



**“Innovative end-to-end management of Dynamic Manufacturing Networks”**

**Deliverable D3.1.2**

**Detailed Design of iMAGiNE Platform Version 2**

**Workpackage:** 3 – Design, Development and System Integration

**Authors:** SAG, INTRA, LOGO, ServTech, REPLY

**Status:** Final

**Date:** 17/12/2013

**Version:** 1.0

**Classification:** Public

**Disclaimer:**

The iMAGiNE IP project is co-funded by the European Commission under the 7<sup>th</sup> Framework Programme. This document reflects only authors' views. EC is not liable for any use that may be done of the information contained therein.

## IMAGINE Project Profile

**Contract No.:** FP7-ICT- 285132

---

<b>Acronym:</b>	IMAGINE
-----------------	---------

<b>Title:</b>	Innovative End-to-end Management of Dynamic Manufacturing Networks
---------------	--

<b>URL:</b>	<a href="http://www.imagine-futurefactory.eu">www.imagine-futurefactory.eu</a>
-------------	--

<b>Start Date:</b>	01/09/2011
--------------------	------------

<b>Duration:</b>	36 MONTHS
------------------	-----------

---

## Document History

Version	Date	Author (Partner)	Remarks
0.1	1/10/2013	SAG	Draft outline for D3.2.1
0.2	17/11/2013	SAG, NISSA	Discussion of EDA
0.3	25/11/2013	SAG, LOGO, INTRA	Including Partner Contributions
0.4	06/12/2013	SAG, INTRA, NISSA, ServTech	Incorporating Partner Contributions
0.5	09/12/2013	SAG, INTRA, LOGO, REPLY	Updates of Partner Contributions
0.6	11/12/2013	SAG, INTRA	Preparing the Document for internal review
0.7	13/12/2013	SAG, ServTech	Incorporating review comments from LAAS and IPA
1.0	16/12/2013	SAG	Finalizing deliverable

## Executive Summary

The detailed design description provides an explicit, in-depth display of all software components along with their interface specifications which will be implemented in the iMAGINE Platform R3. This work is based on the intermediate detailed design v1 and additionally considers as input the Architecture in version 3, the iMAGINE platform implementation R2, and the requirement document of iMAGINE\_enlarged (D8.1), which introduces the production analysis and dynamic monitoring.

In particular, this deliverable addresses a two-fold purpose:

1. It delivers an update of D3.1.1 iMAGINE Detailed Design v1 by aligning design with the implementation of the iMAGINE Platform R2, while addressing changes –imposed by technical and resource limitations - that occurred during the implementation. This can for example be as simple as an update of existing adapters but also addresses new concepts and feature implementations that have taken place since the release of the iMAGINE platform R2. In this sense it is not substituting the existing design but rather complementing the version 1 and providing the technical details like adapter specifications, event definitions, which allow developers to implement the iMAGINE Platform R3.
2. Secondly, this document factors the new concept of the Event-Driven Architecture introduced in the iMAGINE Architecture V3 into the detailed design, and providing a detailed level of understanding by defining event types, event participants, and the flow of events. It further describes components as part of event-driven production analysis and dynamic monitoring as part of the iMAGINE\_enlarged (WP8) that introduce principles like event transformation, pattern detection & - management, reporting mechanisms and means to pro-actively enable on-the-fly production analysis.

Thus, the second version of the detailed design takes all mentioned work from before and refines and re-defines the components, interfaces, database requirements, adapters, and user interfaces by producing a document that will be handed over to the developers for implementation.

The present document deliverable is considered as the base for the implementation of the iMAGINE platform R3 and serves as the reference point for all developing partners to implement the detailed descriptions.

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>11</b>
1.1	Purpose and Scope.....	11
1.2	Structure of the Document .....	12
<b>2</b>	<b>iMAGINE System Architectural Design .....</b>	<b>13</b>
2.1	3 <sup>rd</sup> Version of the Architecture.....	13
2.2	Simplified Architecture v3 .....	13
<b>3</b>	<b>Development Infrastructure and Deployment of i_Platform 15</b>	
3.1	Roadmap .....	15
3.2	Virtualization Clustering .....	17
<b>4</b>	<b>Detailed Description of Components.....</b>	<b>18</b>
4.1	Component Overview .....	18
4.2	Event-Driven Architecture Concept .....	21
4.2.1	Event Types.....	23
4.3	Structural Components .....	24
4.3.1	iMAGINE Enterprise Service Bus .....	24
4.3.1.1	Status Update Possibilities .....	24
4.3.1.2	Implementing Services .....	25
4.3.2	Terminate/Reconfigure DMN Service .....	27
4.3.3	Blueprint Repository System .....	29
4.3.3.1	Blueprint Operations Logging.....	29
4.3.3.2	Blueprint Backup and Restore .....	29
4.3.3.3	Blueprint Cached Read Interface.....	29
4.3.3.4	Blueprint Recursive Instance Fetching .....	31
4.3.4	Production Repository System.....	31
4.4	Administration and On-Boarding .....	35
4.4.1	Portal .....	35
4.4.1.1	Implementation technology .....	35
4.4.1.2	Organizations and Users.....	35
4.5	DMN Analysis and Configuration .....	36
4.5.1	Production Requirements Composer .....	36
4.5.1.1	History .....	36

4.5.1.2	<i>Detailed Logging</i> .....	36
4.5.1.3	<i>Templates</i> .....	36
4.5.1.4	<i>Production Requirements editor</i> .....	36
4.5.1.5	<i>Database Schema</i> .....	37
4.5.1.6	<i>Web Service Interfaces</i> .....	37
4.5.2	Partner Search .....	38
4.5.2.1	<i>Evaluate Search Interface</i> .....	38
4.5.2.2	<i>WEB UI</i> .....	39
4.5.2.3	<i>Search Engine</i> .....	42
4.5.3	Availability Search Engine .....	43
4.5.4	DMN Evaluation and Final Selection.....	44
4.5.4.1	<i>History</i> .....	44
4.5.4.2	<i>Detailed Logging</i> .....	44
4.5.4.3	<i>Reverse DMN Lifecycle Support</i> .....	45
4.5.4.4	<i>Database Schema</i> .....	45
4.5.4.5	<i>Web Service Interfaces</i> .....	45
<b>4.6</b>	<b>DMN Design</b> .....	<b>46</b>
4.6.1	DMN Design Toolset.....	46
4.6.1.1	<i>Production Measurements Editor</i> .....	46
4.6.1.2	<i>Overall Orchestration Editor</i> .....	47
4.6.1.3	<i>History</i> .....	47
4.6.1.4	<i>Detailed Logging</i> .....	47
4.6.1.5	<i>Reverse DMN Lifecycle Support</i> .....	47
4.6.1.6	<i>Database Schema</i> .....	48
4.6.1.7	<i>Web Service Interfaces</i> .....	48
<b>4.7</b>	<b>DMN Execution, Monitoring &amp; Management KIT</b> .....	<b>49</b>
4.7.1	KPI Log and Monitor .....	49
4.7.1.1	<i>Production Metrics and Measurements</i> .....	51
4.7.2	Trouble Shooter .....	53
4.7.2.1	<i>Nagios: extensible architecture</i> .....	53
4.7.2.2	<i>Implementation of the Monitoring System in the iMAGINE Platform</i> .....	57
4.7.3	Dashboard.....	60
<b>4.8</b>	<b>Production Analysis and Dynamic Monitoring Toolset</b> .....	<b>60</b>
4.8.1	Complex event detector.....	61

4.8.2	Pattern Editor for modeling.....	62
4.8.3	Event Storage Component .....	63
4.8.4	IMAGINE-Enlarged Advanced visualization .....	64
<b>5</b>	<b>User Interface Design .....</b>	<b>65</b>
<b>5.1</b>	<b>Production Requirements Composer .....</b>	<b>65</b>
5.1.1.1	Main Screen.....	65
5.1.1.2	History Screen .....	65
5.1.1.3	Detailed Log Screen .....	66
5.1.1.4	Templates Screen .....	67
<b>5.2</b>	<b>Partner Search .....</b>	<b>67</b>
5.2.1	Listing Products .....	67
5.2.2	Listing Requirements.....	68
5.2.3	Dynamic Criteria for Partner Search and List Creation.....	68
5.2.4	Listing Partner Search Results.....	69
<b>5.3</b>	<b>DMN Evaluation and Final Selection .....</b>	<b>70</b>
5.3.1.1	History Screen .....	70
5.3.1.2	Detailed Log Screen .....	70
<b>5.4</b>	<b>DMN Design Toolset.....</b>	<b>71</b>
5.4.1.1	Overall Orchestration View .....	71
5.4.1.2	Production Measurements Editor.....	72
5.4.1.3	History Screen .....	72
5.4.1.4	Detailed Log Screen .....	73
<b>5.5</b>	<b>Dashboard.....</b>	<b>74</b>
<b>5.6</b>	<b>IMAGINE-Enlarged Advanced visualization.....</b>	<b>75</b>
<b>6</b>	<b>Conclusion.....</b>	<b>77</b>
6.1	Summing up.....	77
6.2	Future work and Outlook.....	77
<b>Annex A:</b>	<b>References .....</b>	<b>78</b>
<b>Annex B:</b>	<b>Organization Customization Parameters .....</b>	<b>79</b>
<b>Annex C:</b>	<b>List of Acronyms.....</b>	<b>83</b>

## List of Figures

Figure 1-1: iMAGINE Detailed Design v2 relation to other deliverables .....	11
Figure 2-1: Simplified iMAGINE Architecture v3 .....	13
Figure 3-1: Development Roadmap as of M36.....	16
Figure 3-2: Overview of the iMAGINE virtual infrastructure as M28 .....	17
Figure 4-1: Administration and On-Boarding Components.....	18
Figure 4-2: DMN Analysis and Configuration Components.....	19
Figure 4-3: Design Components .....	20
Figure 4-4: Components and Interfaces for the Execution, Monitoring, and Management .....	21
Figure 4-5: Event Consumer and Producer of the EDA.....	22
Figure 4-6: Definition of Event Types .....	23
Figure 4-7: Possible Status Updates .....	24
Figure 4-8: Reminding Partners to Update Status.....	25
Figure 4-9: Process Chart for Notifying and Reminding Partners.....	26
Figure 4-10: Alerting Managers About Status Change .....	26
Figure 4-11: Update Status Flow Service .....	27
Figure 4-12: Result of an Attempt to a Forbidden Status Update .....	27
Figure 4-13: Components involved in the DMN Termination and Reconfiguration .....	28
Figure 4-14: Production Repository Extension - Production Metrics.....	33
Figure 4-15: Production Repository Database Schema .....	34
Figure 4-16: Message flow of the Availability Search Extension .....	44
Figure 4-17: DMN Evaluation and Final Selection Database Schema .....	45
Figure 4-18: DMN Design Toolset Database Schema .....	48
Figure 4-19: Information flow of the NAGIOS service .....	50
Figure 4-20: Production Repository Extension - Production Metrics.....	53
Figure 4-21: User Interface of the inGraph Service.....	56
Figure 4-22: Information and Data dependencies of the Monitoring Server .....	57
Figure 4-23: Component Overview of the Production Analysis and Dynamic Monitoring Toolset .....	61
Figure 4-24: The CEP Engine with its environment .....	62
Figure 4-25: Pattern and Statistics Components.....	63
Figure 4-26: Event Storage Components .....	64
Figure 5-1: Production Requirements Composer – Main Screen.....	65
Figure 5-2: Production Requirements Composer – History Screen .....	66



Figure 5-3: Production Requirements Composer – Detailed Log Screen .....	66
Figure 5-4: Production Requirements Composer – Templates Screen .....	67
Figure 5-5: Product Lists .....	68
Figure 5-6: Listing Production Requirements .....	68
Figure 5-7: Dynamic Partner Search Criteria .....	69
Figure 5-8: Search Result Lists.....	69
Figure 5-9: DMN Evaluation and Final Selection- History Screen.....	70
Figure 5-10: DMN Evaluation and Final Selection – Detailed Log Screen .....	71
Figure 5-11: DMN Design Toolset – Overall Orchestration View.....	71
Figure 5-12: Design Toolset Production Measurements Editor .....	72
Figure 5-13: DMN Design Toolset - History Screen .....	73
Figure 5-14: DMN Design Toolset – Detailed Log Screen .....	73
Figure 5-15: Extension of the DMN Dashboard.....	74
Figure 5-16: Production Metrics and Measurement.....	75
Figure 5-17: Dynamic Monitoring Dashboard .....	76

## List of Tables

Table 4-1: Status Update Matrix.....	25
Table 4-2: Parameters used for calling the IS upon a Termination/Reconfiguration request .....	28
Table 4-3: Blueprint Repository Cached Read Interface .....	30
Table 4-4: Blueprint Repository Recursive Instance Fetching Interface .....	31
Table 4-5: Parameters of the DMNAccess Table .....	32
Table 4-6: Update of the DMN Status IDs .....	32
Table 4-7: Submit Production Requirements Interface .....	38
Table 4-8: Get Production Requirements Interface .....	38
Table 4-9: Evaluate Selection Parameters .....	39
Table 4-10: Parameters for External Marketplace Search .....	41
Table 4-11: Parameters for Saving Search Criteria .....	41
Table 4-12: Parameters for Saving a DMN and its Results.....	41
Table 4-13: Parameters for Sending a Notification to Partners .....	41
Table 4-14: Parameters for Sending a Notification to DMN Manager .....	42
Table 4-15: Parameters for the DMN Evaluation and Final Selection .....	42
Table 4-16: DMN Evaluation And Final Selection – Submit Shortlists Interface .....	46
Table 4-17: Get Product Blueprint Interface.....	46
Table 4-18: DMN Design Toolset - Submit DMN Configuration Interface .....	48
Table 4-19: Get End To End Blueprint Interface .....	49
Table 4-20: NAGIOS Database Table of iMAGINE_BP_partners.....	51
Table 6-1: Organization Customization Parameters .....	79

# 1 Introduction

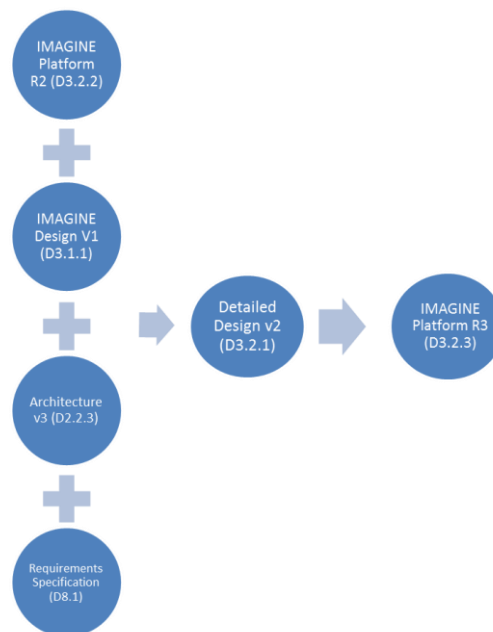
## 1.1 Purpose and Scope

This deliverable shall provide the 2<sup>nd</sup> version of the iMAGINE Detailed Design document giving detailed specifications of individual components, interfaces while considering interoperability of the underlying technology. In this sense the document has a two-fold purpose:

Firstly, it delivers an update of D3.1.1 iMAGINE Detailed Design v1 by concretizing the implementation of the iMAGINE Platform R2 and addressing changes that have taken place. This can for example be as simple as an update of existing adapters but also addresses new concepts and feature implementations that have taken place since the release of the iMAGINE platform R2. In this sense it is not substituting the existing design but rather complementing the version 1 and providing the technical details like adapter specifications, event definitions, which allow developers to implement the iMAGINE Platform R3.

Secondly, this document describes the new concept of the Event-Driven Architecture introduced in the iMAGINE Architecture V3 and provides a detailed level of understanding by defining event types, event participants, and the flow of events. It further describes components as part of event-driven production analysis and dynamic monitoring as part of the iMAGINE\_enlarged (WP8) that introduce principles like event transformation, pattern detection & - management, reporting mechanisms and means to pro-actively enable on-the-fly production analysis.

An overview about the different inputs considered in the Detailed Design v2 document and the output it sketched in Figure 1.



**Figure 1-1: iMAGINE Detailed Design v2 relation to other deliverables**

## 1.2 Structure of the Document

This document is structured in the following way. Section 2 provides a wrap-up of the iMAGINE Architecture V3 by highlighting aspects of production analytics and production monitoring on the basis of complex events. Thereafter Section 3 illustrates general design concepts provides an update to the detailed design v1 and explains a development roadmap and design decisions which influence the setup of the virtual machines. In Section 4 the detailed description of all components is accomplished by providing a description per component on adapters, interfaces, and their interrelation. Section 5 gives an outlook on the graphical user interfaces for software items, enabling the use and control of the i\_platform. This is done for each component which experienced an update as well as for newly introduced components. Design issues and the essential parts of this document are summarized in Section 6, giving an outlook on the development roadmap, component interfaces and relevant considerations used for future work and the implementation of the i\_Platform v3 including the extension of event driven monitoring and dynamic production analysis.

## 2 iMAGINE System Architectural Design

This chapter summarizes the third version of the i\_Platform architecture, and introduces its ramifications for the second version of the design.

### 2.1 3<sup>rd</sup> Version of the Architecture

Objective of the iMAGINE architecture v3 is to achieve accurate, complete manufacturing data and events, visibility in a timely fashion and production-wide consistency according to stipulated production plans, deadlines and performance criteria. In this way, the architecture v3 allows tracking of material movement, consumption, asset and resource utilization, and pull data from various manufacturing sources to transform it into information suitable for analysis to gain “intelligence” to gain DMN visibility and support improved production decisions. It further allows the processing of real-time production data to support DMN managers and factory-based employees in making improved decisions.

### 2.2 Simplified Architecture v3

Based on the complexity of the iMAGINE Architecture, a simplified cornerstone architecture has been devised which sets the foundation for the second version of the design of the i\_platform and the subsequent implementation of the iMAGINE platform that will be realized in the scope of the project.

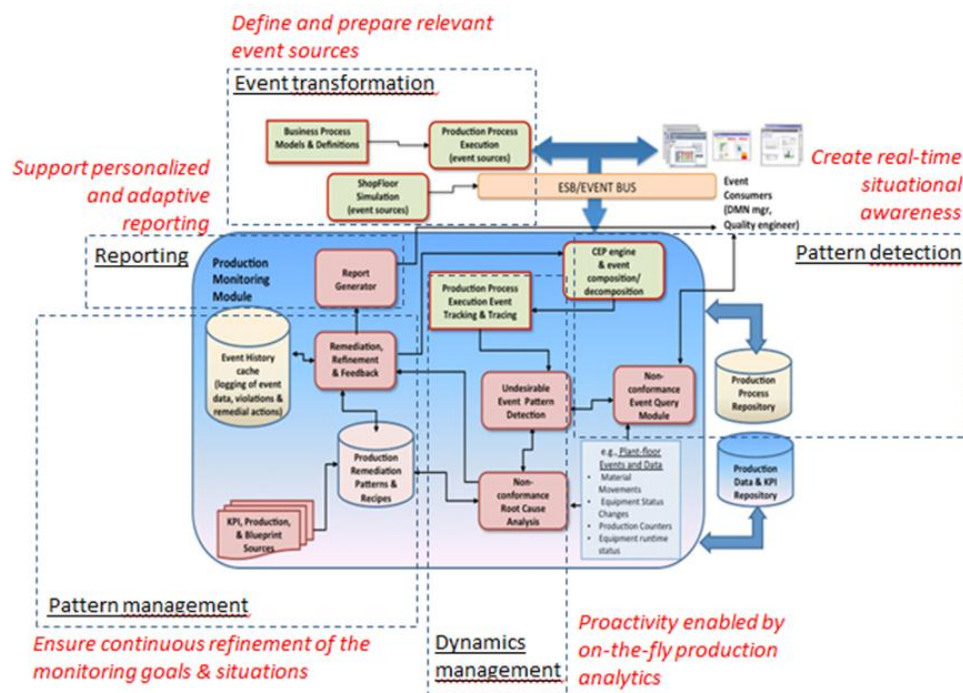


Figure 2-1: Simplified iMAGINE Architecture v3

The main requirement for selecting and highlighting the key subset of the functionalities was to enable end-to-end monitoring, i.e. the complete pipeline but with streamlined functionality:

1. Event transformation

It is related to defining and preparing relevant event sources required for monitoring and production analysis. Definition consists of modeling event types which correspond to each event sources and the preparation is related to the software adaptors that enable the transformation of the signals sensed from event sources in the proper format.

2. Pattern detection

This block enables the detection of the patterns (situation of interests). This is "classical" pattern recognition task, realized by a CEP engine. It results in creating real-time situational awareness (in the given manufacturing context).

3. Pattern management

It is related to ensuring continuous refinement of the monitoring goals & situations. The block is very relevant for dynamic environments, like manufacturing, where plenty of factors can influence business processes (production efficiency and quality). This is an advanced processing block that is based on different methods for extracting knowledge from operative data and the usage data in order to refine the situations that should be detected and monitored.

4. Reporting

This block is related to supporting personalized and adaptive presentation of the results from the Pattern detection process. The block will be specified with more details once the requirements for presenting/delivering information are clarified in use cases.

5. Management of the dynamics

This is the most crucial processing block in the dynamic management process. It is related to sensing the dynamic changes in the input streams in order to detect situations which might be exceptional, unusual, or misleading. Moreover, this block is responsible for detecting the business opportunities as well.

The ramifications for the second version of the design can be summarized as follows. Firstly, the second version of the detailed design has to incorporate the above-mentioned functionalities, revolving around the event-driven architecture that will be factored into the i\_platform. In particular, this has implications for the DMN design, execution and management. The detailed design should define the i\_platform components and their interfaces to allow for designing event-driven end-to-end processes. Also, the second version has to amend the IMAGINE bus with event-broker functionalities, that support the execution of event-driven end-to-end processes in the DMN. Thirdly, from a management perspective, the second version of the design will detail on the logging monitoring of events as well as its dynamic management, including assisting on corrective actions to the DMN given production metrics and measures based on KPIs.

---

## 3 Development Infrastructure and Deployment of i\_Platform

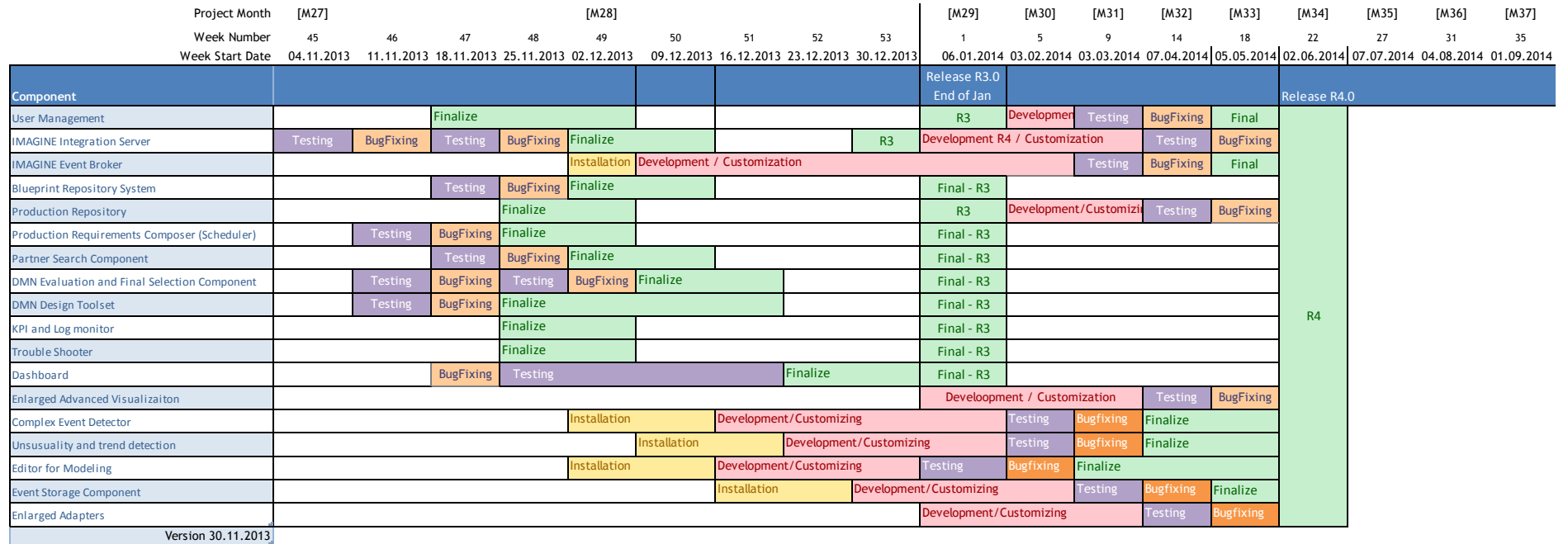
---

This chapter covers the basic technological concepts and design requirements that lead the design decisions for the i\_Platform. As many of the concepts underlying the i\_platform were already introduced in Detailed Design v1 (D3.1.1), we refrain from repeating them but rather provide an update where concepts and planning have changed.

