Cloud-based **R**apid **E**lastic **MA**nufacturing

WP5 – Cloud Manufacturing Process and Optimisation Framework

D5.10 – T5.5 – CREMA Runtime Optimisation - Prototype II

Deliverable Lead: DFKI

Contributing Partners: DFKI, IKER, TCO

Delivery Date: 30.06.2017

Dissemination Level: Public

Version 1.0

This deliverable provides a description of the second prototype implementation of task T5.5 Runtime Optimisation. As stated in the Description of Action (DoA), this deliverable is a prototype (software) deliverable. As such, this document is reduced in length and its only purpose is to briefly describe the prototype functionality as well as to give installation instructions and usage clarifications. This document is delivered with instructions to obtain the software itself.

Document Status	
Deliverable Lead	Matthias Klusch, DFKI
Internal Reviewer 1	Jessica Gil, TCO
Internal Reviewer 2	Olena Skarlat, TUV
Type	Other
Work Package	WP5: Cloud Manufacturing Process and Optimisation Framework
ID	D5.10 – T5.5 – CREMA Runtime Optimisation – Prototype II
Due Date	30.06.2017
Delivery Date	30.06.2017
Status	For Review | Approval

Note

The type of the official deliverable is OTHER, as it is a software deliverable. This document mainly describes how to obtain the software and how to run it.

Disclaimer

The views represented in this document only reflect the views of the authors and not the views of the European Union. The European Union is not liable for any use that may be made of the information contained in this document.

Furthermore, the information is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user of the information uses it at its sole risk and liability.

Project Partners

Ascora GmbH, Germany

Information Catalyst, United Kingdom

TechnischeUniversität Wien, Austria

Technology Application Network Limited, United Kingdom

German Research Center for Artificial Intelligence, Germany

IKERLAN S. Coop., Spain

Ubisense, United Kingdom

Tenneco automotive Europe bvba, Belgium

FAGOR ARRASATE S. Coop., Spain

Goizper, Spain

Executive Summary

This deliverable provides the requirements, installation, and execution details of the second prototype of the Design Time and Runtime Optimisation (ODERU) component with focus on its functionality for process optimisation at runtime (ORU). The ORU part of ODERU is responsible for functional and non-functional optimisation of service-based process models in BPMN at their time of execution.

The second prototype of ODERU extends the functionality of the first prototype (see deliverable D5.9) by means of (a) an integrated functional and non-functional process optimisation of process service plans at runtime, (b) optional approval of re-optimised process service plans by a human user before they are returned to PRU for continued execution, and (c) the integration of the CREMA security concept into the respective part of the internal workflow of ODERU.

The CREMA use case specific constraint optimisation problems (COP) to be solved by ODERU for a non-functional optimisation of the given process service plan with actual input data at runtime have been re-checked with and approved by the user partners for both use cases.

The software of the first prototype is available at https://go.abelssoft.de/oderu_prototype2 (password: crema_reviewers_1234).

Table of Contents

List of Figures and Tables

Figures

Tables

1 Introduction

CREMA – Cloud-based Rapid Elastic MAnufacturing – is a project funded by the Horizon 2020 Programme of the European Commission under Grant Agreement No. 637066.

Within this deliverable, the process of installation and execution of the software prototype is described to support administrators and users.

1.1 CREMA Project Overview

CREMA aims at simplifying the establishment, management, adaptation, and monitoring of dynamic, cross-organisational manufacturing processes following Cloud manufacturing principles. CREMA will also provide the means to integrate data from distributed locations as if the complete manufacturing was carried out on the same shop floor, by integrating extra- and inter-plant manufacturing assets and making them “mobile”.

CREMA will be built upon concepts and methods from the fields of Virtual Factories, Service-oriented Computing, Ubiquitous Computing, Cyber-Physical Systems, the Internet of Things and the Internet of Services, and naturally and most importantly Cloud computing. To achieve its goals, the project will define tools and approaches in these areas:

- Manufacturing Virtualisation& Interoperability
- Cloud Manufacturing Process and Optimisation Framework
- Cloud Manufacturing Collaboration, Knowledge and Stakeholder Interaction Framework

Thus, to achieve its goals, CREMA conducts original research and applies technologies from the fields of full end-to-end integration of Cloud manufacturing, integration of manufacturing assets and corresponding data sources, the design and execution of manufacturing processes, to the end user support via collaboration and interaction tools. For more information, please refer to the project Website[1].

