure. After this, the user can (de-)select items of the hierarchy. Any selected leaf Component automatically selects the parent Component.
As can be seen in the figure, identified Core Components are highlighted with different colours. The tool user is also able to edit, remove or add new annotations accomplished in the central pane in which the user highlights a sentence (for example) then clicks on the related Core Component in the hierarchy.

Additionally, the right pane shows the current classification of the document instance in the central pane. Next to each Document Type, a percentage is shown representing the similarity the document instance has to each of the Document Types. Moreover, this pane will eventually allow the user to browse Document Instances which will be shown in the central pane.

Finally, Document Types can be easily created with the help of the bottom pane. Document Type creation uses a Document Instance as the “starting template” which, as explained above, already contains assigned Core Components. These are the Components to be included in the Document Type. If the user is happy with those Components, then a simply click on the button “Save New Document Type” will show a small field for entering the name of the new Document Type. Then after saving it, a new Document Type will appear in the right pane.

However, if the user prefers to use additional Core Component for the Document Type, then by a simple GUI Drag & Drop functionality the user can choose those Components and drop them to the middle box in the bottom pane. By clicking on “Save New Document Type” button the Document Type will be saved as explained above.

A more detailed description of the visual mapping tool can be found in the project public deliverable D5.3, available on the project web site (http://www.commius.eu/).
Interoperability has been defined by the IEEE as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged".

From the point of view of information systems (IS), this definition states very well that an IS is interoperable if it is able to interwork with other ISs. In principle, this definition is also suitable to organizations. However, the transfer to the concept of business processes is not intuitive: applying the COMMIUS approach, process interoperability (PI) is the ability of a business process to interwork with other processes. But what does it mean?

A business process is a synchronized, continuous series of enterprise tasks, undertaken for the purpose of creating output (Scheer 1999). Thus, from the organizational perspective, a business process describes the coordination of enterprise activities. Its purpose is steering and control. Consequently, process interoperability should also be a concept on control and synchronization.

Interoperable business processes have to be able to integrate other business processes. In terms of control, this implies that PI describes a state in which an enterprise is able to steer activities, i.e. business process parts, within other firms or vice versa, to be partially steered by another enterprise.

Process interoperability itself can be incorporated by two prerequisites: On the one hand, enterprises need organizational and technological instruments to ensure an efficient control beyond their own borders. These instruments can comprise e.g. standardized forms at specific points of the processes documenting the transfer of control and responsibility or ICT systems tracking the progress of a business process and arranging the corresponding duties. On the other hand, the process design needs to be adaptive and changeable. Interworking with other firms or business processes mostly mean to change specific aspects of your own business.

As a result, having interoperable processes implies to have mechanisms incorporated within these processes to make them adaptable to the requirements that could come up when initiating new process relationships. Actually, processes are always changeable; it is only a matter of cost, effort and time. In respect to PI, such a broad understanding makes no sense. Rather, PI should imply that processes can be changed and adapted to new process collaborations with almost zero cost, effort and time of change.

In summary, we can agree with the definition of process interoperability recently proposed by Loos (2009): “Process interoperability is the ability of an enterprise to synchronize, partially control and integrate continuous series of enterprise tasks (i.e. business process parts) that are governed and executed outside of its own enterprise borders without the need to modify its own organizational and technical environment.”
The Process Interoperability Layer

COMMIUS Process Interoperability Layer, which included the process interoperability Components, represents an architecture which is the basis to provide the process-oriented concept during the run-time of COMMIUS. On the one hand, the functionalities of the layer offer users the possibility to configure company specific processes and on the other hand establish a basis for an executable process support based on standard email communication. Furthermore, the layer is responsible for processing an incoming email, according to an assigned document type, i.e. recognizing if this message can be attached to an already existing process instance (within the actual company or one of its business partners), determining the current state of that instance and finally provide additional information, as well as advice for further proceedings.
The Process Layer Input

To be able to allocate incoming emails with process steps, they must be identified by the semantic layer and allocated to a concrete document type (see previous issue of this newsletter on Semantic Interoperability). Beside a document type (a bundle of keywords that can be analyzed in the incoming email), process steps are equipped by functionalities which are provided by Modules in the semantically enriched emails. To bundle enriched keywords and integrated Modules, a step output style can be defined that determines how the processed email should look like. Taking these process step related functions into account, process steps can be seen as the integrating element of COMMIUS. The structure of a process step can be seen in the following Figure:

Once the Process Interoperability Layer receives all necessary input data, the actual processing is initiated, monitored and controlled by a Component handler, which is the single Component managing the communication and dataflow along the process of analyzing and enhancing the current message. It invokes the four main Process Layer Components (Detecting, Tracking, Assisting and Advising) and provides them with the necessary data, sequentially. The invocation sequence itself is fixed and will be repeated for each incoming email message, as the single Components depend on the information gathered by their respective predecessors.

