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

WP3 – Architecture, Functional & Technical Specification, Security & Privacy Concept, Integration

D3.7

Software Integration Report II

Deliverable Lead: ASC

Contributing Partners: ASC, DFKI, IKER, TUV, UBI

Delivery Date: 10/2017

Dissemination Level: Public

Version 1.0

This document contains an overview of the CREMA architecture and information on how to deploy the CREMA components. Additionally it provides information about the general CREMA infrastructure and supporting tools. It will also give generic recommendations to other who may perform other similar projects.

Document Status	
Deliverable Lead	Norman Wessel, ASC
Internal Reviewer 1	Simon Osborne, TANet
Internal Reviewer 2	Philipp Waibel, TUV
Type	R
Work Package	WP3: Architecture, Functional & Technical Specification, Security & Privacy Concept, Integration
ID	D3.7: Software Integration Report II
Due Date	31.10.2017
Delivery Date	31.10.2017
Status	For Approval

Note

This deliverable is subject to final acceptance by the European Commission.

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

Technische Universität 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, “Software Integration Report II” (D3.7), is the second of two reports for task “Component Integration and Bug Fixing” (T3.5) providing an overview of the CREMA architecture and information on how to deploy the CREMA components. Additionally, it provides information about the general CREMA infrastructure and supporting tools. It will also give generic recommendations to others who may perform other similar projects.

Table of Contents

List of Figures, Tables and Listings

Figures

Tables

1 Introduction

CREMA – Cloud-based Rapid Elastic MAnufacturing – is a project funded by the Horizon2020 Programme of the European Commission under Grant Agreement No. 637066.

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

This deliverable is the second of two deliverables of task T3.5. Its purpose is to provide an updated overview of the CREMA system, including the information how to deploy the CREMA components. It will also give generic recommendations to other who may perform other similar projects.

1.3 Document Status and Target Audience

This document is listed in the Description of Action (DoA) as "public", since it provides general information about CREMA system and its deployment. Additionally, this document provides generic recommendations to these processes, which can be used by external parties as guidelines for similar projects.

[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 (System Overview): Provides an overview of the CREMA system and its components.
- Section 3 (Source Code Management and Integration): Provides information about the source code management and continuous integration tools, which have been put in place to support the implementation process.
- Section 4 (System Deployment): Provides required information about setting up the CREMA system.
- Section 5 (Server Infrastructure): Provides general information about the server infrastructure.
- Section 6 (Lessons Learned): Provides a collection of useful hints and other recommendations learned in the process of this project.

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

2 System Overview

This section provides an overview of the CREMA system and describes major changes, which have been decided upon since the technical specification (D3.3). Figure 1 shows the current CREMA architecture.