### 3.1 Roadmap

The development roadmap is depicted in Figure 3-1 showing the upcoming development and testing task per component until M34. The majority of components already reached a stable state within the development of Release 2. In the upcoming release 3 only a small update of these components will be provided which contains foremost bug fixing and quick-fixes which also address some feature requests made.

The development until M34 will predominantly involve the implementation of the Event Driven Architecture (EDA) nature which incorporates the IMAGINE enlarged into the existing IMAGINE platform.



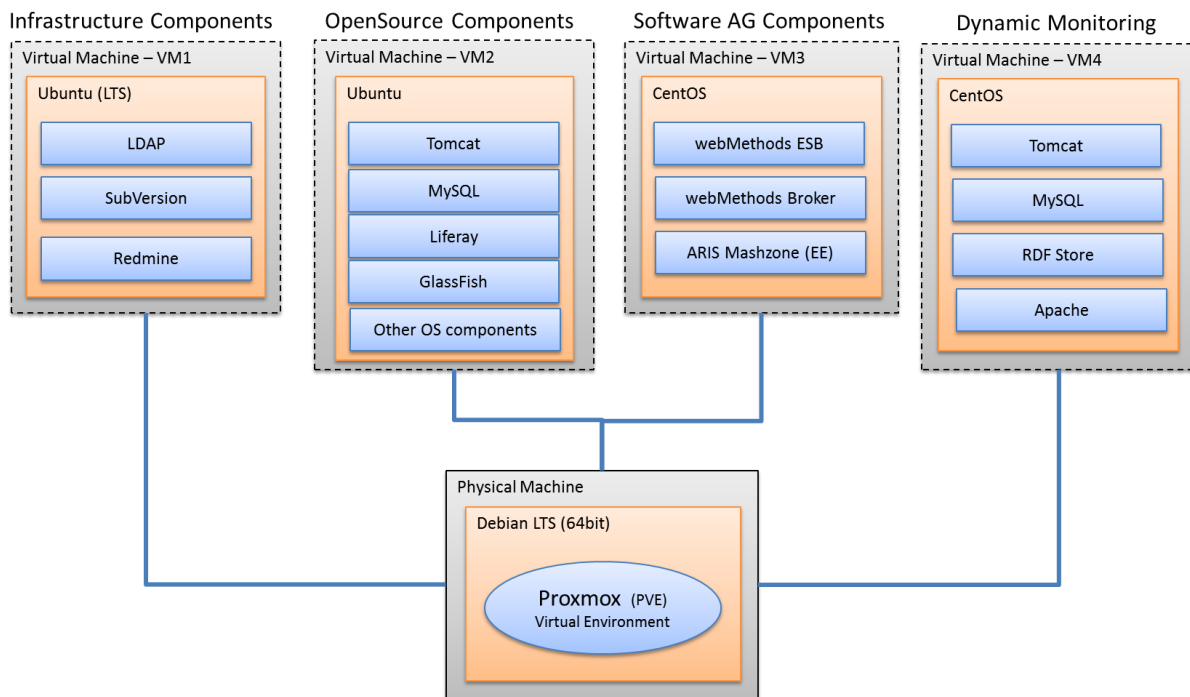
Legend				
Installation	Development/Customizing	Testing	Bugfixing	Finalize

Figure 3-1: Development Roadmap as of M36



## 3.2 Virtualization Clustering

By the use of virtual machines an abstraction layer between the operation system and hardware resources of the computer is created. As the general approach of using virtualization is already introduced in the Detailed Design v1 we refrain from repeating but concentrate on the current deployment which introduces the virtual machine (VM4) for Dynamic Monitoring.



**Figure 3-2: Overview of the iMAGINE virtual infrastructure as M28**

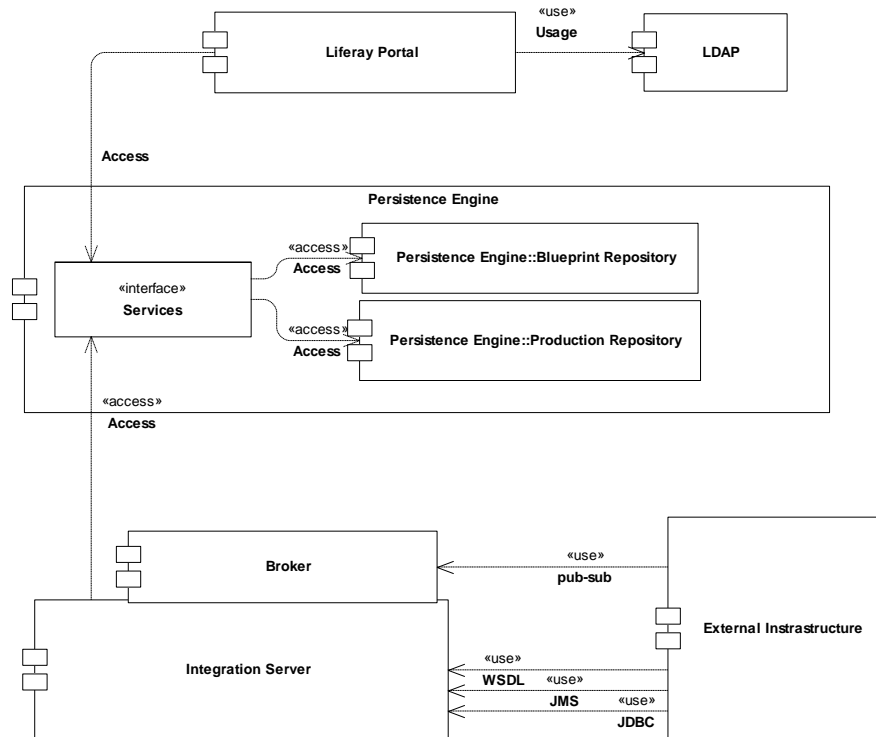
The “Dynamic Monitoring” VM is newly introduced as it provides the bracket for implementing the dynamic pattern management monitoring and event-driven production management, mainly contributed by iMAGINE Enlarged. As the base operation system of this VM CentOS will be used on top a Tomcat Server, Apache Server, MySQL Database along with an RDF store will be deployed and integrated within the existing virtual machines.

## 4 Detailed Description of Components

This section provides a detailed design description of all components within the iMAGINE platform v3. It does this by specifying interfaces, their realization, and exchange formats and also introducing the EDA concept which is mainly integrated within the Design and the Manufacturing and Monitoring phase. In this sense, the following sub-sections are perfectly aligned with the iMAGINE DMN lifecycle and the production oriented flow from the iMAGINE Architecture v3.

### 4.1 Component Overview

The global design is illustrated per DMN phase in the following figures. Figure 4-1 shows the interfacing of the *Administration and Onboarding* phase where mainly the components iMAGINE Liferay Portal, Persistence Engine, Integration Server and External Infrastructures from the Living Labs are participating. By *External Infrastructure* software systems are meant which are hosted by the different iMAGINE Living Lab partners and directly connect to the i\_Platform using services offered by Integration Server and Broker.

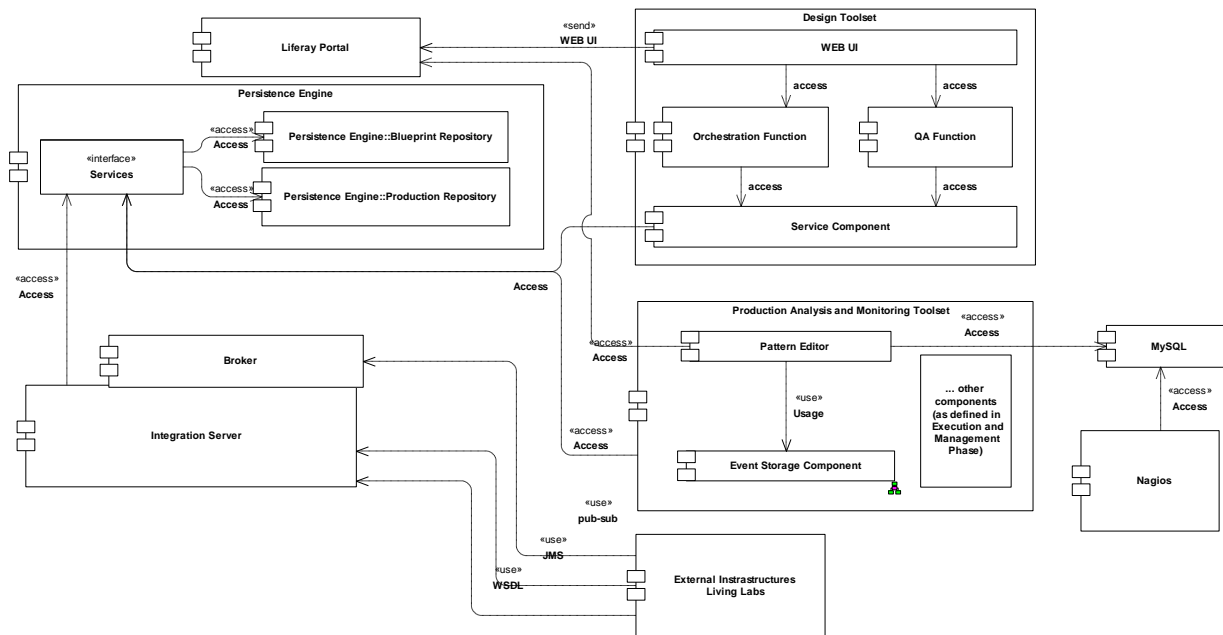


**Figure 4-1: Administration and On-Boarding Components**

Phase 1 in charge of the *DMN Analysis and Configuration* is depicted in Figure 4-2. It describes the detailed component layout of the production requirement composer, the search component, and the DMN evaluation & final configuration component. All access paths are displayed and with respect to the iMAGINE Liferay portal, their integration using the standardized portlet technology.



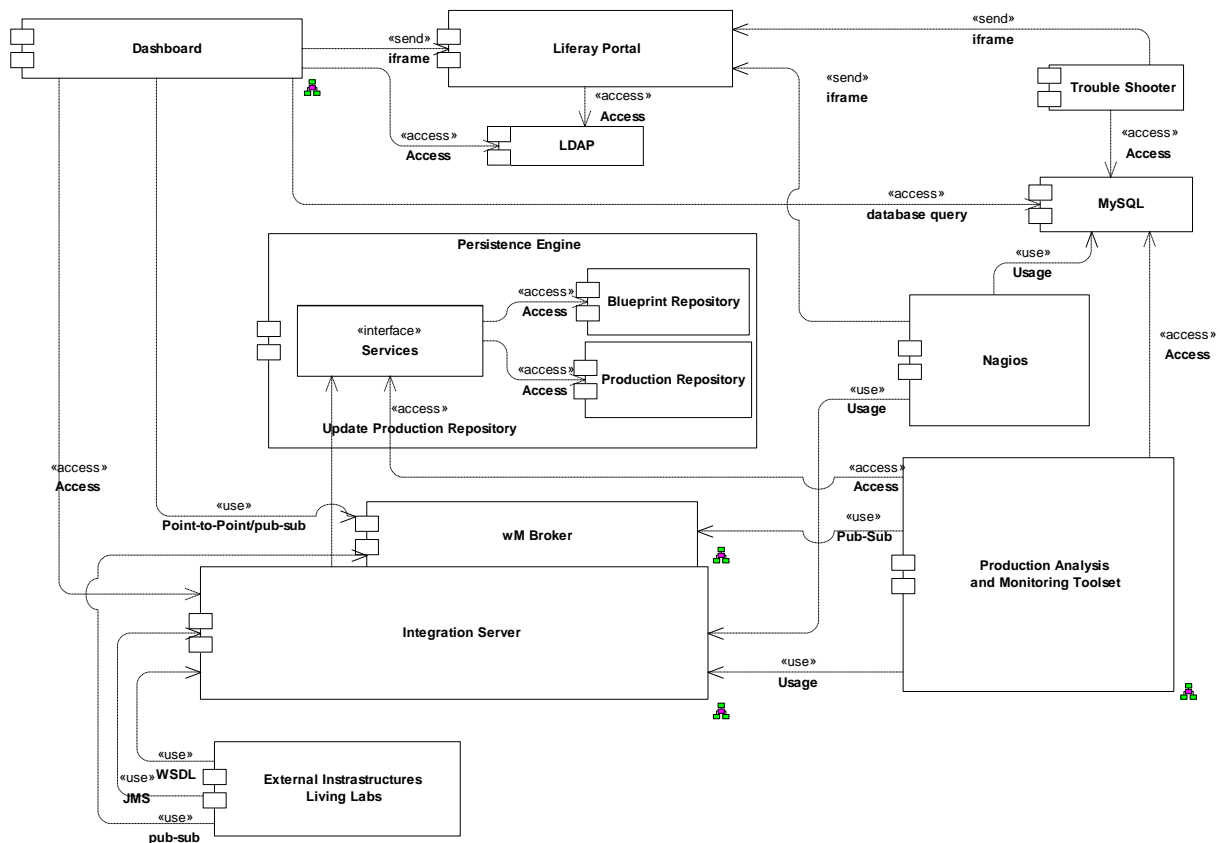
In comparison to version 1 of the detailed design, the production analysis and monitoring toolset is introduced. In particular the use of the Pattern Management Component needed for the modeling the event pattern is required as this stage, giving the DMN manager the opportunity to define specific pattern which will be monitored during the production phase.



**Figure 4-3: Design Components**

A design of the DMN *Execution and Monitoring* phase is depicted in Figure 4-4. In addition to Phase-3 of the DMN lifecycle it introduces Nagios tool along with the Trouble Shooter to accomplish a monitoring functionality of iMAGINE version 3. In addition to the detailed design version 1, the Production Analysis and Monitoring Toolset handling dynamic monitoring is defined which is considered to be part of WP8 (iMAGINE\_enlarged) and introduces concepts for event-driven messaging.

In this figure, the wM Broker and the Production Analysis and Monitoring toolset will be the main players which implement the EDA. This is accomplished by offering different topics on the broker to which components can subscribe, hereby receiving the events.



**Figure 4-4: Components and Interfaces for the Execution, Monitoring, and Management**

## 4.2 Event-Driven Architecture Concept

Event Driven Architecture (EDA) is a methodology that allows you to process the events that shape the business environment. An event can be something as simple as an electrical component being switched on or off, or more complicated, such as a bid being made in an auction house for the painting of a great master. An event represents something that has happened, and it may or may not require some follow-up action to be taken. An event can also represent something that was expected to happen but has failed to happen.

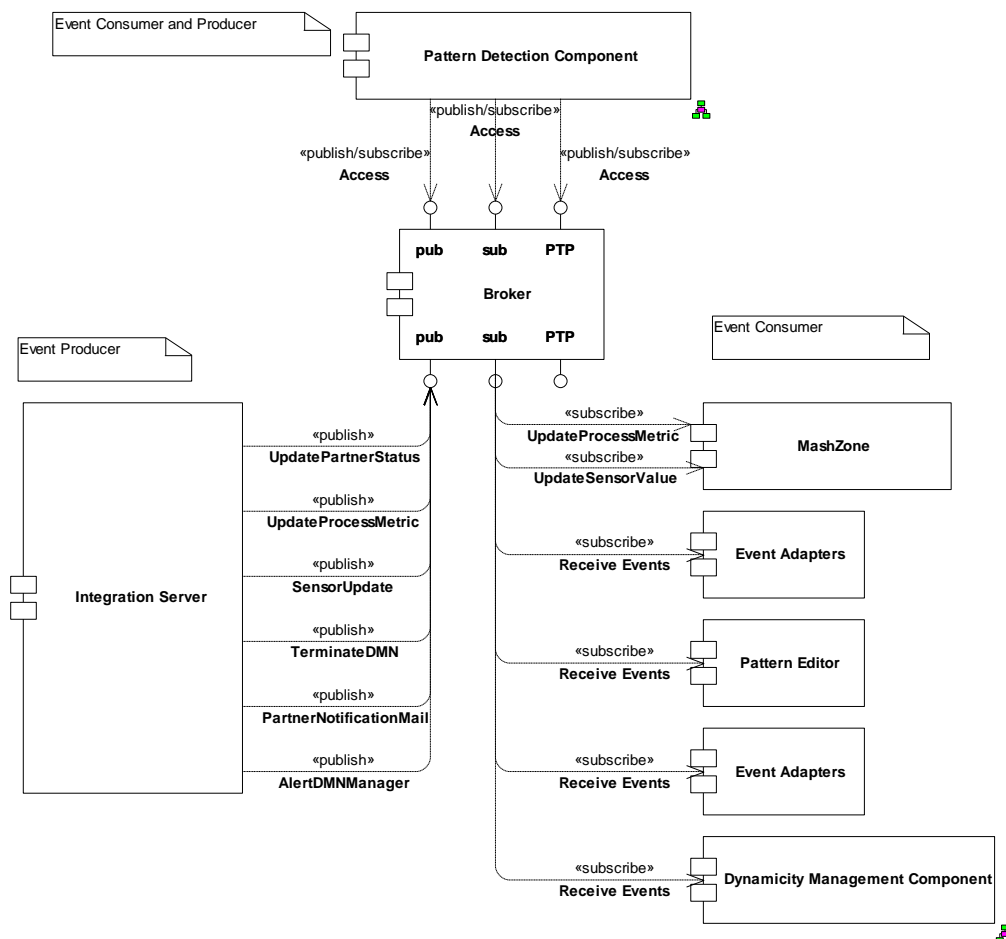
The term “event driven” indicates that when an event happens, it can have a significance which requires some follow-up action to be taken. An event can be noticed by several observers or listeners, and each observer can react to the event differently; for one observer, an event might represent some critical status which requires immediate action, whereas for another observer the same event might not be relevant at all.

The significance of a single event is sometimes only visible when viewed in the context of other events that together form a pattern. For example, if cash is withdrawn at a cash machine in the city center, this is not unusual, but if cash is withdrawn at many different cash machines on the same day throughout the city using the same card, this might raise the suspicion that the card is stolen.

The existence of an event can be the trigger for processes such as the invocation of a service, the initiation of a business process or the publication of relevant information. EDA picks up on these ideas and provides a set of concepts for dealing with events at all stages throughout the processing chain.

The general concept of JMS messaging models is already described in Detailed Design v1. For the sake of completeness we differentiate between Pub-Sub messaging where one publisher produces events consumed by multiple subscribed consumers and Point-to-Point (PTP) messaging where a message is produced and consumed by a single consumer.

In the following a detailed description of events is given, providing a clear overview of the different components participating in the iMAGINE EDA though JMS messages. For better understanding components are clustered according to their functionality into Event Producer, Event Consumer, and components which have both roles like the Pattern Detection Component.



**Figure 4-5: Event Consumer and Producer of the EDA**

The main producer of events is the Integration server and the Pattern detection component. From the perspective of the integration server, a tight integration of the service oriented architecture (SOA) with the EDA is ensured by producing events whenever an existing service adapter (web service interface) is called.

This could be for example an update of the Partner Status by clicking the provided link in the email notification. As in the existent implementation the production repository is updated, the integration server will additionally issue an *UpdatePartnerStatus* event to the webMethods Broker which queues the events and publishes them to subscribed components.

The Pattern Detection Component has a role of an event consumer and provider as it receives the base events from the integration server and provides higher level aggregates which are published and distributed to relevant event consumers.

#### 4.2.1 Event Types

An event type is a schematic definition that describes the structure of events in an event stream. Event Types are considered as first-class objects declared at a high-level in the environment. Events within the same event stream always have the same payload structure. The stream's schema defines which data fields are present in each event, the data type of each field, and the order in which the field appears. Each event stream has exactly one event type associated with it.

One event type can be used as the schema for more than one event stream. All event publishers on a given stream must ensure that their published events comply with the stream's schema, and all subscribers must be aware of the schema that describes the events received.

Lists of events which are defined for the *i\_platform* are depicted in the following figure.

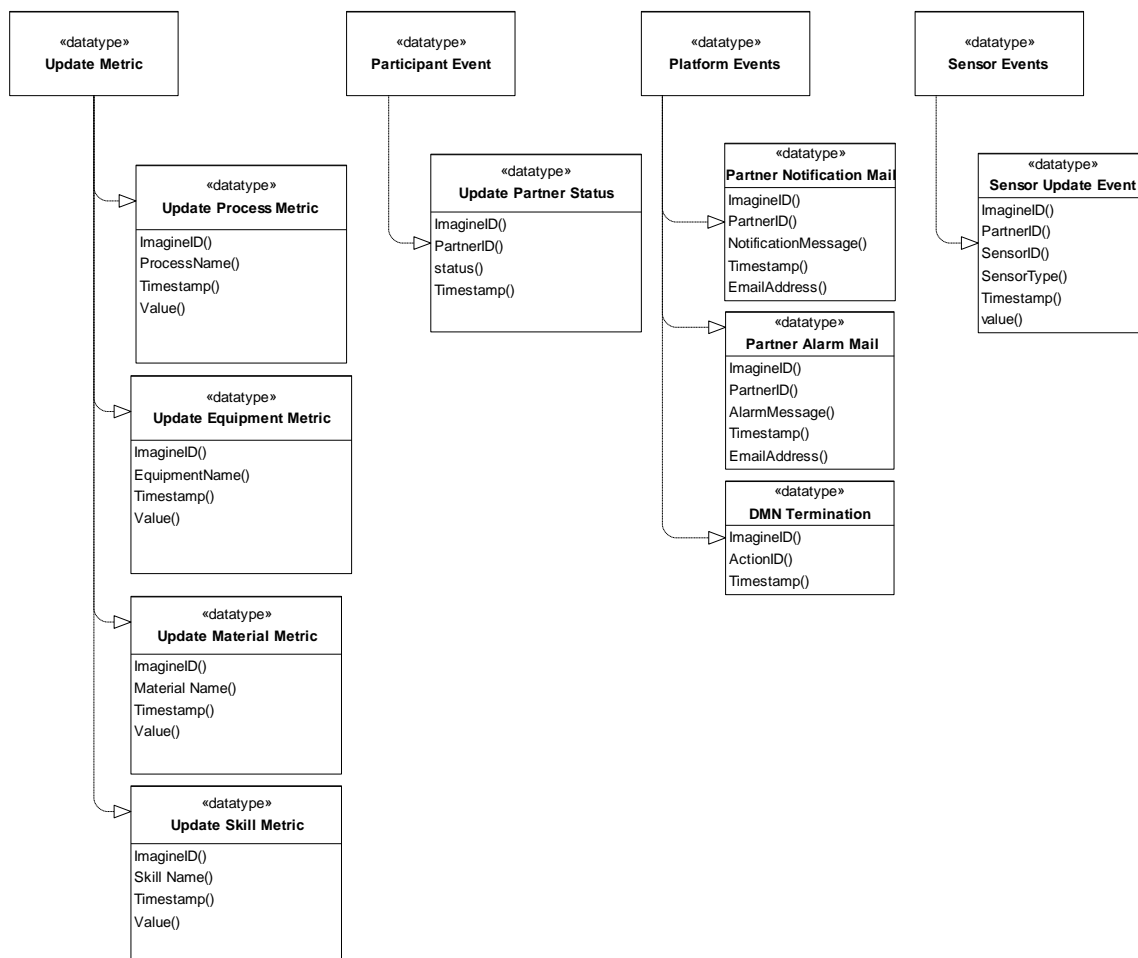


Figure 4-6: Definition of Event Types

## 4.3 Structural Components

### 4.3.1 iMAGINE Enterprise Service Bus

The Integration Server is introduced in detail in the 1<sup>st</sup> version of the detailed design.

After a DMN is started, the partners are notified by an e-mail to start producing their components. In that e-mail they are also asked to update their production status by clicking on the corresponding links. As an example, the status transition from pending to active shows that the production has just started. If a partner does not reply to the notification e-mail in some predefined time (currently two minutes), he/she is reminded to update their status in a reminder e-mail once again.