1.2 Deliverable Purpose, Scope and Context

The purpose of this deliverable is to accompany the second prototype implementation of the CREMA component ODERU for runtime optimisation (ORU) in T5.5. As such, its main purpose is to briefly clarify the scope of the relevant part of the ODERU prototype, and (as in D5.7) to show the download, installation instructions and the use of the API of the ODERU software. The document is limited in length as the main focus of the task is the software itself rather than its accompanying document.

1.3 Document Status and Target Audience

This document is listed in the Description of Action (DoA) as “public”, which means “Restricted to other programme participants (including the Commission Services)”, primarily since the audience of the document is largely internal. It is true, of course, that the largest audience for dissemination itself is external, but this document covers only the planning around this and not the outputs of doing this and hence its non-public nature.

[1] http://www.crema-project.eu/

1.4 Abbreviations and Glossary

A glossary of common terms and roles related to the realisation of CREMA as well as a list of abbreviations is provided as an online glossary[2] / abbreviations list[3].

1.5 Document Structure

This deliverable is broken down into the following sections:

- Section 1 (Introduction): Provides an introduction for this deliverable, including a general overview of the project, and outlines the purpose, scope, context, status, and target audience of this deliverable
- Section 2 (Scope and Relationship): Clarifies the context and scope of the first software prototype deliverable for runtime optimisation, and its relationship with other CREMA components for this purpose
- Section 3 (Requirements and Preparations): Describes the requirements of and preparations for installing the software component ODERU
- Section 4 (Deployment): Describes the deployment of the component for its execution and usage.
- Section 5 (Execution and Usage): Describes the execution and usage of the deployed component ODERU.
- Section 6 (Feature Status): Provides an overview of the implemented features of the ORU part of ODERU matching its final technical and functional requirements.

[2] http://crema-project.eu/glossary
[3] http://crema-project.eu/abbreviations

2 Scope and Relationship

The ODERU component implements the complete set of functionalities for process optimisation at design time and runtime as specified in the functional and technical architecture (see deliverables D3.3 and D3.4). This section summarizes the ODERU part for the optimisation at runtime (ORU) with focus on functionalities that have been added to the first prototype (see deliverable D5.9).