The next sessions discuss in details these four main Components of Process Interoperability Layer: the Detection Component, the Tracking Component, the Assisting Component and the Advising Component.
The Detection Component

The first step along the execution of the Process Interoperability Layer is the Detecting Component which utilizes the incoming message with its containing information as well as the Enterprise Process Repository to determine if the process, the email refers to, already exists or if the process is a new one and a new process instance has to be initiated. This means that either the incoming email is already tagged with a corresponding process instance ID and the email can immediately be assigned to its overall process, or that the detection operation has to assign the incoming email to an appropriate overall process based on the information detected by the semantic layer. In the second case, a new process instance ID will be created and implemented into the email as a reference for follow-up emails concerning this case. Future incoming emails concerning this particular process will be assigned to this initial process instance henceforth.

The Tracking Component

As second step along the Process Interoperability Layer’s execution, the Tracking Component utilizes also the Enterprise Process Repository, as well as the document type and the semantic information gathered from the original incoming email, to track which process step within the given overall process is triggered by an email. It also updates the assigned process instance within the Enterprise Process Repository with all important data that can be applied in future monitoring and analyzing processes. Following this approach, every performed step and conjoined information within a process instance is documented and comprehensible for further disposal. This high amount of transparent information constitutes the precondition for further beneficial functionalities of COMMIUS.

Each performed step concerns two occurrences, actions and events. Actions signify human or application triggered activities, which, in the context of email interoperability, mainly occur sending an email. Events on the other hand have no active part and match, in the context of COMMIUS, incoming emails. Transcribing this concept on a order transaction process reference scenario, an order process instance will be derived and subdivided into several action or events, i.e. “sending order confirmation” or “order received”. Since every performed step is related to its unique process instance, it can be tracked and on this basis recommendations for further steps can be obtained.

In case the COMMIUS system is applied in a collaborative scenario, it may be possible, that the incoming email belongs to an overall process, where previous steps have been executed by other COMMIUS instances. In this case, the Tracking Component offers a synchronization functionality, which allows to synchronize already executed steps of an overall process throughout several COMMIUS instances.
The Assisting Component

In the third step of the Process Interoperability Layer's execution, the Assisting Component is being invoked. Its main goal is to add to the original mail additional context-sensitive business information that are useful to support the user for the given business process step.

This functionality consists of two segments: First, it provides case related information about the particular process step, like customer history, contact information or useful web links, and second, relevant process history is procured from the Enterprise Process Repository. In order to perform the needed operations the Assisting Component, through a well-defined interface, invokes and exploits the process step that has already been identified by the Detection Component before.

How needed operations are really executed depends on the internal implementation of the process step and its user-defined configuration/customization. Exploiting the process step manager the user can update process steps, also changing their internal behaviour, without modifying the Assisting Component and so the COMMIUS core.

The Advising Component

As the fourth and last step in the enrichment of an incoming email, the Advising Component utilizes both the internal databases, as well as previously gathered information on the incoming email, to provide the adequate and, in the course of the present process step, logical advice on further proceedings. If the incoming email has, for example, been identified as a confirmation of payment of an order transaction, COMMIUS would recommend triggering the shipment and procure reasonable options like forwarding the confirmation to the shipping department.

The gathered information is then being forwarded to the message post processing, which includes the auxiliary information into one email, allowing the respective user accessing the ascertained information. Using the method of embedding the information directly into an email, allows COMMIUS to be applicable as well in a collaboration scenario in which only one partner deploys the system, due to the wide spreading of the email standard.

A second functionality of the Advising Component, which is not directly invoked while processing an incoming email, but later via the embedded hyperlinks, is to provide advice in actually executing the next process step once the user has chosen one of the provided actions.

Like the Assisting Component, the Advising Component also depends on the correctly chosen process step description retrieved from the Reference Model Directory, to gather data about what options to present to the user.

In case of a strictly defined process workflow it presents one determined proposition, as there are no other logical choices of alternatives. As COMMIUS is supposed to support loosely coupled processes as well, the Advising Component will also provide the option to “jump” to process steps farther ahead in the defined process flow, thus supporting the user in his actions only with recommendation for further process steps without forcing him into predefined structures.
The Process Layer’s Internal Database

A vital part of the processing Components consists of two databases, which form the basis for each action executed by the interoperability layer.

The Enterprise Process Repository is the central storage for all business relevant data. The business data managed by this database can be roughly classified into two categories, namely business objects, like articles as well as business persons like customers or business partners.

The second, and most important, class of information stored in the Enterprise Process Repository contains the actual process instances, i.e. all running or already terminated processes including links to the business data (customers, orders,...) and the triggering emails, they are related to.

Furthermore, it contains all customized settings the users apply to the give reference models of the process steps to guarantee that the actual resulting document meets the user’s expectations.

