

# “ELLECTRA-WeB softverski okvir za razvoj open-source aplikacija za podršku procesa sprovođenja elektronskih javnih nabavki”

## “ELLECTRA-WeB Open-source electronic Public Procurement Application Framework”

C.Georgousopoulos<sup>1</sup>, A.Ramfos<sup>1</sup>, B.Elvesæter<sup>2</sup>, G.K.Olsen<sup>2</sup>, P. Stoiljkovic<sup>3</sup>, B. Stoiljkovic<sup>3</sup>

<sup>1</sup>Intrasoft, Greece

<sup>2</sup>Sintef, Norway

<sup>3</sup>CIM College, Serbia

**Sadržaj-** U ovom radu predstavljamo ePP softverski okvir za razvoj aplikacija za elektronsko upravljanje javnim nabavkama. ePP softverski sistem razvijen je u okviru ELLECTRA-Web projekta Evropske komisije i prati MDA pristup i prednosti koje pristup nudi u odnosu na tradicionalne metodologije softverskog inženjerstva.

**Abstract -** In this paper we present the ePP Application Framework for electronic Public Procurement solutions. The ePP Application Framework has been developed within the ELLECTRA-WeB project and it follows the MDE approach due to its advantages in contrast with traditional Software Engineering methodologies.

### 1. INTRODUCTION

The capital investment associated with software development has been identified as one of the main barriers for the implementation of electronic Public Procurement systems in the EU. The required capital investment may even be prohibiting for Public Organisations with less available budget for internal development, such as local authorities, hospitals, education establishments, public museums, public agents in tourism or commerce, NGOs and other civic society associations, unless means for reducing it are found. If the capital investment associated with development time and costs is reduced, the adoption of electronic Public Procurement in the EU will be accelerated.

ELLECTRA-WeB confronts the challenge resulting from the associated capital investment for the software development of electronic Public Procurement systems. ELLECTRA-WeB empowers all interested parties, i.e., EU Public Organisations and ICT solution providers, with an integrated software production environment that enables the automatic generation of required code. The integrated software production environment is offered to the interested parties in the EU as an integrated Open-source Application Framework.

The ELLECTRA-WeB Open-source Application Framework enables the generation of the code required for the production of 3-tier, Service-Oriented Architecture-based electronic Public Procurement systems. The production of electronic Public Procurement systems is guided by the principles of Model-driven Engineering, in which the required code for an electronic Public Procurement system is automatically generated from comprehensive models representing the Public Procurement information and processes, instead of hand-writing code. The models incorporated in the offered Application Framework were developed according to the

IDABC guidelines for the electronic Public Procurement in the EU and, hence, the produced electronic Public Procurement systems are in absolute compliance with the EU legal framework on public procurement.

EU Public Organisations and ICT solution providers can use the ELLECTRA-WeB Open-source Application Framework to automatically generate code based on the incorporated models, or they can easily produce variant electronic Public Procurement solutions by appropriately customising the incorporated models and let the Application Framework to automatically generate the corresponding code.

The rest of the paper is organised as follow. The specifications of the ePP Application Framework are presented in Chapter 2. Chapter 3 covers the development methodology of producing an ePP solution with the utilisation of the framework, and Chapter 4 documents the architecture of the generated ePP solutions. The paper concludes with Chapter 5.

### 2. SPECIFICATION OF EPP APPLICATION FRAMEWORK

The ePP Application Framework is built on top of the Eclipse Platform [1] and several different plug-ins are integrated to form the framework. The architecture of the ePP AF is depicted in Figure 1. The architectural stack is composed of a collection of components. Every component is identified by a solid box; a dashed box represents an optional component that might be introduced into the architecture. A single component may consist of a number of different sub-components and plug-ins. The five major components identified within the architecture are listed below and are described in more details in the following sub-sections.

