ITEA 2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

CREATE

Deliverable 2.1: CREATE Software Architecture

CREATE

Creating Evolution Capable Co-operating Applications in Industrial Automation

Project number: ITEA 2 ip10020

Edited by: Vadim Chepegin (TIE Nederland b.v.)

Contributors: Fernando Perales (TRIMEK) and Silvia de la Maza (INNOVALIA)

Date: 29.08.2012

Document version no.: *0.8*

Revision History

Date	Version	Description	Author
6/12/2011	0.1	Major Use-Cases : the key requirements and initial draft of a software architecture	Vadim Chepegin
24/01/2012	0.2	Revision based on the comments of Mr. Stuart Campbell	Vadim Chepegin
	0.3 – 0.5	Improving content of the document and integrate the latest results of the development into it	Vadim Chepegin
07/08/2012	0.6	Contributions of Spanish consortium	Silvia de la Maza Fernando Perales
13/08/2012	0.7	Aggregation comments from partners, adjusting styles between deliverables, moving technologies review into deliverable D2.2	Vadim Chepegin
30/08/2012	0.8	Final format edition	Fernando Perales

INDEX

LIST OF FIGURES

1. Introduction

CREATE will develop innovative solutions supporting the seamless integration of manual processes into efficient industrial systems through self-configuring, flexible and evolutionary techniques.

CREATE innovates, develop and deliver proof of concepts in the domain of industrial automation and establish cooperative and layered automation systems with a significant exploitation potential. It will deploy technologies in a modular form which have business impact by increasing the flexibility and adaptation of existing systems and allowing their integration with new paradigms such as human-centred design, service orientation, secure and safe distributed control architectures, semantic use, dynamic legacy integration and control system life-cycle engineering support.

It will do this in the areas of material flow, monitoring & QA, metrology, and product tracking.

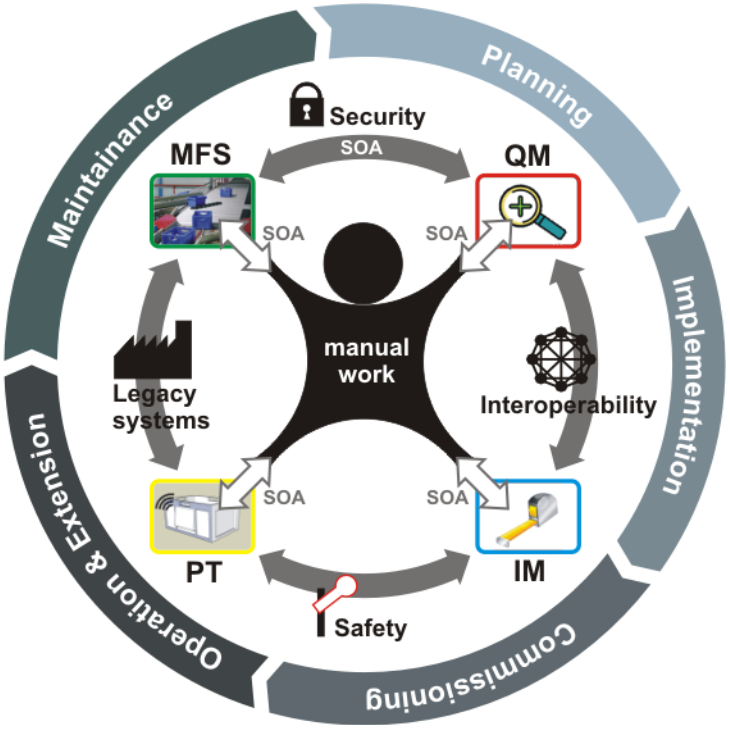

Figure 1: *CREATE*, "the big picture"

This document provides a high level overview of CREATE software architecture based on the understanding of business and technical requirements elicited from use cases presented by different national consortia Dutch, Spanish and Swedish.

The document provides a high-level description of the goals of the architecture, the use cases supported by the system and architectural styles and components that have been selected to best achieve the use cases. This framework then allows for the development of the design criteria and documents that define the technical and domain specifications in detail.

1.1 Purpose

This Software Architecture Document (SAD) provides an architectural overview of CREATE in order to capture service level requirements of the platform. It is intended to capture and convey the significant architectural decisions which have been made on the system.

The architecture is closely coupled with the three general project goals: flexible choreography, integration of manual work and development of engineering methodologies and tools.

