D5.3 REQUIREMENT SPECIFICATION AND FUNCTIONAL DESIGN SPECIFICATION OF THE COPRO INTEGRATION FRAMEWORK

Udo Enste
LeiKon GmbH - Germany

October 2017
www.spire2030.eu/copro
Project Details

PROJECT TITLE
Improved energy and resource efficiency by better coordination of production in the process industries

PROJECT ACRONYM
CoPro

GRANT AGREEMENT NO
723575

INSTRUMENT
RESEARCH AND INNOVATION ACTION

CALL
H2020-SPIRE-02-2016

STARTING DATE OF PROJECT
NOVEMBER, 1ST 2016

PROJECT DURATION
42 MONTHS

PROJECT COORDINATOR
PROF. SEBASTIAN ENGELL (TUDO)

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 Ereunys 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>
Deliverable 5.3
Public

Abstract

To realize advanced distributed optimization and scheduling algorithms, many different IT systems must interact in a smooth and easy to handle manner. The CoPro Integration Framework supports current and future data communication technologies and will be used as a flexible interaction and orchestration master for distributed plant coordination and scheduling applications. This deliverable describes the system used for and the results of the requirement specification of the CoPro Integration Framework. Moreover, the key features of the functional design specification are presented.

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>V0.8</td>
<td>Sept 2017</td>
<td>Creation</td>
<td>Udo Enste (LeiKon)</td>
</tr>
<tr>
<td>V0.9</td>
<td>Oct 2017</td>
<td>Review and contribution</td>
<td>Tim Butters (Sabisu)</td>
</tr>
<tr>
<td>V1.0</td>
<td>Oct 2017</td>
<td>Update</td>
<td>Udo Enste (LeiKon)</td>
</tr>
<tr>
<td>V1.1</td>
<td>Oct 2017</td>
<td>Final revision</td>
<td>Sebastian 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 estoppels 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 ............................................................................................. 6
   2.1 Management of the requirement specification phase ........................................... 6
   2.2 Results of the Requirement Specification .............................................................. 9
      2.2.1 Requirements for the engineering environment ........................................ 9
      2.2.2 Requirements for the runtime environment .............................................. 10

3 Functional Design Specification .................................................................................. 11
   3.1 Base Technology ...................................................................................................... 12
   3.2 Engineering Environment ..................................................................................... 12
   3.3 Runtime Environment - Deployment Strategy ...................................................... 13

List of Figures

Figure 1: Sketch of the role of the CoPro Integration Framework ................................. 5
Figure 2: Requirement specification process managed by "Mantis" ......................... 7
Figure 3: Web based User Interface for an agile Requirement Specification Process ... 8
Figure 4: First preliminary prototype of the engineering environment ...................... 13
Figure 5: Flexible Deployment of data orchestration jobs within distributed runtime environments 14

List of Tables

Table 1: Requirement states ............................................................................................. 7
1 Executive summary

The goal of task 5.2 of CoPro is to develop an engineering and runtime environment to realize data orchestration of site wide and also cross-site executions of distributed model based, scheduling, control and optimisation applications. The framework should have capabilities to integrate different sources and targets as well as different data calculation, transformation and validation functions required in distributed plant coordination solutions.

During the first 12 months of the CoPro project, a systematic approach to specify the requirements for the Integration Framework and based on this specification a first functional design specification were developed. Both tasks were performed by using innovative methods of project management. To produce the requirement specification in an effective and sustainable manner, a web based requirement specification tool was provided to the partners. This requirement tool will also be used to control the further development process. The functional design process was supplemented by an agile software development process. So instead of paper work, preliminary prototypes were provided in order to collect feedback and to steer the development process.
2 Requirement Specification

The development of a requirement specification for the CoPro integration framework was made from the users’ point of view. The requirements of the functionality of the integration framework are based on the concepts of WP1-4 and on the definition of the use cases in WP6 (Task 6.1).

