D5.5 REQUIREMENT SPECIFICATION AND FUNCTIONAL DESIGN SPECIFICATION OF THE COPRO MODEL MANAGEMENT PLATFORM

Udo Enste
LeiKon GmbH - Germany

November 2018
www.spire2030.eu/copro
DELIVERABLE D5.5

Project Details

<table>
<thead>
<tr>
<th>PROJECT TITLE</th>
<th>Improved energy and resource efficiency by better coordination of production in the process industries</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT ACRONYM</td>
<td>CoPRO</td>
</tr>
<tr>
<td>GRANT AGREEMENT NO</td>
<td>723575</td>
</tr>
<tr>
<td>INSTRUMENT</td>
<td>RESEARCH AND INNOVATION ACTION</td>
</tr>
<tr>
<td>CALL</td>
<td>H2020‐SPIRE‐02‐2016</td>
</tr>
<tr>
<td>STARTING DATE OF PROJECT</td>
<td>NOVEMBER, 1ST 2016</td>
</tr>
<tr>
<td>PROJECT DURATION</td>
<td>42 MONTHS</td>
</tr>
<tr>
<td>PROJECT COORDINATOR (ORGANIZATION)</td>
<td>PROF. SEBASTIAN ENGELL (TUDO)</td>
</tr>
</tbody>
</table>

THE CoPRO PROJECT

The goal of CoPro is to develop and to demonstrate methods and tools for process monitoring and optimal dynamic planning, scheduling and control of plants, industrial sites and clusters under dynamic market conditions. CoPro pays special attention to the role of operators and managers in plant-wide control solutions and to the deployment of advanced solutions in industrial sites with a heterogeneous IT environment. As the effort required for the development and maintenance of accurate plant models is the bottleneck for the development and long-term operation of advanced control and scheduling solutions, CoPro will develop methods for efficient modelling and for model quality monitoring and model adaption.

The CoPro Consortium

<table>
<thead>
<tr>
<th>Participant No</th>
<th>Participant organisation name</th>
<th>Country</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Coordinator)</td>
<td>Technische Universität Dortmund (TUDO)</td>
<td>DE</td>
<td>HES</td>
</tr>
<tr>
<td>2</td>
<td>INEOS Köln GmbH (INEOS)</td>
<td>DE</td>
<td>IND</td>
</tr>
<tr>
<td>3</td>
<td>Covestro Deutschland AG (COV)</td>
<td>DE</td>
<td>IND</td>
</tr>
<tr>
<td>4</td>
<td>Procter &amp; Gamble Services Company NV (P&amp;G)</td>
<td>BE</td>
<td>IND</td>
</tr>
<tr>
<td>5</td>
<td>Lenzing Aktiengesellschaft (LENZING)</td>
<td>AU</td>
<td>IND</td>
</tr>
<tr>
<td>6</td>
<td>Frinsa del Noroeste S.A. (Frinsa)</td>
<td>ES</td>
<td>IND</td>
</tr>
<tr>
<td>7</td>
<td>Universidad de Valladolid (UVA)</td>
<td>ES</td>
<td>HES</td>
</tr>
<tr>
<td>8</td>
<td>École Polytechnique Fédérale de Lausanne (EPFL)</td>
<td>CH</td>
<td>HES</td>
</tr>
<tr>
<td>9</td>
<td>Ethniko Kentro Erevnas Kai Technologikis Anaptyxis (CERTH)</td>
<td>GR</td>
<td>RES</td>
</tr>
<tr>
<td>10</td>
<td>IIM-CSIC (CSIC)</td>
<td>ES</td>
<td>RES</td>
</tr>
<tr>
<td>11</td>
<td>LeiKon GmbH (LEIKON)</td>
<td>DE</td>
<td>SME</td>
</tr>
<tr>
<td>12</td>
<td>Process Systems Enterprise LTD (PSE)</td>
<td>UK</td>
<td>SME</td>
</tr>
<tr>
<td>13</td>
<td>Divis Intelligent Solutions GmbH (divis)</td>
<td>DE</td>
<td>SME</td>
</tr>
<tr>
<td>14</td>
<td>Argent &amp; Waugh Ltd. (Sabisu)</td>
<td>UK</td>
<td>SME</td>
</tr>
<tr>
<td>15</td>
<td>ASM Soft S.L (ASM)</td>
<td>ES</td>
<td>SME</td>
</tr>
<tr>
<td>16</td>
<td>ORSOFt GmbH (ORS)</td>
<td>DE</td>
<td>SME</td>
</tr>
<tr>
<td>17</td>
<td>Inno TSD (inno)</td>
<td>FR</td>
<td>SME</td>
</tr>
</tbody>
</table>
Abstract
In order to increase the efficiency of the model development and the deployment process of model-based applications, a model repository and a model catalogue will be developed as part of the CoPro framework. These will help to manage and reuse already developed models and methods, providing formal information about the model descriptions, the algorithms and constraints related to models and model parameters. The model descriptions include amongst others model quality aspects specified in WP1. This report describes the requirement specification and based on that the functional design specification of the CoPro Model Management Platform.