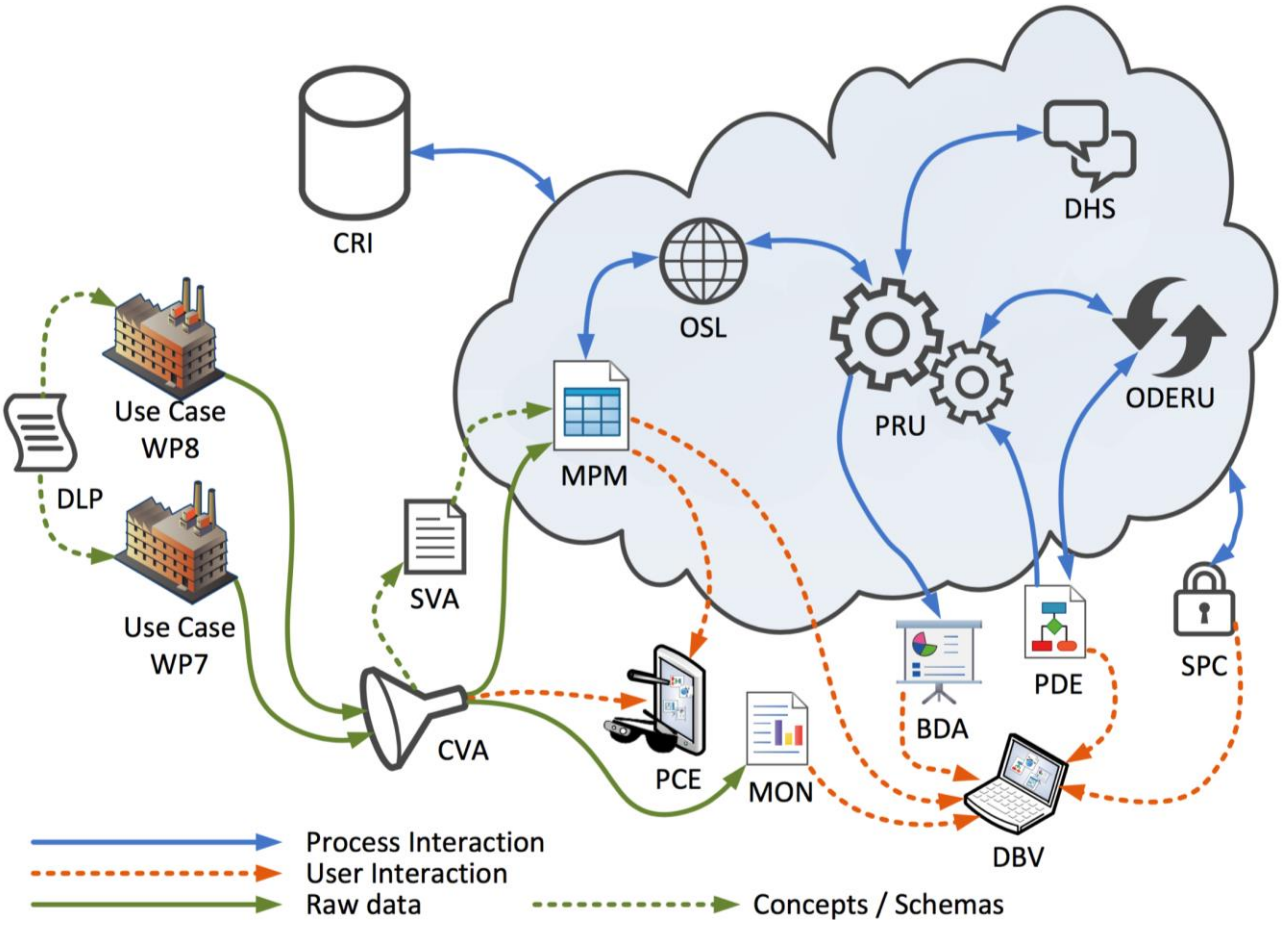

Figure 1: CREMA Architecture

Table 1 provides information about the CREMA components including their acronyms and which task they are connected to.

Table 1: CREMA Components

Acro	Task	Component	Short Name
BDA	T6.3	CREMA Manufacturing Big Data, Knowledge and Analytics	Big Data
CRI	T4.3	CREMA Cloud-based RAID Infrastructure	Cloud Storage
CVA	T4.4	CREMA Cyber-Physical Systems, Sensor Abstraction and Virtualisation	CPS Sensors
DBV	T6.5	CREMA Dashboard and Visualisation	Dashboard
DHS	T4.2	CREMA Data Harmonisation Services	Harmonisation
DLP	T4.1	CREMA Data Model, Model Library and Profiles	Model
MON	T6.2	CREMA Monitoring and Alerting	Monitoring
MPM	T6.1	CREMA Marketplace and Monetisation	Marketplace
ODE	T5.4	CREMA Design Time Optimisation	Optimisation
ORU	T5.5	CREMA Runtime Optimisation	Optimisation
OSL	T5.3	CREMA On-Demand Service Leasing and Releasing	(Re-)Leasing
PCE	T6.4	CREMA Agile Personal Collaboration Environment	Collaboration
PDE	T5.1	CREMA Cloud Collaborative Process Design Time Environment	Designer
PRU	T5.2	CREMA Cloud Process and Messaging Runtime Environment	Process Engine
SPC	T3.4	CREMA Holistic Security and Privacy Concept	Security
SVA	T4.5	CREMA Service Virtualisation and Abstraction	Service Abstraction

Overall, the architecture implementation was executed as envisioned. The only major change was to encourage the use of the lightweight virtualization software Docker[4] not only for the services binaries that are stored in the MPM, but also for the components themselves. Docker will be introduced in Section 4.2.1.1.

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

3 Source Code Management and Integration

This section provides information about the supporting infrastructure and the CREMA API Reference.

3.1 Source Code Management

Source code management (SCM) is the way to store, share and work with the source code generated in the implementation phase of the project. Storing the source code in a secure way is important but also how the source code can be accessed and how changes are tracked is of importance.

In CREMA, the SCM system Git is being used. Git is currently state-of-the-art in this area and combined with GitLab[5], which helps to visualise the different projects/components, it will support the development process of the CREMA components. In addition, Git can easily be integrated in all IDEs used by the partners or be used on the command line by all partners not using IDEs, provides all the necessary features expected by the CREMA technical partners.

3.2 Continuous Integration

Continuous integration (CI) is the process of systematically compiling and deploying software components after changes have been made to the source code. The goal of this process lies in the enhancement of the software quality by running software tests for each build and therefore providing fast feedback to the developer. Changes in connected repositories can trigger builds of the corresponding component and by using integrated software test, which can reveal errors at build time.