#### 4.3.1.1 Status Update Possibilities

Not every status update is acceptable. For example a process that is completed should not be canceled afterwards. Moreover, not every possible status update is a sign of desirable progress in the production. Changing the status of a process from active to cancel is an example of such updates. It is favorable that the DMN manager is notified whenever such changes happen during the production. Figure 4-7 depicts a state machine for the status updates of the processes, where the DMN manager is made aware of the unproductive status changes. Forbidden status changes are implicitly represented by deadlocks (i.e., missing transitions to any successor states).

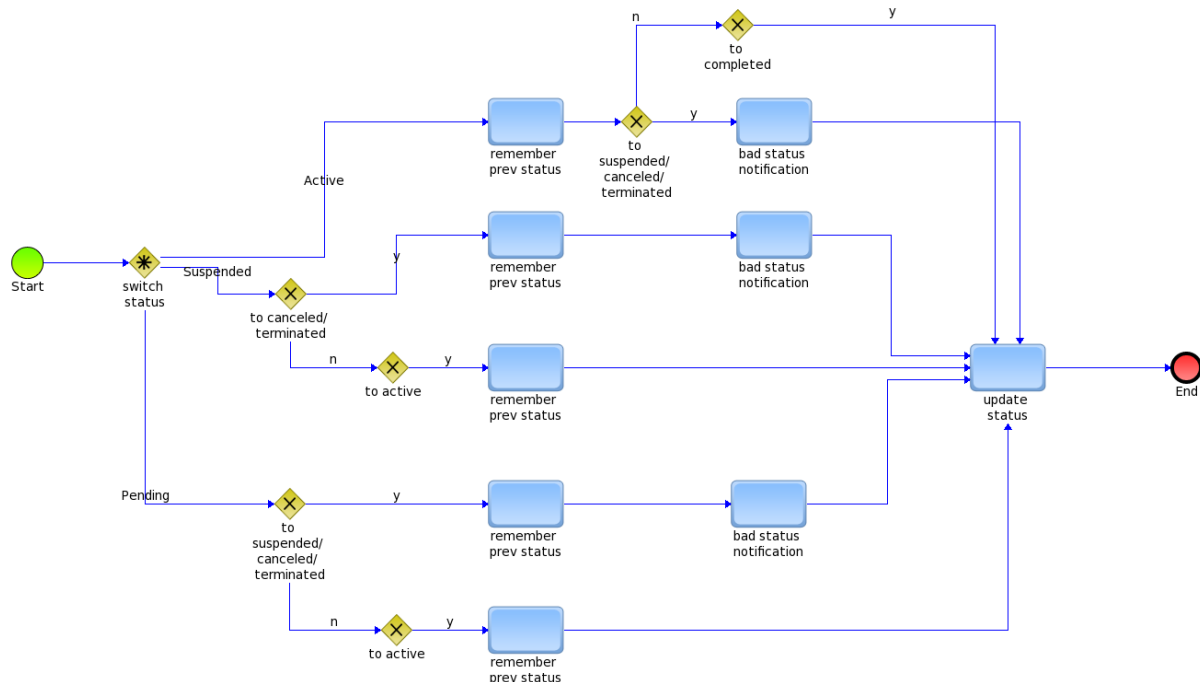


Figure 4-7: Possible Status Updates



Table 4-1 also summarises the status changes in the following manner:

1. Forbidden status changes (e.g., from completed to pending) are shown by the *grey* entries.
2. Status changes from any state to itself (e.g., from active to active) should clearly have no effect, and are shown by the *white* entries.
3. Possible and productive status changes (e.g., from active to completed) are represented by the *green* entries.
4. Possible but unproductive status changes (e.g., from active to suspended) are represented by the *yellow* entries. In such cases, the manager should be informed.

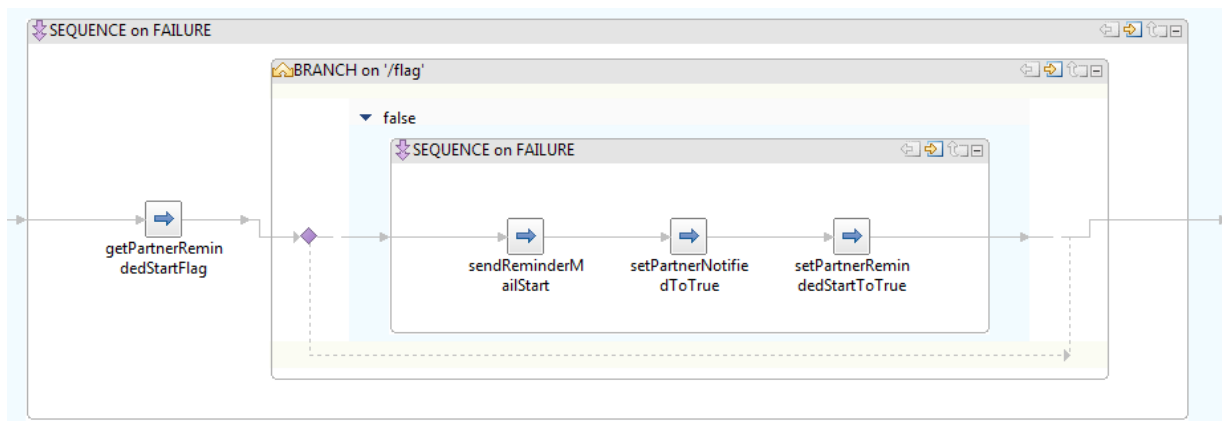
from/to	Completed	Active	Suspended	Cancelled	Pending	Terminated
Completed		-	-	-	-	-
Active	+		+	+	-	+
Suspended	-	+		+	-	+
Cancelled	-	-	-		-	-
Pending	-	+	+	+		+
Terminated	-	-	-	-	-	

**Table 4-1: Status Update Matrix**

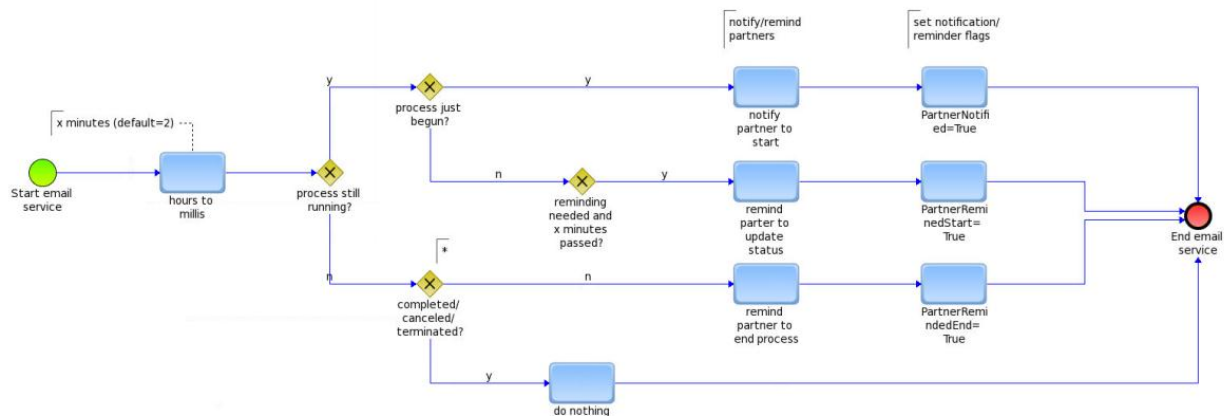
#### 4.3.1.2 Implementing Services

The scenario described above is implemented by three different flow services.

1. ***periodic\_email\_partners\_service***: This flow service is run periodically and is responsible for sending the notification and reminder emails to the partners. Figure 4-8 depicts the part of the flow service, where a partner is reminded to update the process status, if he/she was not reminded before. The complete flow service is rather large-scale. Hence, an abstraction of the entire service is shown in Figure 4-9.

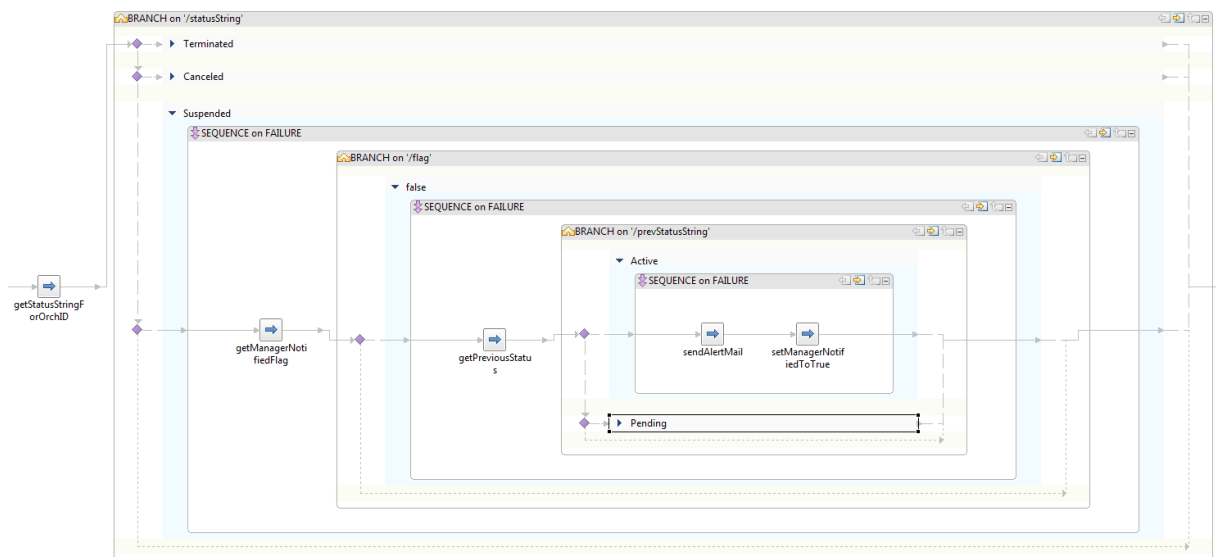


**Figure 4-8: Reminding Partners to Update Status**



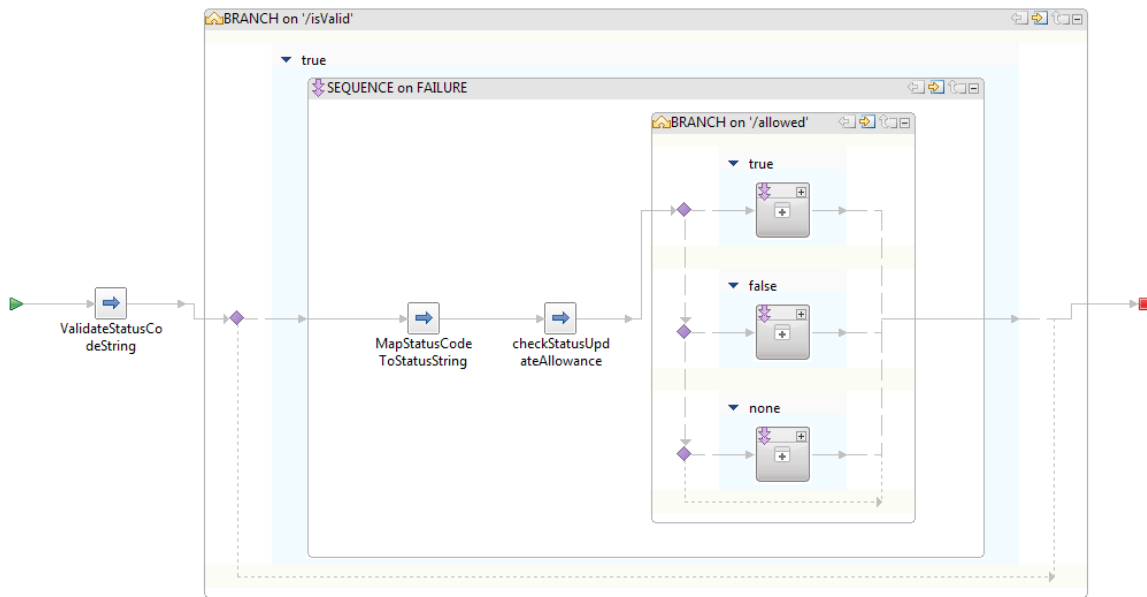
**Figure 4-9: Process Chart for Notifying and Reminding Partners**

2. ***periodic\_alert\_managers\_service***: Every time an unproductive status update is performed, this flow service alerts the corresponding DMN manager. The flow service is partially illustrated in Figure 4-10. It specifically shows the case, where the status of a process is changed from active to suspended, and therefore the DMN manager is notified, if he/she is not alerted already. Other cases are built analogously.



**Figure 4-10: Alerting Managers About Status Change**

3. ***UpdateProcessStatusIfAllowed***: After a partner clicks on a link to update its production status, this service is called, which updates the status if possible according to Table 4-2. The flow service is partially shown in Figure 4-11. Based on the current status and the requested status change it decides whether an update is allowed (case true), forbidden (case false), or it does not have an effect (case none). A status update in the production repository is only performed in the first case. Figure 4-12 depicts the result displayed to a partner, who attempts to perform a forbidden status update from completed to pending.



**Figure 4-11: Update Status Flow Service**

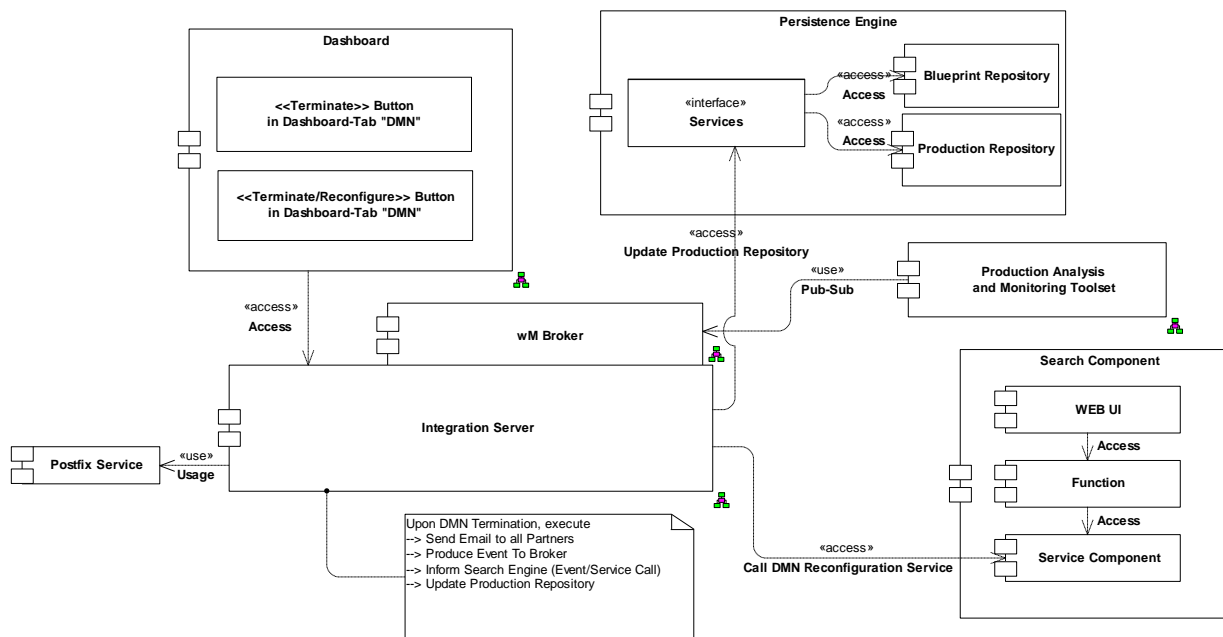
<b>LogMSG</b>	Unsuccessful status update from Completed to Pending.
<b>processID</b>	DemoRepository12.DTC.EquipmentOrchestration.A.060e39a0-9aad-45e3-84b7-a37e9f4934aa
<b>Info</b>	You can close this window!

**Figure 4-12: Result of an Attempt to a Forbidden Status Update**

### 4.3.2 Terminate/Reconfigure DMN Service

The Termination and Reconfiguration Service gives the DMN Manager the possibility to react on issues which prohibit the further execution of the dynamic manufacturing network. Scenarios include the drop-out of an existing DMN producer due to internal problems in the production segment but can also as broad as to react on external conditions by terminating the whole DMN, stopping currently active as well as future processes from being manufactured.

The ability to terminate and reconfigure executed DMNs is introduced in the following design (cf. Figure 4-13) where the termination is triggered the DMN manager from the dashboard by pressing either the termination or reconfiguration button.



**Figure 4-13: Components involved in the DMN Termination and Reconfiguration**

This button-event triggers call on a service adapter implemented on the integration server and passes the ID of the network which needs to be terminated/reconfigured and an action (either termination or reconfiguration). In the Dashboard, the following parameters are used for notifying the integration server:

Name	Type	Description
DMNID	String	the ImagineID of the user that sends the preliminary schedule
ActionID	Int	Indicating 0 for DMN termination and 1 for DMN reconfiguration

**Table 4-2: Parameters used for calling the IS upon a Termination/Reconfiguration request**

Upon the reception of the service call, the following the following actions are sequentially executed on the integration server:

1. An email is sent to all partners participating in the DMN, demanding to cancel the current as well as future production as the DMN is terminated.
2. An event (Event Type: *TerminateDMN*) is issued to the broker and distributed to subscribed clients. This event will carry the *ImagineID* of the DMN network, the *ActionID* describing a termination or reconfiguration, and a timestamp.
3. In case a reconfiguration, the service component is notified that the current DMN execution is terminated but the partner list should be used for another short-listing within the partner search component. In case of a DMN termination, this step is skipped.
4. The production repository is updated accordingly by setting either the "termination" or the "reconfiguration" flag on the considered DMN

A design of the user interfaces for the Dashboard component is proposed in Figure 5-15.

### 4.3.3 Blueprint Repository System

The Blueprint Repository System design and implementation will be enriched with some additional functionality that will enable detailed logging and tracing of operations, backing up and restoring of repositories as well as support for high performance, cached queries and additional functionality that facilitates integration.

#### 4.3.3.1 *Blueprint Operations Logging*

The Blueprint repository will allow the detailed logging of operations for specific repositories. All actions will be recorded in detail when this feature is enabled. This functionality will allow the gathering of valuable information during customization. The logging functionality will be enabled or disabled per blueprint repository id, giving fine grained control on the administrators regarding the monitoring and the allocation of system resources. This configuration option allows the disabling of the logging functionality when not needed in order to save computational power resources and memory.

#### 4.3.3.2 *Blueprint Backup and Restore*

In particular additional interfaces will be added that facilitate user operations. In particular the following interfaces will be added:

- Back Up Repository

This functionality will enable the backing up of the current state of a repository. The current snapshot of the repository will be persisted for reference and it could be restored if needed.

- Restore Repository

This interface will allow the replacement of a repository with a backup version that has been saved. The repository that will be restored will be automatically backed up before it is overwritten by the backed-up version.

#### 4.3.3.3 *Blueprint Cached Read Interface*

The blueprint repository enables the reading and searching of the blueprints by the execution of cached queries specified in the SPARQL language. In IMAGINE Platform R2 the need to create a functionality that will improve performance of specific frequently used queries emerged. In order to address that need and speed up the performance of specific queries the Blueprint repository will be able to support the caching of frequently used query results. In particular when executing a "cached query", the results will be cached using a MRU (Most Recently Used) caching algorithm in conjunction with "dirty cache" indicators. This interface provides very low response times for frequently used SPARQL queries. The aim of the caching interface is to provide a high performance end point for queries that are often required by IMAGINE DMN Lifecycle support components. For this reason the execution of cached queries will only be available for components of the i\_platform and not for external systems (e.g. Supplier systems). Caching algorithms for external systems could be applied inside the adaptors if needed and when appropriate. However the interface will also be available for customized components of the customized IMAGINE Platform versions. A strict requirement for cached queries is that the query string should be identical in every call in order to see performance benefits. Indicative usages of cached queries would be queries such as the queries for all available categories of a blueprint repository or the querying of all available units.

iMAGINE Blueprint repository ensures that only users with appropriate roles and permissions are allowed to query a repository.

Name	Type	Description
Username	String	A valid username
Password	String	A valid password that corresponds to the given username
RepositoryID	String	The unique ID of the repository that would be queried
SPARQLQuery	String	A Query for the blueprint which should be in accordance with the SPARQL Language specification in order to query data that follows the implemented iMAGINE Blueprint RDF Schema definition

**Table 4-3: Blueprint Repository Cached Read Interface**

Calling this interface provides results in XML format, following the published specification "SPARQL Query Results XML Format" [2].

Access to a repository is granted only if the following three preconditions are found to be true:

1. The provided credentials, namely username and password parameters, identify a registered user.
2. RepositoryID refers to a valid, existing repository.
3. The identified user has read access to the repository specified by the repositoryID.

When the three aforementioned preconditions have been verified the cached query specified by the SPARQL query input is executed in the repository. The supplied query needs to comply with the W3C SPARQL Query Language for RDF as described in [4]. An example call would be the following:

```
blueprintCachedRead(User,UserPassword,RepositoryId," PREFIX kb:
<http://imagine.linkedblueprint.com/kb#> SELECT ?unit ?id
WHERE { ?unit kb:hasImagineID ?id . } ;")
```

The result of this sample call would be all the companies and their *ImagineIds* in the repository *RepositoryID*, provided that *User/UserPassword* are valid credentials.

The protocol by the iMAGINE Blueprint repository is:

1. Determine if the username and password provided are valid.
2. Determine if the specified repository exists.
3. Determine if the user specified by username and password has permission to read the specified repository.
4. If the username and password are valid, repository exists , the user has a role that gives him read permission on the repository and the SPARQL Query is valid:
  - a. Check if this query has been cached in the Blueprint Repository
  - b. If it is cached do not execute the SPARQL query on the repository.

- c. If it is not cached, then execute the SPARQL query on the repository, cache the result.
  - d. Get the results.
  - e. Return the results to the user.
5. If the username and password are not valid or repository does not exists or user has not a role that gives him read permission on the repository or the query is not valid:
  - a. Return null;

#### 4.3.3.4 Blueprint Recursive Instance Fetching

Every blueprint class instance has an ImagineID that uniquely identifies it. The blueprint repository will provide an interface that allows the fetching of any class instance by specifying its *ImagineID*. All linked resources and attributes will also be recursively loaded in order to construct and return a complete RDF that contains the requested information.

iMAGINE Blueprint repository ensures that only users with appropriate roles and permissions are allowed to recursive fetch instances from the repository.

Name	Type	Description
Username	String	A valid username
Password	String	A valid password that corresponds to the given username
RepositoryID	String	The unique ID of the repository that contains the specified instance.
ImagineID	String	The Imagine ID of the class instance to be returned.

**Table 4-4: Blueprint Repository Recursive Instance Fetching Interface**

Calling this interface provides results in RDF format.

Access to a repository is granted only if the following three preconditions are found to be true:

1. The provided credentials, namely username and password parameters, identify a registered user.
2. RepositoryID refers to a valid, existing repository.
3. The identified user has read access to the repository specified by the repositoryID.

#### 4.3.4 Production Repository System

##### DMN Access

The production repository is extended by a new view called DMNAccess, which includes the DMN IDs and their managers' e-mail addresses. This information is currently used to alert the DMN managers about undesirable status updates explained in Section 4.2.1. The e-mail addresses will be used in the future to implement the multi-user service (see Section 4.6.3).