- **Eclipse platform:** is the platform that integrates all the required plug-ins for the realization of the ePP Application Framework.
- **Development Environment:** corresponds to the required components for the development of models and code.
- **Generation engine:** is responsible for the transformation of models into code.
- **Optional components:** represent a collection of components that can be optionally integrated into the ePP Application Framework.
- **ePP specific components:** have been developed for the needs of the ePP AF. Examples of such components are fixed web-services to support different security implementation schemes, conversion of ePP forms (represented in XML

standard) into PDF/HTML documents, deployment descriptors etc.

- **ePP Application Framework Integration Plug-in:** is responsible for the integration of all plug-ins under the Eclipse platform.

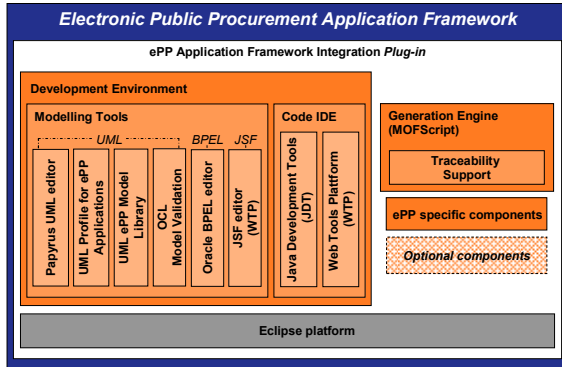


Figure 1: ePP Application Framework Architecture

Apart from the development environment and the rest of the framework’s architectural components that provide Application Developers with all the appropriate functionalities for the development of ePP solutions, the ePP Application Framework also requires a server environment, where the ePP solutions are deployed, as well as a testing environment, where the ePP solutions are tested, before put into operation. This involves the following server applications: Sun Java System Application Server 9.1 and Open ESB 2.0.

All the components and plug-ins of the ePP AF architecture are Open-source and can be downloaded (or updated) directly from the Vendor’s official web-site. The ePP AF is provided as a pre-configured platform of Eclipse with all the necessary plug-ins installed and it is available for public access via SourceForge.net in [3].

### 3. DEVELOPING AN EPP SOLUTION WITH THE USE OF THE EPP APPLICATION FRAMEWORK

The ePP Application Framework follows a hybrid Model Driven Development approach on producing ePP solutions; hybrid in the sense that code is being generated directly from a PIM model, excluding the intermediate step of PIM-to-PSM transformation.

A PIM corresponds to the blueprint of a system and may be composed of a collection of different models expressed in a formal language. Based on the MDD methodology addressed by the ePP AF, the necessary models to constitute the complete specifications of an ePP solution must be expressed in UML and BPEL modelling languages. Figure 2 illustrates the specifications of the ePP PIM, and as it can be observed it involves a collection of different models and diagrams<sup>1</sup>. Information with regard to the models/diagrams depicted in figure is

<sup>1</sup> A diagram may be comprehended as the building block of a model. A model may be consisted of one or more diagrams. Usually, model’s information is distributed into different diagrams for clarity purposes.

provided in the following sections – all of the models are publicly available in [3]

The development of an ePP solution via the framework involves a two-phase generation as it is depicted in Figure 3. The first phase involves the generation of the persistence tier and part of the business logic of the end system, whereas the second phase regards the generation of the business tier and its integration with the web tier.

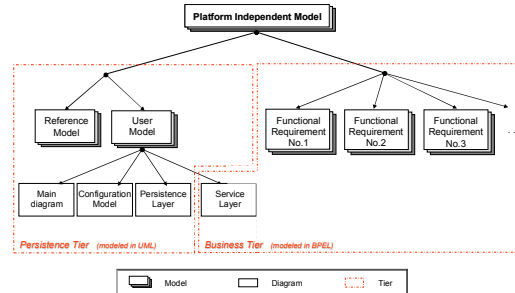


Figure 2: ePP Platform Independent Model specifications

UML is primarily used to model the persistence tier and part of the business logic of the end system - as it is highlighted in Figure 2 by the dashed boxes. BPEL is used for the definition of ePP processes that constitute the business tier. No modelling is required for the web tier as it comes off-the-self, implemented based on IDABC standards [4]. Though its parameterisation is performed via UML and transformation rules are employed for the compilation of the necessary web-pages expressed in JSF.