The Reference Model Directory is responsible for storing and managing the reference models of known process types, i.e. which information is to be gathered if a specific process step is invoked and what the generated output will look like. As soon as an additional process step is being installed its corresponding reference model is being stored within the Reference Model Directory, since the process step itself has no database.

Along the execution of the four process layer Components, all the collected information will be written in a XML-Document. This document constitutes the actual output of the Process Interoperability Layer and will be passed to the message post processing, where it will be refined into the actual email, the user will finally see.

Roll-Back Functionality

As described in the previous sections, a user can change the system defined steps to the type he or she wants to by clicking on the appropriated link. By doing this, the user activates the system provided roll-back functionality. Imagine there is a typical "OrderProcess" running and the order has already been received. The next step would be to ship the ordered products to the client but the system defines the incoming e-mail as regarding to an "OrderStep" (for example an extended order). The user now has the possibility to change the step to a "ShippingStep" by clicking on the name of the step (See Figure 3 on the next page, linked marked with a red box).
Commius Extensions

So far we have examined in a detailed way the parts of Commius architecture that compose the system’s extensible core, the System, Semantic and Process Interoperability layers. There are two basic mechanisms by which this core can be extended and customized:

**Commius Business Modules**, or just Commius Modules for short, allow the definition of Business Case or Business Domain specific functionality. A basic interoperability module is a software component able to fully exploit all the Commius capabilities provided by the System, Semantic and Process layers in order to offer a coherent set of functionalities exploitable by SMEs in one or more business processes.

**Commius System Connectors** can be used to access and use information from existing SMEs infrastructure such as document repositories, relational (SQL) or XML databases, intranet applications or complex ERP systems. Legacy systems utilized by collaborating SMEs are the final endpoints and Commius enables their interoperability.

The following sections give some architectural details about these two extension mechanisms.

Commius Modules and Module Management

As explained, the software Components provided by System, Semantic and Process Layers are used by the Interoperability Modules to provide the functionalities which Commius directly exposes to the users: A Module provides a coherent set of process- or domain- oriented functionalities exploitable in one or more business processes. The Commius Configuration tools allow the user to select which module to activate in each step of his specific business process.

During the project time, according to SME interoperability general issues and to the project end-user requirements, and in order to demonstrate Commius capabilities, we have developed six modules, both to supply an initial set of functionalities available to the users who download Commius, and to serve as reference for the development of further modules. All the modules share a common set of features and functionalities, needful for their use and management by the framework, and at the same time each module provides a well-defined set of custom functionalities useful for its specific goal.

At the end of the project the developed modules, with their source code, have been made available on the Commius Sourceforge site (http://commius.sourceforge.net/). The set of modules implemented during the project time is listed in the table on the following page.
<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search for Partner</td>
<td>Exploiting a set of heterogeneous data source enables a SME to find potential business partners able to provide products, services, expertise related to an incoming mail.</td>
</tr>
<tr>
<td>Mail Template Manager</td>
<td>Provides to the user a set of mail templates and documents suitable to reply/forward an incoming mail. The templates and documents are selected according the business process associated to the mail.</td>
</tr>
<tr>
<td>Attachment Manager</td>
<td>Enables the user to store, version and share incoming mail attachments through the use of one or more external document repositories. Suggests to the user the location where to store attachments according the business process associated to the mail and its sender.</td>
</tr>
<tr>
<td>InfoBridge</td>
<td>Makes a bridge between data contained into incoming mails and external systems. With System Connectors enables the integration of Commius with other software used by SMEs.</td>
</tr>
<tr>
<td>Business Document Generator</td>
<td>Exploiting business information contained into an e mail like products, quantities, weight, customer details etc. enables the user to generate business documents.</td>
</tr>
<tr>
<td>Email Social Network Search</td>
<td>Exploiting a social network built on the user mail archive enables the user to navigate and search for persons, organizations, products, etc. and their relations.</td>
</tr>
</tbody>
</table>

To ensure a common management framework, and to maximize code reuse, all the Modules share a common set of features while each Modules provides its own functionalities. Modules expose the same interface to the rest of the Commius system so that other Components (in particular Configuration tools and the Process layer that activates Modules when needed) interact with them according to predefined standards. Modules do not need any external software (i.e. database management system, web server, etc.), each Module is independent from the other Modules (i.e. the user can freely choose the set of Modules to use), and they provide highly interactive Ajax based GUI.

All Commius modules have been designed in order to be very easy to install; in fact, they are deployed as pluggable independent components. The user will be able to decide the modules that he needs and install them exploiting a visual interface. Modules will start working with a small set of textual configuration parameters. Modules share several parameters (i.e. HTTP port exploited for web interfaces or the system folder where module can store their databases) so the user can define them just one time. Moreover the Commius prototype released at the end of the project provides some default values for these parameters.