Fields of the DMNAccess View are:

Column Name	Remark
End2EndBlueprintImagineID	The ID of the iMAGINE End2End Process
DMNManagerEmail	The Email address of the DMN manager

**Table 4-5: Parameters of the DMNAccess Table**

### DMN Termination/Reconfiguration

To indicate the termination and the reconfiguration a running DMN network, an additional status is introduced to indicate this DMN status. The updated status table looks as follows:

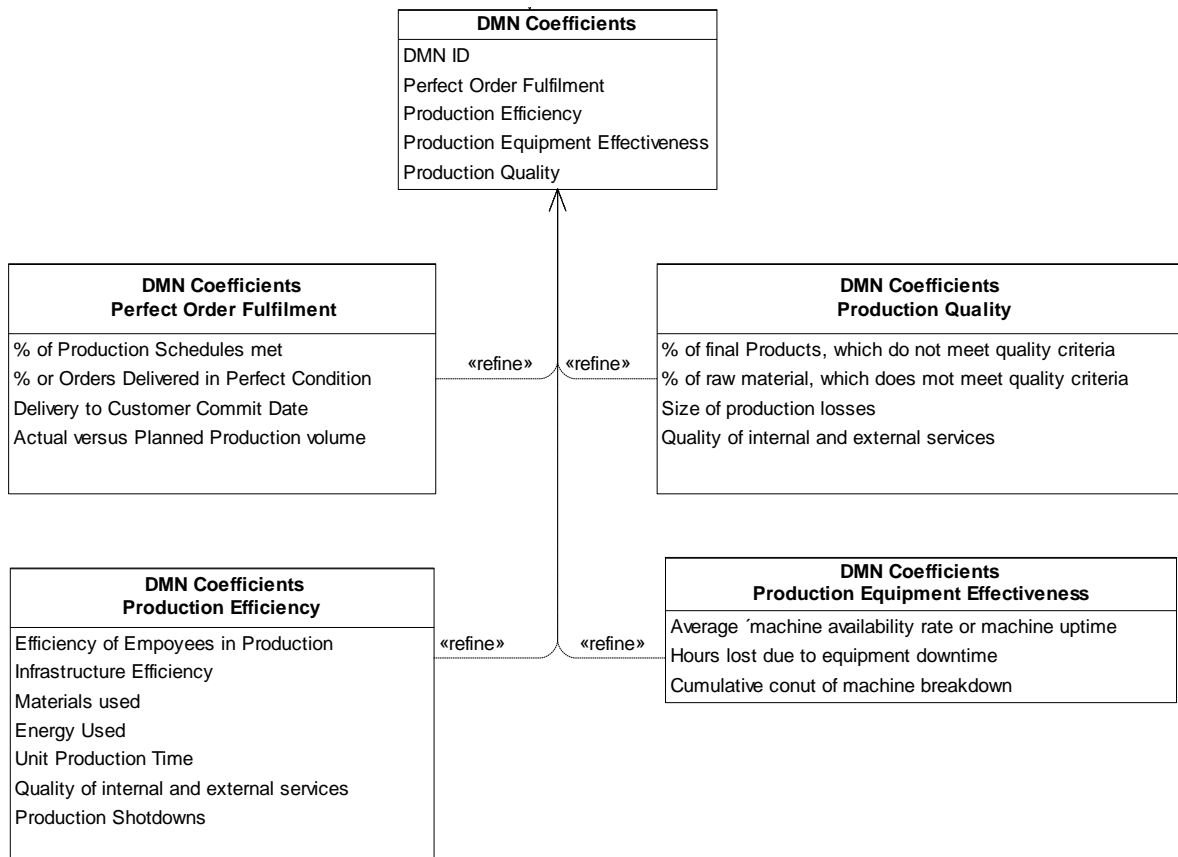
Status ID	Status Name
0	Completed
1	Active
2	Suspended
3	Cancelled
4	Pending
5	Terminated
6	Reconfiguration

**Table 4-6: Update of the DMN Status IDs**

### Production Metrics and Measurement Coefficients

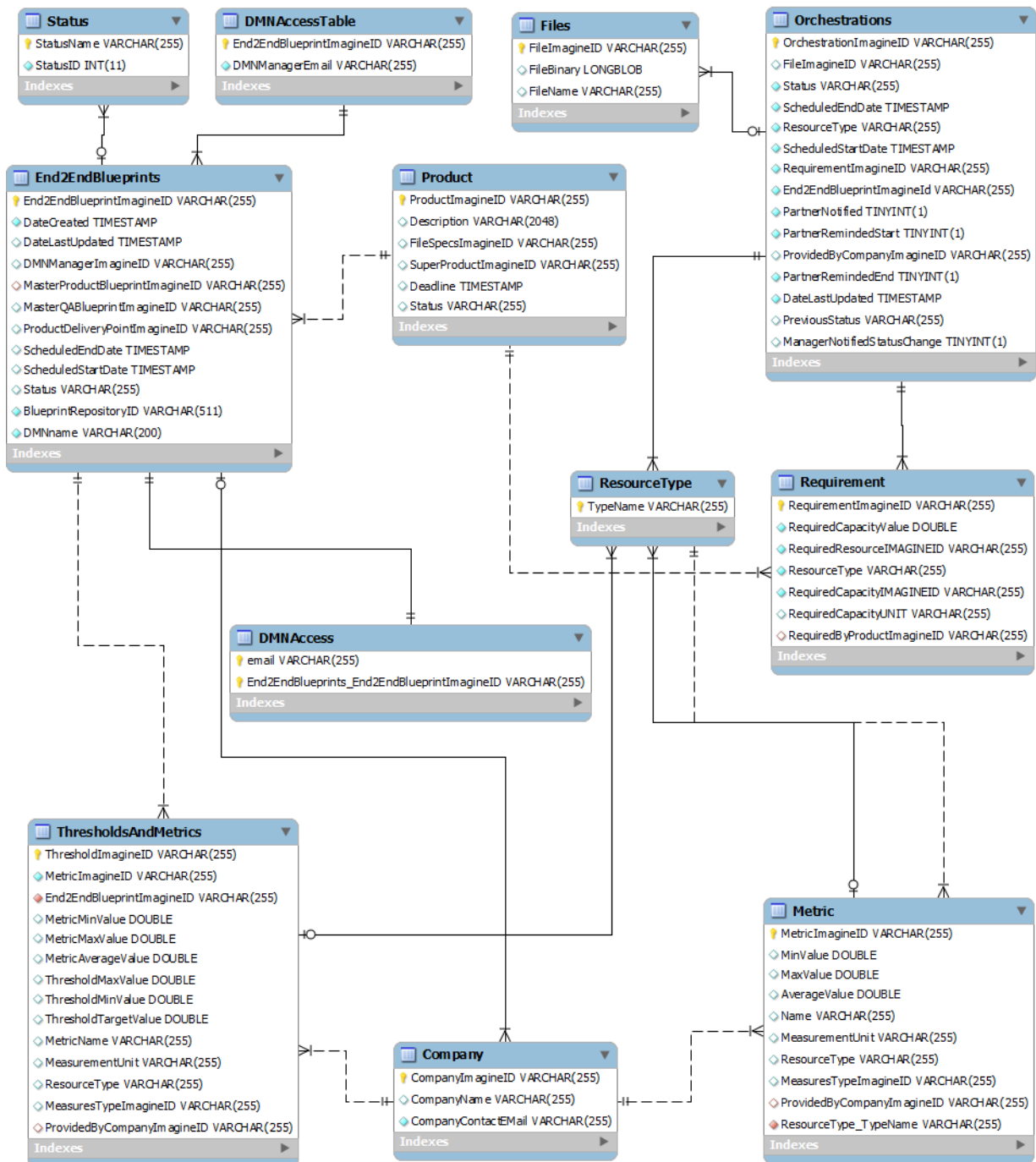
The Production Measures are implemented by a coefficient table which gives a DMN manager the option to specify for each DMN appropriate parameters which we intend to monitor. The Production metric is sketched in the Architecture v3 and relevant figures are displayed below. The Coefficient Table will contain a definition of coefficients for each designed DMN with the following structure:





**Figure 4-14: Production Repository Extension - Production Metrics**

A complete overview of database schema currently deployed in the Production Repository is depicted in the following figure:



**Figure 4-15: Production Repository Database Schema**

## 4.4 Administration and On-Boarding

### 4.4.1 Portal

The Portal usage is extended in order to be able to integrate with the iMAGINE Lifecycle support components. In particular the iMAGINE Platform R3 will be able to support multiple companies and users that use the same installed platform instance to access completely segregated iMAGINE blueprint repository instances.

#### 4.4.1.1 *Implementation technology*

iMAGINE portal is implemented on top of Liferay Portal, CE, 6.1.1 GA2, which will be deployed in Tomcat Server. The original choice of Glassfish was replaced by Apache Tomcat Server because some issues have been noticed in the hosting of Liferay Portal in Glassfish. However the developed portlets will also be compatible with Glassfish hosted Liferay Portal installations. Glassfish will be used for the hosting of DMN Lifecycle components that do not use Liferay Portal, in accordance to D3.1.1.

#### 4.4.1.2 *Organizations and Users*

iMAGINE Platform R3 will be able to support several different and segregated instances of Blueprint Repositories. The administrator of iMAGINE Platform can create, configure and assign users to different, segregated iMAGINE Blueprint Repositories. These repositories are in fact distinct DMN Market Places where Suppliers and DMN Manager may create and deploy DMNs. In terms of the portal these Market Places are organized by using the Liferay technical term of "Organizations". An organization in iMAGINE Portal represents the logical structure of the company or institution where the portal is being used. Each user can be assigned to at most one organization inheriting the permissions and associations of that organization. The name organization was chosen because it's agnostic and matches well with most real world uses. It can be used for industry market places, departments, groups, divisions, partner companies, providers, etc. These all potential uses can support the multiple potential applications of the iMAGINE DMN Management Methodology.

An application scenario of this feature in terms of iMAGINE is that each Living Lab is regarded as an Organization. So in the centrally hosted iMAGINE Platform five completely isolated instances of the generic iMAGINE Platform can coexist, without interfering with each other. The flexibility of iMAGINE Portal, that is implemented by utilizing Liferay portal technology allows multiple other potential scenarios. The aforementioned scenario will be applied for testing and development to the centrally hosted iMAGINE Platform R3 in order allow the interaction with the latest version of the iMAGINE Platform.

A distinct instance for each every iMAGINE Platform R3 components can be used by every Organization. An important prerequisite is that each user should only belong to one organization. The segregation of components can be logical, which means that the components reside in the same physical server or even physical, where the components are completely isolated in different physical machines. In order to configure the Organizations created inside the generic platform, the parameters of Annex B need to be configured for every Organization.

It is important to notice that these values should contain the values that are used to access the components internally from the iMAGINE Platform. Network administrators could also add additional security to the installed iMAGINE Platform instances by restricting access to the addresses that are

used internally by the iMAGINE Platform from external usage. External systems should only be able to access iMAGINE Platform R3 web services only via the iMAGINE Integration Server.

## 4.5 DMN Analysis and Configuration

### 4.5.1 Production Requirements Composer

The Production Requirements Composer will be updated to enable the reuse of submitted production requirements thus providing increased productivity and minimizing the effort needed for creating DMNs for similar products as well the time needed to setup a new DMN in order to manufacture them. In addition the functionality that is required to customize the behaviour of the Production Requirements Composer will be added. The generic design of the user related parameters that need to be taken into account for this reason are detailed explained in section 4.3.3.

#### 4.5.1.1 *History*

The History functionality of the Production Requirements Composer supports the day to day operation of the iMAGINE Platform by the DMN Manager, making available information about all the latest actions performed by the DMN Manager. Entries no longer required in the History can be deleted by the DMN Manager. The design of the History GUI of the Production Requirements Composer is provided in section 5.1.1.2.

#### 4.5.1.2 *Detailed Logging*

The Production Requirements Composer will log all created, submitted and deleted production requirements in a dedicated database schema. The logging will provide transparency to Production Requirements related actions performed in every company as well as detailed reporting in case it is needed. The major difference between the logging and the history information is that records cannot be deleted from the user interface of the logging mechanism; rather they are permanently stored while the history functionality is mostly used to allow the DMN Manager to keep up with the daily operation of the Production Composer. For this reason entries can be deleted from the history if needed. The design of the GUI that supports the logging mechanism of the Production Requirements Composer is provided in section 5.1.1.3.

#### 4.5.1.3 *Templates*

In order to increase productivity, the Production Requirements Composer will support the creation, editing and submission of production requirements based on templates. The DMN Manager will be able to save a production requirement as a template that can be reused multiple times. The design of the GUI Screen that supports the usage of templates is provided in sections 5.1.1.1 and 5.1.1.4.

#### 4.5.1.4 *Production Requirements editor*

The Production Requirements Composer provides four different editors for production resources. In particular it provides the Material, Equipment, Skill and Process front ends that are dedicated in editing specific parts of the Production Requirements. These interfaces are enhanced with the following functionality in order to better support the customisation of the platform, by allowing the dynamic selection of customized units, categories and other types of customized instances via the GUI, as well as to ensure the conformance of the created Production Requirements with respective blueprint schema that has been defined in detail in Deliverable D3.1.1. The Production Requirements

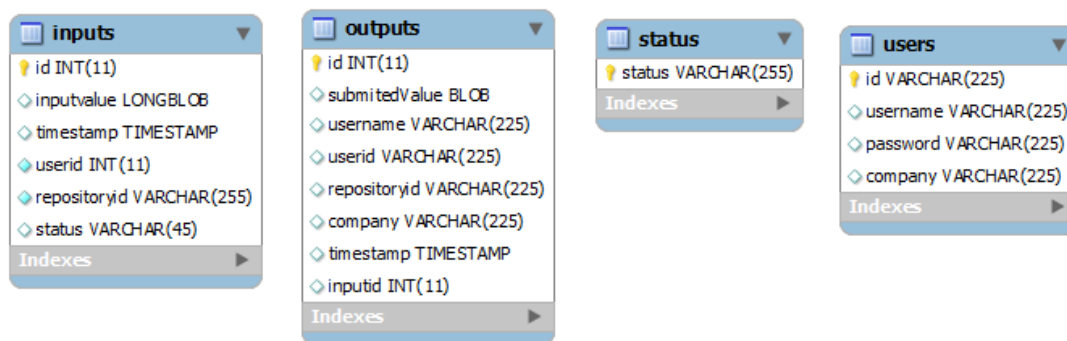
Composer will enforce the compliance with the Production Requirements by preventing the DMN Manager to submit any Production Requirements that are not fully compliant with the Production Requirements Blueprint and providing appropriate hints that would help the DMN Manager.

#### 4.5.1.4.1 Customized Instances Selection

The Production Requirements Composer editors allow the DMN Manager to select among the Units, Categories, Certifications, Skills and Attributes that are available in the Blueprint Repository. These instances that are made available for selection in the editors are the distinct instances that should have been already been made available by the Administrator of the iMAGINE Platform and are used by all the submitted Partner Participant Blueprints.

#### 4.5.1.5 Database Schema

In order to support the aforementioned functionality, the Production requirements composer will also be supported with the database schema that is depicted in Figure 4-10: Production Requirements Composer Database Schema. This database contains information that is used for the Production Requirements Composer functionality.



**Figure 4-10: Production Requirements Composer Database Schema**

#### 4.5.1.6 Web Service Interfaces

##### 4.5.1.6.1 Submit Production Requirements

Production Requirements Composer provides a WS interface that allows other components to signal that a Production Requirements Blueprint has been inserted in the Blueprint Repository and is pending for editing by the Production Requirements User Interface. In case the indicated Production Requirements has already been submitted to the Partner Search component in the past, it is opened for editing and resubmitting.

This interface is called by the Partner Search, the DMN Evaluation and Final Selection Component, and the Design Toolset in order to signal that the indicated Production Requirements will be changed by the DMN Manager. In a customized version of the i\_platform this can also be made available for usage by external systems via the iMAGINE Integration Server.

Name	Type	Description
Username	String	A valid username
Password	String	A valid password that corresponds to the given username
RepositoryID	String	The repositoryID of the iMAGINE blueprint repository that contains the Production Requirements Blueprint.
ProductionRequirementsDMNID	String	The iMAGINE Id of the Production Requirements instance that should be edited.

**Table 4-7: Submit Production Requirements Interface**

#### 4.5.1.6.2 Get Production Requirements

Production Requirements Composer provides a WS interface that allows other components to retrieve the complete RDF of a Production Requirements Blueprint that has been inserted in the Blueprint Repository.

Name	Type	Description
Username	String	A valid username
Password	String	A valid password that corresponds to the given username
RepositoryID	String	The repositoryID of the iMAGINE blueprint repository that contains the Production Requirements Blueprint.
ProductionRequirementsDMNID	String	The iMAGINE Id of the Production Requirements instance that should be returned.

**Table 4-8: Get Production Requirements Interface**

### 4.5.2 Partner Search

The Partner Search Component is improved in order to address the requests of Living Labs as well as to comply with the iMAGINE Platform overall planning. In this sense, a new and dynamic search requirements structure is designed for increasing the search quality. Additionally, a better grid structure for displaying product requirements and search results has been designed to lower the effort of DMN managers, during search criteria setting and short/long list creation.

#### 4.5.2.1 *Evaluate Search Interface*

This interface is called by Production Requirements Composer Component (referred as PRC), after preliminary schedule has been created. PRC calls the following Web Service *markProductSearchReady* of Search Engine. This Web Service stores the provided data to a separate MySQL database to be used by other Search Engine services. A detailed description of parameters used is shown in the following table.

Name	Type	Description
UserID	String	the ImagineID of the user that sends the preliminary schedule
Password	String	Password of the user that sends the preliminary schedule
repositoryID	String	the iMAGINE Blueprint repository that this preliminary schedule resides
ProductID	String	the ImagineID of the super product sent for Search
DMNID	String	the ImagineID of the DMN created for the super product by PRC

**Table 4-9: Evaluate Selection Parameters**

#### 4.5.2.2 WEB UI

Web User Interface (WEB UI) provides an easy listing for Production Requirements Products. The Web User interface is illustrated in detail in section 5.2 and supports the following functionalities:

#### Display Products by status as "Product Type"

As the count of products increase, the need to separate them by their status occurred.

##### **Product Types:**

- **New Products:** New products sent for partner listing from Production Requirements Composer.
- **In Process:** Products that have their requirements started to be matched with companies but not yet sent for evaluation.
- **Sent for Evaluation:** This is the final stage of a product after all its requirements are matched with a company list and sent to DMN Evaluation and Final Selection Component.
- **Archived:** In any stage, users are able to send a product to archive. It removes the product from that stage list, lowering the number of products to be displayed in each stage.

On any stage, super products are listed and are selectable.

Once a super product is selected, the product and its sub-products are listed below "Product Type" as "Selected Product".

#### Quick Partner Match

As for per product resources needed may vary in numbers from a few to lots, a "Quick Partner Match" button is added. This button allows users to define a specific number of companies to be matched automatically to each required resource.

#### Display Requirements

When any product listed in "Selected Product" tree is selected, its requirements are listed to the right grouped by their type: Equipment, Material, Process and Skill. Requirements can be selected one at a time for partner search.

Once selected, clicking "Search for Partners" button will call the related search interface: Long List Search or Short List Search.

- **Long List Search:** Long List Search interface provides criteria related to common information of companies such as employees, turnover, location, company category, production category.
- **Short List Search:** Short List Search interface provides criteria related to dynamic information of companies: Capacity rate, duration, fixed cost, variable cost. It also allows prioritizing these criteria to better filter companies from long list search.

### External Partner Search

DMN manager will be able to search for additional partners outside the network if he/she finds match results unsatisfying. This will require specific external sources, namely B2B Matchmaking Marketplaces such as Diyalogo.com or Alibaba.com to develop a service which will be called with a specific, predefined xml structure such as used currently between PSC interface and Search Engine. Return of the WS will be the same. Each B2B Marketplace is responsible for providing output for the required criteria.

- **searchExternalMarketPlaces** Sends the search criteria for the resource being searched to specified external marketplace. Reforms the output of External Marketplace WS to be displayed in search results. Candidate partners will be saved to a separate database with their information.
- **acceptExternalCandidatePartners** If DMN Manager accepts proposed candidate partners, an invitation email to iMAGINE platform will be sent to the provided emails of these partners.

### Store

DMN Managers stores the final long list of companies and annotated preliminary production schedule.

- **Dynamic and Preset Search Criteria**

Users can save their search criteria choices before the search. This will allow them to choose their repetitive searches easily and apply the criteria again.

Additionally default RDF and SPARQL structure allows us to see fields of a record. Using this feature, search criteria will be converted to dynamic search criteria. PSC will display only related, existing fields of the resource being searched. This will provide an ease of use to DMN managers.

- **Preset DMN Schemas**

DMN Managers will be able to save and view their previously accepted DMNs. This will allow them to easily pick a predefined DMN while creating a new network for another product.

### Confirm Long List

The DMN manager confirms current long lists and proceeds to short list. All requirements must have a list of partners.

### Confirm Shortlist

After the partner list is created, the manager can call the DMN Evaluation and Final Selection.



Name	Type	Description
Marketplace	String	The marketplace to be searched for possible partners
SearchCriteria	String	Search criteria specified by the DMN Manager for the resource
ResourceId	String	The iMAGINE Id of the resource being searched
DMNID	String	The iMAGINE Id of the DMN that is under creation

**Table 4-10: Parameters for External Marketplace Search**

Name	Type	Description
UserID	String	The ImagineID of the user that sends the preliminary schedule
Password	String	Password of the user that sends the preliminary schedule
RepositoryId	String	The repositoryID of the iMAGINE blueprint repository
SearchCriteria	String	Search criteria that is used for the search
ResourceId	String	The iMAGINE Id of the resource being searched
SearchType	String	The type of the search made: Long List or Short List Search
DMNID	String	The iMAGINE Id of the DMN that is created

**Table 4-11: Parameters for Saving Search Criteria**

Name	Type	Description
PSC_Result ID	String	The iMAGINE Id of the PSC_Result that is saved for the DMN
DMNID	String	The iMAGINE Id of the DMN that is created

**Table 4-12: Parameters for Saving a DMN and its Results**

Name	Type	Description
NotificationType	Int	Type of the notification to be sent to external partner
RelatedDmn	String	ID of the DMN that is searching for partners
NotificationId	String	iMAGINE Id for the notification
relatedProductXml	String	XML that contains information about the product being searched and requirements to be met

**Table 4-13: Parameters for Sending a Notification to Partners**

Name	Type	Description
response	String	XML that contains response of the partner to DMN's request
RelatedDmn	String	ID of the DMN that requested an update
NotificationId	String	iMAGINE Id for the notification
relatedProduct	String	iMAGINE id of the product that update is requested for

**Table 4-14: Parameters for Sending a Notification to DMN Manager**

Name	Type	Description
UserID	String	The user ID
Password	String	Password of the user
repositoryID	String	The repositoryID of the iMAGINE blueprint repository that contains the Potential Supplier annotated Preliminary production schedule
ResultListID	String	The ImagineID of the PSC_Result for related product
DMNID	String	The iMAGINE Id of the DMN that is under creation

**Table 4-15: Parameters for the DMN Evaluation and Final Selection**

#### 4.5.2.3 Search Engine

Search Engine, a.k.a. MATCH component in version 1, is a replaceable subcomponent which is used by the interface. It can be replaced either by the default UI or a custom UI can call services from the Search Engine.

Version 1 was designed to provide search functions for each type of requirement, such as:

- **Part and Material Search:** Search partners for parts and materials required for the production
- **Skill Search:** Search partners with required skills for the production
- **Equipment Search:** Search partners with required equipment for the production
- **Process Search:** Search partners capable of required processes for the production

Due to similar behavior of these requirements, Search Engine now provides functions for each "Search Type": Long List search and Short List search. For the sake of customization, data required by UI is also provided by Search Engine.

- **Search Functionality**

Searches for partners already registered to the system for related functions (Material, Skill, Equipment, Process Search).

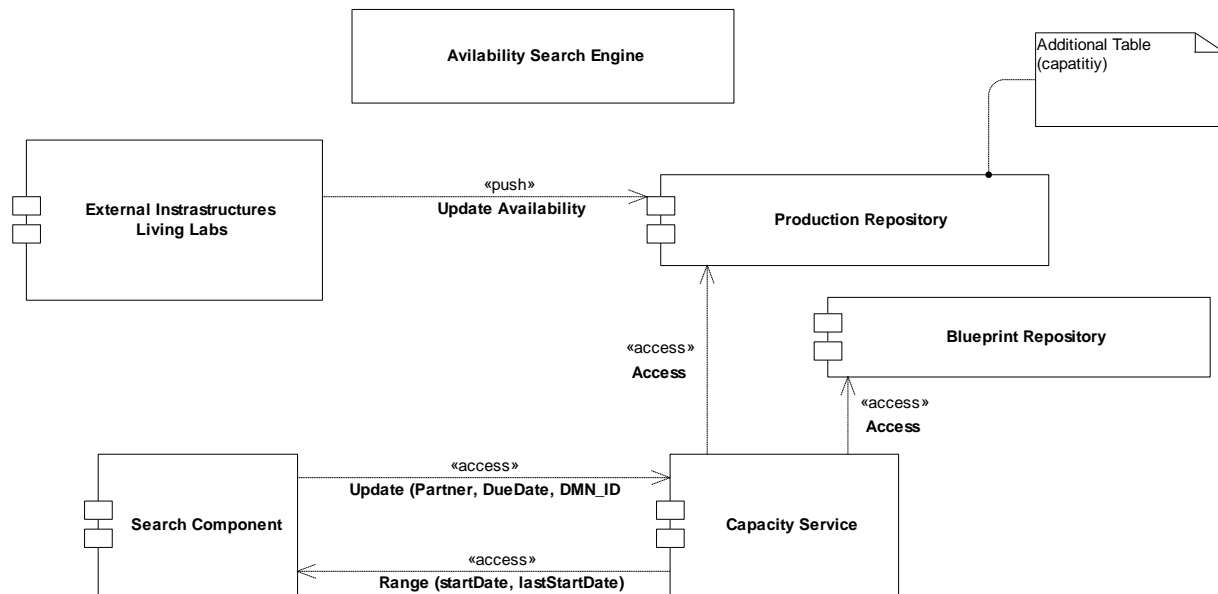
- o **searchPartnersForLongList:** Search partners to create long list for a requirement with given criteria.
- o **searchPartnersForShortList:** Search partners to create short list from a requirement's long lists with given criteria.