Each of the generation phases are described in detail in the following sections.

An Application Developer may build an ePP solution from scratch by defining his own models or by using the pre-existing template models offered by the ePP AF. In the second case, the developer’s effort lies on the parameterisation of the existing models or the definition of new models based on the re-use, extension or optimisation of the pre-made ones.

#### 3.1 First-phase generation

In order to simplify the modelling of the persistence tier as much as possible, we have divided the model that captures the persistence information into two separate models, namely the Reference model and the User model.

##### 3.1.1 Reference Model and User Model

In this instance, an Application Developer may model and make changes to the User model, but not in the Reference model. The Reference model contains specific information pertinent to ePP domain of which an Application Developer can use in order to describe the required functionality in his/her User model. More specifically, the reference model captures information from the UML ePP profile. The primarily objective of the Reference model is to abstract the level of complexity, and as the name suggests its role is to be used as a reference model – no amendments are possible to be performed on this model.

If amendments are required to be carried out in the Reference model this is the responsibility of the Framework Developer.

The User Model is composed of the following four diagrams:

- **Main diagram:** models the main packages of the ePP solution for placing the generated code.
- **Persistence Layer:** models the persistence schema.
- **Service Layer:** models the operations that manage the persistent objects defined in the Persistence Layer.
- **Configuration Model:** models the parameterisation of the ePP solution

The usage of the Reference model may be better comprehended with reference to the example models illustrated in Figure 4. The figure demonstrates a fraction of the Reference model (on the left hand-side) and the User model (on the right). For instance, if an Application Developer designs a Class (in the Persistence Layer diagram of User Model) and stereotypes it with the <<Entity>> stereotype of type 'CallForTender', then the code that will be generated after the transformation rules are executed will include all the information of the 'CallForTender' Class from the Reference model – as they are depicted in figure.