Revision History
The following table describes the main changes done in the document since it was created.

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
<th>Author (Organisation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>Sept 2018</td>
<td>Creation</td>
<td>Enste, LeIKon</td>
</tr>
<tr>
<td>V1.01</td>
<td>Oct 2018</td>
<td>Review</td>
<td>Nauta, PSE</td>
</tr>
<tr>
<td>V1.1</td>
<td>Oct 2018</td>
<td>Final version</td>
<td>Enste, LeIKon</td>
</tr>
<tr>
<td>V1.1</td>
<td>Nov 2018</td>
<td>Final approval</td>
<td>Engell, TUDO</td>
</tr>
</tbody>
</table>

Disclaimer
THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Any liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein. The members of the project CoPro do not accept any liability for actions or omissions of CoPro members or third parties and disclaims any obligation to enforce the use of this document. This document is subject to change without notice.
Table of contents

1 Executive summary ........................................................................................................ 5
2 Requirement Specification ........................................................................................... 5
  2.1 Management of the requirement specification phase ............................................. 5
  2.2 Results of the Requirement Specification ............................................................ 7
3 Functional Design Specification ................................................................................. 8
  3.1 Base Technology ....................................................................................................... 8
  3.2 Data Model ............................................................................................................... 8
  3.2 User Interface .......................................................................................................... 12
  3.3 API ......................................................................................................................... 15

List of Figures

Figure 1: Web based User Interface for an agile Requirement Specification Process .............. 6
Figure 2: Core of the Data Model .................................................................................... 9
Figure 3: Project View .................................................................................................... 12
Figure 4: Details of a Model .......................................................................................... 13
Figure 5: Search View .................................................................................................... 14
Figure 6: Model upload .................................................................................................. 15
DELIVERABLE D5.5

1 Executive summary

The goal of task 5.3 within WP 5 of CoPro is to develop a model management platform in order to provide information about catalogues and repositories of models. The goal is to increase the transparency of all model based applications within a company and to support the reuse of models. Using a web based platform, relevant information of model based applications should be given in an intuitive and easily explorable way.

Since month 12 of the CoPro project a systematic approach to specify the requirements for the Model Management Platform and based on this a first functional design specification were developed. Both tasks were performed by using innovative methods of project management. To obtain a requirement specification in an effective and sustainable manner, a web based requirement specification tool was provided to the partners. This approach was proved and tested successfully before in task 5.2 to develop the CoPro Integration Platform.

2 Requirement Specification

The development of a requirement specification for the CoPro model management platform was made from the users’ point of view. In order to collect and prioritise requirements for the usage of the CoPro Model Management Platform the same interactive web based Requirement Specification Platform « Mantis » was implemented as we used for the CoPro Integration Framework.

2.1 Management of the requirement specification phase

The resulting web based user interface is shown in the screenshot below. The requirement specification platform can be used by the partners for all tasks within WP5 during the whole project.
**Figure 1:** Web based User Interface for an agile Requirement Specification Process.
2.2 Results of the Requirement Specification

The following requirements were specified by the CoPro team, in particular by the partners involved in task 5.3.

