Deliverable 4.1: Evaluated Demonstrators v1

CREATE

Creating Evolution Capable Co-operating Applications in Industrial Automation

Project number: ITEA 2 ip10020

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

Contributors: DATAPIXEL, CBT, CEESA, EPC, INNOVALIA, Malardalen Unieverisy, TIE, TRIMEK, STAUBLI

Date: 30 April 2013

Document version no.: *0.6*

List of document contributors

Version	Contribution	Name	Company
0.0	Table of content and contributors	Antonio Ventura-Traveset	DATAPIXEL
0.1	Industrial Metrology demonstrator	Antonio Ventura-Traveset Silvia de la Maza Fernando Perales	DATAPIXEL INNOVALIA TRIMEK
0.2	Monitoring and Quality control demonstrator	Peter Funk	Malardalen University
0.3	Material Flow demonstrator	Vadim Chepegeim Anastasios Martidis Armando Miraglia	TIE
0.4	Merging information and format	Fernando Perales	TRIMEK
0.5	Final review and conclusions	Silvia de la Maza Anastasios Martidis Peter Funk	INNOVALIA TIE Malardalen University
0.6	Final document	Fernando Perales	TRIMEK

INDEX

LIST OF FIGURES

1. Material Flow Demonstrator

1.1. Introduction

Manufacturing domain, e.g. automation of a shop floor, is a hot spot for research and innovations in order to keep up with the market, its dynamicity and demands. In the context of automation systems, software has been extensively used to support the automatic execution of tasks. In fact, low level operations are often supported by specialized software. Lately the usage of the web service technology for networked intelligent machines or Smart Objects[1] has been taken into consideration to develop Internet of Things (IoT) for manufacturing, see e.g. FP6 SOCRADES project (Moreira, et al., 2008). However, automation of workflows for high level tasks has received much less attention. Consequently, the integration of multi-vendor software intensive solutions is still limited. For this reason, the CREATE project aims to provide a distributed and easily configurable modular automation system based on web services. Moreover, the components of the production line and their interrelations are (semantically) described as resources in a machine understandable way and will provide a knowledge base for device matchmaking algorithms and artificial intelligence applications for reconfiguration. The results of the project will enable dynamic, flexible and re-configurable production systems.

The deliverable presents the implemented demonstrator as output of the CREATE project. The demonstrator aims to provide a minimal but useful system to demonstrate the approach of the CREATE project. The demonstrator resembles the principles of Service Oriented Architectures (SOA). The devices installed in the addressed production line are transformed into virtual objects by means of service wrappers. The service wrappers export the interfaces of the devices in a SOA fashion. In this way, the devices can be monitored and controlled over internet by means of standardized and reusable protocols and technologies.

1.1.1 Purpose and scope

The goal of this section of the deliverable is to present the demonstrators based upon the flexible material flow use case conceived and developed in the Dutch consortium. It describes the technologies and tools that were used in the implementation of the prototype and how they are related to the defined CREATE architecture and a set of selected technologies. For this reason, this deliverable is closely related to the previous work documented in the deliverables “D2.1 CREATE Architecture” and “D3.1 Technologies Description”.

[1] RFID, Smart Embedded Devices, Sensor Networks

Furthermore, this document will present the tests and evaluations performed based on the demonstrator. Based on the evaluation and the planning defined in the work packages, the conclusions will be presented. The status of the CREATE platform realization will be shown. Moreover, the results congruency with the expected CREATE approach value will be discussed.

1.1.2 Relation of the demonstrator to the CREATE architecture

The CREATE project aims to enhance industrial automation via hierarchical networks of smart objects and services – Smart Neighbourhood Modules (SNM). It employs a fully distributed software architecture for industrial automation systems that are composed of cooperating modules (SNMs). These modules consist of mechanical parts (e.g. devices in a production line) and their controls accomplished by means of automation software. CREATE will increase the flexibility and adaptation of existing systems and will allow their integration with new paradigms such as human-centred design, service orientation, secure and safe distributed control architectures, semantics, dynamic legacy integration and control system life-cycle engineering support.

The following scenarios consider three actors.

1. An *operator* is the actor which interacts with the production line. The tasks of the operator are the monitoring and control of the whole production line and of each component.
2. The *system integrator* is an actor who integrates different hardware parts into a production line.
3. The *platform administrator* is the actor responsible to maintain the CREATE system.