- **External Match:** Results from External Partner Search will be reformed with respect to repository data structure. If the DMN manager accepts the match, an invitation email will be sent to external potential partners to join the iMAGINE platform and send response to RFI, RFP, RFQ.
- **savePartnerListForRequirement:** creates a partner list for given search type (Long List or Short List) with given companies for a requirement.
- **generatePSCResultForProduct:** When a user confirms current lists (Long List or Short List) a PSC\_Result is created.
- **UI Functionality**
  - **getSuperProductsList:** Queries MySQL database for super products and returns data in xml format.
  - **getProductAndSubProductsWithName:** Queries the given repository for a selected product and returns it with its sub-product tree.
  - **getProductRequirements:** Queries the given repository for a product's *ALL* requirements.
  - **markProductSearchReady:** Web Service called by PRC to send a product for partner search. Provided product data is saved to MySQL database.
  - **updateProductStatus:** Updates the status of a given product depending on its current stage (See: WEB UI – Display Products)
  - **archiveProducts:** Change status of given products in MySQL database to Archived (See: WEB UI – Display Products)
  - **getDmnId:** Get DMN ID of a super product from MySQL database
  - **getAllUnits:** Currently retrieves all unit data from repository. When dynamic category function is finished, this will be removed.
  - **getSearchResultListsForProduct:** Get result lists (PSC\_Result) of a super product.
- **Dynamic Categories**

All categories (units, production categories, cities and similar data will be named as a "category") will be received by the same function. As categories will be created by administrators, all users will use the same category data. The Blueprint Repository can cache SPARQL queries so queries will be executed only if there is an update. Even though this function will be used quite frequently, it will not cause performance issues.

### 4.5.3 Availability Search Engine

The Availability Search Engine enhanced the present implementation of the Partner search with more recent data which will improve the quality search results and partner matches. This is achieved by a frequent update of Availability updates from the External Infrastructures (Living Labs) to the platform using an *UpdateAvailability* Service. The service is implemented using a push-paradigm to give every external partner the freedom to update its availability according to its needs.



**Figure 4-16: Message flow of the Availability Search Extension**

The Partner Search Component (PSC) communicates with Capacity Service via WSs to check availability of partners provided by the Long List according to search criteria provided in Short List creation phase. Listing of will made using the specific up-to-date dynamic data provided by partners in accordance to a specific data structure that will be developed for that purpose. This will consist of a calendar like structure with availability per day or week for a specific period.

After partners listed with their current data, DMN manager will be able to create a better partner short list for the related requirement.

#### 4.5.4 DMN Evaluation and Final Selection

The DMN Evaluation and Final Selection component will be updated to enable the reuse of submitted production requirements thus providing increased productivity and minimizing the effort needed for creating DMNs for similar products as well the time needed to setup a new DMN in order to manufacture them. In addition the functionality that is required to customize the behaviour of DMN Evaluation and Final Selection component will be added. The generic design of the user related parameters that need to be taken into account for this reason are detailed explained in section 4.3.3.

##### 4.5.4.1 History

The History functionality of the DMN Evaluation and Final Selection component supports the day to day operation of the iMAGINE Platform by the DMN Manager, making available information about all the latest actions performed by the DMN Manager. Entries no longer required in the History can be deleted by the DMN Manager. The design of the History GUI of the DMN Evaluation and Final Selection component is provided in section 5.3.1.1.

##### 4.5.4.2 Detailed Logging

The DMN Evaluation and Final Selection component will log all created, submitted and deleted short lists in a dedicated database schema. The logging will provide transparency to DMN Evaluation

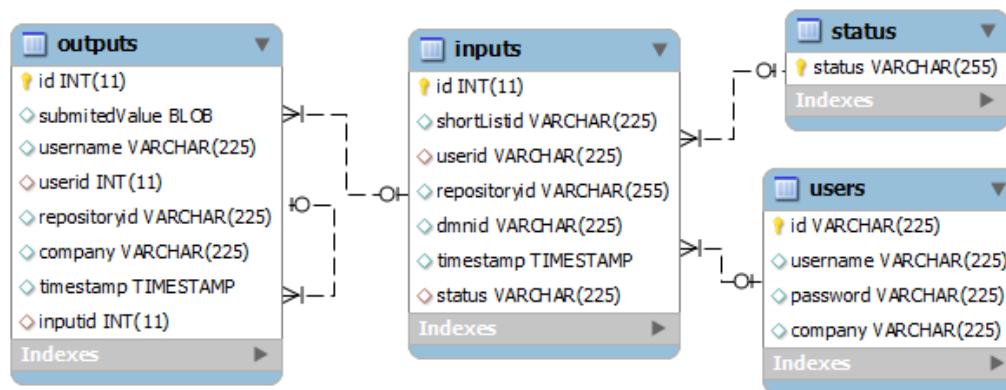
related actions performed in every company as well as detailed reporting in case it is needed. The major difference between the logging and the history information is that records cannot be deleted from the user interface of the logging mechanism via the GUI; rather they are permanently stored for reference. The history functionality is mostly used to allow the DMN Manager to keep up with the daily operation of the DMN Evaluation and Final Selection. For this reason entries can be deleted from the history if needed. The design of the GUI that supports the logging mechanism of the DMN Evaluation and Final Selection component is provided in section 5.3.1.2.

#### 4.5.4.3 Reverse DMN Lifecycle Support

The DMN Evaluation and Final Selection component will allow the DMN Manager to go back in the DMN Lifecycle and resubmit a DMN that is currently in progress either to the Partner Search component or the Production Requirements Composer. This functionality will be implemented by calling the "Submit Production Requirements" interface of Production Requirements Composer, which is specified in section 4.5.1.6.1, or the respective interface of Partner Search Component, which is specified in section 5.2 .

#### 4.5.4.4 Database Schema

In order to support the aforementioned functionality, the DMN Evaluation and Final Selection component will also be supported with the database schema that is depicted in Figure 4-17: DMN Evaluation and Final Selection Database Schema. This database contains information that is mostly used for the History and Logging Functionality.



**Figure 4-17: DMN Evaluation and Final Selection Database Schema**

#### 4.5.4.5 Web Service Interfaces

##### 4.5.4.5.1 Submit Short Lists

DMN Evaluation and Final Selection component provides a WS interface that allows other components to signal that Short Lists have been inserted in the Blueprint Repository. In this case they are pending for editing by the Production Requirements User Interface. A prerequisite is that the Production Requirements Blueprints should also reside inside the Blueprint repository. In case the Short Lists have already been submitted to the DMN Evaluation and Final Selection component in the past, it is opened again for evaluation.

This interface is called by the Design Toolset as well as the Partner Search component. In a customized version of the i\_platform this interface can also be made available for usage by external systems via the iMAGINE Integration Server.

Name	Type	Description
Username	String	A valid username
Password	String	A valid password that corresponds to the given username
RepositoryID	String	The repositoryID of the iMAGINE blueprint repository that contains the Short Lists and Production Requirements Blueprint instances.
PartnerSearchResultsID	String	The iMAGINE Id of the Short Lists instance that should be edited.

**Table 4-16: DMN Evaluation And Final Selection – Submit Shortlists Interface**

#### 4.5.4.5.2 Get Product Blueprint

DMN Evaluation and Final Selection Component exposes a WS interface that allows other components to retrieve the complete Product Blueprint of a DMN that has been inserted in the Blueprint Repository.

Name	Type	Description
Username	String	A valid username
Password	String	A valid password that corresponds to the given username
RepositoryID	String	The repositoryID of the iMAGINE blueprint repository that contains the Production Requirements Blueprint.
ProductImagineId	String	The iMAGINE Id of Product Blueprint instance that should be returned.

**Table 4-17: Get Product Blueprint Interface**

## 4.6 DMN Design

### 4.6.1 DMN Design Toolset

The DMN Design Toolset component will be updated to provide additional functionality in accordance to the Architecture of iMAGINE Platform v3 that enables the usage of DMN level production measurements. An overall orchestration view will also be added in order to provide better visibility of the overall DMN Orchestration schedule.

#### 4.6.1.1 *Production Measurements Editor*

The Production Measurements Editor aims to assist the DMN Manager to effectively and efficiently define the Production Measurement KPIs as defined in the Architecture of iMAGINE Platform v3. KPIs assist the organization to define and measure progress toward organizational goals and objectives.

Once the organization has analysed its mission and defined its goals, it needs to measure progress towards those goals. The Production Measurements Editor is responsible to define the required coefficients and thresholds that allow the calculation of the Production Measurements that are stored in the updated iMAGINE Quality Assurance Blueprint schema as defined in section 11.2 of deliverable D2.2.3. The design of the Production Measurements Editor GUI of the DMN Design Toolset is provided in section 5.4.1.2.

#### *4.6.1.2 Overall Orchestration Editor*

The design of the Overall Orchestration View Screen of the DMN Design Toolset is provided in section 5.4.1.1.

#### *4.6.1.3 History*

The History functionality of the Design Toolset supports the day to day operation of the iMAGINE Platform by the DMN Manager, making available information about all the latest actions performed by the DMN Manager. Entries no longer required in the History can be deleted by the DMN Manager. A unified history window will be provided for all the DMN Orchestration editors and Quality Assurance Editors. The design of the History GUI is provided in section 5.4.1.3.

#### *4.6.1.4 Detailed Logging*

The Design Toolset component will log all created, submitted and deleted DMNs in a dedicated database schema. The logging will provide transparency to DMN Design related actions performed in every company as well as detailed reporting in case it is needed. Similarly to the logging and history functionality of other components, the major difference between the logging and the history information is that records cannot be deleted from the user interface of the logging mechanism via the GUI; rather they are permanently stored for reference. The history functionality is mostly used to allow the DMN Manager to keep up with the daily operation of the DMN Design. For this reason entries can be deleted from the history if needed. A unified history screen will be provided for all the DMN Orchestration editors and Quality Assurance Editors.

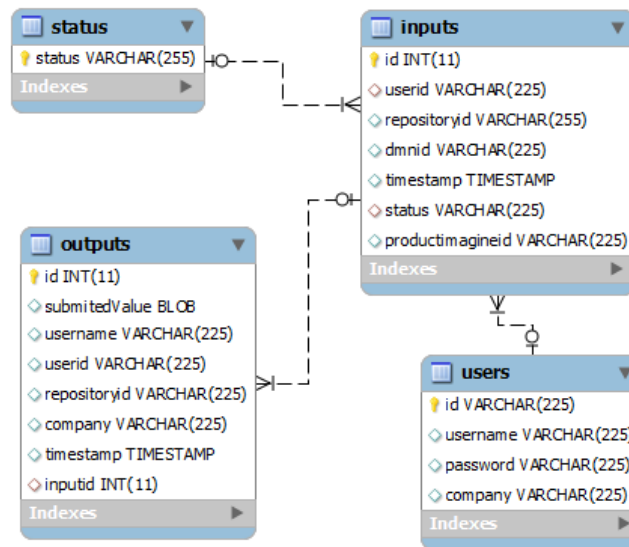
#### *4.6.1.5 Reverse DMN Lifecycle Support*

The DMN Design component will allow the DMN Manager to go back in the DMN Lifecycle and resubmit a DMN that is currently in design progress to one of the following components:

- The Partner Search component. The DMN will be resubmitted to the Partner Search Component by calling the appropriate interface of Partner Search Component, which is specified in section 5.1.1.2 .
- The Production Requirements Composer. This functionality will be implemented by calling the "Submit Production Requirements" interface of Production Requirements Composer, which is specified in section 4.5.1.6.1. Database Schema
- The DMN Evaluation and Final Selection Component. This functionality will be implemented by calling the "Submit Short Lists" interface of DMN Evaluation and Final selection component, which is specified in section 4.5.4.5.1 .

#### 4.6.1.6 Database Schema

In order to support the aforementioned functionality, the DMN Design Toolset component will be supported with the database schema that is depicted in. This database contains information that is mostly used for the Design Toolset's History and Logging Functionality.



**Figure 4-18: DMN Design Toolset Database Schema**

#### 4.6.1.7 Web Service Interfaces

##### 4.6.1.7.1 Submit DMN Configuration

DMN Toolset component provides a WS interface that allows other components to signal that a DMN has been configured and is ready to be designed. A prerequisite is that all the Blueprint instances created in the DMN Analysis and Configuration Phase of the DMN Lifecycle should also be in the Blueprint repository. In case the Product Blueprint has already been submitted to the DMN Design Toolset component in the past, it is opened again for design.

This interface is called by the DMN Evaluation and Final Selection component. In a customized version of the i\_platform this interface can also be made available for usage by external systems via the iMAGINE Integration Server.

Name	Type	Description
Username	String	A valid username
Password	String	A valid password that corresponds to the given username
RepositoryID	String	The repositoryID of the iMAGINE blueprint repository that contains the required blueprint instances.
ProductImagineId	String	The iMAGINE Id of the DMN Product Instance that should be designed.

**Table 4-18: DMN Design Toolset - Submit DMN Configuration Interface**



#### 4.6.1.7.2 Get End To End Blueprint

DMN Design Toolset exposes a WS interface that allows other components to retrieve the complete End-To-End Blueprint of a DMN that has been inserted in the Blueprint Repository.

Name	Type	Description
Username	String	A valid username
Password	String	A valid password that corresponds to the given username
RepositoryID	String	The repositoryID of the iMAGINE blueprint repository that contains the Production Requirements Blueprint.
EndToEndBlueprintiMAGINEID	String	The iMAGINE Id of the End To End Blueprint instance that should be returned.

**Table 4-19: Get End To End Blueprint Interface**

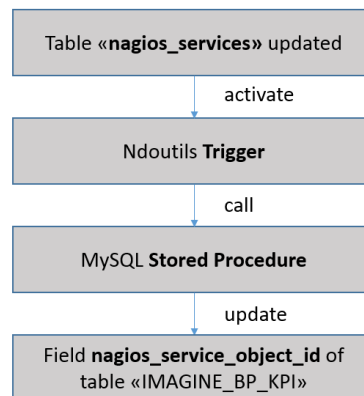
## 4.7 DMN Execution, Monitoring & Management KIT

### 4.7.1 KPI Log and Monitor

In order to collect the KPIs of all partners, in the NDOUtils database (the one used by the monitoring system Nagios to store its data), it has been created a table *iMAGINE\_BP\_KPI* that will contain the required information about hosts (partners) and metrics:

- DMN\_Imagine\_ID
- Partner\_Imagine\_ID
- Metric\_Name
- Metric\_ID
- nagios\_service\_object\_id
- list of KPIs

The first four fields are useful to identify the specific metric of a given partner; the *nagios\_service\_object\_id* has been introduced to match the standard Nagios protocol and the "Imagine syntax": this is the ID number Nagios gives to each service. When the Nagios table *nagios\_services* is updated, a trigger is fired and a stored procedure is called to set the ID number in the *nagios\_service\_object\_id* field.



**Figure 4-19: Information flow of the NAGIOS service**

Finally, there is the list of KPIs with their values normalized between 0 and 1 passed through a specific web service (detailed in the next paragraph). Thus, this table will collect all the KPIs of a specific metric of a partner.

Table “*IMAGINE\_metric\_last*” is intended to be a view-table merging data coming from the Nagios tables and the “*IMAGINE\_minmagavg\_last*” table (that is a table containing for each metric the minimum, maximum and average values of the performance data):

- DMN\_Imagine\_ID
- display\_name
- metric\_name
- Metric\_ID
- Partner\_Imagine\_ID
- Perfddata
- status\_update\_time
- min
- max
- avg

Data stored in *IMAGINE\_minmagavg\_last* is written in the MySQL database through a scheduled stored procedure (to be executed every 10 minutes) which is in charge of synchronizing the NDOUTils tables and inGraph tables (databases of the graphing tool). Due to different IDs given to services and hosts by Nagios and inGraph, a match between the two tools needs to be implemented.

Finally, table *IMAGINE\_BP\_partners* aims to collect the list of all the partners using the Imagine platform (*DMN\_Imagine\_ID*, *display\_name*, *Partner\_Imagine\_ID*, *Full\_Name*).

DMN_Imagine_ID	display_name	Partner_Imagine_ID	Full_Name
DMN_001	DMN_001_Partner_001	Partner_001	Wood_Wholesales_inc
DMN_001	DMN_001_Partner_002	Partner_002	Chair_Manufacturing_Company
DMN_001	DMN_001_Partner_003	Partner_003	Tissue/patcher
DMN_001	DMN_001_Partner_004	Partner_004	Furniture_Packaging_and_Shipment_GmbH
DMN_001	Partner_1	Partner_005	
Car_Door	Car_Door_43	43	Chair_Mfg_2
Car_Door	Car_Door_44	44	Partner_002
Car_Door	Car_Door_45	45	Partner_001
Nagios_Test	Nagios_Test_iMAGINE_ID_My_PartnerName	My_PartnerName	FullName
123	123_SAG-test	SAG-test	SAG-new
Packaging.E	Packaging.EquipmentOrchestration_blueprints2.0_Cl...	blueprints2.0_Class1	companyX

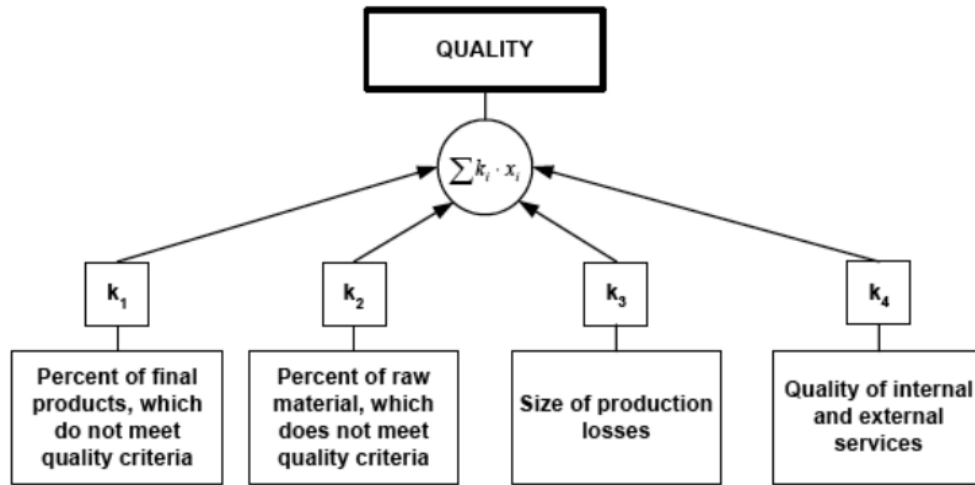
**Table 4-20: NAGIOS Database Table of iMAGINE\_BP\_partners**

#### 4.7.1.1 Production Metrics and Measurements

In order to provide Production metrics and measurements, the KPIs have been grouped into four categories; each of these production measures has additional sub-coefficients:

- **Perfect Order Fulfillment:**
  - % of Production schedules met
  - % of Orders delivered in perfect conditions
  - Delivery to customer commit date
  - Actual versus planned production time
- **Production Efficiency:**
  - Efficiency of employees in production
  - Infrastructure efficiency
  - Materials used
  - Energy used
  - Unit production time
  - Quality of internal and external services
  - Production shutdowns
- **Production Quality:**
  - % of final products, which do not meet quality criteria
  - % of raw material which does not meet quality criteria
  - Size of production losses
  - Quality of internal and external services
- **Equipment Effectiveness:**
  - Average machine availability rate or machine uptime
  - Hours lost due to equipment downtime
  - Cumulative count of machine breakdown

Each of these sub-coefficients will be provided to the monitoring machine through a web service that will require also the name of the host and the associated service. The four main metrics are computed as weighted sum of coefficients (all  $k_i$  fixed for a specific DMN) with their KPIs ( $x_i$ ). For the Production Quality metric this turns out to be:



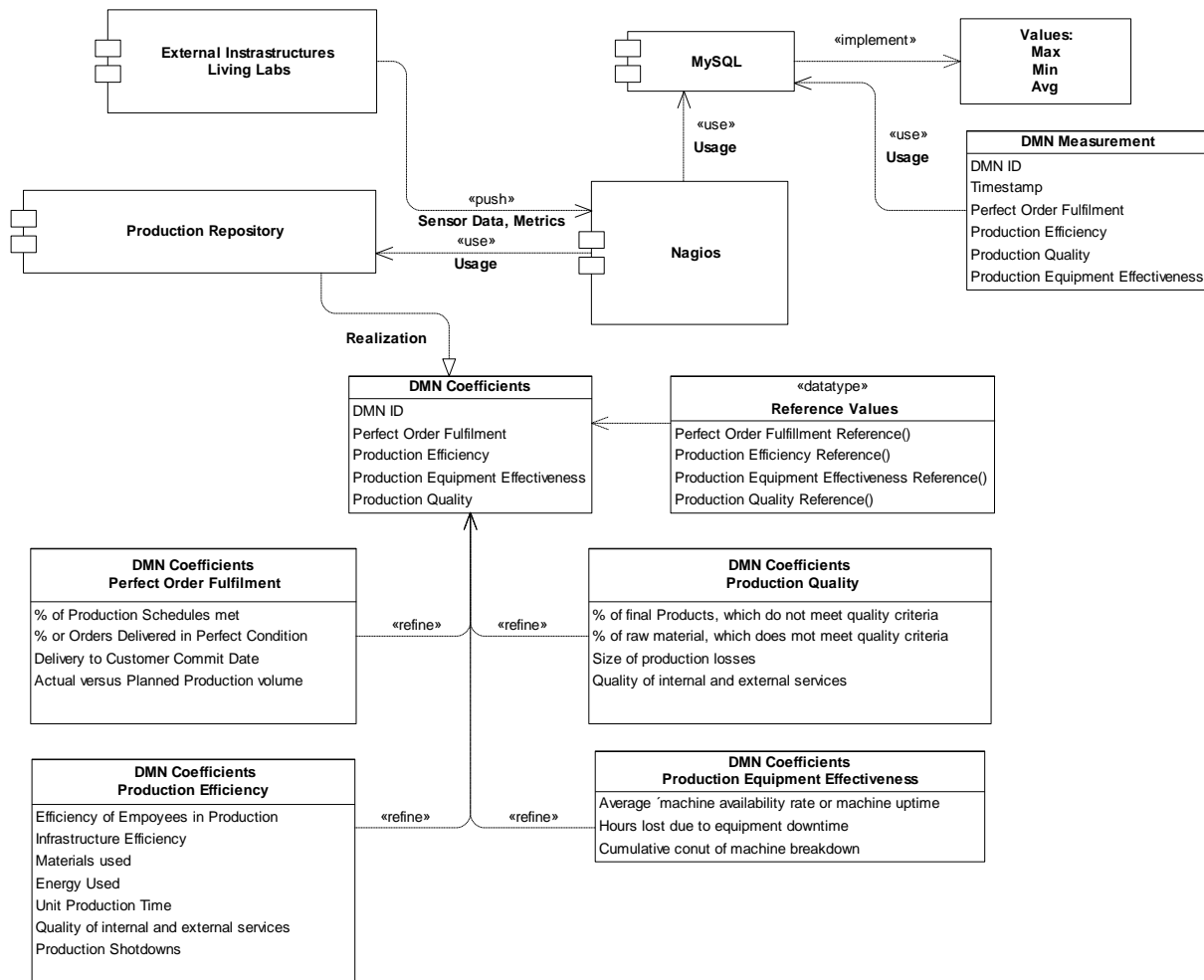
The Production Quality metric (as the other metrics) will be equal to:

$$Production\ Quality\ Metric = \sum_{i=1}^4 k_i \cdot x_i$$

with  $0 \leq x_i \leq 1$ ,  $i = 1, 2, 3, 4$ .