Flexible choreography with dynamic service composition at runtime:

1. Determine how fundamental and higher-level services are defined
2. Define machine-readable rules and design patterns for service composition
3. Design automatic service binding mechanism (FIND – EXPLORE, NEGOTIATE – BIND, CO–OPERATE) to compose service ensembles at runtime based on predefined rules and design patterns.
4. The service set must be extensible by new services, rules and design patterns.

Integration of manual work:

1. Description and integration of services that involve manual work steps fulfilled by human workers.
2. Defining appropriate integration patterns for humans. For example, design of a general SOA/HMI-mediator concept. A mediator translates service calls into human-readable working instructions and translates the human feedback into service responses or events.
3. Consideration of safety-related and ergonomic aspects relevant to services including human workers

In order to depict the software as accurately as possible, the structure of this document is based on the “4+1” model view of architecture [KRU41].

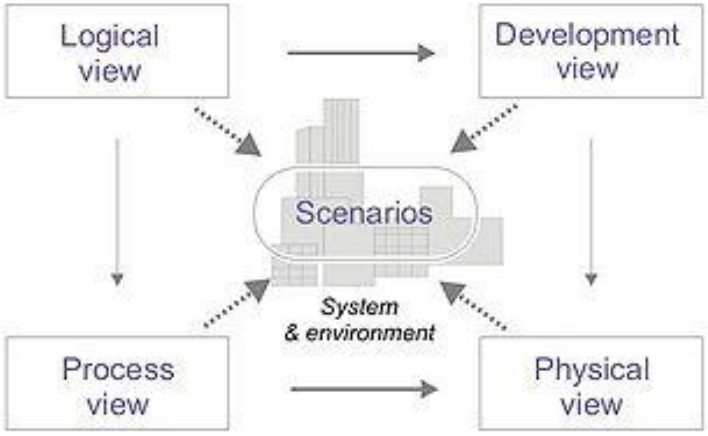

Figure 2: "4+1" approach to architecture documentation

The "4+1" View Model allows various stakeholders to find what they need in the software architecture.

1.2 Scope

This SAD presents the structure and behaviour of the entire software stack on the high level. It only covers the first level of decomposition of the CREATE platform into its major components. Further decomposition of the major components themselves will be included in the High-Level Design, which also includes the APIs.

1.3 Abbreviations, Acronyms and Definitions

Abbreviation	Description
MySQL	Free Relational Database Management System (RDBMS)
HTTP	Hypertext Transfer Protocol
WWW	World Wide Web
Tomcat	Web Server that enables Java technologies

SAD	Software Architecture Document
RUP	Rational Unified Process
UML	Unified Modeling Language
SOA	Service Oriented Architecture

Term	Definitions
Polyglot application	Application that can work under different programming languages
Cloud	Ubiquitous network access
Generic user	User with no special knowledge or rights
Administrator	User with platform maintenance tasks
Operator	User that interacts with the system when it is running

1.4 References

[CRT-DOW]: ITEA2 CREATE ip10020 amended FPP, December 2011.

[INT-GSC]: Interfaces of the Generic Substrate Carrier.

[KRU41]: The "4+1" view model of software architecture, Philippe Kruchten, November 1995, (http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf).

[RWAES] S. Meyer, K. Sperner, C. Magerkurth "Towards Real-World Aware Enterprise Systems", IEEE MASS 2011, Valencia, Spain.

[IoT-IRM] Internet-of-Things Architecture IoT-A Project Deliverable D1.2 – Initial Architectural Reference Model for IoT (http://www.iot-a.eu/public/public-documents/documents-1/1/1/d1.2/at_download/file).

2. Software architecture overview

CREATE architecture follows principles of state-of-the-art *SOA* and *Cloud* styles and guidelines for building state-of-the-art *polyglot* information platforms.

Support for web services and *REST* (Representational State Transfer) interfaces will allow CREATE to construct service compositions or so-called mashups. CREATE platform will be realized as an extension of the Enterprise Service Bus (ESB) concept which facilitates seamless distributed connectivity through binding components and service composition through service components (which may implement a specific service or logically compose services). To facilitate loose coupling and great concurrency between different threads, CREATE platform will communicate via asynchronous message passing which means senders can send messages even in the absence of receivers.

2.1 Goals and Objectives

The goal of the architecture is to describe, in sufficient detail, all of the software components, their responsibilities and behaviour, and their interfaces to allow a more detailed design to proceed forward. The architecture will not define the exact APIs. The APIs will be documented in the High Level Design specifications for each software component.