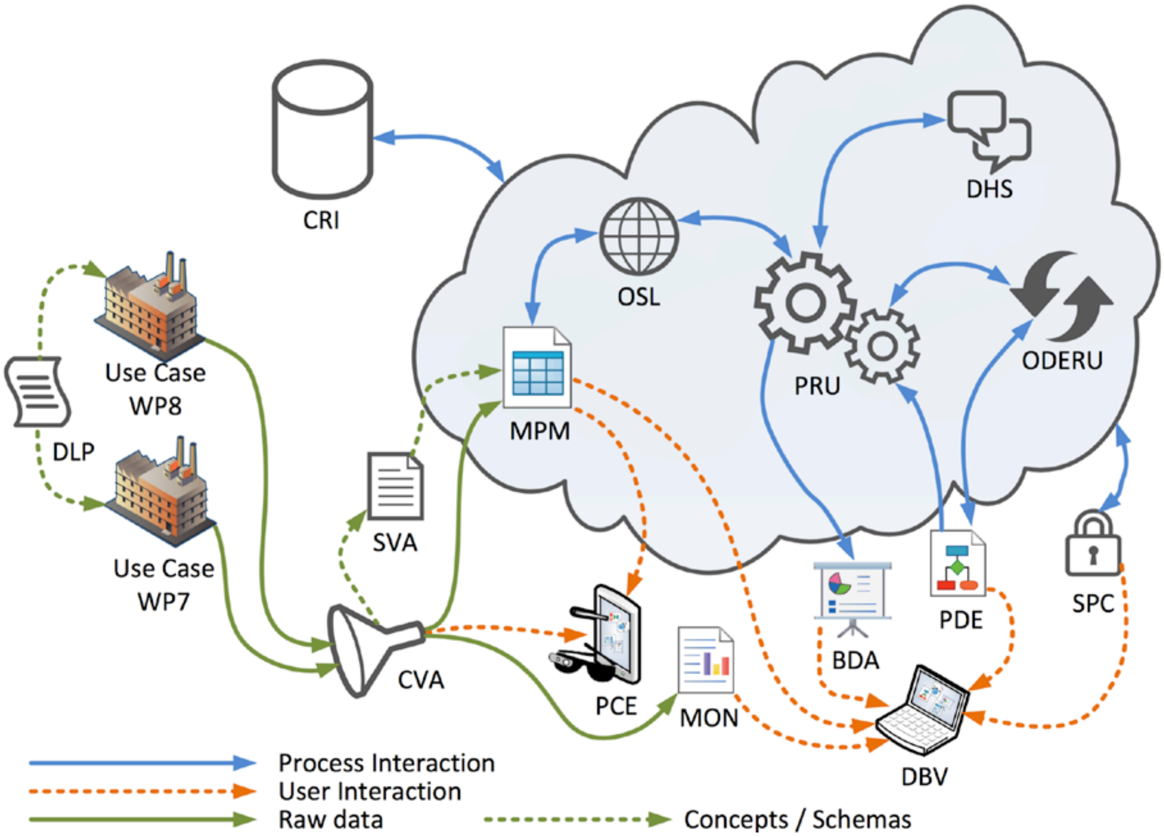

Figure 1: ODERU position in the CREMA platform architecture

The main improvements of the ORU part of the second prototype of ODERU compared to its first prototype are the same as for the ODE part:

(a) The COP solving software modules (for non-functional optimisation) are fully integrated with the internal component workflow of ODERU. The optimisation result in terms of optimal service parameter values and bindings required for the optimal execution of the re-optimised process service plan (PSP) are embedded into the plan which is returned to the PRU for its continued execution.

(b) Function for approval of re-optimised process service plans by a human user before they are returned to PRU for continued execution

(c) The CREMA Security and Privacy component (SPC) was integrated into the internal workflow of ODERU. Since the aspect of authentication of inter-component requests required a major revision of previously implemented interaction of ODERU with MPM and CRI, a significant amount of efforts from the final implementation phase of ODERU was put into enabling this integration. This integration was concerned with addressing user-specific access/visibility constraints in the internal computation of optimal process service plans.

The implemented modules of the ODE part of ODERU and its interactions with CREMA components for process optimisation at design time are shown in Figures 2 and 3.