Based on the operations that need to be performed by the identified actors, some scenarios were identified.

1. The *operator* starts the production line.
   a. The operator wants to start the production line and opens a browser with a view on the preferred device.
   b. When the browser is opened, the operator goes to the main URL of the GUI web application installed at their site.
   c. The operator proceeds to login into the GUI by providing username and password.
   d. When the dashboard is loaded in the browser, the operator navigates to the production line management widget that provides the initialize and start functionality.
   e. The production line starts.
2. The *operator* monitors the operation of the production line.
   a. The operator wants to monitor the operation of the production line which was started and opens a browser on the preferred device.

   b. When the browser is open, the operator goes to the main URL of the GUI web application installed at their site.
   c. After the login, the operator navigates its dashboard and checks the state of the production line observing the provided monitoring graphs.
3. The *system integrator* needs to replace a device.
   a. The system integrator receives a communication for the CREATE system that a device is not working correctly and needs to be replaced.
   b. It access the suggestion facility based on semantics and artificial intelligence algorithms to select the best replacement for the defected piece.
   c. After selecting the replacement component, the system integrator replaces the device and updates the workflow status provided by the CREATE system.
   d. The CREATE system informs the other actors associated with the event.
4. The *platform administrator* set the system in place.
   a. The platform administrator installs the server needed to have the GUI available on the network.
   b. The platform administrator connects each device with the related service wrapper to enable the GUI to actually query the device interface.
   c. When the system is up and running, the platform administrator is then able to insert the needed workflows which fit the requirements of the devices.

The CREATE platform reaches the above described functionality and offers solutions building on top of the following tools and technologies.

- *Thing as a service (TaaS):* Internet-of-Things representatives 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 for semantics and metadata.
- *Stack of infrastructural services:* several services are provided, e.g. 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.
- *Configuration services:* these services will represent configuration jobs and they will effectively create and update descriptions over the performed actions (and associated conditions) on the particular components and their composites.

The defined architecture for the CREATE platform is presented below (fig. 1).