A goal of the architecture is to provide support for integrators of hardware systems, e.g. for printing shop floor, which consists of parts coming from multiple vendors to easily integrate them, monitor and (re)configure.

2.2 Compatibility and Interoperability

The architecture employs the use of industry standard interfaces, where available, for communication among the software and hardware components. This supports the greatest possible interoperability with third party hardware. A special interoperability software layer will be developed to:

- Support integration of a convey belt (Substrate Carrier by CCM) with different vendors' products
- Facilitate the integration and communication of dimensional, quality control machines.

2.3 Extensibility

The architecture places a requirement on each component to provide a mechanism for the user of the interface to determine the revision of an interface. This allows the user of an interface to determine if it is compatible with a particular implementation and allows the interface to a particular component to be extended over time.

2.3.1 Portability and Platform Independence

The architecture will promote portability of the client side (GUI) to various operating systems and platforms, e.g. Web, iOS (iPad, iPhone), Android, etc., following the principles of Cloud computing where user should be enabled to work seamlessly from different business and personal devices he has in possession. For portability of a GUI HTML5 and JavaScript will be used as much as possible.

For the server side it will be used mostly Java and JavaScript, which have virtual machines for most of the existing computer platforms and Operating Systems. However, due to the SOA nature of the architecture it will be possible to integrate software components developed with other technologies with reasonably low efforts.

In the case of metrological software, for example, the relevant data that is required to perform an efficient monitoring will be determined and adapted to the different operating systems and platforms.

As a general rule for portability the interface processing should be separated from the logic in different classes.

2.3.2 Interaction with human worker

The architecture will be designed in order to facilitate the integration and interaction with the human task. Every software interface has to be developed with the aim to improve the communication between the automated components of the system. These user interfaces will be implemented as user-friendly as possible guarantying that no mistakes will be caused due to an insufficient communication message.

Verification and validation test for the human – machine interactions are going to be done.

The attention will be specially focused on the way data are displayed (colours, interactive screens or traditional buttons, graphics, etc.) The goal is to inform as much as possible in the simplest way possible.

Moreover, individual safety aspects will be considered for each of the machines.

2.4 Key user roles on CREATE platform

In this document the term "system" will refer to the complex hardware and software system that includes e.g. CCM Substrate Carrier and other hardware components such as such as inspection machines, or other kind of sensors and actuators assembled together on a shop flow and involved in a production cycle.

The following key roles of users that will be interacting with CREATE platform have been identified.

- System integrator: a person who integrates different hardware parts into one production system. The same person is responsible for re-configuration of the hardware system.
- Operator: a person who interacts with the system when it is running in a production mode.
- Business user: a user who is interested in business aspects of the production system, e.g. costs, production value, time for maintenance, etc.
- Platform administrator: a person who maintain CREATE platform.

3. High level architecture: logical view

The ultimate goal and challenge of CREATE is revolving around the idea of the Smart Neighbourhood Module (SNM) which embodies an enhanced Multi-Level-Composite-Module approach. SNM aggregates associated production and transportation mechanics, dimensional quality control machines, control hardware as well as of automation software. It is mechanically coupled with its physically neighboured components. Moreover it is connected with the data network. The productive operations of an SNM contribute to production workflows and implement steps of material flows.

To implement a CREATE vision it is necessary to facilitate merging of the physical world with a virtual networked world. This could be illustrated with help of the graphics provided by the EU FP7 project called Internet of Things – Architecture that depicts the relationship between *services*, *resources,* and *devices* (Figure 3).