Requirements for User Interface:

R 1. A web based user interface to explore model base applications should be provided without need to install additional software

R 2. It should be possible to browse models and projects using model based applications

R 3. It should be possible to search models by attributes using different filter

R 4. It should be possible to upload new models and versions

R 5. It should be possible to edit attributes as meta data describing models

Model Description Requirements:

R 6. The model management platform should differ between model metadata and application data

R 7. Models should be described by at least the following information:
   - ID
   - Author [Name, Email, Company]
   - Description
   - Version History
   - Functions the model consists of
   - For each Function
     a. Set of Input Parameter [Name, Datatype, Unit]
     b. Set of Output Parameter [Name, Datatype, Unit]
   - Link to source code (e.g. link to Subversion repository)
   - Format (e.g. Matlab 2014a Compiler 32bit DLL)

R 8. Version history of models should be transparent

Platform Functional Features:

R 9. Model description data should be stored on a central data base

R 10. The platform should provide a user and policy management

R 11. Only users with an appropriate permission can add or import models
R 12. Simulation/Calculation server should periodically check repository for new versions

R 13. The platform should provide an open interface to provide model information for other tools

3 Functional Design Specification

Based on the first requirement specifications provided by the partners, a functional design specification was proposed by partner LeiKon and approved by the partners who are involved in task 5.3.

3.1 Base Technology

The model management platform will be implemented using the Java programming language and the Eclipse framework. The user interface will be web-based using the Eclipse RAP (Remote Application Platform). With this design all data is stored on a central server and all authorized users can access the application using their web browser without installing new software on their computers. An SQL database will be used to store model metadata and application data (e.g. user management information). The model files itself will be stored in the filesystem of the server.

3.2 Data Model

The core of the model management platform is the database design. A database of process models and model based application was developed in several iterative design loops. The database contains ready-to-use model files, searchable metadata and version history. An important design decision was that the database shouldn't contain source code of models (e.g. “.m files” of Matlab). The main reason is that most of the model development tools itself contain features for version control and management. An additional source code version management would imply risks of inconsistencies and additional effort. Instead existing source code repositories can be linked, so that an unambiguous access to a valid source code can be obtained.

The figure below describes the core of the developed data model of the model management platform. The main elements, their most important attributes and the relationships between the data model elements will be described in more detail.
Figure 2: Core of the Data Model
Objects and their attributes:

Model

The Model is the central object class of the model management platform. It is identified by a unique name but can also have a shorter label for displaying purposes. A describing text and a list of associated keywords (e.g. “Furnace”) can be added by the user. So users can explore the database and find models that suit their application needs. Further documentation and the source code repository of the model can be referenced as hyperlinks.

The ModelOrigin of a specific model class refers to the software that created it (e.g. Matlab). A model consists of one or more Functions and can be referenced by any number of Projects. The actual Files that comprise a model can change over time and are therefore referenced by a ModelVersion.

Function

A Function is an entry point of a model – data is passed to the function, then a computation or simulation is performed on the data which results in new data that is returned from the function. The data that is passed to the function is referred to as (the list of) Parameters. The returned data is called (the list of) results. Results are in a data model point of view also parameter. A Function has a name (which matches the technical name in the model file) and a description. Each parameter and result has a name, description and optionally a unit (e.g. kg/m³). Also the DataType (e.g. integer or floating point number) and array or matrix dimensions are noted.

Version

A Model has at least one (the initial) ModelVersion, e.g. “1.0”. In general a model can be improved and refined over time. Each time a set of changes intended for deployment is made, a new version is created. A version is created by one or multiple Authors, each identified by name, email address and company. The authors of all versions of a model comprise the authors of the model. When creating a new version, the authors give it a name (e.g. “1.3”) and describe the changes made since the last version. The timestamp of the version is automatically added by the model management platform. Optionally, the corresponding version in a source control system can be referenced as a hyperlink.