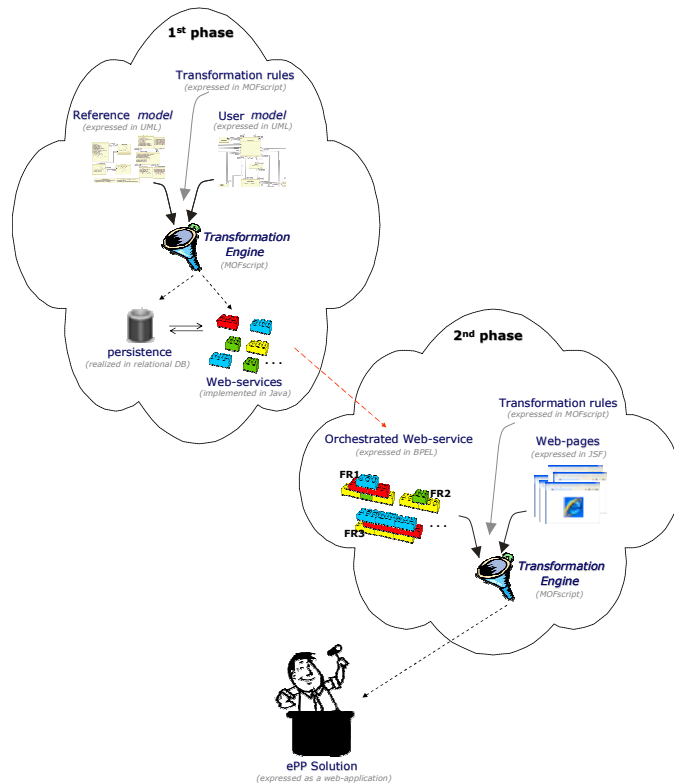


Figure 3: Procedure of generating ePP solutions

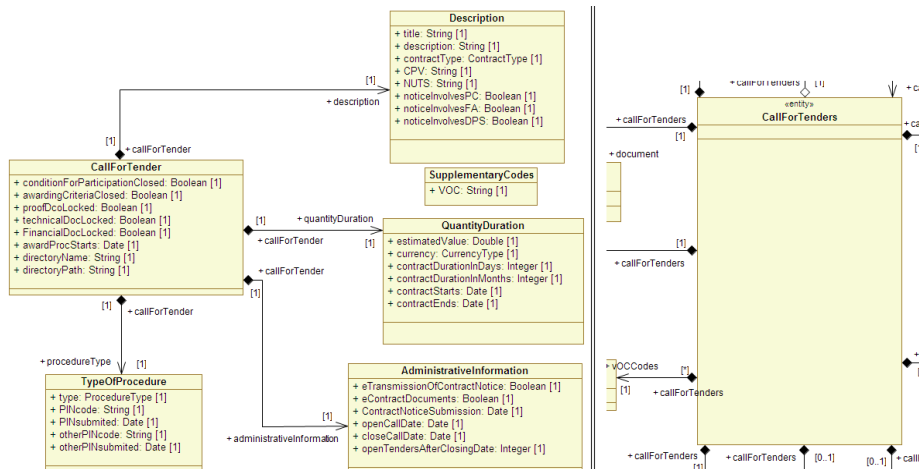


Figure 4: Fraction of the Reference model (on the left) and User model (on the right)

On one hand, from the Application Developer's perspective, complexity is decreased since transformation rules use information from the Reference model on properties defined within the User Model. On the other hand, a more abstract and simpler interface to the framework is provided. In addition we models are used to represent something that would normally have been specified in source code.

### 3.1.2 Generation of the Persistence tier and part of business logic

Once the design of the User Model is complete it is fed as input to the MOFscript Generation Engine along with the Reference model. Appropriate server-side<sup>2</sup> transformation rules are required by the Generation Engine for the conversion of the information captured in the aforementioned models into Java code. The resulting code regards the creation of persistence objects and CRUD operations<sup>3</sup> for managing those objects in the form of web-services. The procedure of the first-phase generation is illustrated in Figure 5.

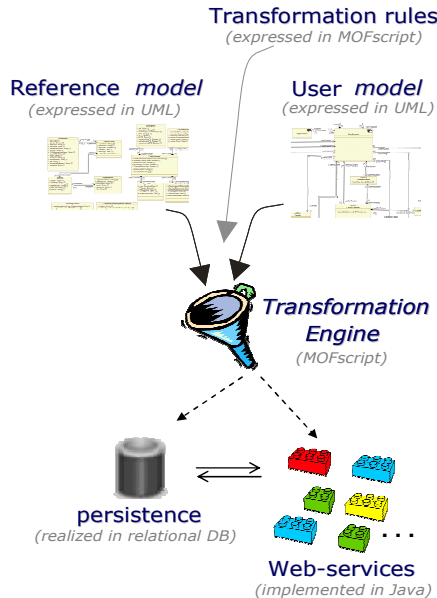


Figure 5: Visualisation of first-phase of generation process

After the first-phase of generation is completed, an Application Developer may manually add code into the generated Java Classes. Therefore, the developer's effort on manually writing code involves the code which is not possible to be generated from models. This is anticipated to be the 5-15% of the total code that constitutes a complete ePP solution – of course this is depended upon the complexity of the end application.

<sup>2</sup> A collection of transformation rules have been defined for the needs of the ePP AF. The server-side transformation rules are utilised for the generation of the persistence tier and part of the business logic (modelled in UML). All transformation rules are available in **Error! Reference source not found.**

<sup>3</sup> CRUD are the basic operations for managing a persistent entity. This involves the Create, Read, Update and Delete operations.

### 3.2 Second-phase generation

Once the persistence layer of the ePP solution has been created along with the necessary operations for managing the persistent objects, the next step is to define the business tier of the architecture and integrate it with the web tier.