2.1 Management of the requirement specification phase

In order to collect and prioritise requirements for the usage of the CoPro Integration Framework an interactive web based Requirement Specification Platform was implemented. The advantages of using such an interactive way to collect requirements are:

- Requirements can be collected in a systematic manner.
- Each requirement can be classified as one of a set of predefined requirement classes. This helps to structure the list of requirements, to organize the development process and later on to test and validate the implementation.
- During the whole development process it is clear who initiated a specific requirement. This helps to clarify open questions directly between the responsible development teams and the users.
- Additional information like notices, web links, drawings or documents can be linked to each individual requirement.
- Developers can give feedback such as remarks or questions either directly to the initiator of a request or to another member of the platform. Misunderstandings are can be solved in early phases of a development process.
- The collection of requirements can be expanded during the whole project phase in order to react to modified constraints.

As an interactive web based requirement specification tool LeiKon chose the open source platform for bug and ticket tracking “Mantis”. This tool, including the GUI, was customized by LeiKon in order to manage the requirement specification process in CoPro. Also a short manual was written by LeiKon that includes the most important aspects of the use of the tool.

The state chart in Figure 2 illustrates the requirement specification process that is provided by “Mantis”. The process is initialized by reporting a new idea which is assigned to the state “NEW”. A developer has to document the realization of requirement and to change the requirement state to “PROVIDED”. If a reporter does not agree on the realization of the requirement provided by a developer, a reporter can enhance the requirement description, which resets the requirement state to “NEW” and reinitializes the realization process. These steps can be repeated until both the reporter and the developer agree on the technical realization.
Figure 2: Requirement specification process managed by "Mantis".

An overview of available states of the requirements is provided in Table 1.

**Table 1: Requirement states**

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW (NEW)</td>
<td>A reporter posted a new idea or a feature request. A developer has to review the issue and provide documentation for the execution, development efforts etc.</td>
</tr>
<tr>
<td>PROVIDED (PRO)</td>
<td>The requirement was documented and given to the members of WP5, who will make a decision whether it will be used and whether now or in the next release.</td>
</tr>
<tr>
<td>DEFERRED (DEF)</td>
<td>A core team switched the requirement to the next release.</td>
</tr>
<tr>
<td>DECLINED (DEC)</td>
<td>A core team declined the requirement.</td>
</tr>
<tr>
<td>CONFIRMED (CON)</td>
<td>A core team checked and accepted the requirement.</td>
</tr>
<tr>
<td>SCHEDULED (SCH)</td>
<td>The requirement is ready and planned for developing.</td>
</tr>
<tr>
<td>DEVELOPED (DEV)</td>
<td>The requirement is developed and incorporated.</td>
</tr>
<tr>
<td>TESTED (TST)</td>
<td>The requirement is tested.</td>
</tr>
<tr>
<td>DELIVERED (DEL)</td>
<td>The system is in production or delivered to customer.</td>
</tr>
<tr>
<td>CLOSED (CLS)</td>
<td>The requirement is archived for possible later use.</td>
</tr>
</tbody>
</table>
The resulting web based user interface is shown in the screenshot below. The requirement specification platform can also be used for the other tasks within WP5.

Figure 3: 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 5.2. The requirements are divided into two parts. The first part describes the requirements for the engineering environment and the second part specifies requirements for the runtime environment of the integration framework.

2.2.1 Requirements for the engineering environment

R.1.1 Orchestration of different systems and model environments

R.1.2 Managing a proper integration of data scattered throughout the company’s information systems

R.1.3 IT systems must interact in a smooth and easy to handle manner

R.1.4 Data orchestration should provide ETL functionalities (ETL: Extract, Transform, Load). Modular interacting functions must be provided in order to read, transform and write data within an ETL-chain. Each specific chain must be storable with a specific ID and Version as “jobs”.

R.1.5 A graphical editor with drag & drop functionality to specify workflows and data flows between different IT systems by using workflow notation through an intuitive graphical library of shapes and links is needed. With a chain of connected functions a dataflow management processes should be realizable without need of programming skills. Designing a workflow it should be possible to:

R.1.5.1 Put in place data integration actions using a library of connectable components

R.1.5.2 Set connections and relationships between components in order to define the sequence and the nature of actions (= building interaction jobs)

R.1.5.3 Create and add items to the repository/library for reuse and sharing purposes (in other projects, jobs or with other users)

R.1.5.4 Change default settings of components or create new components or family of components to match specific needs.