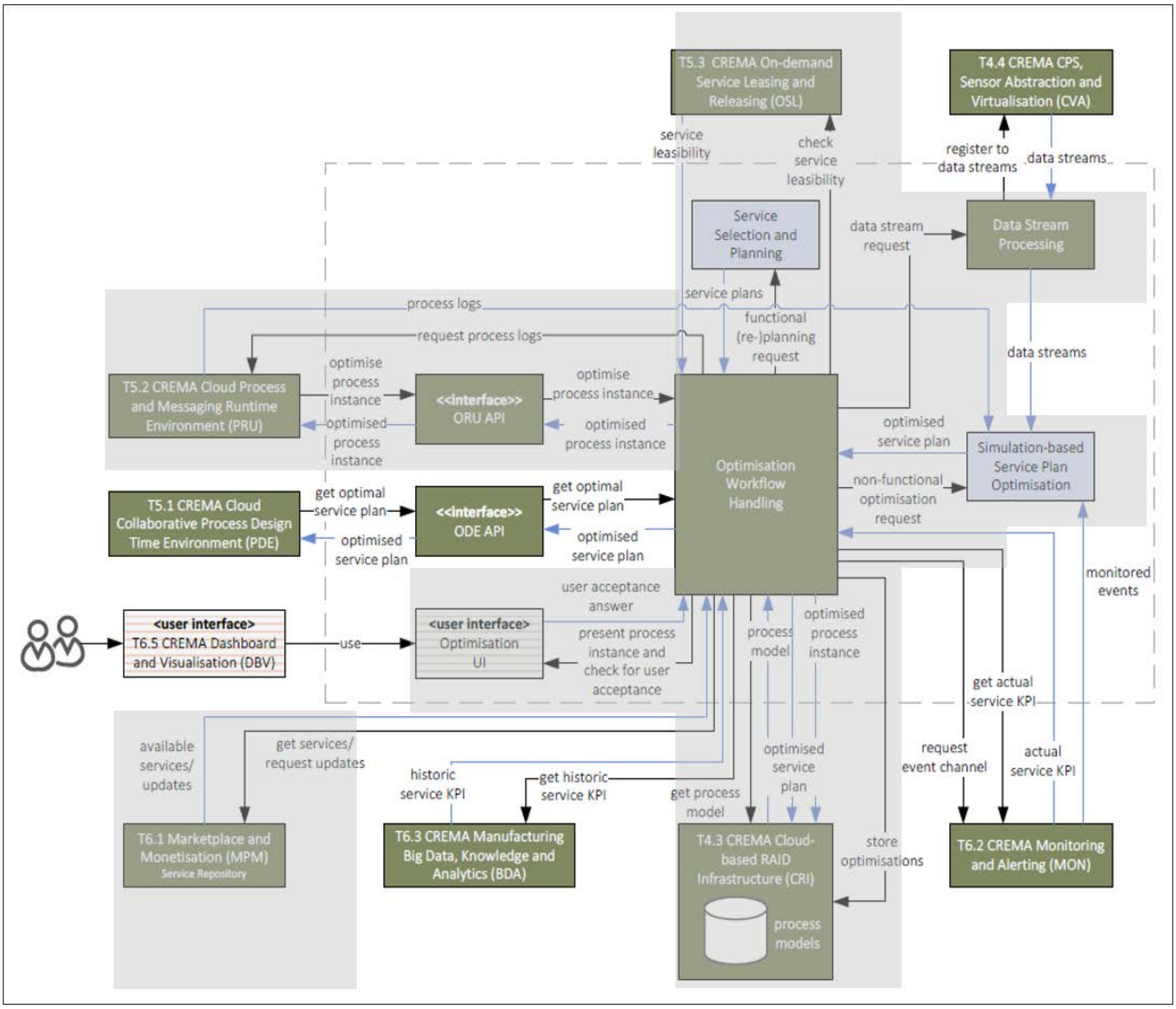
T5.3 CREMA On-demand Service Leasing and Releasing (OSL)
T4.4 CREMA CPS, Sensor Abstraction and Virtualisation (CVA)
service feasibility
check service feasibility
register to data streams
data streams
Service Selection and Planning
data stream request
Data Stream Processing
service plans
functional (re-)planning request
process logs
request process logs
data streams
T5.2 CREMA Cloud Process and Messaging Runtime Environment (PRU)
optimise process instance
optimised process instance
<<interface>> ORU API
optimise process instance
optimised process instance
optimised service plan
Simulation-based Service Plan Optimisation
Optimisation Workflow Handling
non-functional optimisation request
T5.1 CREMA Cloud Collaborative Process Design Time Environment (PDE)
get optimal service plan
optimised service plan
<<interface>> ODE API
get optimal service plan
optimised service plan
monitored events
user acceptance answer
<user interface> T6.5 CREMA Dashboard and Visualisation (DBV)
use
<user interface> Optimisation UI
present process instance and check for user acceptance
process model
optimised process instance
get actual service KPI
available services/ updates
get services/ request updates
historic service KPI
get historic service KPI
get process model
optimised service plan
request event channel
actual service KPI
store optimisations
T6.1 Marketplace and Monetisation (MPM)
Service Repository
T6.3 CREMA Manufacturing Big Data, Knowledge and Analytics (BDA)
T4.3 CREMA Cloud-based RAID Infrastructure (CRI)
process models
T6.2 CREMA Monitoring and Alerting (MON)