#### 3.2.1 Generating the business tier

The business tier regards the support of all the ePP processes required by an ePP solution. IDABC [1, 2] has documented the Functional Requirements (FR) for building an ePP solution compliant with the legislative framework of EC. An FR describes simple ePP process such as the user authentication/authorisation, or complex ones like the submission of tenders which is basically involves the composition of other ePP processes. The orchestration of FRs realise a complete phase of ePP such as the notification, tendering, contract conclusion etc.

Within the ePP Application Framework those FRs are formally expressed in BPEL; in general each FR is represented by a single BPEL model. From the technical point of view, every BPEL model orchestrates a collection of web-services in order to provide the necessary functionality i.e. an ePP process. Those web-services involve the (i) ones generated from the first-phase of generation (discussed in section 0) that manage concepts from the ePP domain, (ii) other BPELs exposed as web-services, and/or (iii) manually implemented web-services not possible to be generated via the framework that handles complex procedures. In Figure 3 it is represented how the web-services produced from the first-phase generation - which are visualised as Lego bricks - form the building blocks of a single BPEL model required in the second-phase of generation.

An Application Developer may define his/her own BPEL models to describe the required ePP processes, or re-use/extend and optimise the pre-made template models provided by the ePP AF. The framework prerequisites predefined interfaces of BPELs (that represent FRs) for supporting recursion and loose-coupling. Although, the BPEL models provided by the ePP AF are in accordance to the IDABC guidelines, their content i.e. logic is possible to be modified by an Application Developer. This is a twofold advantage. Firstly, it enables the enhancement of an ePP solution if the pre-made models are used as the basis for the generation, and secondly it enables the option of supporting future amendments on the current EC legislative framework.

The BPEL designer **Error! Reference source not found.** bundled with the ePP Application Framework provides the necessary functionality for converting the BPEL models into appropriate code<sup>4</sup> which may be deployed on an Application server.

#### 3.2.2 Integrating the business tier with the web tier and generating the end ePP solution

<sup>4</sup> The orchestration of web-services within a BPEL model is formally expressed in XML language.

The final step of generating an ePP solution involves the integration of the business tier with the web tier. In simpler words, associating ePP web-pages with appropriate ePP processes, which they are either being invoked by clicking on a web link/button, or used for populating the content of a web-page with information retrieved/aggregated by the execution of a specific process.

The web tier of the ePP solution is provided by the ePP AF off-the-self, implemented based on IDABC standards and therefore is compliant to the EC legislative framework. The front-end interface shares the same look and feel as with the IDABC demonstrator [1] but has been implemented with the latest technology JSF. Although, no modelling is required for the web tier, its parameterisation is performed via UML and transformation rules are employed for the compilation of the distinct user interface components (implemented in JSF) into the necessary web-pages that constitutes an ePP solution.

Due to the fact that the interfaces of BPELs are predefined, the necessary code to invoke a BPEL from a web-page is integrated within the latter. Having specified the logic of the required BPEL models (business tier) for the corresponding web-pages (web tier) that constitutes the ePP solution, appropriate client-side<sup>5</sup> transformation rules undertake the integration of the business with the web tier. The MOFscript Generation engine is responsible for the execution of those transformation rules. The procedure of the second-phase generation is illustrated in Figure 6.

After the completion of the second-phase of generation, the Application Developer's effort on manually writing code which has not been automatically generated from the ePP AF is anticipated to be between 5-10% of the total code that constitutes a complete ePP solution. This involves enhancement modifications on the generated front-end interface and integration of the web-pages with extra ePP processes defined by the developer. Though, the percentage of the necessary code that needs to be manually written is minimised increasingly due to the fact that the front-end interface may be modified via the visual JSF editor bundled with the ePP Application Framework.