Each module works in a transparent way for the user supporting the enrichment of incoming mails and providing useful functionalities. Our end users clearly specified that, in order to avoid an excess
of additional information into the mail (they must read everyday many mails and they want to have less to read and not more), modules have to provide information and functionalities only “on-demand”. Following this suggestion, all modules have been designed only to add limited information (in many cases they do not add anything) into the mail but to add a link to their web interface. The user will be able to fully exploit modules functionalities following the link provided by modules and accessing the web interface of the modules. These links embed some information enabling the web interface to be initialised according the considered mail.

The advantages of this approach (to embed a link into the mail enabling the user to exploit the module web interface) are summarized below:

- Today the Web is surfed everyday by many users that are able to exploit web browsers and web interfaces for several purposes (work, hobby, communicate, etc.).

- Modules provide simple and well-designed web interface enabling the user to exploit all the functionalities of the modules with a graphical user interface.

- When the user accesses the web interface of a module following a link embedded into a mail, the web interface is initialised according the specific mail so for the user it is simple to keep in mind the relation between the module and the mail without being disoriented.

- The web interface enables the user to have full control over the actions performed by the modules improving his/her comfort with the system. For example, the user will be able to confirm that an order must be inserted into the company system by a module avoiding automatic insertion (which could not be a good solution if the order was sent by an unknown customer).

- The web interface enables modules to provide the same interface both if Commius is installed on the single user machine or on a company server.

Modules developed during the project have been licensed as Open Source software, enabling the SMEs to install them freely and without costs. Another advantage of releasing modules as open source solutions is that the open source community is free to extend them and that SME needing specific custom functionalities and wanting to invest some money are able to extend and customize the modules developed during the project.

Modules are designed not only to enhance SME interoperability but also to simplify the work of the user and improve his/her productivity in his/her company providing benefits in terms of money and time spent to accomplish daily tasks.

The consortium’s end user partners, as other SMEs spread over Europe, have very heterogeneous features in terms of owned technical infrastructure. Modules are designed to cover the major parts of possible scenarios ranging by the SME with any specific technical infrastructure (i.e. a SME without a Local Area Network, without a dedicated Web Server, etc.) to SMEs with well-structured software infrastructures (i.e. having a dedicated Local Area Network, Web server, DBMS, company document repository, etc.).
How Modules Interact with Core Layers’ Components

Commius modules are built on components belonging to the System, Semantic and Process layers. In order to define modules and their functionalities, an analysis about how the functionalities of the components could be exploited inside modules is presented here.

Components belonging to the System Layer enable the modules to access in a standard and simple way all the incoming mails independently of the underlying mail technologies. Moreover, the system layers’ components enable modules to access all the information related to the incoming mails such as the sender, the receiver(s), its subject, etc. Looking at this information it is possible to associate some users with the mail (sender and receivers), to categorize the mail looking at their subjects and so on. Finally the system layers components provide to the modules the mail attachments in a well-structured and uniform way. In other words, modules can easily access the attachments of a mail that can represent technical documents, business documents etc.

Components belonging to the Semantic Layer enable modules to access the mail content exploiting a well-defined data model, namely the core component schema, and enabling modules to understand and exploit the information contained into a mail. To make some examples, modules will be able to understand if in the mail products, services, people, organizations are contained, or if the mail contains a particular document such as an order, a request for quotation, an invoice etc.

Components belonging to the Process Layer enable the modules to associate to the incoming mail a context considering the mail as part of a business process. This enables the modules to exploit information about the particular type of business process associated to the mail, the business process step to which the mail refers and finally to understand if two or more mails belong to the same business process.

All modules provide the same set of OSGi services (see section on modules’ enabling technology later). This service enables other components to find installed modules and apply the specific functionalities of a module on a given mail. The Process Layer components are able, considering the business process associated to the mail and the content of the mail, to select the modules to apply.

The user can configure the modules to invoke for a given business process exploiting the Process Configuration Tool. When a module is invoked on a certain mail the other Commius components dealing with information extraction and semantic and process analysis (i.e. components belonging to the System, Semantic and Process layers) have been already applied to the mail and their results have been stored into the MailMetadataStore.

The MailMetadataStore is an OSGi bundle offering some services enabling Commius components to store/retrieve information about a mail. When a module is invoked, it queries the MailMetadataStore to retrieve all the available information about the mail, produced by other components and summarized in the previous section.
Then, the module performs its work and returns its results as a set of XML documents. The component that invoked the module is in charge of collecting the results of all the modules that it invoked and forwarding these results to the Message Post Processing (MPP) component. Finally, the MPP will exploit the results of modules to enrich the mail.

The figure above summarizes the whole process supposing that the module is invoked by the Process Layer.