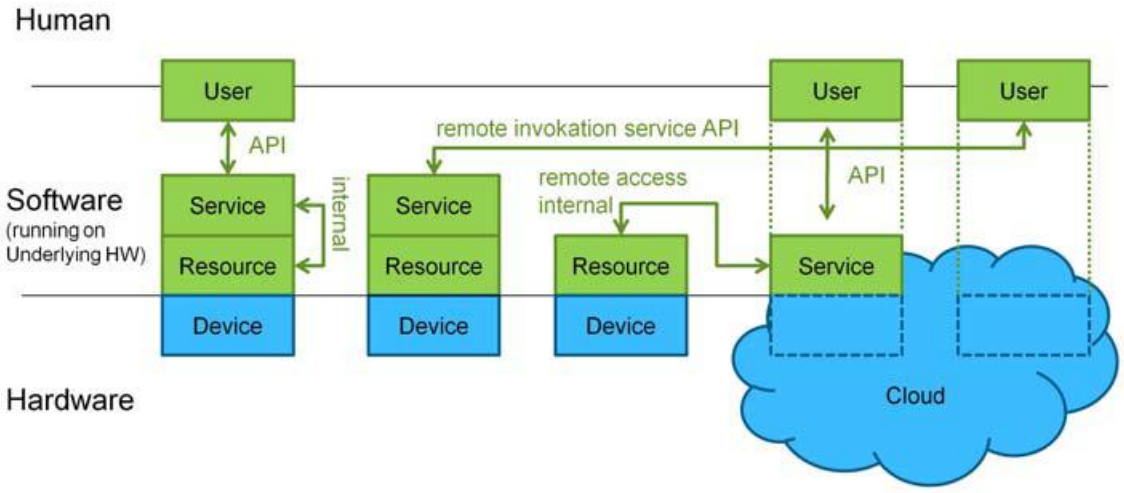

Figure 3: Devices, resources and services, from [IoT-IRM]

How it is beneficial for the companies such as CCM, DATAPIXEL or TRIMEK and what it means for them? Besides the software functions performing control and automation tasks, an SNM contains neighbourhood awareness and adaptation functions. They detect and identify the physically neighboured SNMs, their location, type and services. They adapt the services offered by the module to the current neighbourhood and assist the neighboured modules in their

neighbourhood recognition. For example, when an inspection machine is installed in a production line after a milling process it will analyse its surrounding machines and, taking into account multiple variables (such as the following processes, maximum time allowed for the inspection, required scanning resolution, etc), it can focus its inspection process on the most interesting part to be scanned. Once this part is scanned, it can be compared with a virtual pattern. Following the results of the comparison, different decisions can be taken on real time. These decisions can be on line monitored and easily changed, as well as the pattern for comparison.

By adding abstraction layers, e.g. resource and service descriptions to the hardware modules, CREATE will allow to assemble, discover, monitor and (re-)configure modules produced by various vendors seamlessly and on the fly. For instance, for CCM generic substrate carrier every time has to be mounted with several other modules which vary from case to case and typically shipped by different vendors. Assembling such a line is time consuming and error prone procedure and usually it consumes sometimes up to 20% of the overall project budget. Then connecting a new line to the shop floor control and monitoring system needs to be done and if something changed in the hardware (or sometimes even software) interface the existing system should be re-configured and re-programmed.

In the case of the aforementioned inspection machine case, the abstraction layer will facilitate the automatic “understanding” among the different surrounding devices in order to quickly and easily adapt to the requirements thus lowering the assembling and configuration periods and the likelihood of errors.

Moreover, CREATE will provide infrastructure for monitoring and control, coherent graphical user interface on top of Enterprise Integration pattern based on services. CREATE will also provide frameworks and guidelines for how to expose hardware components and modules as services. This way, starting from the mostly used and typical hardware pieces (per industry, per customer) CREATE will collect a library of resource and service descriptions in order to support re-usability of those components. With expanding of the library an assembling of a new line will be much faster and will not involve programmers because assembling and configuration will be made based on existing and tested service descriptions and software artefacts.

Figure 4 demonstrates CCM use case in a way how it will look like on the CREATE platform.