R.1.5.5 Separate data and event flows within a workflow/data flow chain

R.1.6 It must be possible to build a repository of pre-built components and connectors. It should be possible to group function types into type libraries.

R.1.6.1 Library of elements to trigger event flows (Cyclic Trigger, Cron Trigger, Event Driven Trigger, …)

R.1.6.2 Library of data preparation functions

R.1.6.3 Connectors and Interfaces to standardized data communication technologies (like ODBC, OPC DA, OPC UA, OPC HDA) and as well to common system
specific interfaces (like OSI-PI API, IP.21 API, SAP RFC) and also connectors to Web Services and Cloud Infrastructures

R.1.6.4 Library of transformation and data re-routing components
R.1.6.5 Library of data mapping components
R.1.6.6 CSV data import and export driver

R.1.7 A common UI-Interface should provide a list of available type libraries, function types including a brief summary of the functions.

R.1.8 All configurations and specified ETL-chains (jobs) should be managed within projects. In order to resume the development process, the projects including configurations, designs and current development states should be persisted to local/network file storages.

R.1.9 It should be possible to integrate templates of typical data workflows e.g. to validate data including exception handling in case of anomaly detections, to recalculate data or to provide substitute data.

R.1.10 Generic interfaces should be provided to integrate plant models designed by common model development tools like Matlab.

R.1.11 Generic interfaces should be provided to integrate big data analysis tools like DIVIS ClearVu Analytics.

R.1.12 It should be possible to activate a whole chain but also each component of a chain individually into an online mode in order to support the start-up and commissioning phase of a data integration application. In the online mode it should be possible to monitor input and output parameters of each component of a chain inside the engineering environment.

R.1.13 If there is the need to connect to database tables or you have to establish service connections like a ftp-connection several times inside a job or project, it should be possible to centralize the connection information details once in a metadata folder.

R.1.14 The Engineering Environment should provide an extensible help system for accessing common knowledge base and/or context sensitive topics.

R.1.15 There should be an integration path of the engineering environment with the former developed MORE Deployment Platform.

2.2.2 Requirements for the runtime environment

R.2.1 It must be possible to download and to activate workflows from the engineering environment in order to bring it into an online runtime mode.

R.2.2 All functions must lead to robust data integration solutions.

R.2.3 It must be possible to implement event driven distributed software scheduling solutions.
R.2.4 IT security aspects must be taken into account. Especially the runtime engines must be connectable by one or at least a few number of specific TCP-IP ports in order to fulfil firewall demands.

R.2.5 In an online mode, activation or deactivation of workflows or of single workflow components must be possible at any time.

R.2.6 The framework should provide a web based user interface to monitor the state of the workflow activities. Therefore the runtime environment should provide a common service interface which is connectable by common web based user interface solutions.

R.2.7 The framework should provide a flexible deployment of data orchestration jobs into distributed runtime environments. Dependent on the IT-system landscape of a company and dependent of their network architecture it must be possible, to load the workflow applications to different PCs at different network levels (e.g.: DCS level, MES level, office level, ERP level).

R.2.8 The runtime environment should work as a stand-alone application running in a user-mode as well as a service of the operating system.

R.2.9 Any data passed from within a company environment to an external location (e.g., cloud infrastructure or services) must be secured (e.g., using SSL over HTTPS).

3 Functional Design Specification

Based on the requirement specification, a functional design specification was proposed by partner LeiKon and approved by the partners who are involved in task 5.2. The CoPro Integration Framework includes an engineering environment and runtime environments which can be deployed within a distributed IT-landscape.

The Engineering Environment will support all programming and engineering steps which are necessary to build data flows between several IT-systems in order to realize well connected distributed IT-solutions for plant control and plant coordination applications. As a fundamental engineering methodology a modern “model based software design” process will be supported. In this context “model” is used in the sense of “data model”. In order to use the wide range of its existing functionalities the “Eclipse Modelling Framework” (EMF) was chosen as a base implementation technology for developing IDEs (Integrated Development Environment). The Eclipse Modelling Framework is an open software design framework and code generation facility for building tools and other applications based on structured data models. Within the engineering environment ready to use workflow applications can be realized. This includes the integration of existing systems like DCS, PIMS and ERP systems but also tools like Matlab, gPROMS, or ClearVU as representatives of model development and model generation tools. The result of the design process can be transferred to decoupled runtime environments using tool chains.