**Business Modules Enabling Technologies**

In the Commius project we have chosen to exploit the OSGi technology in order to develop a system where it is easy to add or update components and in particular basic interoperability modules. In fact a user must be able to decide the set of needed interoperability modules and install them in an easy way and without shutting off the system. Moreover the user must not run into problems of incompatibilities or dependencies.

The **OSGi technology** is a set of specifications enabling to define a dynamic component system for Java. By exploiting a framework compliant with these specifications, it is possible to develop
applications that are dynamically composed of many different reusable components. In Commius all components, including basic interoperability modules, are developed as OSGi software components called bundles.

The OSGi specifications define a set of rules and mechanisms enabling bundles to fully hide their internal implementations from other components. This feature prevents dependency problems and makes the development and testing of the system easy. At the same time bundles are able to communicate between them through services, which are objects that are specifically shared between bundles. Moreover, bundles can declare the Java packages that are able to export and that need to import.

A bundle can provide a set of services that other bundles can exploit and a bundle can listen for the availability of new services as depicted in Figure 2. Moreover bundles can be installed and uninstalled also at run time through specific mechanisms enabling the OSGi framework to manage the whole system correctly avoiding inconsistencies.

These OSGi features are really useful for Commius modules. In effect, a module can be installed at run-time. Each module provides a set of functionalities as OSGi services and search for functionalities (OSGi services) provided by components belonging to the System, Semantic and Process Layer. In other words, the OSGi framework supports the installation of a module and the interaction between this module and other components handling these issues in a clean way.

Another advantage of the OSGi specification is that it is possible to specify dependencies between bundles (i.e. bundle A is not started until bundle B is installed and started) and the class loading mechanism enables to embed into bundles also incompatible versions of the same libraries. This avoids incompatibility problems between modules and other Commius components.

Today there are several available frameworks compliant with the OSGi specification. Even if we have decided to exploit Apache Felix1, the basic interoperability modules are developed as OSGi compliant bundles that can be installed inside any OSGi compliant framework (in particular we have considered the last release, namely R4, of the OSGi specification).
An OSGi Http Service allows other bundles in the OSGi environment to dynamically register Web resources and servlets into the URI namespace of Http Service. A bundle may later unregister its resources or servlets. This service is needful to enable bundles to provide web interfaces. Also Interoperability Modules exploit this service to publish their user interface (if needed).

We have decided to develop user interface as Web interface to enable users to exploit modules in a common way both if Commius is installed on the user machine or if it is installed on a company web server.

To develop Basic Interoperability Modules the PAX-WEB2 implementation of the HTTPSERVICE has been exploited. PAX-WEB provides a standard OSGi Http Service and extends it providing advanced servlet support, filters, listeners, error pages and JSPs and some others features in order to meet the latest versions of Servlet specs and support some AJAX mechanisms.

In order to develop high interactive web user interfaces for interoperability modules we have chosen to use the AJAX technology. In particular we have used Google Web Toolkit (GWT)3, an open source framework that enables developers to easily develop AJAX web applications exploiting Java language without going into details of JavaScript development or browser compatibility. AJAX front-end is developed with the Java programming language, which GWT then cross-compiles into optimised JavaScript that automatically works across all major browsers.

This approach is compatible with the OSGi development model and the Http Service. In fact the web interface of a module can be developed independently by the module core, compiled into JavaScript and published by the module as any other Web resource exploiting the Http Service provided by PAX-WEB. Moreover web pages based on java script communicate with the module through servlets that are fully supported by the Http service.

In order to obtain a good look & feel for interoperability modules without spending too many resources in the customisations of GWT components we have exploited two GWT extensions namely GWT-Ext and GWT-FX. GWT-Ext4 is a powerful widget library (a widget is a generic GWT visual component) providing rich widgets with a professional look & feel. Gwt-Fx5 has been used to add some simple animations and effects to web interfaces.

Basic Interoperability Modules developed during the project exploit the H2 database engine to store their internal data and configuration. H2 is a fast open-source database engine written in Java supporting standard SQL and JDBC API. The H2 engine can be executed in “embedded” or in “server mode”. In “embedded mode”, an application opens a database from within the same JVM using JDBC. This way, the user is not requested to install an external database engine and to configure/admin it.

In the scenario of the project where Commius must be very easy to install and configure this feature is a great advantage. Each module can configure, execute and manage its internal database without requiring the user support. An H2 database is stored as a set of files that can be encrypted using AES (Advanced Encryption Standard) or XTEA (eXtended Tiny Encryption Algorithm) algorithms.

Another advantage of H2 is that database (structure and data) can be exchanged simply exchanging the db files (database files do not depend by the execution environment). This feature can be exploited to release basic interoperability modules that can be distributed together with their pre-initialised databases. Basic interoperability modules needing an internal database implement the following steps:
1. Look into the OSGi framework to find H2 required packages. The H2 database engine is distributed as a Java Archive (JAR) compatible with OSGi specification and if installed in the OSGi framework exports all H2 needed packages for other bundles.