In CREMA, the CI platform Jenkins[6] is being used and provides a central overview of all components and their status viewable by all project partners. The mentioned overview will also help to recognize issues at an earlier stage.

3.3 CREMA API Reference

To support the implementation phase, a CREMA API website has been published, which contains helpful information about the different CREMA components and its services. The website is available under http://docs.crema-project.eu/ and is hosted by the partner IKERLAN. The documents are all uploaded to a Git repository, can be directly edited by using the GitLab website if necessary, and a build script creates the documents website via Slate[7].

[5] https://about.gitlab.com/
[6] https://jenkins.io/
[7] https://github.com/lord/slate

4 System Deployment

This section provides information about the requirements, deployment and configuration of the different CREMA components. Specific information about the deployment and configuration of each CREMA component is being provided by the prototype deliverables, so this section links to the different CREMA deliverables, which contains further information.

4.1 Requirements

CREMA components are running on different hardware specifications and therefore require different setups to work. Most CREMA components are available as Docker images, so they provide their own environment and tools; only a running Docker environment is required. A short tutorial on how to set up a Docker environment on a Linux-based system can be found in Section 4.2.1.

The following table provides an overview of server environments, which are required to run all CREMA components.

Table 2: System Environment Table

Server Environment	Component
Docker Engine on Linux-based system	BDA, CRI, DBV, DHS (Server), MON, MPM, ODE, ORU, OSL, PCE, PDE, PRU, SPC, SVA
Windows server (Windows 7+) running Microsoft SQL server	CVA – IndustreWeb Global & Collect Server
Embedded board computer (e.g. Raspberry Pi or Odroid C2) running Debian	CVA – IndustreWeb Display Thin Client
Windows client (Windows 7+) with Java Runtime Environment and Talend	DHS (Client)
Windows or Linux system with .NET Framework 2.0+/Mono installation	CVA – Smart factory Web CVA – Smart factory CPS services CVA – Location engine platform

4.2 Installation and Configuration

The installation process for each CREMA component is being provided in the connected prototype deliverable. Therefore, besides the following information this section only provides a table, which links the components with the connected sections and an installation guide for setting up Docker on Linux.

To make sure, that CREMA components can be set up in different environments, all CREMA components have been fog-enabled by the second prototype version. This enables the component deployment to be more flexible to infrastructure changes by adding configurable endpoints for each used service/component instead of having one point of access for each service.

Table 3: System Installation Table[8]

Component	Deliverable
BDA	T6.3 – D6.6 – CREMA Manufacturing Big data, Knowledge and Analytics – Prototype II
CRI	T4.3 – D4.6 – Cloud-based RAID Storage – Prototype II
CVA	T4.4 – D4.8 – CREMA Cyber-Physical Systems, Sensor Abstraction and Virtualisation – Prototype II
DBV	T6.5 – D6.10 – CREMA Dashboard and Visualisation – Prototype II
DHS	T4.2 – D4.4 – CREMA Data Harmonisation Services – Prototype II
MON	T6.2 – D6.4 – CREMA Monitoring & Alerting Framework – Prototype II
MPM	T6.1 – D6.2 – CREMA Marketplace and Monetisation – Prototype II
ODE	T5.4 – D5.8 – CREMA Design Time Optimisation – Prototype II
ORU	T5.5 – D5.10 – CREMA Runtime Optimisation – Prototype II
OSL	T5.3 – D5.6 – CREMA On-Demand Service Leasing and Releasing – Prototype II
PCE	T6.4 – D6.8 – CREMA Personal Collaboration Environment – Prototype II
PDE	T5.1 – D5.2 – CREMA Cloud Collaborative Process Design Time Environment – Prototype II
PRU	T5.2 – D5.4 – CREMA Cloud Process and Messaging Runtime Environment – Prototype II
SVA	T4.5 – D4.10 – CREMA Service Virtualisation and Abstraction – Prototype II

4.2.1 Docker

The following subsections provide a short introduction of Docker and the required steps to set up the Docker environment.

4.2.1.1 Docker Introduction

Docker is a virtualisation platform, which allows the user to run lightweight containers containing any software component the user would like to run. These containers are wrapping a complete filesystem, which can be customized by the user like any other system. As the containers are not bound to hardware, it makes relocation, testing and later scaling of the software components much easier.

[8] The actual URLs of the services have been removed from this table, as this is a public deliverable. If needed, please contact service@ascora.de.