To conclude, the Application Developer's effort on manually writing code which is not possible to be generated automatically from the framework is less than 25% of the total code required for the development of a complete ePP solution pertinent to EC directives. The deployment of the end ePP solution to an Application Server is handled by specialised deployment descriptors offered by the framework.

#### 4. ARCHITECTURE OF GENERATED EPP SOLUTIONS

The guiding design rational behind the platform's architecture is to facilitate interoperability and

extensibility, as well as preserving in this sense a considerable degree of independence regarding the implementation of the end platform. Key factors to achieving this objective are the adoption of widely accepted as well as emerging standards and the usage of Open-source Software (OSS). Java and Web standards and technologies such as Java Enterprise Edition (JEE), XML, WSDL and BPEL are utilized to bring forth the desired objective.

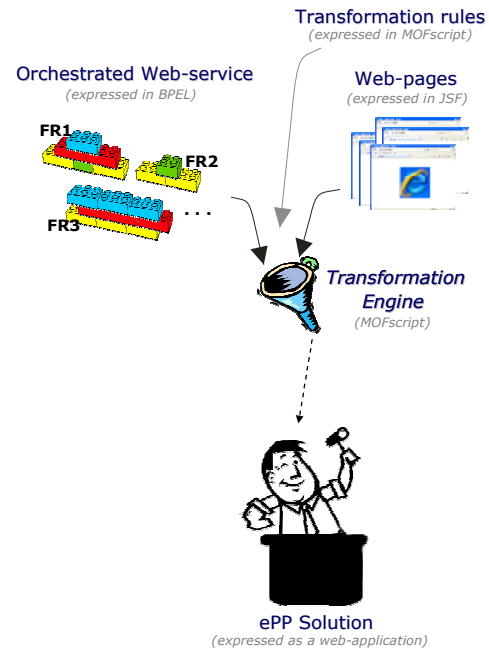


Figure 6: Visualisation of second-phase of generation process

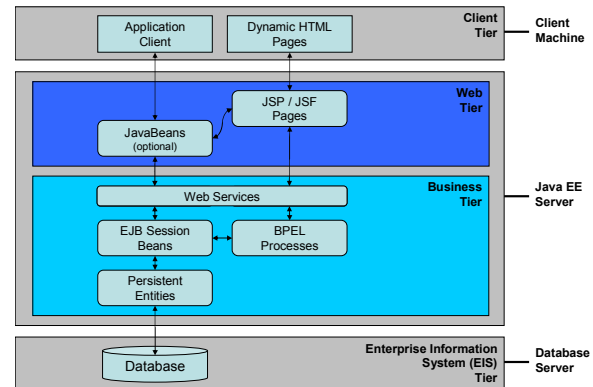


Figure 7: Overview of the ePP solution architecture

The ePP AF utilizes a distributed multi-tiered (4-tier<sup>6</sup>) application model for ePP solutions. An ePP solution will be implemented as a Java EE application on a Java EE platform. A Java EE application is made up of Java EE components which are self-contained functional software

<sup>5</sup> A collection of transformation rules have been defined for the needs of the ePP AF. The client-side transformation rules are utilised for the generation of the web tier and its integration with the business tier. All transformation rules are available in **Error! Reference source not found.**

<sup>6</sup> The term *tier* is used both to describe division of application logic and describe distribution on different locations (machines). Though a Java EE application typically consists of the 4 (application logic) tiers shown in Figure 7, Java EE applications are generally considered to be 3-tiered applications because they are distributed over three locations; client machine, Java EE server and database server.

units. Application logic is divided into components according to function, and the various components are installed on different machines depending on the tier to which the application component belongs. Figure 7 illustrates the tiers and their respective components.