2. Look in a pre-configured folder to check if its database files exist
   a. if the files exist then, exploiting H2 features and JDBC, the module opens the database represented by these files and use it
   b. if the files do not exist then, exploiting H2 features and JDBC, the module create an empty database, opens it, initialises it with a set of pre-defined data and finally uses it.

The whole process is transparent for the user.

**Commius System Connectors**

The second extension mechanism offered by the Commius architecture is represented by the so-called System Connectors, namely generic legacy system connectors which can be used to access and use information from existing SMEs infrastructure, document repositories and legacy systems.

Legacy systems utilized by collaborating SMEs are the final endpoints and Commius enables their interoperability. Document repositories, relational (SQL) or XML databases, intranet applications, complex ERP systems are just a few examples of legacy systems used by SMEs to store their data.

Commius System Connectors are components providing the rest of the Commius system deployed at SME an access to those legacy systems. Other Commius components can utilize System Connectors via unified interfaces, which provide necessary transparency hiding the complexities of interactions with legacy systems. Commius exploits system connectors and SME legacy systems on the one side to enrich incoming e-mails with relevant data coming from SME legacy systems to and on the other side to enable the user to update legacy systems with data received by mail.
The usage of System Connectors is depicted in the above Figure. System Connectors contribute in enriching an initial email with active links that allows the end-user to interact with the company’s IT systems. Interaction refers either to internal database interactions or RPC calls since Commius is able to invoke external procedures as well, with the purpose of performing specific business logic, which is the reason for the RPC connector.

Technologies used in large enterprise environments to interact with legacy systems and data integration, such enterprise information integration (EII) technologies or Enterprise Service Bus (ESB) technologies, are often too heavy for SMEs. While inspired by Web mashups approaches, data mashup frameworks are increasing their diffusion within research as well as within enterprises. Data mashups aim at gathering data from multiple, independent sources with a light-weight approach. Gathered data is then merged and presented to the user in a coherent form.

System Connectors had to be easily extensible and configurable to provide the needed flexibility. Moreover, two security issues have to be considered for systems connectors: Usage of security tokens required to access legacy systems and authentication and authorization to invoke system connectors.

The concrete goals that we wanted to achieve with the System Connectors were the following:

- **Common description of connector capabilities**: connector capabilities are described in a uniform way to define available functions/input/output data and provide a common way of utilization in the COMMIUS modules.

- **Extensibility**: due to the heterogeneity of legacy systems, it is very difficult to provide 'universal' connectors for legacy application classes; the effort was to provide connectors able to adapt to concrete systems by connector configuration in order to minimize the need to modify/rewrite the connectors source code.

- **Security**: connectors must be able to exploit legacy systems where the security tokens are mandatory to perform required operations. More specifically, System Connectors should be flexible and configurable enough so as to operate in line with a company’s pre-installed security scheme (e.g. SSO etc).

**System Connectors Architecture**

The System Connector component architecture comprises three subcomponents: System Connector Logic, Module Specific Configuration and Legacy System Client.

The **System Connector Logic** subcomponent is responsible for the pre-processing of input data, retrieval of data/information from legacy systems and post-processing of extracted data to the required form. This subcomponent is responsible for performing operations over a given legacy system defined in the connector configuration file. One implementation class of System Connector Logic can be used to interact with a wider class of legacy systems (e.g. implementation of a System Connector Logic subcomponent for relational databases can be used with variety of available database systems).
**Module Specific Configuration** defines parameters required by Legacy System Client as well as System Connector configuration. For instance RDMS System Connector configuration may contain RDMS client configuration (server, port, database) and a definition of a query to be issued from the System Connector. The input metadata and the procedure that is executed over a legacy system are defined in the SC configuration file. Thus, if multiple operations should be accessible for a single legacy system, there will be a separate SC configuration file for each of them. The configuration of the system connectors is still pending the inclusion of the privacy classification of each field. But this feature is subject to changes.

The **Legacy System Client** provides mechanisms for connecting and retrieving data from legacy systems. This is usually a client library provided by legacy system vendor/creator (e.g. for relational database system, the client subcomponent would be a JDBC driver).
During the project time, three basic types of connectors have been identified and implemented; the database System Connector, the RPC System Connector and the Web System Connector.

The **Hibernate System Connector** (HSC) is the connector that is used to interact with all kind of databases. The key enabling technology regarding this connector is Hibernate. Hibernate is a powerful, high performance object-relational persistence and query engine. Hibernate lets you develop persistent classes following object-oriented idiom - including association, inheritance, polymorphism, composition, and collections. Hibernate allows you to express queries in a portable manner using (internally) an SQL extension addressed as HQL. A HSC can be used for both providing input to a database and getting information out of the database.