In general it is not possible to track a source control system with the intent of automatically creating new versions in the model management platform from versions in the source control system. The reason is that the results of packaging steps (e.g. Matlab library export) are typically not stored in the source control system.

When creating a new version the authors upload one or more Files to the model management platform. In most cases the platform can automatically detect which Functions of a model are contained in the uploaded file(s). A file has one or more specific Formats
which describe how the file can be loaded by e.g. a simulation platform and access the contained functions.

Auxiliary files related to a model version can be given a description and uploaded or referenced as hyperlink (if a storage system for these files already exists). Examples for such files include documentation, test and simulation logs as well as data files used for model generation.

Examples describing file formats:

**Matlab**

A format describing a Matlab export consists of:

i. The Matlab version (e.g. 2014a)
ii. The export type (Compiler or Coder)
iii. The system architecture (Windows 32bit, Linux 64bit, ...)

In order to enable uploading multiple formats of a Matlab model to the model management platform it is allowed to upload multiple files for one version as long as each function is uniquely available for each format.

A Matlab file could have the format “Matlab Compiler 2014a 32bit”

**FMU (functional mockup interface) model**

An FMU format consists of the system architecture (Windows 32bit, Linux 64bit, ...). Since an FMU file can bundle model implementations for multiple system architectures the model management platform allows one file to have multiple formats.

One FMU file could have the formats “FMU Windows 32bit”, “FMU Windows 64bit” and “FMU Linux ARMv7”.

The model management platform can validate whether an uploaded version of a model contains all functions of the model in all supported formats.

**Project**

A *Project* is an organisational object that allows grouping models for a specific application, e.g. “Efficiency Improvements of R20 in 2018”. Projects are organized in *Directories* which behave like directories on a computer hard disk. In addition to their name projects can have a *description text*. The relationship between *Projects* and *Models* is many-to-many, i.e. a project can have multiple models and a model can belong to (or be used by) multiple projects, facilitating reusability.
3.2 User Interface

The functional design specification of the required User Interface will be described by sketches. The sketches were discussed and iterative improved with partners of task 5.3.

Once a user is authorized at the web-based user interface, he can use a customizable home screen. From here, a user can reach the projects view (see left area in Figure 3) and the model search area (see main view in Figure 3).

In the projects view Figure 3: Project View the directory structure is represented as a tree. Directories can be expanded and collapsed as in a typical file explorer. When a project is selected its description and also a list of all models used by the project will be displayed. The list of models can be used as an entry point to view details of specific models. In addition users with the appropriate permission can add or import models (the latter extracts metadata from an already existing model file).
The model view (c.f. Figure 4) displays all important information about a model at one glance. The description, links, keywords and contributing authors are displayed at the top. The next section lists the functions that comprise the model. Upon selection, the details of a function are displayed. These include its description, parameters and results. Parameters and results are displayed in a table format so that their attributes (unit etc.) are visible. The last section consists of a table of all available versions of the model. Here, the changes since the previous version and the available format can be seen. Also the download page of that version can be reached by a link. A new version can be uploaded via a button.
The search view (c.f. Figure 5) consists of an input field and a list of search results. The user enters one or more search terms in the input field and gets a list of all models whose name, description or key words match the search terms. A result can be selected in order to navigate to its detail page.
The upload view (c.f. Figure 6) is used to upload a new version of a model. As described above, a version has a name, a description of changes and a list of contributing authors. After inputting these, the user can upload the files of the version. The model management platform automatically detects the file format and the contained functions. When all files are uploaded the user saves the version which is then available for other users of the platform.

### 3.3 API

In order to provide model information for other tools or to enable deployments of models to other systems (e.g. simulation platforms, online data coupler ...) a web-based API (Application Programmers Interface) will be implemented in the model management platform. This API allows listing and searching of the model meta data and also downloading of the model files, thus enabling fully automatic deployment of models once they are uploaded to the model management platform. A simulation platform with an appropriate module can periodically check the web-API for new versions of used models and download them when they are available.