4.2.1.2 Docker Installation

Most CREMA components are available as a Docker[9] image, so setting up a Docker environment is necessary for setting up a CREMA system. The following commands are required to set up the Docker Engine on Ubuntu 14.04:

1. `sudo apt-get update`
2. `sudo apt-get install apt-transport-https ca-certificates`
3. `sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D`
4. `echo "deb https://apt.dockerproject.org/repo ubuntu-trusty main" | sudo tee /etc/apt/sources.list.d/docker.list`
5. `sudo apt-get update`
6. `sudo apt-cache policy docker-engine`
7. `sudo apt-get install docker-engine=1.12.1-0~trusty`
8. `sudo apt-mark hold docker-engine`

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

5 Server Infrastructure

CREMA uses a service-oriented architecture, which means that all services can be set up anywhere, provided that the URI endpoints are known for all systems. Currently, all partners host their own software, which speeds up development, however alternatively, a setup will be reviewed for running the platform in live production. This is important since latency times must be considered in a REST-based architecture, as each HTTP call might add seconds to a successive step in a call chain.

For instance, the SPC (Security and Privacy Component) will always be called whenever data in the CRI (Cloud-based RAID Infrastructure) will be requested, therefore, ASC has set up the SPC and the CRI on the same server, in order to minimize the added latency.

5.1 Component Instances

The following table provides an overview of running/active instances of CREMA components and infrastructure. As this deliverable is public, no URI can be provided.

Table 4: Overview of CREMA Component Instances

Component	Partner	Description
T4.1 DLP	ICE	DLP instance with capability to add concepts and updated with UC2 concepts
T4.2 DHS	ICE	DHS Services, used by PDE and ICE Data Gateway (Talend Open Studio for Data Integration)
T4.5 SVA	TUV	Service Virtualization and Abstraction
T5.1 PDE	ASC	Process Designer
T5.2 PRU	TUV	Cloud Process and Messaging Runtime Environment
T5.3 OSL	TUV	On-Demand Service Leasing and Releasing
T5.4 ODERU	DFKI	Design Time Optimisation and Runtime Optimisation
T6.1 MPM	ASC	MPM Backend Services
T6.1 MPM	ASC	MPM Frontend (deprecated)
T6.1 MPM	ASC	MPM Frontend
T6.3 BDA	ICE	BDA Frontend
T6.3 BDA	ICE	Manufacturing Big Data Analytics – Create various graphs for the Fagor UC
T6.3 BDA	ICE	BDA Map
T6.4 PCE	ASC	PCE Notification API
T6.5 DBV	ASC	CREMA Dashboard
T6.5 TIKKI	ASC	Tikki Ticketing System
Jenkins	ASC	CREMA Jenkins (Continuous Integration)
GitLab	ASC	CREMA GitLab (Code repositories)

6 Lessons Learned

This chapter will provide a collection of useful hints and other recommendations learned in the process of this project. Most of the recommendations are linked to technologies, which have been used in the daily work.

- **GitLab[10] as source code management system**
  The GitLab project provides a free community edition, which has been used to store the code of the CREMA components. It provides many helpful features but in CREMA the main usage was to manage our code and make it available to our continuous integration system Jenkins. As the development speed of this tool is fast, we used GitLab in a Docker environment, which made it easy to maintain and to update.
- **Jenkins[11] as continuous integration system**
  The open-source project Jenkins was used for creating software artefacts from CREMA components like the PCE Android application. It has also been set up in a Docker environment to be able to deploy new versions in a short time to keep our code secure.
- **ownCloud[12] as document repository**
  The open-source file server ownCloud has been used for managing deliverables and other documents. As software clients are available for free, syncing documents between clients is possible and the server could be hosted by ourselves, it was our first choice.
- **Slate[13] as API documentation**
  The open-source project Slate has been used to document services provided by our CREMA components as it helps to maintain the documentation.
- **Configuration files to support fog environment**
  The CREMA platform consists of different components, which provide its functionalities only in combination with each other. When setting up a new CREMA platform instance, it needs to be able to run under different environments. To make sure that CREMA components can communicate with each other, every component provides a configuration file containing the location of other CREMA components required for its tasks. Having a shared repository for these configuration files would also make it possible to automatically update the configuration in all different components and instances, but this idea was not yet realized in CREMA.
- **Technical leadership and go-to-market approach**
  Having a technical coordinator in a research project seemed to be very useful in the driving of the overall solution. Also, having a strong focus on commercialization and user partners who support this, make a huge difference in the quality of the human interfaces and the overall technical maturity of the developed solutions.

[10] https://about.gitlab.com/
[11] https://jenkins.io/
[12] https://owncloud.org/
[13] https://github.com/lord/slate