The principle behind HSC is very simple, yet very powerful. As if you provide (declaratively) the meta-model of a desired source (is a more appropriate term than database) you can execute queries in an SQL-agnostic manner. You can do that by defining the type of the database that this schema refers to (hibernate internally undertakes the task of translating this query to the internal schema-dependent representation).

System Connectors are described in details in the project’s public deliverable D4.2.3 (*System Interoperability Components and Methods*), which can be found on the project public website.
Conclusions

Potential Impact

Commius is addressing the needs of more than 19 million enterprises, the EU SMEs, representing 99.8% of all registered businesses, and aims at having a relevant socio-economic impact in Europe. SMEs have some strong advantages in flexibility and responsiveness but they face challenges to address complex products or services where efficient and effective collaboration with others is required, moreover the present economical crisis of the globalised market is forcing SMEs to undertake radical transformations to gain competitiveness.

In this context new interoperability tools exploiting the full potential of Internet technologies, need to be conceived and brought to the market to support SMEs. In fact, enterprise interoperability has been addressed by research for decades without real success and little impact on SMEs and the market still lacks an affordable, easy to integrate solution: SMEs instead face costly investment, extensive integration effort and significant revision of their working tools and systems.

Commius main objective was to address this lack of solutions for SMEs supporting them with a zero, or very low-cost, entry into interoperability, based on non-proprietary protocols. allowing them to reuse existing and familiar applications for electronic communication.

Commius potential impact is therefore relevant and its position in the interoperability tools area is quite unique since there are no products in the market with similar features, and competitors within the traditional interoperability segment, such as ERPs, provide more expensive and complex products.

Commius will be easy to install and use, the initial investments limited, and the investments required for the adoption of a the Commius interoperability platform will be gradual and justified by the expected reduction of the production cost.

Of course within the rapidly growing SaaS sector the market characterisation is more complex and dynamic. In this sector there are already several competitors which propose differentiated services with “pay as you go” model and more will probably join in the near future; nevertheless, Commius, tanks to its features, can play a central role in the SaaS sector, since its innovative features and global vision fit quite well within the this model, and its functionality is presently not provided by any other competitor.

In essence, the adoption of Commius will both increase business opportunities for SMEs and reduce their production costs, thereby supporting the increased economic growth and creation of employment opportunities in Europe.

The expected impact will be based on the project tangible results and demonstrators, which are expected to support the full development of a set of Commius products. These will be based on Commius adaptable software architecture for interoperability and collaboration, easy to be adopted by the SMEs, suitable for integration within network enterprise environments. The SME user of Commius will not be forced to install new interoperability solutions, adopt new working tools nor change its organisation or business model. Commius interoperability solutions will evolve with the
organisation, adapting to the user needs and according to existing ICT infrastructures. Moreover, Commius will enable SMEs to react to the market requests as a net of collaborating companies, each providing specialised products, technologies and services.

Another important aspects which will contribute to the overall impact of the project achievements, is the Open Source approach which Commius has adopted. By creating an Open Source infrastructure for interoperability, based on email communication, other enterprises or individuals will be able, without the need of paying license fees, to use, extend or provide services based on it, either commercially or as Open Source. In fact, Open Source is a key issue for the COMMIUS interoperability solution and will support new business opportunities for SMEs in Enterprise Interoperability markets: From the point of view of the SMEs end users, the choice to adopt Open Source allows to reduce the software licensing costs, contributing to the project goal of almost-zero cost adoption. From the point of view of software development, COMMIUS release as Open Source will allow consortium partners as well as third parties in all Europe to exploit and extend the platform, or to supply services based on it, in order to produce business-specific applications. All this will contribute to the overall strengthening of Europe position in interoperability industry and science. Licensing for the COMMIUS framework has been defined in a way to allow future interoperability Modules developers and providers to decide on licensing and related policies for their new interoperability Modules which may be pluggable to the open source Commius interoperability infrastructure, or obtained modifying and extending the Commius Modules.

Commius is a user-driven project, the end-users have actively participated in all the project activities (the picture below shows an end-user session) and the project results have been demonstrated and optimised within three use-cases, performing a set of piloting events with real end-users. The project outcomes are addressing the real, concrete needs of SMEs, and the partners believe that one of the major achievements of the project has been to apply advanced research results to actual industrial requirements. Though the project outcomes are, of course, results of a challenging research activity, and still need to be completed and industrialised, in order to become products and services ready for the market, the end-user centred approach allowed to produce a usable technology which may have an extremely relevant impact in the largest business sector in EU.
Dissemination activities

The dissemination have been seriously addressed throughout all the project life, and the concerned actual dissemination activities, which played a crucial role within the project. The strategy defined takes into account the overall dissemination process and has been based on four main steps:

- Definition of the key messages,
- specification of the target user groups,
- definition of the main contents to disseminate to the identified potential target user groups,
- identification of media and actions to be adopted for the dissemination.