The  $x_i$  values will be given to the monitoring machine through a web service as values normalized between 0 and 1 (as ratio between the actual performance value and the planned performance data) while the  $k_i$  coefficients (whose sum should be unitary) will be stored into the Production Repository database. In this way, the production measures will always vary between 0 and 1.

The  $k_i$  parameters are useful to give different weights to each metric, while the  $x_i$  values to calculate the performance of a single partner for a specific service. This web service will compute the metrics from the MySQL databases and create an RRD (Round-Robin-Database) file – using the RRDTool – with the four production metrics (perfect order fulfillment, production efficiency, production quality, and equipment effectiveness) which will be provided to the Imagine dashboard to be displayed. The four production metrics may also be obtained through the max, min and avg performance data stored in Nagios tables.



**Figure 4-20: Production Repository Extension - Production Metrics**

## 4.7.2 Trouble Shooter

The Trouble-shooter is implemented in the iMAGINE platform through the open source monitoring tool Nagios Core. The main aims of Nagios are data logging based on events and triggers and storage of data in a MySQL database for further reporting, extraction and analysis. Nagios Core is the monitoring and alerting engine that serves as the primary application around which the Nagios projects are commonly built. It serves as the basic event scheduler, event processor, and alert manager for elements that are monitored.

### 4.7.2.1 Nagios: extensible architecture

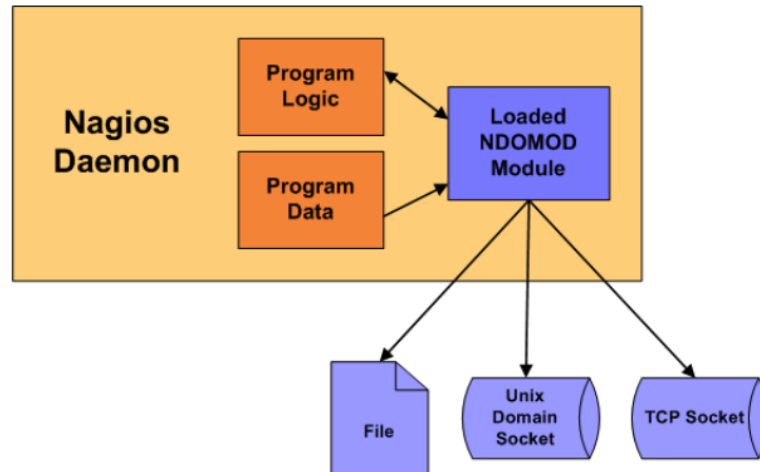
One of the best features of Nagios is its modular architecture that allows other software "Nagios-based" to be integrated. In the iMAGINE platform, the monitoring tool will be connected with three components:

- **NDOutils**
- **Nagios Service Check Acceptor (NSCA)**
- **inGraph**

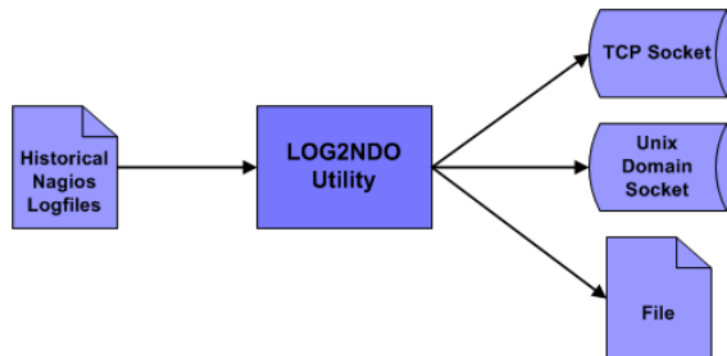
#### 4.7.2.1.1 NDOutils

Given that Nagios is a monitoring tool (not a metering one) it does not store data and results coming from checks: for this reason, the NDOUtils add-on is designed to store all configuration and event data from Nagios in a database. MySQL databases are currently supported by the add-on while PostgreSQL support is still in development. Actually, data from Nagios process can be stored in just one database. There are four main components that make up the NDO utilities:

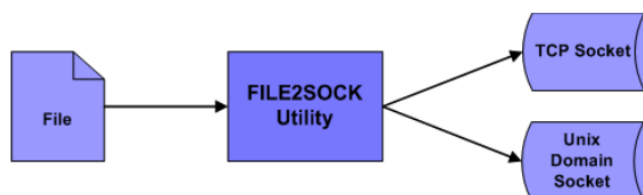
1. **NDOMOD Event Broker Module:** exports configuration data from the Nagios daemon and sends them to a standard file, a Unix domain socket or a TCP socket;



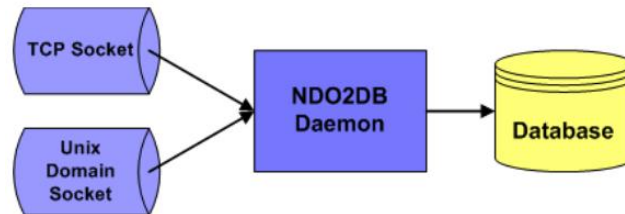
2. **LOG2NDO Utility:** allows you to import historical Nagios log files into a database via the NDO2DB daemon (described later);



3. **FILE2SOCK Utility:** reads input from a standard file and writes all of that data to either a Unix domain socket or TCP socket;



4. **NDO2DB Daemon:** takes the data output from the NDOMOD and LOG2NDO components and store it in a MySQL or PostgreSQL database. When it starts, the NDO2DB daemon creates either a TCP or UNIX domain socket and waits for clients to connect. NDO2DB can run either as a standalone, multi-process daemon or under INETD (if using a TCP socket).



#### 4.7.2.1.2 Nagios Service Check Acceptor (NSCA)

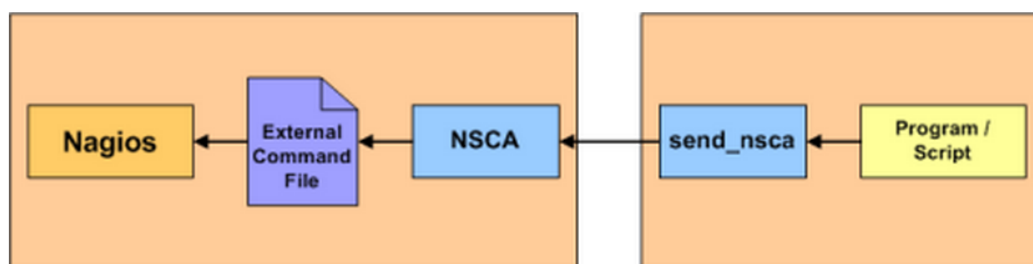
The second component integrated in the Trouble-shooter is the Nagios Service Check Acceptor (NSCA); this is an add-on that allows you to send passive check results from remote Linux/Unix hosts to the Nagios daemon running on the monitoring server. In most cases Nagios will monitor hosts and services using regularly scheduled active checks; active checks can be used to "poll" a device or service for status information every so often. Nagios also supports a way to monitor hosts and services passively instead of actively. The key features of passive checks are as follows:

- Passive checks are initiated and performed external applications/processes (i.e. partners);
- Passive check results are submitted to Nagios for processing;

The major difference between active and passive checks is that active checks are initiated and performed by Nagios, while passive checks are performed by external applications. This is very useful in two main cases:

- distributed and redundant/failover monitoring setups;
- an asynchronous communication between Nagios server and hosts is preferred (hosts send their data to the monitoring server);

The diagram of the NSCA's role is as follows:



In Imagine R2.0, passive checks are preferred due to the highly-distributed, loosely coupled DMN configurations: only infrastructural hosts and metrics will follow the active-check paradigm.

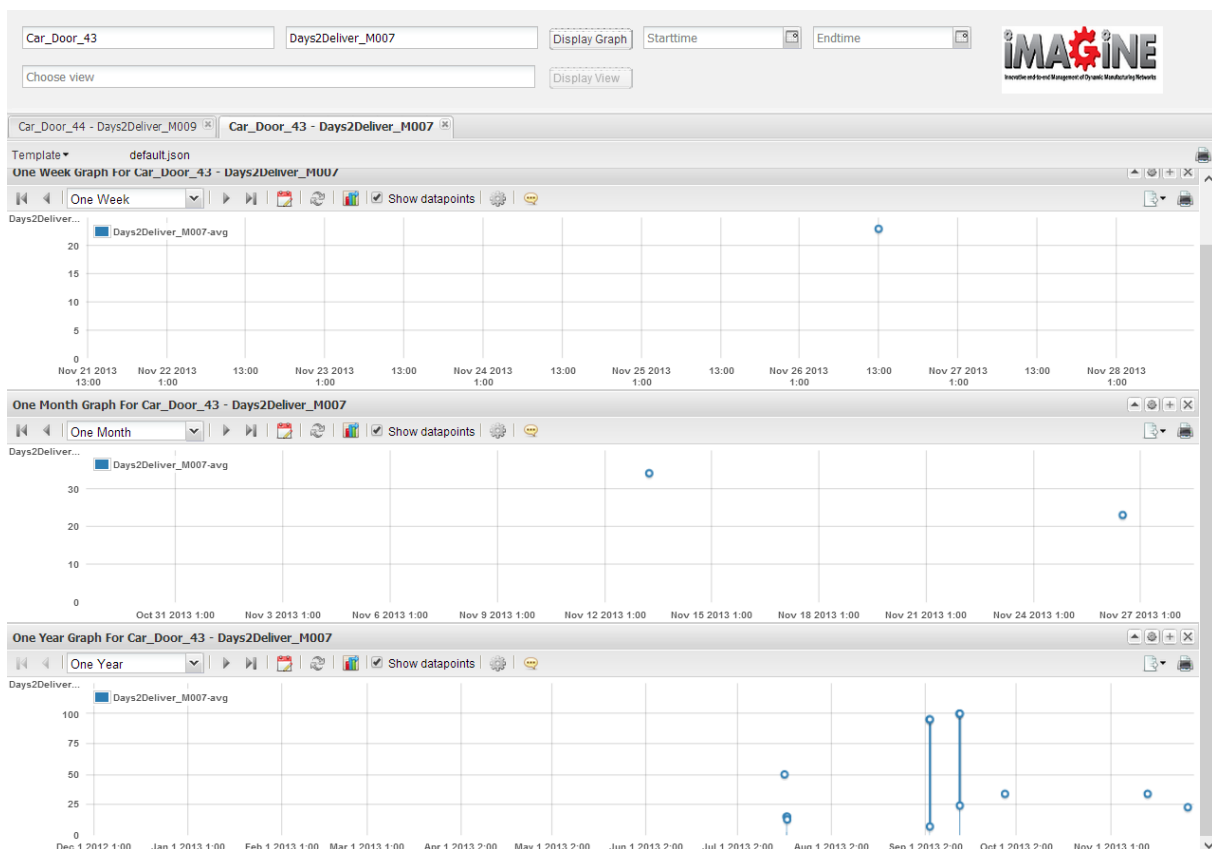
#### 4.7.2.1.3 inGraph

The third component of the Nagios installation is inGraph: this is a flexible, open source charting tool for Nagios, which collects performance data in a database and displays the results in a web interface. inGraph collects Nagios performance data files using the "ingraph-collector" daemon. After pre-

processing the files it forwards the performance data to the "ingraph" daemon which takes care of storing it in a relational database. The inGraph daemon is used by the web interface as well as the check\_ingraph plugin to retrieve performance data for generating graphs. Server side rendering of graphs is realized through Node.js. This allows generating images directly via URL, e.g. without the use of a web browser. This may be useful when generating reports based in inGraph data; so you can remotely access plots directly via URL, as below:

- <http://localhost/ingraph/?host=localhost>
- <http://localhost/ingraph/?host=localhost&service=PING>
- <http://localhost/ingraph/?view=localhost>

The web interface supports exporting data as CSV or XML files. Unlike with traditional RRD-tools, graphs are generated starting from the MySQL databases. The graphs appearance can also be changed: you may define templates which hold render and data manipulation information for monitored services. Written in JSON, a template is created by placing a comma-separated list of key: value pairs within curly braces: if your template contains syntactic errors, an error will be issued to apache's error log and the template will be ignored.

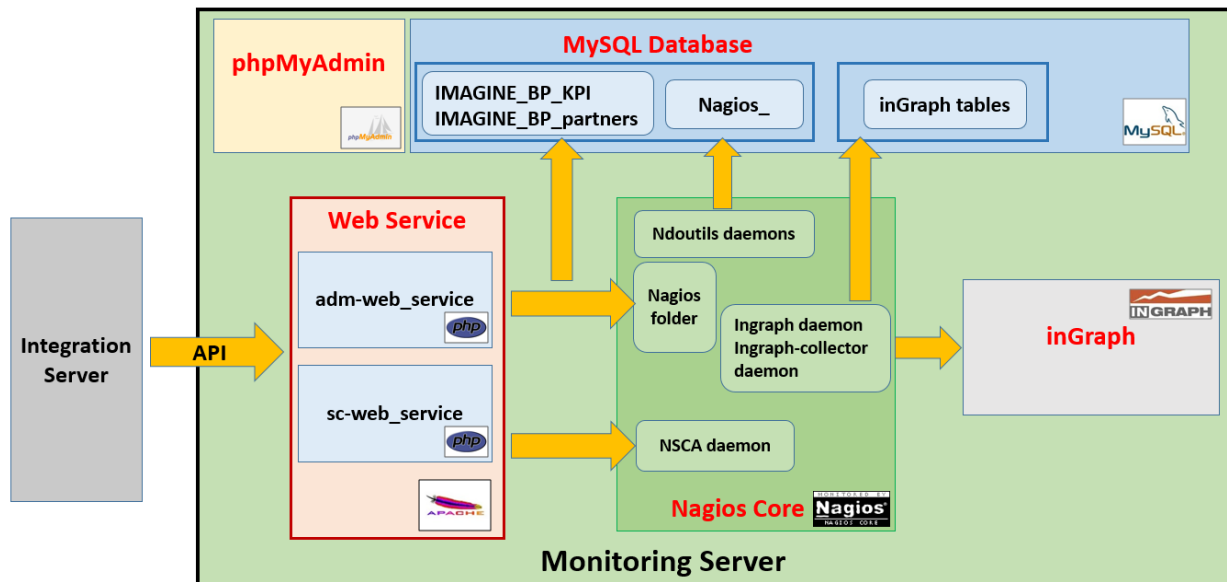


**Figure 4-21: User Interface of the inGraph Service**



#### 4.7.2.2 Implementation of the Monitoring System in the iMAGINE Platform

In the iMAGINE Platform, Nagios has been implemented with the three components detailed above. The whole architecture is as follows:



**Figure 4-22: Information and Data dependencies of the Monitoring Server**

The Integration Server communicates with the Nagios installation through two web services created on the Monitoring Server. There are two SOAP web services hosted by an Apache 2.0 Web Server: **adm-web\_service** and **sc-web\_service**, based on PHP code.

The first one is useful to Add/Delete/Modify hosts and/or services (metrics). For this reason, it exposes six functions:

- **AddHost:** this is useful to add a new host in Nagios. It requires three input parameters:
  - DMN\_Imagine\_ID
  - Partner\_Imagine\_ID
  - Full\_Name

The function checks the Nagios folder to verify it does not contain any other file called as the host you want to insert (each Nagios host configuration file will be called <DMN\_Imagine\_ID>\_<partner\_Imagine\_ID>.cfg). If this is ok, the function adds the host to table IMAGINE\_BP\_partners of the NDOUTils database.

In output the function provides two strings:

- status = this can be "success" or "error";
- description = this is a complete description of the status;

When you add a host, you will not see the host in the Nagios web interface until you associate at least one service to the host.

- **AddService:** this is useful to associate a service to an existing host. It requires four input parameters:
  - DMN\_Imagine\_ID
  - Partner\_Imagine\_ID
  - Metric\_Name (the name of the service to add)
  - Metric\_ID (the ID to identify the Metric)

Firstly, the function checks the Nagios folder to verify the host exists. If this is ok, it checks the NDOUtils database in order to verify there is no any existing service (with the same Metric\_Name and Metric\_ID) associated to the host. Then, the function adds the host and the service in the table IMAGINE\_BP\_KPI of the NDOUtils database. In the Nagios web interface, the service will be displayed as <Metric\_Name>\_<Metric\_ID>.

In output the function provides two strings:

- status = this can be "success" or "error";
  - description = this is a complete description of the status;
- **RenameHost:** this is useful to rename an existing host. It requires four input parameters:
    - DMN\_Imagine\_ID
    - Partner\_Imagine\_ID
    - New\_DMN\_Imagine\_ID
    - New\_Partner\_Imagine\_ID

Firstly the function checks the existence of the host in the Nagios configuration folder: if this is ok, it changes the name of the host in the Nagios configuration file and in the database.

In output the function provides two strings:

- status = this can be "success" or "error";
  - description = this is a complete description of the status;
- **RenameService:** this is useful to rename an existing service. It requires six input parameters:
    - DMN\_Imagine\_ID
    - Partner\_Imagine\_ID
    - Metric\_Name
    - Metric\_ID
    - New\_Metric\_Name
    - New\_Metric\_ID

The function verifies the existence of the host and service; if these are ok, it modifies the Nagios configuration files and the database.

In output the function provides two strings:

- status = this can be "success" or "error";
  - description = this is a complete description of the status;
- **DeleteService:** this is useful to delete an existing service. It requires four input parameters:
    - DMN\_Imagine\_ID
    - Partner\_Imagine\_ID
    - Metric\_Name
    - Metric\_ID

The function checks the existence of the selected host and service. If these are ok, it deletes the host from the Nagios configuration folder and from the database.

In output the function provides two strings:

- status = this can be "success" or "error";
- description = this is a complete description of the status;
- **DeleteHost:** this is useful to delete an existing host. It requires two input parameters:
  - DMN\_Imagine\_ID
  - Partner\_Imagine\_ID

The function checks the existence of the selected host. If this is ok, it deletes the selected host from the Nagios configuration folder and from the database.

In output the function provides two strings:

- status = this can be "success" or "error";
- description = this is a complete description of the status;

The second SOAP web service *sc-web\_service* will be used to update the status of a host or one of its metrics. The web service contains just one function:

- **SendCheck:** it requires some input parameters:
  - DMN Imagine ID: DMN of the host to update;
  - Partner Imagine ID: name of the host to update;
  - Metric Name: name of the service to update;
  - Metric ID: ID of the service to update;
  - status (any kind of comment);
  - performance data to update;
  - upper warning threshold to use;
  - upper ok threshold to use;
  - lower ok threshold to use;
  - lower warning threshold to use;

After checking there are no white spaces in the name of hosts and services, the web service checks the order of the thresholds ("upper warning" greater than "upper ok", "upper ok" greater than "lower ok", "lower ok" greater than "lower warning"), then it calls a bash script (useful to compare the performance data with the thresholds) and finally it sends the result to Nagios.

In output, the function gives an object containing:

- DMN Imagine ID: DMN of the updated host;
- Partner Imagine ID: name of the updated host;
- Metric Name: name of the updated service;
- Metric ID: ID of the updated service;
- status: updated status;
- performance data: updated performance data;
- check sent: it's a Boolean value ("True" means the check has been sent to Nagios, "False" means error);
- description: comment about the success/failure of the update;

This is the function calling the NSCA daemon: the NSCA daemon acts like a tunnel between the web service and Nagios.

Once the *adm-web\_service* is called, the functions exposed will update the tables "IMAGINE\_BP\_KPI" and "IMAGINE\_BP\_partners", while the NDO2DB will keep the "NAGIOS\_" tables up.

In the monitoring system's architecture, the NDOUtils is in charge of storing the performance and log files in MySQL "ndoutils" database, so that all data can be taken by other tools to be processed. The inGraph daemons engage the Nagios performance files, plot the data at and store them in their own MySQL database. Even if the same performance data are stored both in NDOUtils and inGraph databases, this is necessary because the inGraph daemons can work just with properly formatted information.

In addition to hosts and metrics added by the Integration Server, the monitoring tool will set active checks to monitor the performance of infrastructural servers and functionalities; for this reason presently the following checks have been defined on localhost, about the status of:

- web service "adm-web\_service"
- web service "sc-web\_service"
- NDO2DB daemon
- NSCA daemon
- inGraph daemon
- Apache daemon (if this daemon stops working, the Nagios web interface will not be reachable but Nagios daemon will work as always)

Furthermore, whenever Nagios takes notice of a warning or critical state about one of these metrics, it will send an email to customized contacts describing the situation. In addition, the monitoring tool could be programmed to handle particular events (such as a warning state), taking actions like an execution of a script. In order to monitor other servers of the platform, another Nagios plugin should be installed: Nagios Remote Plugin Executor (NRPE).

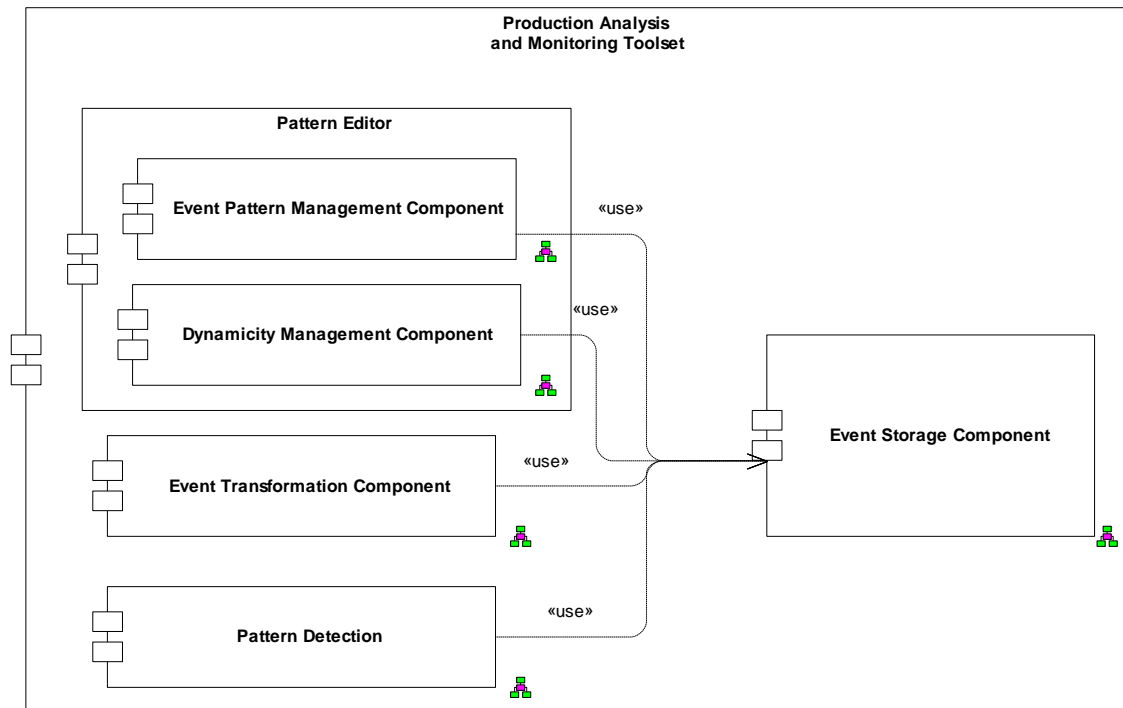
The infrastructural and partner hosts will be visible in separate screens accessible in the Nagios Web Interface with two different accounts: "platform" and "partner" (also an administrator account is present and it is able to see both platform and partner hosts).

#### **4.7.3 Dashboard**

The dashboard will be extended to support multi-user usage. In the current version 2 of the i\_platform, the monitoring information of all DMNs is available to all users, regardless of whether they have created the DMNs or not. Clearly, the DMN managers should only be able to see the relevant information to their own DMNs. In order to provide only the relevant information to each user, the e-mail address, which a user is logged in with, will be used to extract the relevant DMN IDs from the DMNAccess view of the production repository.

### **4.8 Production Analysis and Dynamic Monitoring Toolset**

The production analysis and dynamic monitoring tool set uses real-time production information in a way which enables the DMN manager to effectively manage order flow and production execution. The tool-set is much related to the EDA nature of the IMAGINE Architecture v3 and uses the following major components as shown in Figure 4-23.