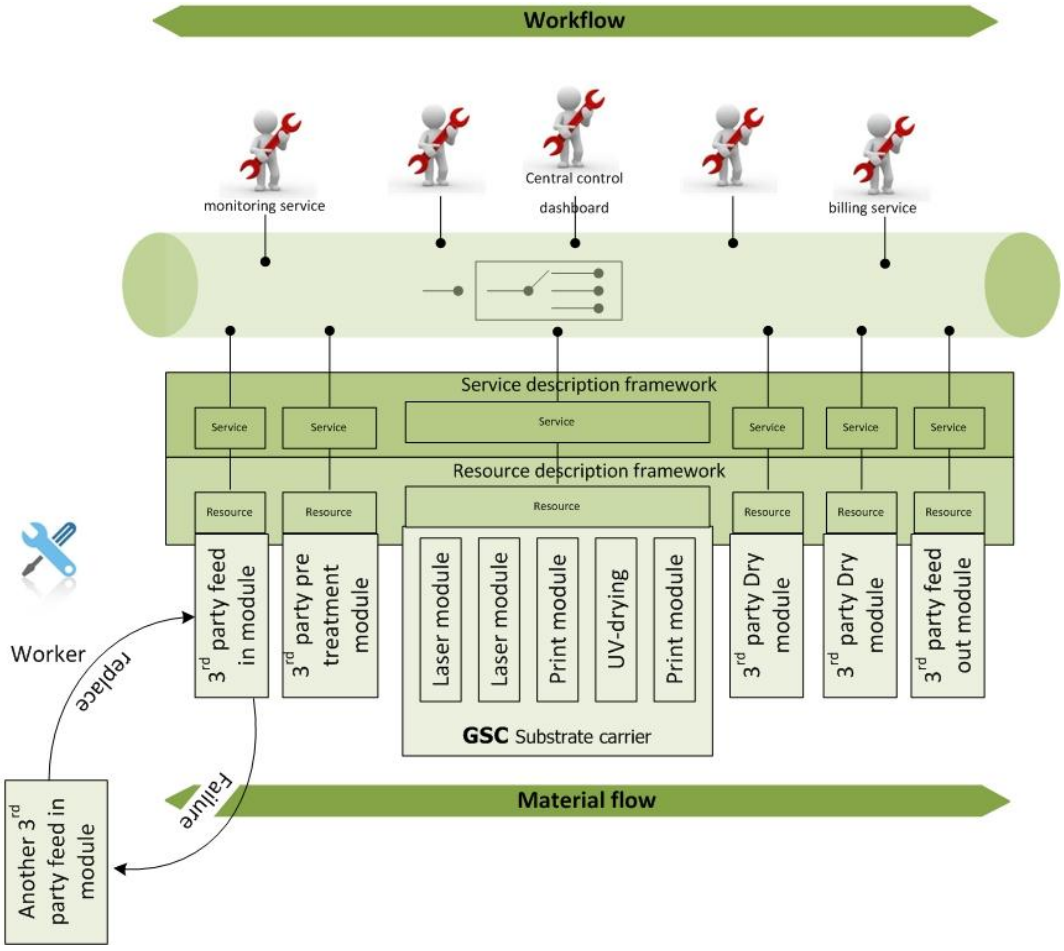

Figure 4: CCM use case in CREATE paradigm

And Figure 5 shows industrial metrology use case inside CREATE platform.