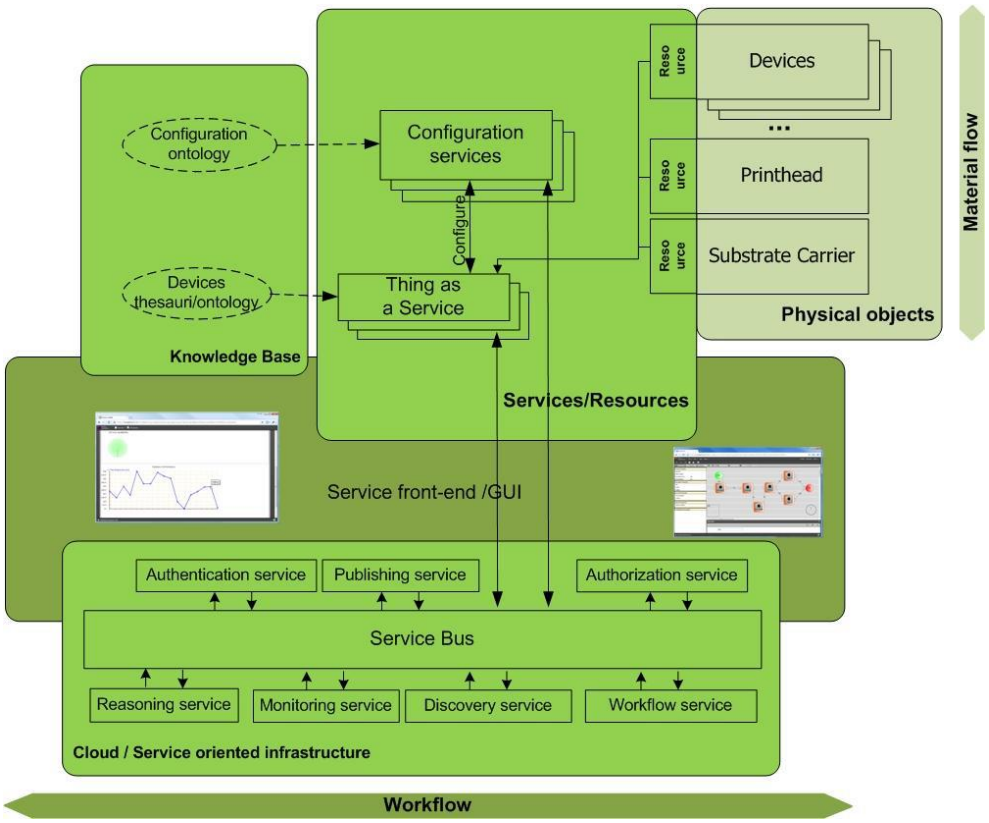

Figure 1: CREATE architecture overview

The application of the CREATE approach in the flexible material flow use case and the expected added value is tested on a production line composed of the Generic Substrate Carrier (GSC[2]), which is a high precision conveyor belt, together with associated cooperating modules such as in/out-feeders that are used as input and output of products in the conveyor belt, sensors that are used to identify the position of products on the conveyor belt and printers (e.g., inkjet print head, industrial and solar panel printers) that operate on the products on the conveyor belt (fig. 2).

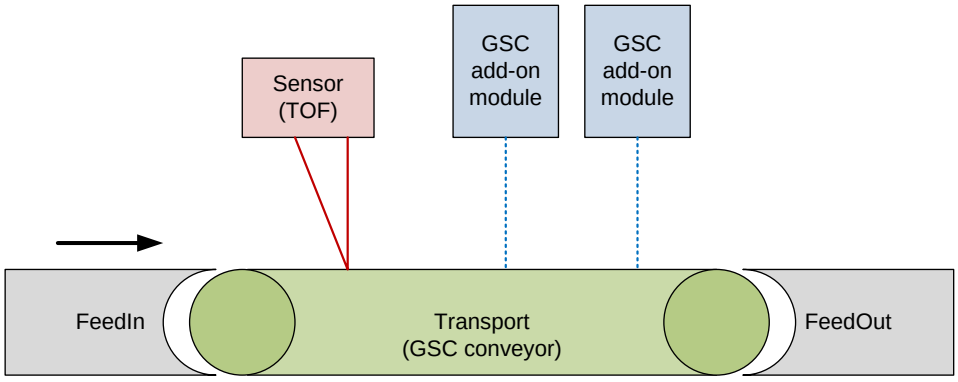

Figure 2: GSC conveyor belt in a production line

[2] Generic Substrate Carrier (GSC) is a high precision conveyor belt, developed by CCM

The *prototype* is composed by physically and logically interconnected devices and software that are involved in an automated production line of the flexible material flow domain. It is developed following the CREATE architecture and achieves the exposure of the devices as web services [3]into the cloud [4]in a modular approach. These services expose the capabilities provided by the automation software for controlling the state, the behaviour and the operations of the device. Additionally, capabilities to reconfigure the production line based on the description of all actors involved in the production line as resources and their relations, knowledge base and device matchmaking algorithms performed on triple stores[5] allow faster and more effective reconfiguration of the production line.

The *prototype* provides a friendly GUI which allows users to consume the exposed services and all the available features in an interactive and intuitive way, accessing the CREATE portal which offers a dashboard of functionality-dedicated widgets that can be customized according to the privileges and preferences of users. The communication between the exposed services and the CREATE portal is performed via an enterprise service bus which allows the exchange and mapping of heterogeneous messages containing the information.

As showed by the high level architectural view (fig. 1) and the description of the prototype, the definitions of previous deliverables (D.3.1 CREATE Architecture) were respected and followed in the implementation of the first prototype for the CREATE platform. The physical objects/ devices in the flexible material flow use case were exposed as web services. The web services can be used via the Cloud to control the state, the behaviour and the operation of the devices. This can be performed via a GUI (CREATE portal). The portal is designed to make the interaction between users and GUI intuitive and effective.

The resource descriptions of the devices as well as descriptions of associated actors in the production line and their relations are stored in triple stores. The information stored in these triple stores are used by device matchmaking algorithms that allow faster reconfiguration of the production line based on a knowledge base. The re-configuration features of the CREATE platform are also accessed through the CREATE portal. The exposure of the devices as web services and the re-configuration features developed are being integrated as new services of the TIE Smart Bridge (TSB), which is a secure and

[3] A software system designed to support machine-to-machine interaction over a network
[4] Computing resources (hardware and software) that are exposed as services over a network (usually the internet) and can be consumed by a variety of devices that can access this network are referred to exist on the “cloud”. Cloud is the “infrastructure” where such services can be found.
[5] Semantic based repositories

flexible messaging solution. It provides routing and mapping/transformations capabilities for the messages exchanged among services.