The Runtime Environments ensure the robust execution of the data flows that were specified in and downloaded from the Engineering Environment. A Runtime Environment runs as a black box independently of the Engineering Environment. To be able to monitor the runtime application, each Runtime Environment offers a built in web service interface. Based on the web service interfaces, all data integration applications can be monitored online.
3.1 Base Technology

Model-driven architecture (MDA) is a software design approach for the development of software systems. It provides a set of guidelines for the structuring of specifications which are expressed as data models. Model-driven architecture is a kind of domain engineering. It was originally launched by the Object Management Group (OMG) in 2001. Model-driven architecture focuses on forward engineering, i.e. producing code from abstract, human-elaborated modelling diagrams (e.g. class diagrams). To develop the Engineering Environment of the CoPro Integration Framework, the open source Eclipse Modelling Framework (EMF) was selected as the most suitable framework to develop the engineering environment and to provide a robust and high-performance runtime support.

Based on a data model specification, EMF provides tools and runtime support to produce a set of software classes for the model, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. Models can be specified using for example UML or domain specific XML documents. Most important of all, EMF provides the foundation for interoperability with other EMF-based tools and applications. So a wide range of ready to use functionalities that are needed in an efficient software design process can be used.

3.2 Engineering Environment

The base for specifying a graphical engineering environment is a generic data model for data flow descriptions. The data model to describe data flow descriptions is derived from the Systems Modelling Language (SysML). SysML is a general-purpose modelling language for systems engineering applications. It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. SysML was originally developed by an open source specification project, and includes an open source license for distribution and use. It is defined as an extension of a subset of the Unified Modelling Language (UML) using UML’s profile mechanism.

The Engineering Environment of the CoPro Integration Framework will offer to the user an editor to specify simple and complex data flows between distributed IT-solutions.

With this editor, a graph of data flows between several function modules can be designed. As base elements, nodes, which represent transaction or data manipulation modules, and edges, which represent either data flows or event flows, were defined. The modules can be described by module types which can be reused and customized as individual module instances inside a data flow specification (job). Module types can be stored and organized in libraries. Each module has 0-n data interfaces and at least one event input or output interface. The event interfaces can be used to specify pre-known or state dependent transaction logics.

In order to validate the engineering concepts and the usability of the engineering environment, a first preliminary prototype of the CoPro engineering environment was realized in an early project phase of CoPro (see Figure 4). This approach supports an agile software development process. Agile software development describes a set of principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing teams. It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement. It encourages also a rapid and flexible response to changes and supports the definition and continuing evolution of software solutions.
A first preliminary prototype of the engineering environment was provided by month 12 of the CoPro project by partner LeiKon. Now the involved partners of task 5.2 have a “look & feel environment” to discuss further the needs of the users in more detail. The needs and new requirements can be provided using the requirement specification tool described in chapter Fehler! Verweisquelle konnte nicht gefunden werden. So the software validation and hints for targeted software improvements can be parallelized and reinforced by several partners. Using agile software development with preliminary prototypes also supports the involvement of the industrial end-user companies in the requirement specification and the evaluation process in an early project phase.

### 3.3 Runtime Environment - Deployment Strategy

The modular architectural design allows a clear separation between the engineering environment and several runtime engines. This leads to a flexible and decentralized deployment of applications. Typically the engineering environment will be installed inside an office local area network. Depending on the IT-systems which are involved inside a distributed interaction logic, it makes sense to deploy the runtime applications inside specific sub nets of a company. If a data flow application includes solely data from and to systems inside a DCS network, e.g. when used for advanced control strategies, it is advisable to run this application exclusively inside this network domain. This increases the performance and avoids complex IT security measures. On the other hand, if data flows only between IT-systems inside the ERP level, e.g. between SAP, plant scheduling and optimization tools are needed, it is advantageous to run the data interaction application inside the ERP or office network. Figure 5 illustrates the flexible deployment strategy which will be realized in the design of the framework.
Figure 5: Flexible Deployment of data orchestration jobs within distributed runtime environments.