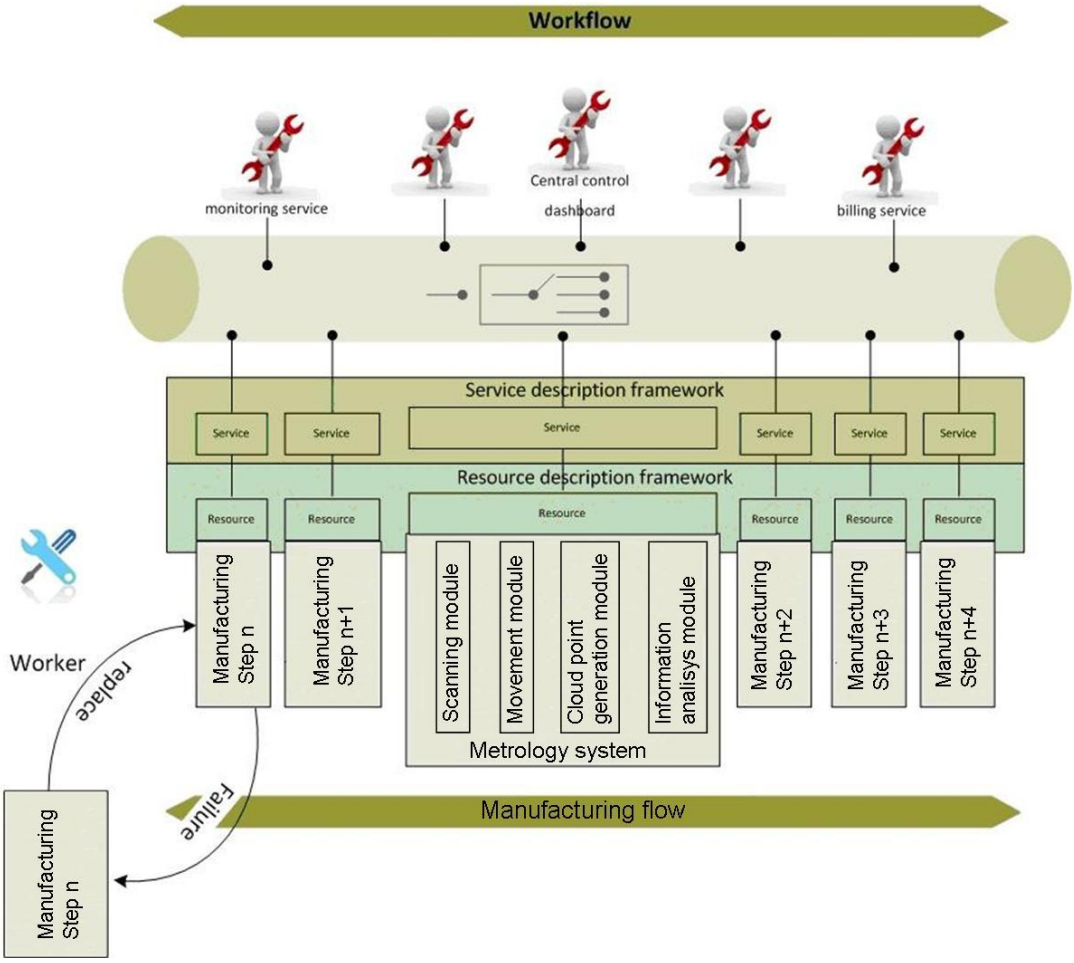

Figure 5 Spanish consortium´s use case in CREATE paradigm

Thus, taking into consideration the above mentioned concerns and thoughts, the high level CREATE architecture is presented on figure 6 below.

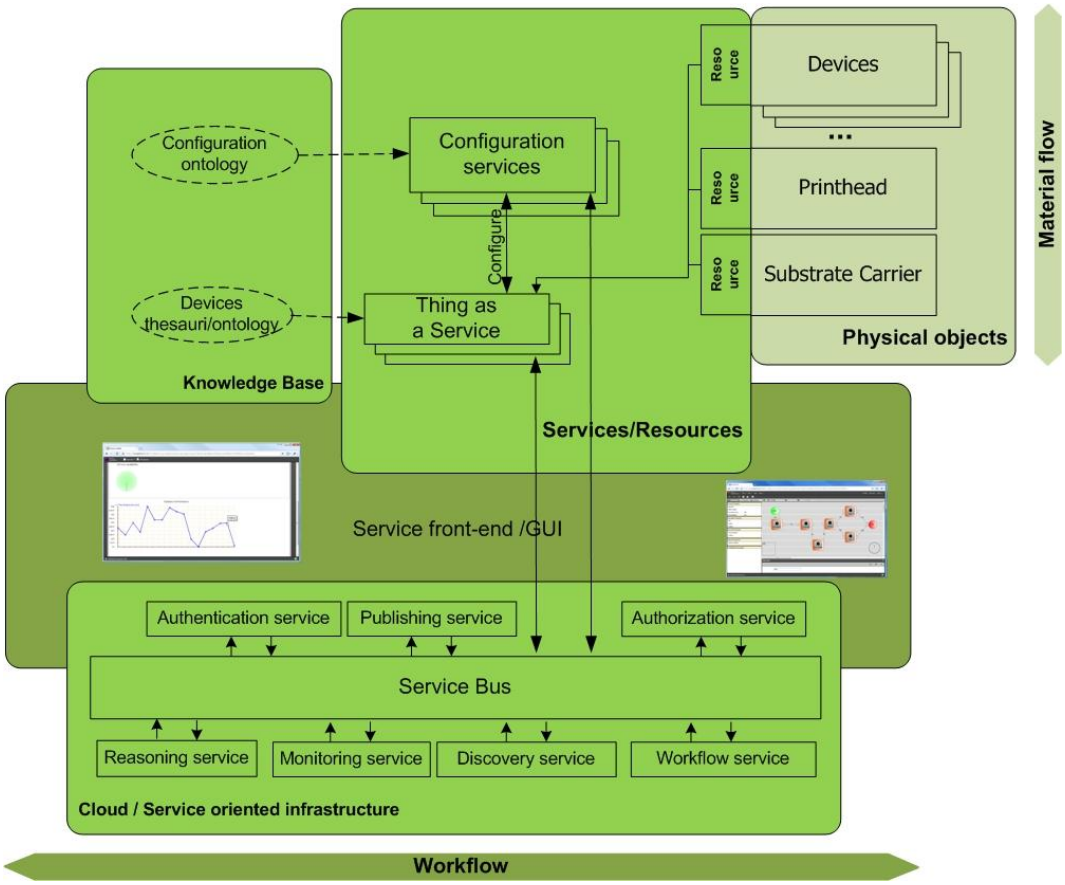

Figure 6: CREATE high-level architecture

The following main concepts are part of the CREATE platform:

- **Configuration services:** These services will represent configuration jobs and effectively they will create/update descriptions over the performed actions (and associated conditions) on particular components and their composites.
- **Thing as a service (TaaS)**: Internet-of-Things representatives that expose different devices (and their parts) in the virtual world of CREATE.
- **Knowledge base:** Machine-readable knowledge bases used for the purpose of having automated reasoning over devices and their possible configurations
- **Semantic Service Bus**: A distributed bus for message exchange with enhanced support of semantics/metadata.

- **Stack of infrastructural services**: For example publishing services, annotation services, authentication and authorization services, etc.
- **User interface**: A presentation tier based on the advancements made in the EU projects such as OMELETTE and SOA4ALL.

4. Architectural Constraints and non-functional requirements

This section describes the software requirements and objectives that have some significant impact on the architecture.

4.1 Real-time system constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

- The GSC is a module in a complete machine, e.g. in an industrial printer, in a thin film solar cell production machine or in an inspection machine. This implies that the interfaces of the GSC with other modules of the machine can be very divergent.
- More sophisticated and application dedicated GUI's are required for the current GSC system.
- The current performance of the GSC should not be affected. CREATE platform should be built on top of the current system. The connection point between CREATE and existing legacy GSC system is a Host PC (GSCHC).
- Real time decisions will require coordination between the mechanical components of the metrology machines or distribution belts and the processing software, in order to avoid mistakes associated to delays in the information communication.

4.2 Persistence

Data persistence will be addressed using a Polyglot approach to persistence, which means a persistence for each module will be defined based on the set of requirements for each individual module and can vary from non-transactional distributed repositories such as Cassandra to DB with native JavaScript interface such as MongoDB.

The more detailed view on persistence will be specified in the next iterations of this document.

4.3 Reliability/Availability (failover)

The scalability and reliability of the system is a key requirement as it pertain the nature of this system.

The server side solution will reside in the Cloud in order to support elasticity and high

availability requirements.

Targeted availability is 95% (out of 24 hours a day, 7 days a week).

In the event of the server side failure CREATE application might result in becoming temporarily unavailable. All relevant user should be made aware of all problems occurred on the platform with visual feedback along with notifications via telecommunication channels, i.e. SMS, automated telephone call.

5. Quality

As far as CREATE system is concerned, the following quality goals have been identified.

Scalability: CREATE system will be conceived to be an evolution-capable system which will have the built-in ability to handle change as well as to keep the planning and the configuration data synchronised in combination with an adequate change tracking.

- **Description**: The server goes down due to a sudden increase in the number of operations.
- **Solution:** An analysis of the processing needs of each new module have to be done before its installation and given the software inside the architecture. Apart from that, the continuous monitoring and performance measurement solutions will provide real time controlling to prevent surprises.

Reliability, **Availability**:

- **Description** : Transparent failover mechanism, mean-time-between-failure
- **Solution:** : application server supports load balancing through clusters

Portability: The information generated and all the possible communications will be available seamlessly from and to all the different platforms that come into play.

- **Description** : Data is not available, its quality is not good or the security policies are not homogenous depending on the platform used.
- **Solution:** :Validation, quality assurance and security tests are going to be performed by SQS at all the levels of the architecture and in all the different platforms available.

Security:

- **Description 1**: Stolen business data, cloud security issues or malware in the system.

- **Solution: 1**: The architecture will take the security and compliance policies at all levels.

Safety: Two main quality goals have been identified from the security point of view.

- **Description 1**: Human injuries due to a wrong interaction with human.
- **Solution 1:** As a general rule, the risk for the human health should be catalogued for its of the machines involved in a production line. Standard aspects will be considered for each type of machines but there will be some points will have to be programmed individually.
- **Description 2**: Human risk due to system general failure.
- **Solution 2:** The first step for the configuration of the software of the machine will set the actions in case of system failure. Therefore in the case of general failures this safe configuration will be loaded.