The demonstrator described in this deliverable shows the feasibility and applicability of the CREATE architecture and selected technologies by instantiating a system derived from that architecture in order to fulfil requirements of the real-life industrial use case presented by CCM, a company that provides advanced solutions in the field of precise mechatronics.

1.2. Brief tool description and integration

1.2.1 Technologies involved

1.2.1.1 SOA paradigm

In the Service Oriented Architecture (SOA), systems are built by composing independent well-defined units of functionality called services. In this type of architecture, services are loosely coupled. This means that each service acts independently from the others and the functionalities provided are also decoupled. From a service point of view, the full system is unknown. The service is only aware of its functionalities and the functionalities which it requires but it is derived from the overall system view. Before their deployment, applications (consumers of services) do not know which concrete service will deliver the concrete functionalities. In this manner, on one hand the system is easier to extend with additional functionalities. On the other hand, dynamic discovery and reconfiguration can be set in place.

The dynamic discovery of the services and their functionalities is performed at run-time. In fact, applications discover available services and bind to them after being loaded. Dynamic reconfiguration is also possible. At run-time, new service providers can be added as well as existing ones can become unavailable or be replaced. Applications can dynamically change their bindings to services based on service availability and the quality they can provide.

To practically achieve dynamic discovery and reconfiguration, service discovery is required. Service consumers do not know which service is available and provides the required functionalities. Because of that there must exist a mechanism for consumers to learn what service providers can deliver the functionality they need. Service discovery can be centralized or distributed. Centralized service discovery rely on a single service, addressed as service registry, which acts as a broker which service consumers can query to obtain information related to the needed service. On the contrary, distributed implementations depend on an overlay network on which multiple service providers can be contacted. Providers send responses for the known services which can satisfy the query.

Once a consumer discovers the services that implement the desired functionality, it can request the related service descriptions. The descriptions contain both functional and non-functional attributes based on which the consumer selects the best service provider to which to bind.

The CREATE demonstrator is tightly coupled with the SOA paradigm. To make SNMs communicate between each other, flexibility and re-configurability are required. These requirements make the SOA paradigm the right design choice. SOA is inspired by agent based system and ultimately enables manufacturing systems that are robust and adapt to dynamic changes in their environment and resolve internal and external disturbances in an intelligent way.

1.2.1.2 Devices as web services

The term "Internet of Things" was first used by Kevin Ashton (That 'Internet of Things' Thing, 2009) while addressing the usage of RFID in the supply chain to identify objects and transpose them in the virtual space. Nowadays this concept has expanded its meaning including all the technologies used to transpose objects in the virtual space. IoT addresses real objects which are linked to the virtual world by means of hardware components, e.g. wireless cards and wired connections, which make them communicate over the internet. When the objects are connected, the communication is obtained by means of standards for message exchange. An approach used in the industry to automate the production line composition and monitoring is the usage of Service Oriented Architectures. In such scenarios the objects are interfaced and exposed to the rest of the virtual world by means of *web services* which export standardized interfaces.

The exposure of devices in the virtual world is accomplished by the means of *web services*, which grow on top of *message exchange standards* and *architectural styles* (SOAP, REST), *semantic descriptions* of the web services (WSDL, WADL), approached for *publishing the services* (UDDI) and data formats (JSON) used by the exchange protocols. These concepts will be described briefly in the following section, as well as a framework for developing web services and their building blocks described above.

Web Services: A Web Service (Web Service Definition, 2004) is a software system designed to support machine-to-machine interaction over a network. It is based on the technologies used for web navigation like HTTP, XML and so on. The interface is described in a machine-processable format. This guarantees the ability to dynamically adapt to an interface provided and using it to request the service to perform operations.

Web services are based on standardized formats for message formatting, processing and data communication. We use web services to expose *devices*.

Web Services for Devices: Is an umbrella term that is used for a wide range of products and services that have been boosted by the use of Internet technology. In practice, many artefacts/devices that have already been developed on technologies unrelated to the World Wide Web are now improved by the use of Internet technologies. Web services on devices allow access of remote devices and associated services from computers through a network. Web service for devices is simply the notion of taking systems that already exist and function and using them better and more efficient by means of connection over the Internet (230ht). The communication between services is achieved with various message exchange formats such as *SOAP* and *REST*.