The dissemination material produced includes:

- The Commius project logo (see below), which, was chosen (by partners’ vote) among a set of logos proposed by all the partners.

- The consortium has published a web site (The picture below shows the home page) where all the relevant public aspects of the project can be accessed, including the events related with the project and the public deliverables.
Several PowerPoint Project presentations.

A Fact Sheet document.

Three versions of short paper documents have been developed, each of them aiming at one of the target groups identified in the dissemination strategy:

- Scientific stakeholders: Research communities who will be interested in the project results to support their research.
- ICT companies and ICT communities: European industries (mainly technology and service provider in the ICT area).
- The Final user: SMEs and associations of SMEs.

Leaflets/Brochures.

The consortium has prepared three different versions of Commius leaflet, to provide people with an attractive project overview with a summary of the main project objectives and characteristics, each of them aimed at one of the specific target groups.

A Newsletter, distributed to the Commius community, with the description of the project main objectives and achievements.

Moreover several paper and articles have been published, both in specialised scientific and technical publications, and in non-technical newspaper and magazines.

Commius dissemination actions, based on the dissemination strategy conceived, included a variety of initiatives, focused both on the scientific and the industrial sectors, such as:

- Presentation of the project to end-users and potential business partners.
- Workshops organisation.
- Academic Courses including a presentation of the project conceptual achievements within courses held by Commius partners.
- Contributions of the Commius partners to research activities and participation at international meetings focused on Commius research themes.
- Participation to international events and fairs in order to disseminate the Commius results to a broader audience.
- Open Source software Components release including software guides.
- Set up and maintenance of the Commius community.

The Community is meant to be a major dissemination channel for the project, and, even after the end of the project, through the community, the Commius consortium will encourage, receive and incorporate opinions and feedbacks coming both from scientific and technical partners and end-users.
Commercial exploitation

The following results which the partners are planning use for the commercial exploitation have been identified:

1. Commius OS software framework, available at Sourceforge, which can be used as the basic Component for the development of products focused on interoperability and collaboration, addressing the specific needs of SMEs.
2. Commius OS software framework, as an opportunity to deploy services.
3. Specific OS software Components, available at Sourceforge, addressing functionality developed within the piloting phases, to support the three project pilots in Spain, Greece and Italy.
4. Connectors, enabling integration of the Commius software platform with legacy systems (such as ERP or company databases).
5. Commius Architecture, based on innovative concepts and technologies, which can be applied to other production or research areas.

The policy adopted to exploit the mentioned results includes several steps, partially performed within the project:

- A Market analysis which contains both a delineation and overview of the target market segments for the planned products.
- Definition of the actors, roles and scenarios, including a customer analysis and a competitor analysis.
- Definition of the main functionality to be provided, and of the Value Proposition and the Value Chain for the Commius products/services within the selected market segments.
- Definition of the core services which can be built over the Commius product, and of the potential enhancements that could be done either by the Commius partners, by the Open Source community, or by other industrial organizations.
- The overall commercialisation strategy and licensing policy of the products, which will include the following aspects:
  - The potential partnerships within the existing consortium, which may rise for the promotion of the exploitation assets (e.g. SingularLogic together with Greek users, Softeco with AITEK, Atos with FEDIT).
  - The specific point of strength of the new products and the position of the main competitors in the market segments individuated.
  - The targeted market segments and their characteristics, the countries involved, the main groups of potential customers/users.

Moreover, the partners have defined specific exploitation plans with an indication of their intended pricing policies and a first estimation of the costs and revenues for the defined products and services.

Commius leverage a few key concepts which proved to have large commercial potential and the partners have elaborated a simple and effective value proposition based on few concrete messages. According to the Value Proposition, which is the starting point for the project commercial exploitation, SMEs deciding to integrate Commius in their organizations will:

- Need a very small initial investment, in terms of effort and budget;
- be able to gradually configure and deploy the Commius platform;
- not change the way they are conducting their business;
- be able to collaborate with other SMEs and exploit new business opportunities.
Knowledge and know-how exploitation

Commius’ approach towards knowledge and know-how exploitation concerns mainly the research partners but it is relevant also for the industrial partners. In order to avoid the results of the projects research remaining in an ivory-tower, knowledge transfer is of major relevance. Knowledge transfer is described as the underlying an actor uses to transfer experience to another unit (e.g., group, department, or division).

Exploitation of knowledge and know-how mainly included:

- Transfer of the know-how to other application areas.
- Further Research & Development.
- Participation in R&D & Standardization activities.
- Diffusion of the results in the research and standardization community.
- Preparation and provision Education & Tutorials.
- Enhancement of the developers’ know-how and decrease of the production time.