Figure 2: Modules of ODERU Prototype II for optimisation at runtime

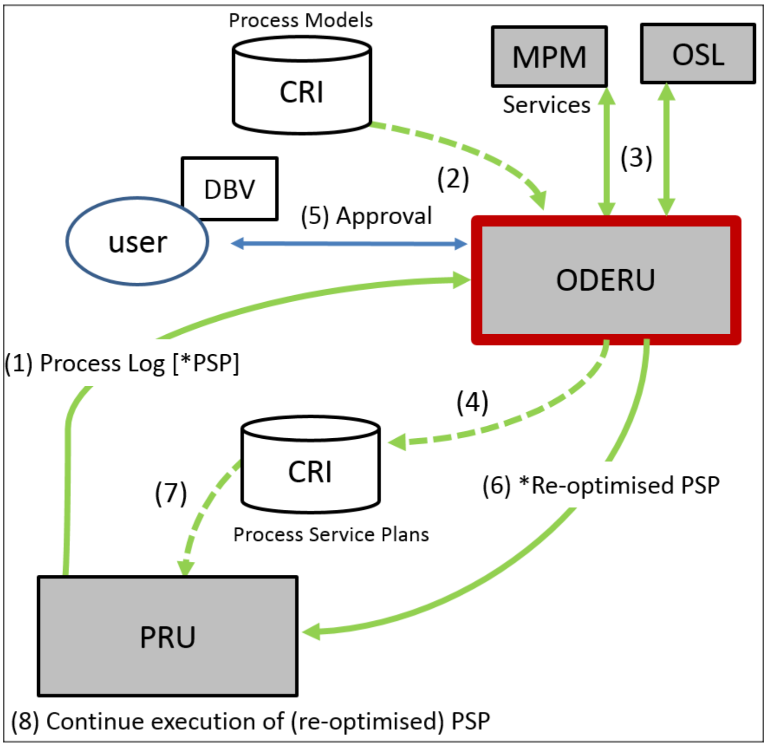

Figure 3: CREMA component interactions with ODERU for optimisation at runtime

The general interactions of ODERU with CREMA components for process optimisation at runtime is shown Figure 3, and the specific interactions for this purpose in both CREMA use case demonstrators are shown in Figures 4 and 5.[4]

Note: In the past reporting period, the implementation of interactions of ODERU with MON, BDA, and CVA as originally envisioned in D3.3 turned out to be not required anymore for the ODERU prototype II in both CREMA use case demonstrators. This change of interaction requirement was agreed with the partners. In fact, in CREMA the optimisation requests are passed to ODERU through its RESTful API by PDE and PRU, or any other proprietary program such as those developed by the user partners for the use cases.

Any dynamic changes of service parameter values by some component like the MPP (Maintenance Planning Program) require the latter to update the affected service descriptions and COPs in the process models. These changes are visible to ODERU via its push-notification/interaction with the MPM and used for its re-optimisation of process service plans (PSPs). Any updated COP becomes known to ODERU as soon as an optimisation request with it is issued. There is no need for an individual interaction with MON for this purpose. Furthermore, any analysis result of historic and forecasted service parameter values by the BDA is, if available and appropriate, inherently reflected in the specification of the optimisation requests that are passed to ODERU in the CREMA use cases. An online detection of optimisation relevant events by means of semantic reasoning/query answering with C-SPARQL on sensor data that may be provided by CVA and/or MON is basically supported by ODERU with its RDF stream processing module.

[4] An asterisk (*) before a named entity in figures 3, 4 and 5 denotes the link to this entity stored in the CRI.