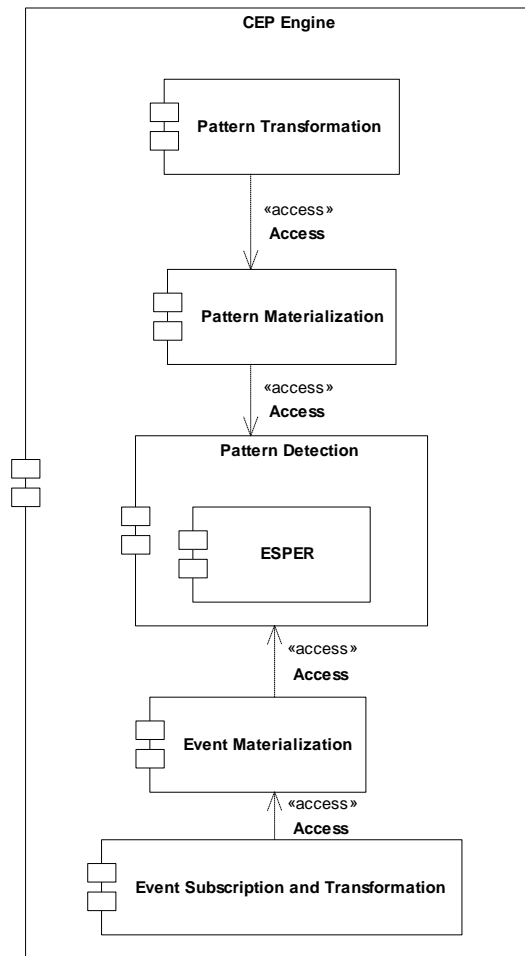


**Figure 4-23: Component Overview of the Production Analysis and Dynamic Monitoring Toolset**

The advanced visualization component implemented in the dashboard is also part of the production analysis and dynamic monitoring tool set in a broader sense, but described in detail in Section 4.8.4 and its user interfaces is sketched in section 5.6.

#### 4.8.1 Complex event detector

Event processing engine is the core of the EDA and the heart of the dynamic manufacturing and monitoring framework. The main purpose of an event processing engine is to detect a situation of interest. In iMAGINE this will be done by considering the background knowledge as well as historical events. The CEP Engine is further described in deliverable D8.1 along with its interfaces to the dynamic monitoring part.



**Figure 4-24: The CEP Engine with its environment**

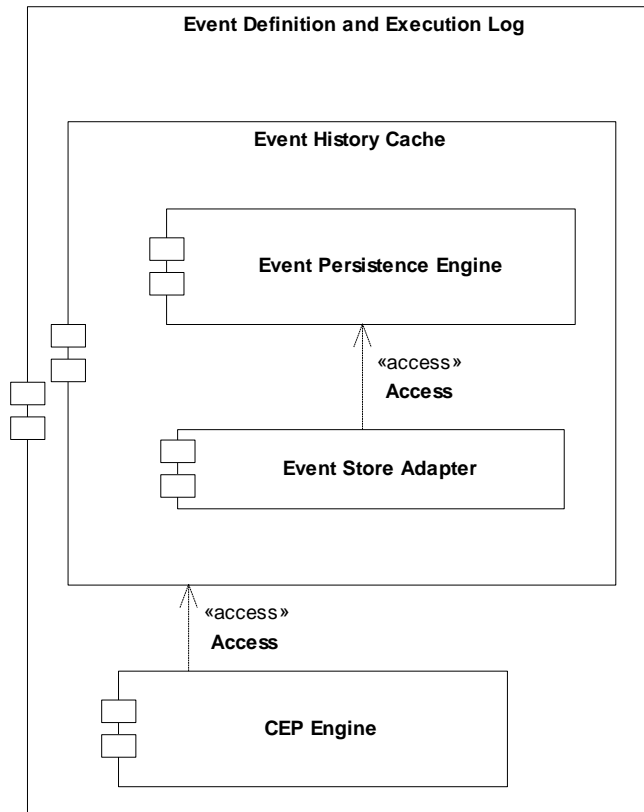
#### 4.8.2 Pattern Editor for modeling

The Pattern editor allows modeling and management of the description of the situations of interests in order to be able to cope with the dynamicity in the internal and external context of a monitored process. It has already been described in Section 5.3 of D8.1.

The Pattern editor will provide support for defining patterns at different levels of abstraction which are Event filtering (i.e. putting constraints on one or more event parameters), combining two and more events without considering time aspects (e.g. sequence, conjunction, disjunction, negation, etc.), and support combining two and more events with temporal aspects and aggregation functionalities (e.g. event aggregation for count, average, sum, min, max etc.).



At design time, a local copy of the predefined event types of the Event Type Store which is part of the Event Pattern Store is available for reference. User-defined event types can be created using the Event Pattern Management Component and stored in the local copy. Event types in the local copy must be deployed to the run-time store, so that EDA participants that process an event stream can retrieve the schema definition of the event. For more information about deploying Event Types, see [Deploying EDA Assets](#). The Event Storage Component is depicted in Figure 4-26.



**Figure 4-26: Event Storage Components**

#### 4.8.4 iMAGINE-Enlarged Advanced visualization

The event-driven, real-time capabilities of our visualization solution enables dashboards to be rapidly constructed which consume business events from the CEP Engine and render this data instantaneously in the form of a wide variety of graphical indicators and charts to help business and operations staff alike to have a real-time insight into key business processes and their performance. As an advanced way of visualizing the highly dynamic data, the existing dashboard will be extended to show real-time information by either receiving events issues from the CEP engine or by continuously polling data from a database. This extension will be based on top of the existing visualization by supporting pattern monitoring, anomaly detection, and trend analysis.

A sketch about the dynamic event visualization is depicted in Section 5.6.



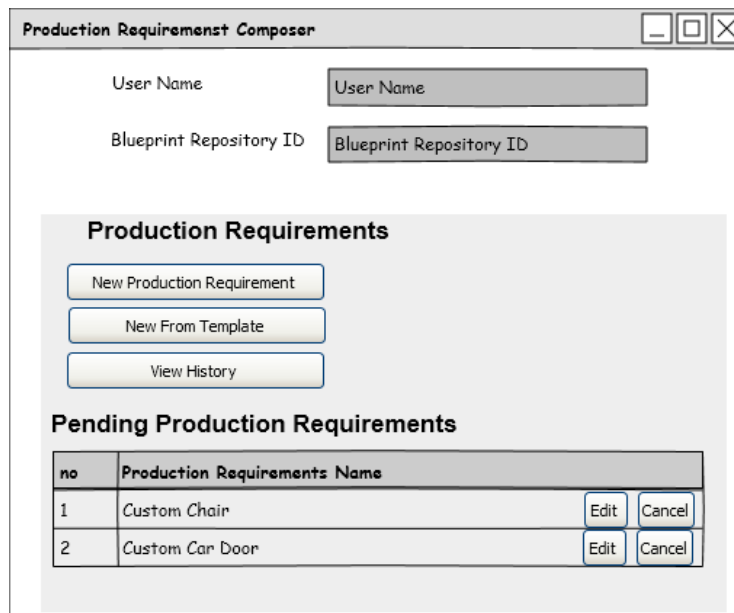
## 5 User Interface Design

This chapter covers the design of the user interfaces as they are delivered by the individual components. The provided interfaces specify in a comprehensive visual way the basic functionalities that each component shall provide. The actual implementation of the interfaces may vary and more screen and options will be available in the developed components. In order to avoid duplicate definitions to previously defined UIs in Detailed Design V1 only newly introduced components are considered. Major improvements and changes of V2 or the platform will be documented in the iMAGINE Platform R3 deliverable D3.2.3.

### 5.1 Production Requirements Composer

#### 5.1.1.1 Main Screen

The introduction of History, Detailed Log and Templates functionality to the Production Requirements Composer required for a new landing screen that gives the DMN Manager the option of choosing the desired functionality. It also provides a quick overview of the pending Production Requirements. These are the Production Requirements that have been created and saved, but have been not yet submitted to the Partner Search Component neither have been canceled.

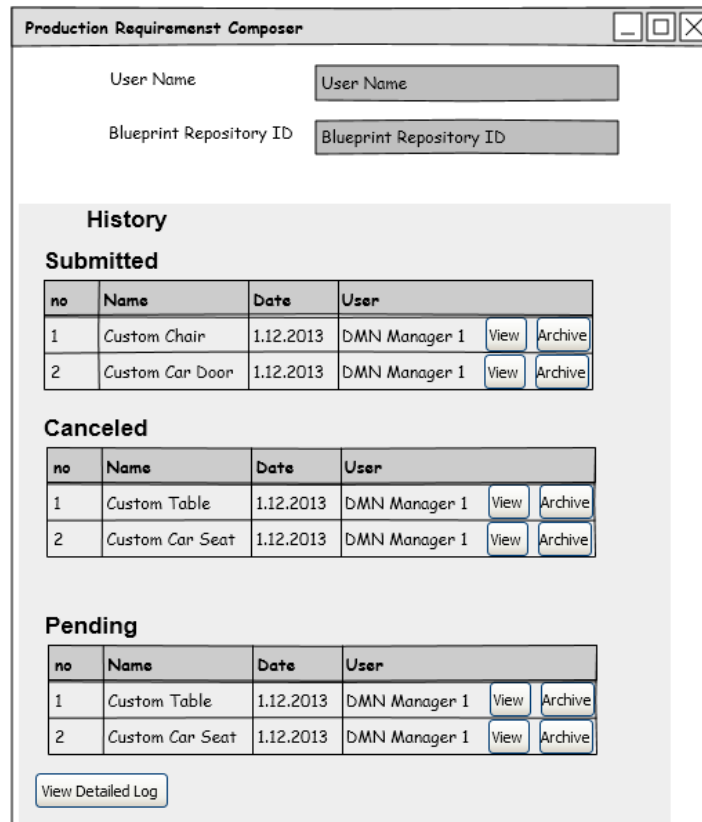


no	Production Requirements Name	Edit	Cancel
1	Custom Chair	Edit	Cancel
2	Custom Car Door	Edit	Cancel

**Figure 5-1: Production Requirements Composer – Main Screen**

#### 5.1.1.2 History Screen

The History Screen allows the DMN Manager to see the latest Production Requirements that have been submitted to the Partner Search component, that are Pending for submission (or cancelation) and the ones that have been canceled. It also provides access to the Detailed Log screen.



Production Requirements Composer

User Name

Blueprint Repository ID

### History

#### Submitted

no	Name	Date	User		
1	Custom Chair	1.12.2013	DMN Manager 1	<a href="#">View</a>	<a href="#">Archive</a>
2	Custom Car Door	1.12.2013	DMN Manager 1	<a href="#">View</a>	<a href="#">Archive</a>

#### Canceled

no	Name	Date	User		
1	Custom Table	1.12.2013	DMN Manager 1	<a href="#">View</a>	<a href="#">Archive</a>
2	Custom Car Seat	1.12.2013	DMN Manager 1	<a href="#">View</a>	<a href="#">Archive</a>

#### Pending

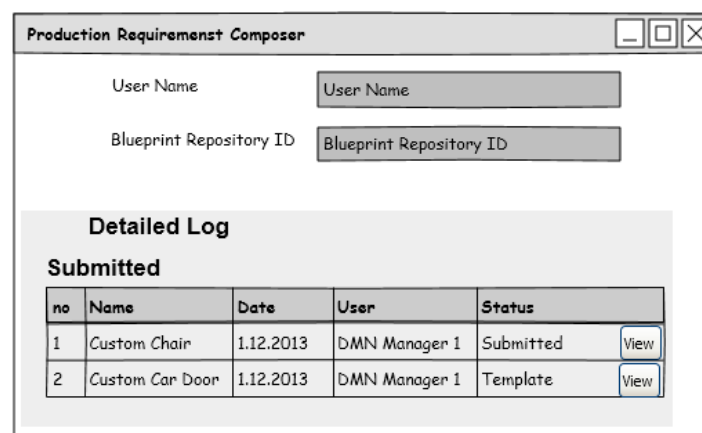
no	Name	Date	User		
1	Custom Table	1.12.2013	DMN Manager 1	<a href="#">View</a>	<a href="#">Archive</a>
2	Custom Car Seat	1.12.2013	DMN Manager 1	<a href="#">View</a>	<a href="#">Archive</a>

[View Detailed Log](#)

**Figure 5-2: Production Requirements Composer – History Screen**

### 5.1.1.3 Detailed Log Screen

The Detailed Log Screen allows the DMN Manager to see all the activity that has been performed via the Production Requirements Composer. This enables a complete overview over all the actions performed using the Production Requirements Composer.



Production Requirements Composer

User Name

Blueprint Repository ID

### Detailed Log

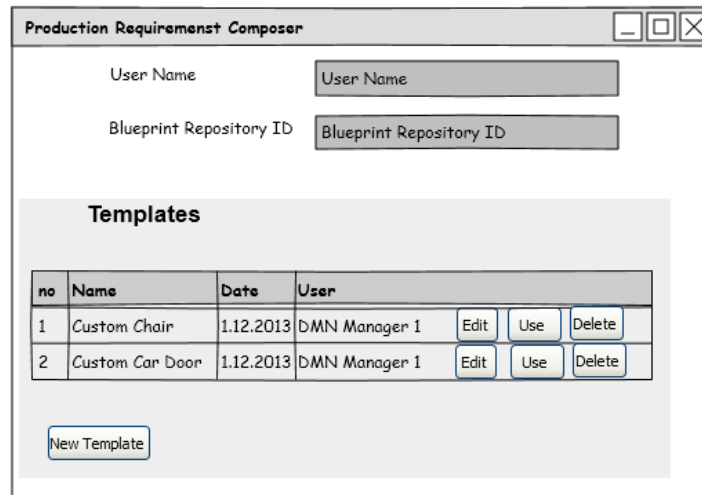
#### Submitted

no	Name	Date	User	Status	
1	Custom Chair	1.12.2013	DMN Manager 1	Submitted	<a href="#">View</a>
2	Custom Car Door	1.12.2013	DMN Manager 1	Template	<a href="#">View</a>

**Figure 5-3: Production Requirements Composer – Detailed Log Screen**

#### 5.1.1.4 Templates Screen

The Templates Screen allows the DMN Manager to create, view, edit, and use Production Requirements that can serve as templates. These are reusable configurable Production Requirements that allow the DMN Manager to save frequently used Production Requirements for the creation of DMNs thus increasing productivity by minimising operations that would have been otherwise repeated.



no	Name	Date	User			
1	Custom Chair	1.12.2013	DMN Manager 1	Edit	Use	Delete
2	Custom Car Door	1.12.2013	DMN Manager 1	Edit	Use	Delete

Figure 5-4: Production Requirements Composer – Templates Screen

## 5.2 Partner Search

### 5.2.1 Listing Products

Product Listing allows a DMN manager to see all products currently in his system. Products are divided into 4 parts:

- **New Products:** Displays products that are newly created by PRC and sent to PSC in order to create long and short lists.
- **Products in Process:** Products that are currently being worked on. An authorized person started creating lists for these products but they are not sent for evaluation yet.
- **Sent for Evaluation:** Products that are sent for final evaluation phase.
- **Archived Products:** Products that are removed from other three lists.

New Products			▲
Product Name	Creation Date	DMN	
DemoRepository12.brc1.ProductionRequirementsProduct.A.4deab...	10-12-2013	DMN-DemoRepository12.brc1.ProductionRequirementsProduct.A.4dea...	
DemoRepository12.brc1.ProductionRequirementsProduct.A.325cb...	09-12-2013	DMN-DemoRepository12.brc1.ProductionRequirementsProduct.A.325c...	
Page <input type="text" value="1"/> /1 <input type="button" value="Send to Long Listing"/> <input type="button" value="Send to Short Listing"/> <input type="button" value="Archive"/>			
Products in Process ▼			
Products Sent for Evaluation ▼			
Archived Products ▼			

**Figure 5-5: Product Lists**

## 5.2.2 Listing Requirements

Once a product is selected for Long List or Short List Creation, its requirements are listed with their details. DMN manager selects each requirement and sends them to create appropriate list.

Required Equipment					▲
Requirement Name	Capacity		Long List Created?	Has Partner Update?	
	Amount	Unit			
Packaging Industrial Machine	22.0	Pieces per Hour	Created	false	
Powder Coating Station	20.0	Pieces per Hour	Created	true	
Required Material ▼					
Required Process ▼					
Required Skill ▼					

**Figure 5-6: Listing Production Requirements**

## 5.2.3 Dynamic Criteria for Partner Search and List Creation

List creation screen will provide DMN managers flexibility for the search where they will be able to pick criteria of their choice which belong to related requirement and search accordingly.

## Partner Long Listing for Equipment

**Search Partners for:** Powder Coating Station

**Description:** Powder coating station and oven

### Production Requirements

**Product Category**

Selection...

**Add Criteria:** Select a criteria ... Add

### Partner Requirements

**Partner Category**

Selection...

**Location**

Istanbul Athens

**Add Criteria:** Select a criteria ... Add

Search Partners

**Figure 5-7: Dynamic Partner Search Criteria**

#### 5.2.4 Listing Partner Search Results

Improved UI for partner search results will allow DMN managers to see partners more accurately and sort them using any criteria after results are displayed.

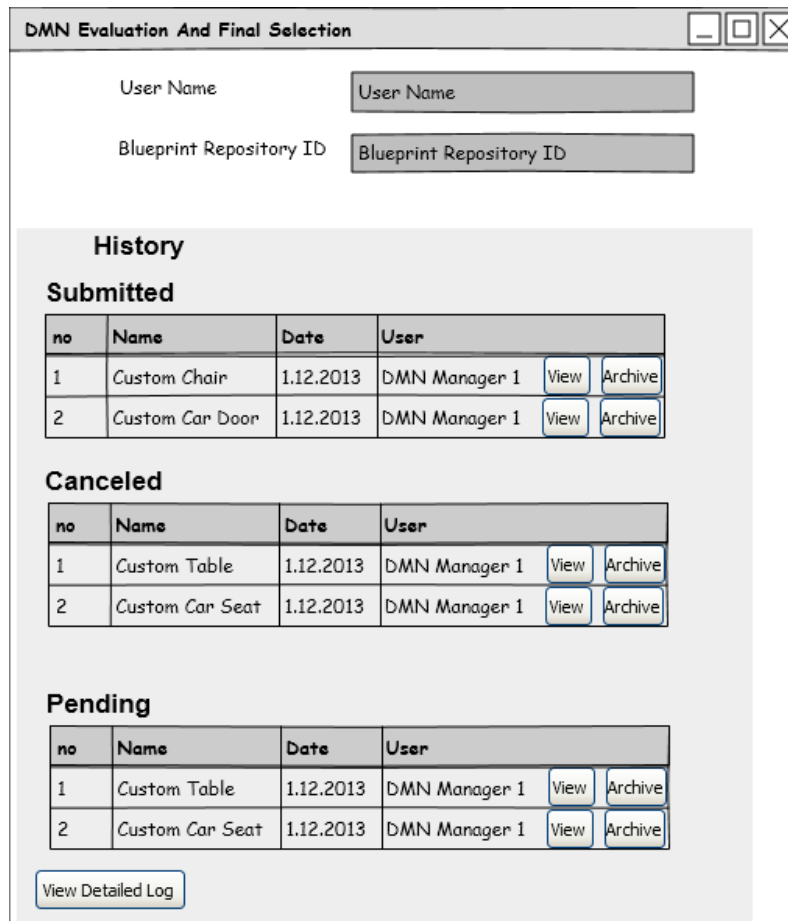
Search Results						
Company Name	Company Category	Location	Company Provides		Annual Turnover	Number of Employees
			Equipment	Category		
Seating Design S.L.	Manufacturer	Istanbul	Powder Coating Station	Painting	8.000000	35
Iron Structures Manufacturing S.A.	SME	Athens	Powder Coating Station	Painting	12.000.000	50
Aluminium Structures Manufacturing S.A.	SME	Athens	Powder Coating Station	Painting	10.000.000	40

**Figure 5-8: Search Result Lists**

## 5.3 DMN Evaluation and Final Selection

### 5.3.1.1 History Screen

The History Screen allows the DMN Manager to see the latest DMN Shortlists that have been submitted to the DMN Evaluation and Final Selection component, that are Pending for submission (or cancelation) and the ones that have been canceled. It also provides access to the Detailed Log screen.



**DMN Evaluation And Final Selection**

User Name

Blueprint Repository ID

**History**

**Submitted**

no	Name	Date	User
1	Custom Chair	1.12.2013	DMN Manager 1 <a href="#">View</a> <a href="#">Archive</a>
2	Custom Car Door	1.12.2013	DMN Manager 1 <a href="#">View</a> <a href="#">Archive</a>

**Canceled**

no	Name	Date	User
1	Custom Table	1.12.2013	DMN Manager 1 <a href="#">View</a> <a href="#">Archive</a>
2	Custom Car Seat	1.12.2013	DMN Manager 1 <a href="#">View</a> <a href="#">Archive</a>

**Pending**

no	Name	Date	User
1	Custom Table	1.12.2013	DMN Manager 1 <a href="#">View</a> <a href="#">Archive</a>
2	Custom Car Seat	1.12.2013	DMN Manager 1 <a href="#">View</a> <a href="#">Archive</a>

[View Detailed Log](#)

**Figure 5-9: DMN Evaluation and Final Selection- History Screen**

### 5.3.1.2 Detailed Log Screen

The Detailed Log Screen allows the DMN Manager to see all the activity that has been performed via the the DMN Evaluation and Final Selection component. This enables a complete overview over all the actions performed using the DMN Evaluation and Final Selection component.

DMN Evaluation And Final Selection

User Name

User Name

Blueprint Repository ID

Blueprint Repository ID

Detailed Log

Submitted

no	Name	Date	User	Status	
1	Custom Chair	1.12.2013	DMN Manager 1	Submitted	<a href="#">View</a>
2	Custom Car Door	1.12.2013	DMN Manager 1	Submitted	<a href="#">View</a>

Figure 5-10: DMN Evaluation and Final Selection – Detailed Log Screen

## 5.4 DMN Design Toolset

### 5.4.1.1 Overall Orchestration View

The DMN Design Toolset will provide a comprehensive overview of all processes, skills, materials and equipment that are required for a specific DMN. This screen will also allow editing the orchestrations by giving access to the appropriate editor screens. The Overall Orchestration Screen aims to increase productivity by giving a more intuitive visualization and organization of the orchestration information.

Design Toolset - Orchestration

Overall Orchestration

Requirement Name	Requirement Name	Scheduled Start Date	Scheduled End Date
Cutting Machine	Equipment	14.02.2013	14.08.2013
Wood Sheet	Material	14.02.2013	14.08.2013
Cuttin Machine	Process	15.08.2013	14.11.2013
Wood Carving	Skill	15.08.2013	14.11.2013
Packaging Machine	Equipment	14.11.2013	14.12.2013

Save

Figure 5-11: DMN Design Toolset – Overall Orchestration View

#### 5.4.1.2 Production Measurements Editor

The Quality Assurance Editor will be enhanced with the required functionality that allows for the editing and configuration of the Production Measurement KPIs that have been defined in detail in iMAGINE and iMAGINE Enlarged Platform Architecture, version 3.

Design Toolset - Quality Assurance - Production Measurements

**Define QA Thresholds**

Product 1 > Product 1.1 > Product 1.1.1 > Equipment 1

Description:

Required Equipment Capacity:

**Equipment Attributes Requirement**

no	Name	Value	Unit
1	Max Shaft Processing Length	110	mm

**Equipment Available Metrics (by Supplier)**

Name	MinValue	AverageValue	MaxValue	Unit	Production Measurement Type	Coefficient Value	MinValue	TargetedValue	MaxValue
Utilisation	0	50	100	%	Production Efficiency	0.70	-	30	70
MTBF	1000	1500	2000	weeks	Production Efficiency	0.30	1000	1500	-

**Equipment Production Measurement Coefficients (by DMN Manager)**

Production Measurement Type	Coefficient Value
Production Efficiency	0.25
Production Quality	0.25
Perfect Order Fulfilment	0.25
Production Equipment Effectiveness	0.25

**Figure 5-12: Design Toolset Production Measurements Editor**

#### 5.4.1.3 History Screen

The History Screen allows the DMN Manager to see the latest DMN network configurations that have been submitted to the DMN Design Toolset, that are Pending for Execution (or rejection) and the ones that have been rejected. It also provides access to the Detailed Log screen with more detailed information.



Design Toolset

User Name

Blueprint Repository ID

### History

#### Submitted

no	Name	Date	User		
1	Custom Chair	1.12.2013	DMN Manager 1	View	Archive
2	Custom Car Door	1.12.2013	DMN Manager 1	View	Archive

#### Canceled

no	Name	Date	User		
1	Custom Table	1.12.2013	DMN Manager 1	View	Archive
2	Custom Car Seat	1.12.2013	DMN Manager 1	View	Archive

#### Pending

no	Name	Date	User		
1	Custom Table	1.12.2013	DMN Manager 1	View	Archive
2	Custom Car Seat	1.12.2013	DMN Manager 1	View	Archive

View Detailed Log

Figure 5-13: DMN Design Toolset - History Screen

#### 5.4.1.4 Detailed Log Screen

The Detailed Log Screen allows the DMN Manager to see all the activity that has been performed via the the DMN Design Toolset. This enables a complete overview over the key actions can be performed using the DMN Design Toolset.