- The client tier typically contains Web clients. A Web client consists of two parts: (1) dynamic Web pages (e.g. HTML and XML), which are generated by Web components running in the Web tier, and (2) a Web browser that accesses server-side components in the Web tier using HTTP or SHTTP and renders the pages received from the server. It is also possible to access server-side components from an application client running on the client machine. Application clients can accommodate a richer graphical user interface than what can be provided by Web browsers.
- The Web tier components are either servlets or pages created using JavaServer Pages (JSP) and/or JavaServer Faces (JSF) technologies. Servlets are Java programming language classes that dynamically process requests and construct responses. JSP pages are text-based documents that execute as servlets but allow a more natural approach to creating static content. JavaServer Faces technology is built on servlets and JSP technology and provides a user interface component framework for Web applications.
- The business tier contains Enterprise JavaBeans (EJB) components that implement the ePP business logic. EJB components are exposed as Web Services that use open XML-based standards and transport protocols to exchange data with calling clients. Session beans represent a transient conversation with a client. When the client finishes executing, the session bean and its data are gone. Persistent entities represent persistent data stored in one row of a database table. If the client terminates, or if the server shuts down, the persistence manager ensures that the entity data is saved. The Web Services Business Process Execution Language (BPEL) processes are used for the composition and orchestration of Web Services. BPEL specifies business processes and business interaction protocols using Web Services interfaces. An executable BPEL process is itself exposed as Web Service on the platform.
- The enterprise information system tier handles EIS software and includes enterprise infrastructure systems such as enterprise resource planning (ERP), mainframe transaction processing, database systems, and other legacy information systems. Resource adapters (e.g. JDBC or ODBC drivers) provide access, search and update services to databases and their data is stored in database management systems (DBMS).

## 5. CONCLUSION

The development of ePP solutions in Europe today is driven by low-level bespoke design and coding. Whether an incremental and iterative process or the traditional waterfall process is followed, documents and diagrams that express the specifications of the solutions are produced only during requirements collection, analysis and design. These documents and diagrams, however, rapidly lose their value as soon as the coding starts, and, typically, the connection between the diagrams and the

code fades away as the coding phase progresses. Also, documentation has always been a weak link in the Software Development process, and this is also true for electronic Public Procurement solutions. The above issues have long been recognised by the software industry as the main factors that contribute to the cost in deploying ePP solutions.

An alternative to code-centric development is the model-centric approach followed by the Model-Driven Engineering (MDE) paradigm. MDE refers to the systematic use of models as primary engineering artefacts throughout the engineering lifecycle, and therefore preserves the investment on requirements descriptions, design and analysis (which are expressed as models). The ePP Application Framework developed within the ELLECTRA-Web project follows the MDE approach due to its advantages in contrast with traditional Software Engineering methodologies.

As any Internet-based solution nowadays, ePP solutions need to be implemented as a distributed Internet architecture in which component-based programming and increased sophistication of middleware lessens some of the overall complexity of previous multi-tier client-server environments. The latest advancement in implementing distributed Internet architectures is the Service-Oriented Architecture (SOA) approach. By following the SOA paradigm, the ePP functions become modular and can be exposed as Web-services that are, typically, specified using standard languages (XML, WSDL) and interoperate through standard protocols (SOAP).

The innovative combination of MDE and SOA approaches selected for the cost-effective development of ePP solutions facilitates integration, interoperability, easy maintenance and management of possible change in the European ePP environment.

## REFERENCES

- [1] IDABC, "Software demonstrators for eProcurement". <http://ec.europa.eu/idabc/en/document/3488/5874> (last visited 2008).
- [2] eclipse.org, "Eclipse platform". <http://www.eclipse.org> (last visited 2008).
- [3] "Official web-site of ELLECTRA-Web in SourceForge.net", [Sourceforge.net. http://sourceforge.net/projects/ELLECTRA-Web](http://sourceforge.net/projects/ELLECTRA-Web) (last visited 2008)
- [4] IDABC, "IDABC guidelines, standards and Functional Requirements for eProcurement". <http://europa.eu.int/idabc/en/document/4721/5874> (last visited 2008).
- [5] Eclipse.org, "BPEL Designer Editor", <http://www.eclipse.org/proposals/bpel-designer/>, (last visited 2008).