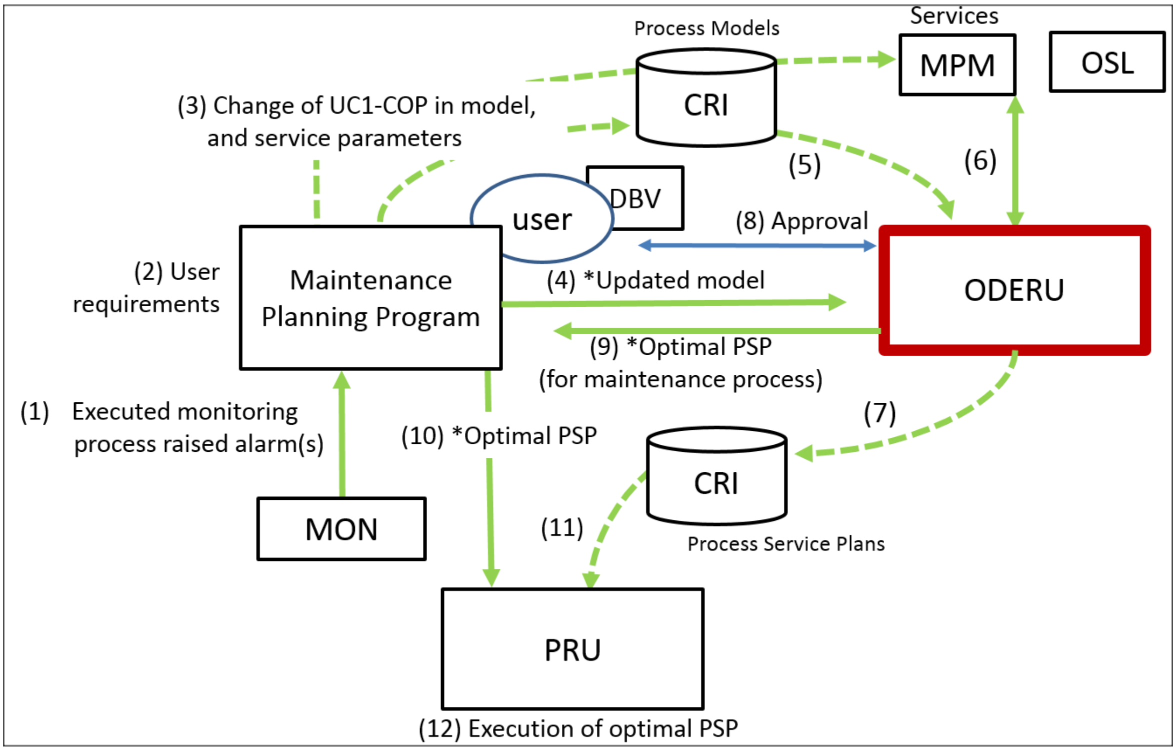

Figure 4: Component interactions for process optimisation by ODERU in Use Case I

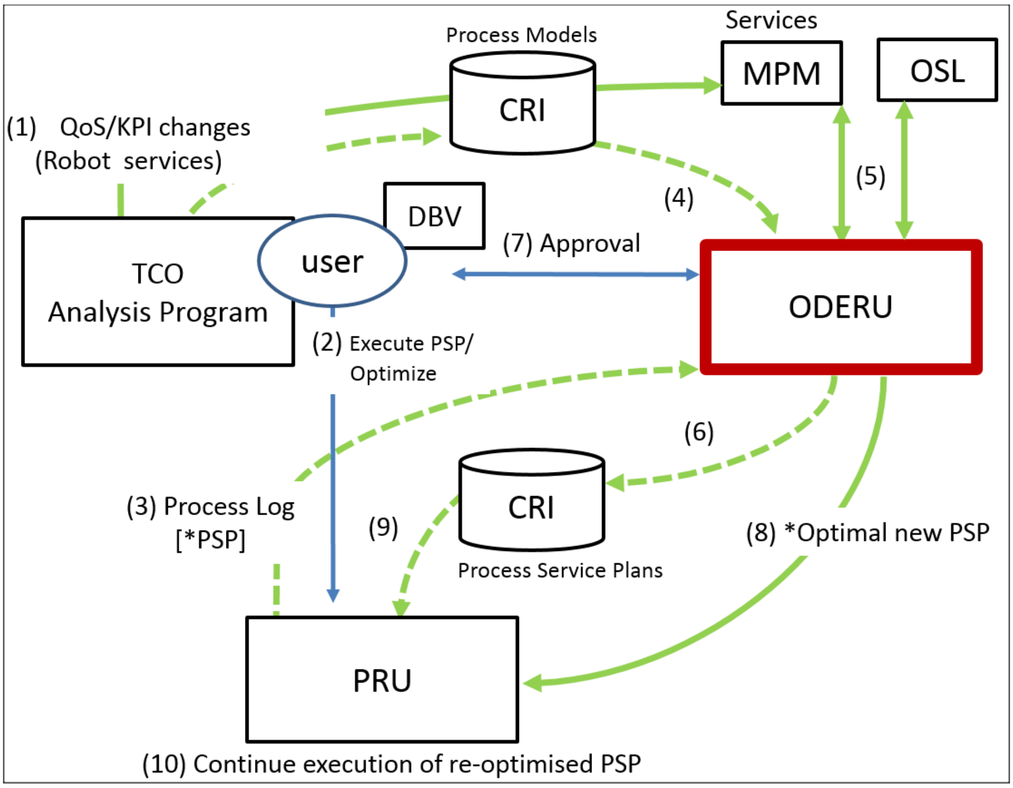

Figure 5: Component interactions for process optimisation by ODERU in Use Case II

3 Requirements and Preparations

There are no changes of requirements and preparations to be done for installing ODERU prototype II compared to its first prototype (see D5.9 Sect. 3.1). For reasons of self-containment of this document, the following sections are the same as in the deliverable D5.8 (Section 3).