Design Toolset

User Name

Blueprint Repository ID

### Detailed Log

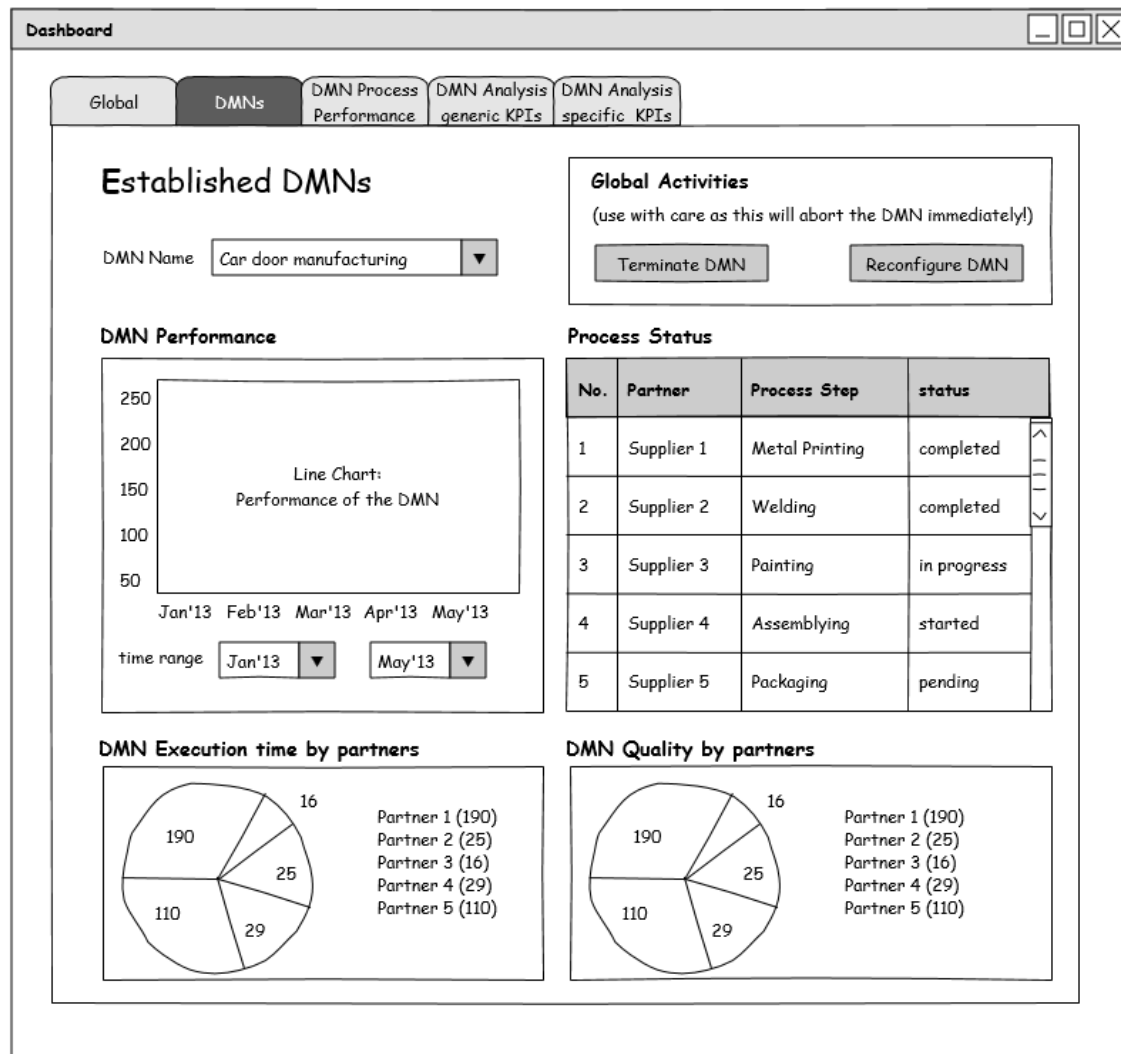
#### Submitted

no	Name	Date	User	Status	
1	Custom Chair	1.12.2013	DMN Manager 1	Submitted	View
2	Custom Car Door	1.12.2013	DMN Manager 1	Rejected	View

Figure 5-14: DMN Design Toolset – Detailed Log Screen

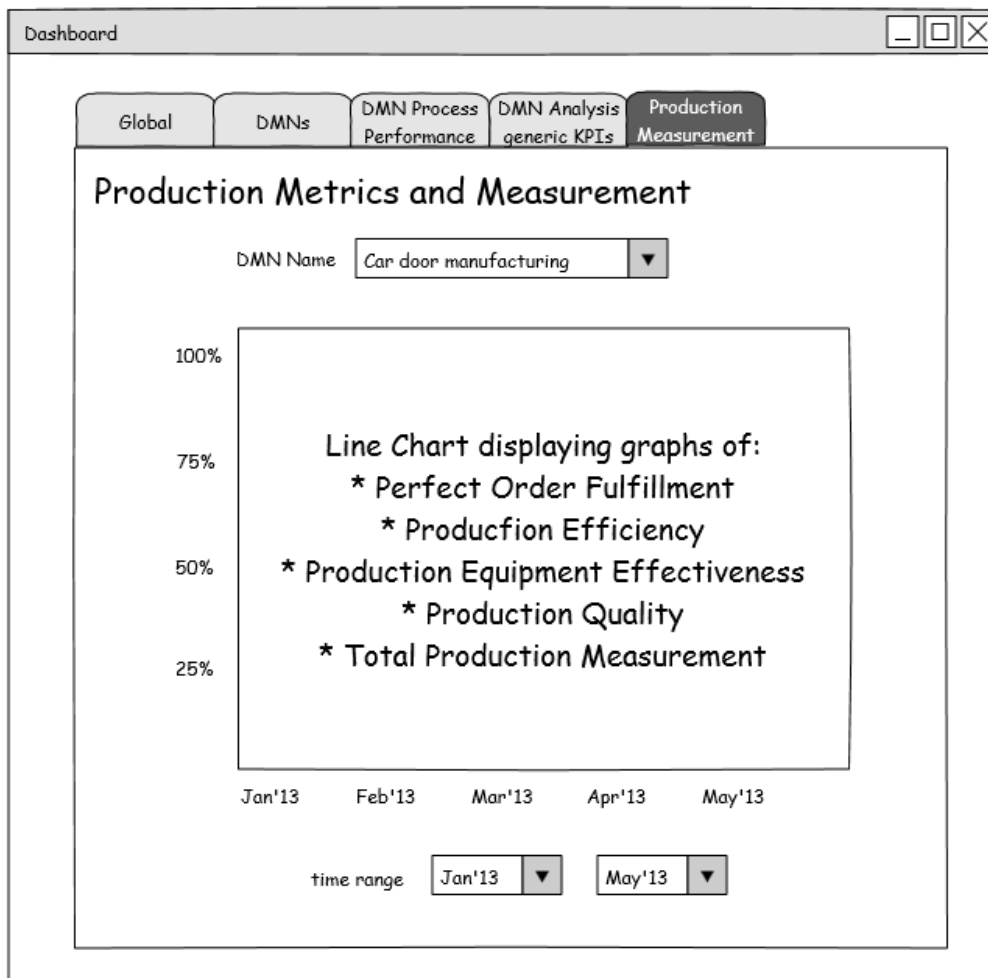
## 5.5 Dashboard

The Dashboard view on DMNs is extended by a global activity box containing two buttons for “Terminate DMN” and “Reconfigure DMN”. Buttons are only available if previously a specific DMN has been selected from the drop-down menu. The two additional buttons are explained in detail in Section 4.3.2. A draft of the user interface design is shown in Figure 5-15.



**Figure 5-15: Extension of the DMN Dashboard**

Production metrics and measurements generated by the component introduced in Section 4.7.1.1 is displayed in the *Production Measurement* tab of the dashboard (cf. Figure 5-16). Upon selection of a DMN, the user obtains graphs for the available production metrics, taken from the NAGIOS MySQL Database. With the use of the drop-down menus a specific time range can be selected to investigate a certain point in time in more detail.

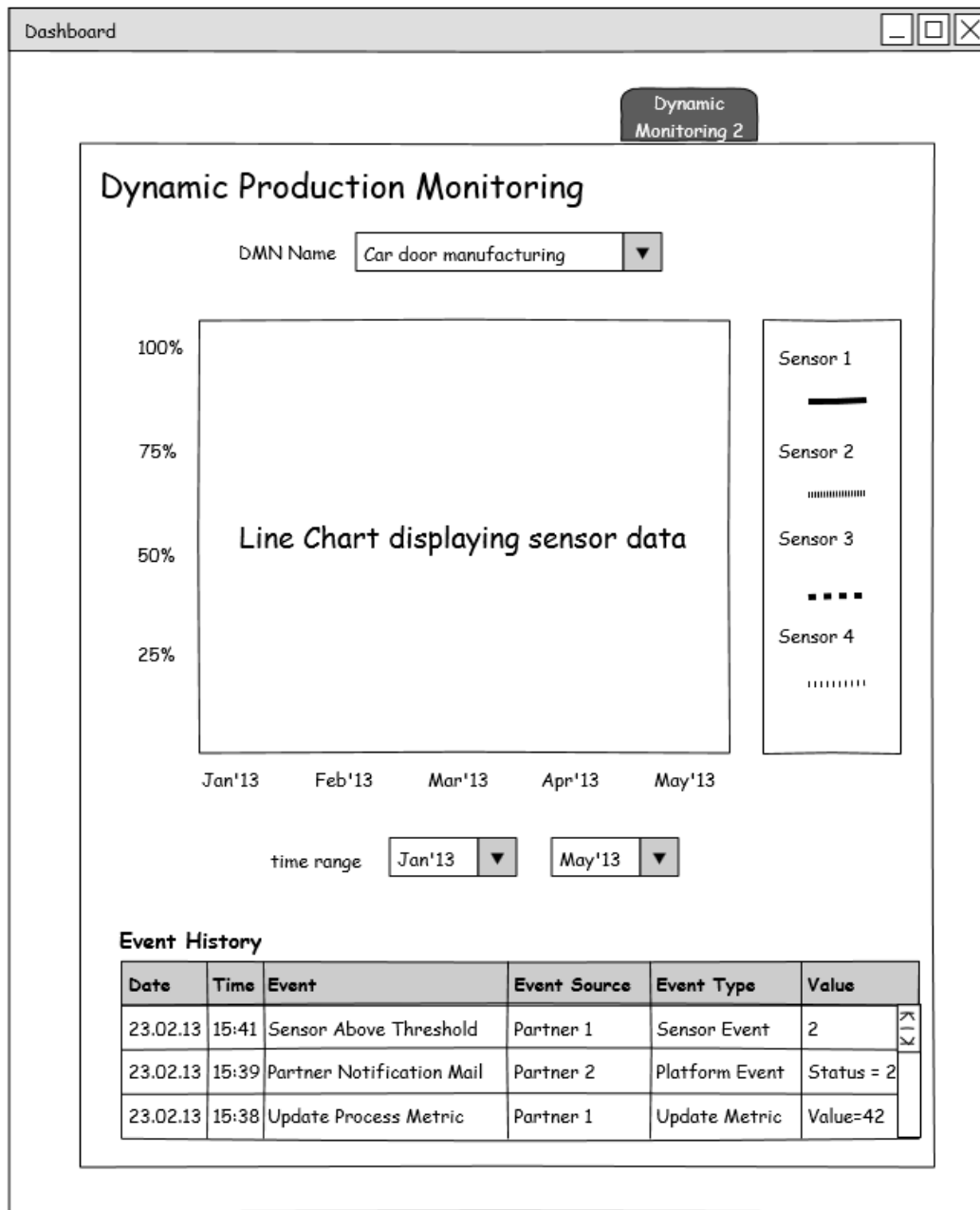


**Figure 5-16: Production Metrics and Measurement**

## 5.6 iMAGINE-Enlarged Advanced visualization

The Dynamic Monitoring is enabled by the dashboard in the “Dynamic Monitoring” tab which offers the selection of a specific DMN by the drop-down menu provided. By using a defined DMN Network, available sensor data is displayed. Underlying this graphs representation are events which are processes in real-time and allow the DMN manager an up-to-date view of available sensor data.

In addition a history of cached events is displayed by the table below which includes the date and time of the event, the event name, event source and type, and the concrete value of the captured event.



**Figure 5-17: Dynamic Monitoring Dashboard**

As the "Advanced Dashboarding" is mainly driven by the work in work package 8, there will be more user interfaces available in the future which will be provided in the upcoming tasks.

## **6 Conclusion**

---

### **6.1 Summing up**

This deliverable concludes the IMAGINE Detailed Design by translating the IMAGINE architecture v3 into a design needed for the upcoming implementation of the IMAGINE Platform R3.

As the main contribution of this work, the component overview is refreshed by introducing aspects from the event driven architecture and reflecting the IMAGINE\_enlarged production analysis and dynamic monitoring toolset. This document further strives to introduce a service for termination and reconfiguration existing DMNs interfacing the execution phase with the partner search component. As living labs experienced a need for a more sophisticated search on close-to-date partner capacities, the existing search engine is extended to better account for current partner capabilities by the Availability Search Engine.

With respect to the execution phase, there are manifold changes which e.g., offer the use of production metrics and measures based on KPIs. The largest impact however is introduced with the production analysis and dynamic monitoring tool set based on the IMAGINE enlarged components aligned with the current development of the IMAGINE platform.

### **6.2 Future work and Outlook**

In the next step the proposed design is implemented in the IMAGINE Platform R3 which is the final version of the IMAGINE generic platform. As this platform will be made public to all living lab partners, they can take full advantage of the current implementation by introducing customized adapters and services which are necessary to support their use cases.

---

## Annex A: References

---

- [1] Software Design Description, IEEE 1016-2009 Standard; Recommended Practice for Software Design Descriptions; Reference 2, <http://standards.ieee.org/findstds/standard/1016-2009.html>
- [2] SPARQL Query Results XML Format, W3C Recommendation 15 January 2008, <http://www.w3.org/TR/rdf-sparql-XMLres/>
- [3] SPARQL 1.1 Update, W3C Proposed Recommendation 08 November 2012, <http://www.w3.org/TR/sparql11-update/>
- [4] SPARQL Query Language for RDF W3C Recommendation 15 January 2008, <http://www.w3.org/TR/rdf-sparql-query/>
- [5] Apache Jena Official Web Site, <http://jena.apache.org/>, accessed 01.03.2013
- [6] Akao, Yoji (1994). "Development History of Quality Function Deployment". The Customer Driven Approach to Quality Planning and Deployment. Minato, Tokyo 107 Japan: Asian Productivity Organization. p. 339. ISBN 92-833-1121-3.
- [7] SixSigma magazine, <http://www.isixsigma.com/dictionary/QFD-103.htm>
- [8] A. Freier, P. Karlton, P. Kocher, The Secure Sockets Layer (SSL) Protocol Version 3.0, Internet Engineering Task Force (IETF), RFC 6101, August 2011
- [9] Prohibiting Secure Sockets Layer (SSL) Version 2.0, S. Turner, T. Polk, Internet Engineering Task Force (IETF), RFC 6176, March 2011

## Annex B: Organization Customization Parameters

The following table offers a configuration per Organizations introducing multi-client capability.

**Table 6-1: Organization Customization Parameters**

Parameter Name	Description
BP_REPOSITORY_WS_SERVER_IP_N_PORT	<b>The address</b> and port of the server hosting the Blueprint Repository that will be used by this Organization. E.G. 127.0.0.1:8080
BP_REPOSITORY_ID_MASTER_PASSWORD	The master password for the Blueprint Repository ID that will be used by this Organization.
BP_REPOSITORY_ID	The Blueprint Repository ID that will be used by this Organization.
BP_REPOSITORY_USERNAME_FOR_PRC	The username for the Blueprint Repository ID that will be used by this Organization's Production Requirements Composer.
BP_REPOSITORY_PASSWORD_FOR_PRC	The password for the Blueprint Repository ID that will be used by this Organization's Production Requirements Composer.
BP_REPOSITORY_USERNAME_FOR_PSC	The username for the Blueprint Repository ID that will be used by this Organization's Partner Search Component.
BP_REPOSITORY_PASSWORD_FOR_PSC	The password for the Blueprint Repository ID that will be used by this Organization's Partner Search Component.
BP_REPOSITORY_USERNAME_FOR_FEC	The username for the Blueprint Repository ID that will be used by this Organization's DMN evaluation and Final Selection Component.
BP_REPOSITORY_PASSWORD_FOR_FEC	The password for the Blueprint Repository ID that will be used by this Organization's DMN evaluation and Final Selection Component.
BP_REPOSITORY_USERNAME_FOR_DTC	The username for the Blueprint Repository ID that will be used by this Organization's Design Toolset Component.
BP_REPOSITORY_PASSWORD_FOR_DTC	The password for the Blueprint Repository ID that will be used by this Organization's Design Toolset Component.
BP_REPOSITORY_USERNAME_FOR_DMC	The username for the Blueprint Repository ID that will be used by this Organization's Dynamic Monitoring Components.

BP_REPOSITORY_PASSWORD_FOR_DMC	The password for the Blueprint Repository ID that will be used by this Organization's Dynamic Monitoring Components.
BP_REPOSITORY_USERNAME_FOR_EPC	The username for the Blueprint Repository ID that will be used by this Organization's Execution Phase Components.
BP_REPOSITORY_PASSWORD_FOR_EPC	The password for the Blueprint Repository ID that will be used by this Organization's Execution Phase Components.
PRODUCTION_REQUIREMENT_COMPONENT_WS_IP_N_PORT	The address and port of the server hosting the Production Requirements Component Web Service that will be used by this Organization. E.G. 127.0.0.1:8080
PRODUCTION_REQUIREMENT_COMPONENT_WS_USERNAME	The username of the Production Requirements Component Web Service that will be used by this Organization.
PRODUCTION_REQUIREMENT_COMPONENT_WS_PASSWORD	The password of the Production Requirements Component Web Service that will be used by this Organization.
PARTNER_SEARCH_WS_COMPONENT_IP_N_PORT	The address and port of the server hosting the Partner Search Web Service that will be used by this Organization. E.G. 127.0.0.1:8080
PARTNER_SEARCH_WS_COMPONENT_USER_NAME	The username of the Partner Search Web Service that will be used by this Organization.
PARTNER_SEARCH_WS_COMPONENT_PASSWORD	The password of Partner Search Web Service that will be used by this Organization.
DMN_EVALUATION_FINAL_SELECTION_COMPONENT_IP_N_PORT	The address and port of the server hosting the DMN Evaluation and Final Selection Web Service that will be used by this Organization. E.G. 127.0.0.1:8080
DMN_EVALUATION_FINAL_SELECTION_COMPONENT_USERNAME	The username of the DMN Evaluation and Final Selection Web Service that will be used by this Organization.
DMN_EVALUATION_FINAL_SELECTION_COMPONENT_PASSWORD	The password of the DMN Evaluation and Final Selection Web Service that will be used by this Organization.
KPI_LOG_AND_MONITORING_WS_IP_N_PORT	The address and port of the server hosting the DMN KPI Log and Monitoring Web Service that will be used by this Organization. E.G. 127.0.0.1:8080
KPI_LOG_AND_MONITORING_WS_USERNAME	The username of the DMN KPI Log and Monitoring Web Service that will be used by this Organization.
KPI_LOG_AND_MONITORING_WS_PASSWORD	The password of the DMN KPI Log and Monitoring Web Service that will be used by this Organization.
DMN_DESIGN_TOOLSET_WS_IP_N_PORT	The address and port of the server hosting the DMN



	Design Toolset Web Service that will be used by this Organization. E.G. 127.0.0.1:8080
DMN_DESIGN_TOOLSET_WS_USERNAME	The username of the DMN Design Toolset Web Service that will be used by this Organization.
DMN_DESIGN_TOOLSET_WS_PASSWORD	The password of the DMN Design Toolset Web Service that will be used by this Organization.
PRODUCTION_REPOSITORY_DATABASE_IP_N_PORT	The address and port of the server hosting the DMN Production Repository Database that will be used by this Organization. E.G. 127.0.0.1:8080
PRODUCTION_REPOSITORY_DATABASE_NAME	The database name of Production Repository Database that will be used by this Organization.
PRODUCTION_REPOSITORY_DATABASE_USERNAME	The database username of the Production Repository Database that will be used by this Organization.
PRODUCTION_REPOSITORY_DATABASE_PASSWORD	The database password of the Production Repository Database that will be used by this Organization.
PRODUCTION_REQUIREMENT_COMPOSER_DATABASE_IP_N_PORT	The address and port of the server hosting the DMN Production Requirements Composer Database that will be used by this Organization. E.G. 127.0.0.1:8080
PRODUCTION_REQUIREMENT_COMPOSER_DATABASE_NAME	The database name of the Production Requirements Composer Database that will be used by this Organization.
PRODUCTION_REQUIREMENT_COMPOSER_DATABASE_USERNAME	The database username of the Production Requirements Composer Database that will be used by this Organization.
PRODUCTION_REQUIREMENT_COMPOSER_DATABASE_PASSWORD	The database password of the Production Requirements Composer Database that will be used by this Organization.
PARTNER_SEARCH_DATABASE_IP_N_PORT	The address and port of the server hosting the Partner Search Database that will be used by this Organization. E.G. 127.0.0.1:8080
PARTNER_SEARCH_DATABASE_NAME	The database name of the Partner Search Selection Database that will be used by this Organization.
PARTNER_SEARCH_DATABASE_USERNAME	The database username of the Partner Search Database that will be used by this Organization.
PARTNER_SEARCH_DATABASE_PASSWORD	The database password of the Partner Search Database that will be used by this Organization.
DMN_EVALUATION_FINAL_SELECTION_DATABASE_IP_N_PORT	The address and port of the server hosting the DMN Evaluation and Final Selection Database that will be used by this Organization. E.G. 127.0.0.1:8080

DMN_EVALUATION_FINAL_SELECTION_DATABASE_NAME	The database name of the DMN Evaluation and Final Selection Database that will be used by this Organization.
DMN_EVALUATION_FINAL_SELECTION_DATABASE_USERNAME	The database username of the DMN Evaluation and Final Selection Database that will be used by this Organization.
DMN_EVALUATION_FINAL_SELECTION_DATABASE_PASSWORD	The database password of the DMN Evaluation and Final Selection Database that will be used by this Organization.
DESIGN_TOOLSET_DATABASE_IP_N_PORT	The address and port of the server hosting the DMN Design Toolset Database that will be used by this Organization. E.G. 127.0.0.1:8080
DESIGN_TOOLSET_DATABASE_NAME	The database name of the DMN Design Toolset Database that will be used by this Organization.
DESIGN_TOOLSET_DATABASE_USERNAME	The database username of the DMN Design Toolset Database that will be used by this Organization.
DESIGN_TOOLSET_DATABASE_PASSWORD	The database password of the DMN Design Toolset Database that will be used by this Organization.

## Annex C: List of Acronyms

---

API: application programming interface  
BPM: Business Process Management  
CMS: Content Management System  
cPLM: collaborative Product Lifecycle Management  
DMN: Dynamic Manufacturing Network  
ERP: Enterprise Resource Planning  
ESB: Enterprise Service Bus  
HTTP: Hypertext Transfer Protocol  
ICT: Information & Communication Technology  
JSP: JavaServer Pages  
JMS: Java Messaging Service  
MES: Manufacturing Execution System  
MRP: Material Resource Planning  
OS: Operating System  
PLM: product lifecycle management  
PTP: point-to-point  
QFD: Quality function deployment  
QoS: Quality of Service  
RDBMS: relation database management system  
RDF: Resource Description Framework  
RDFS: Resource Description Framework Schema  
SCM: Supply Chain Management  
SDD: Software Design Description  
SOA: Service Oriented Architecture  
SOAP: Simple Object Access Protocol  
SPARQL: SPARQL Protocol and RDF Query Language  
SSH: secure shell  
SSL: Secure Socket Layer  
UML: Unified Modeling Language  
UUID: Universally Unique Identifier  
VPN: virtual private network  
W3C: World Wide Web Consortium  
XML: Extensible Markup Language