Note on licensing[5]: The code of the ODERU is Open Source under the Apache 2.0 license[6]. The state of the code at the end of the project CREMA represents/constitutes the final state of this component. Please note that ODERU makes use of the Java constraint programming solver JaCoP which code is released under GNU Affero General Public License v3[7] by the core developers who are the JaCoP code copyright holders. If ODERU is used for commercial purposes the embedded JaCoP code can be released by the copyright holders under different license schemes[8].

The ODERU component provides process optimisation at design time (ODE, see deliverable D5.7) and runtime (ORU, see deliverables D5.9 and D5.10). This section provides requirements for running the second prototype of ODERU, its installation and instructions to make the component available for interconnected components at design time of process models, and describes the process how to execute the installed instance of this CREMA component.

Since there is no graphical user interface (GUI) the component can only be used via the ODERU REST API, which is documented at http://docs.crema-project.eu/#oderu. The complete software of the first prototype of ODERU is available at https://go.abelssoft.de/oderu_prototype2 (password: crema_reviewers_1234).

A few preparations to install and run the ODERU component and the COP solving software in the first prototype software package have to be made. It is recommended to use a Linux-based operating system (e.g., Ubuntu 16.04) to host the Docker[9] VM. The Docker version used for the testing is 17.03.1-ce-win12 (build number 12058). The installation of Java JDK version 8 and respective setting of the PATH environment variable is required.

[5] https://choosealicense.com/licenses/

[6] http://www.apache.org/licenses/LICENSE-2.0.html

[7] http://www.gnu.org/licenses/agpl-3.0.de.html

[8] https://osolpro.atlassian.net/wiki/display/JACOP/JaCoP+Licence

[9] https://www.docker.com/

4 Deployment

There are no major changes of actions required to be performed for deploying ODERU prototype II compared to its first prototype (see D5.7 Sect. 3.2).

The complete software of the second prototype of ODERU is available in the file ODERU_Prototype2.zip which is available at https://go.abelssoft.de/oderu_prototype2. The ODERU is delivered in a self-contained form as a Docker image and can be obtained from https://go.abelssoft.de/oderu_prototype2. The Docker image is in the file oderu.zip which is part of the complete software of the second prototype of ODERU in the downloaded file ODERU_Prototype2.zip.

Once deployed in a local Docker virtual machine every required component for ODERU is in place and no external dependencies exists with the host OS. To install the software, the Docker image has to be downloaded, expanded, and built. Following these steps, the ODERU software is launched:

```
zip –x oderu.zip .

cd oderu/oderu-app

docker build -f Dockerfile -t oderu .

docker run -p 8080:80 -t oderu
```

For more information, please refer to the *README.md* file text in the main directory of the release. The ODERU answers to the external port 8080. The basic configuration is already present in the deployed Docker image. For a final release, some configurations will be available, for example to customise the locations of the MPM and CRI components.

Note: The software module for solving the flexible job-shop scheduling COP is also included in ODERU_Prototype2.zip as separate zip file shopST.zip. The default COP solver JaCoP is fully integrated in the ODERU prototype II. The code for the second COP solver of ODERU, that is the shopST for approximated optimal flexible job-shop scheduling, is in the file shopST.zip. This file should be extracted and saved in an individual base directory named shopST; the executable solver in this directory is compiled as shopST.jar.

5 Execution and Usage

There are no changes of actions required for executing and using ODERU prototype II compared to its first prototype (see D5.7 Sect. 3.3). The only difference is, that the COP solver JaCoP is now fully integrated in ODERU. Therefore, no stand-alone application of JaCoP is provided anymore.

Once the ODERU component (Docker image) is deployed and launched, the ODERU REST API is exposed automatically through the port 8080, waiting for incoming connections/requests.

The installed COP solver shopST in the directory shopST can be executed for approximated optimal job-shop scheduling with, for example, the Hurink benchmark as follows:

```
java –jar shopST.jar -b benchmarks\Hurink_Data\vdata\la40.fjs -r 10 -si 500 -tl 60 -disp
RNDRND -c TOPC -ml 5 -gui true
```

with parameters (see explanation in params_definition.pdf)

-b path to test data file, otherwise a random scenario is generated

-r number of simulated trading rounds

-si number of iterations per simulated trading round

-tl maximal execution time in minutes

-seed basis of random number generation

-disp disposition rule for initial planning, e.g.:

RND: random, FASFS: first at shop first served, SPT: shortest processing time

-c cost function, e.g.:

TOPC: total operation completion date, TOPL: total operation lateness,

TOPT: total operation tardiness, TOPSL: total operation slack time

-ml maximal number of trading levels

-gui shows GUI (true, false)

6 Feature Status

The status of the implemented functionality of the ODE part of the CREMA ODERU Prototype II matching the respective requirements is shown in Table 1 (marked **Runtime** in the column for comments). The functions and requirements indicated in the table are extracted from the deliverables D3.2 Functional Specification and D3.3 Technical Specification (Sect. 5.4), the deliverables D5.7 and D5.9 for the first prototype, and the agreed changes of interaction requirements (see Sect. 2). In summary, the required functionality ORU part of ODERU has been fully implemented.

Table 1: Status of the ODERU Prototype II

Requirement of ODERU Prototype II	ID/Source	Priority	Status	Comments
Functionally optimal composition of process service plan for given process model in BPMN	ODERU_F 010	Must	done	Design Time REST method PUT
Non-functional optimisation of process model by solving of associated constraint optimisation problem COP	ODERU_F 020	Must	Done	Design Time REST method PUT
Realise (functionally compose and non-functionally optimise) a process service plan for given process model	ODERU_H 510	Must	Done	Design Time REST method PUT
Functionally optimal composition of service plan for process *instance*	ODERU_F 050	Must	Done	**Runtime** REST method PUT
Non-functional optimisation of process *instance*	ODERU_F 040	Must	Done	**Runtime** REST method PUT
Realise (functionally compose and non-functionally optimise) a process service plan for a process instance.	ODERU_H 520	Must	Done	**Runtime** REST method PUT
Approve a newly computed process service plan	ODERU_F 030	Must	Done	Design Time, **Runtime** REST method PUT
Return ordered list of services implementing a given process task (service selection)	ODERU_F 060	Must	Done	Design Time, **Runtime** REST method GET
Retrieve previously computed process service plan for a given process model	ODERU_F 070	Must	Done	Design Time REST method GET
Retrieve previously computed process service plan for a given process instance	ODERU_F 080	Must	Done	**Runtime** REST method GET
Notify about new/updated service in MPM component (push approach)	ODERU_F 100	Must	Done	Design Time, **Runtime** REST method PUT

Requirement of ODERU Prototype II	ID/Source	Priority	Status	Comments
Remove an existing service from MPM component (push approach)	ODERU_F 110	Must	Done	Design Time, **Runtime** REST method DELETE
Notify about new data bucket from an external “buffered stream”	ODERU_F 120	Must	Done	Design Time, **Runtime** REST method PUT, part of module RSP
Implement module RSP “Data Stream Processing” (RDF stream reasoning/processing)	D3.2, Sect. 5.4.1	Must	Done	**Runtime**
Interaction with PRU: Get optimisation request via process log; Return re-optimised *process service plan for continued execution	D3.2, Sect. 5.4.1 D5.9, Sect. 2	Must	Done	**Runtime**
Interaction with MPM: Retrieve/remove service descriptions	D3.2, Sect. 5.4.1 D5.9, D5.7, Sect. 2	Must	Done	Design Time, **Runtime**
Interaction with CRI: Get/Store process models and process service plans	D3.2, Sect. 5.4.1 D5.9, D5.7, Sect. 2	Must	Done	Design Time, **Runtime**
Interaction with OSL: Check for service leasability	D3.2, Sect. 5.4.1 D5.9, Sect. 2	Must	Done	**Runtime**
Interaction with PDE: Get optimisation request; Return optimal process service plan	D3.2, Sect. 5.4.1 D5.7, Sect. 2	Must	Done	Design Time
Support of dynamic and flexible job-shop scheduling in smart factories	D3.2, Sect. 5.4.1 D5.7, Sect. 2	Should	Done	**Runtime**, Design Time Anytime FJSS solver system In addition to CREMA use cases

Note on component testing: The integration (interaction) of the ODERU component with other components has been manually performed according to the component workflows for process optimisation in the CREMA use cases. Regarding the ODE part of ODERU (see Sect. 2, Figure 3) its REST-based interactions with PDE, MPM and CRI were successfully tested and used in the demonstrations of the CREMA use cases at the second project review. In addition, the user partners approved the returned results of process optimisation by ODERU in the considered use cases. Final integration testing will be performed during the validation phase of the final CREMA use case demonstrators.