Simple Object Access Protocol (SOAP): (SOAP 1.2, 2001) is a protocol specifying rules to exchange structured data across the network. Using XML, the protocol prescribes the structure of the exchanged messages between nodes. The messages are characterized by a header which has metadata describing the message itself and a body where the content is put .The protocol specifies bindings with HTTP as well as with SMTP, which are used to perform communication between machines. These bindings do not prevent to use the protocol to process and exchange using a different underlying transport protocol.

Representational State Transfer: also known as REST (Fielding, 2002), is an architectural style. It prescribes rules that describe an abstract model of web-architecture. This architectural style is significantly based on the Hyptertext Transfer Protocol (HTTP) and, in fact, it is characterized by the very same principles. In a very simplistic definition, REST is a structured way of using HTTP. The principles of the architecture design are the following.

- The system must be *Client-Server (Layered System)*. In this way, based on the *separation-of-concerns*, client is not concerned with specific details of the server. The layers implemented on the services side are not visible to the client. In this sense, the client cannot distinguish the end server from possible intermediary services contacted along the way.
- It must be *Stateless*. No state is kept between client and server. This is a strong requirement which is eventually relaxed in many real implementations where authentication and sessions are required.
- It must be *Cacheable.* In the system clients can cache responses which when correctly implemented and managed reduces client-server interactions.

- The availability of a *Uniform Interface* between client and server makes it possible for both sides to evolve independently so far the interface is followed. The Uniform Interface is also significantly based on the HTTP protocol. In fact, it is characterized by the use of URIs and data format like HTML, XML or *JSON*.

REST applications stick to all afore mentioned principles.

JavaScript Object Notation: JSON (json) is a lightweight format to exchange data. This is a programming language independent format that is used to exchange data. It is built on collection of name-value pairs and ordered list of values. With this representation, all available data-structures can be easily described. It is also protocol independent and can be used as payload to represent data in all communication technologies. It is fast to process and easy to manipulate.

Web Services Description Language: also known as WSDL (WSDL, 2007), is an XML format that is used to describe several services characteristics. The main properties that can be described are the location of the service, which identifies where the application can be found; the operations that the service is able to perform and the corresponding messages, protocols used and so on. Since is written in XML, this description can be processed at runtime and accordingly, dynamic requests for the specific operations can be requested.

Web Application Description Language: also known as (WADL, 2009), is an XML format that equivalently to WSDL is used to describe services. This is also automatic processable. The main difference with WSDL is that WADL is fully REST compliant. In respect to WSDL, WADL is much less used in real applications.

Universal Description, Discovery and Integration: also known as (UDDI, 2004), was born as a XML based standard for companies to publicly provide services. It can provide information related to the company-provider, information about services location, interface and content provided. The concept of UDDI has been adapted in other scenarios and is used for automatic discovery of services. It is also identified as service registry, meaning a location where data about the services are published and consumed. The current version of UDDI is the version 3.

Windows Communication Foundation (WCF): WCF [6]is a framework provided by Microsoft to create Service-Oriented applications. It makes it easy to comply with the service oriented standards which characterize Service Oriented Architectures. In fact, WDSL descriptions are automatically generated to provide service metadata. It supports also XML Schema and WS-Policy industry standards. It provides data and operation contracts to publish to users of the service the needed data structures as well as the interfaces needed to use the services provided. It provides several built-in transport protocols and encodings. TSB, the selected ESB for the CREATE platform uses WCF and the services developed using this framework will extend TSB and integrate all components of the architecture to the CREATE platform.

1.2.1.3 Devices as resources

The re-configuration of the production line is accomplished based on semantic description of all components (active and potential) of the production line as resources. Information associated with the devices themselves (e.g., physical specification), functional descriptions (e.g., cooperating devices) and non-functional descriptions (e.g. suppliers) are explicitly described according to an information schema (ontology). The description is implemented in machine understandable format (RDF), and the ontology is developed and populated using an ontology language built on that format (OWL). The populated ontologies with the information associated with the production line is stored in semantic based repositories (triple stores) and appropriate query language (SPARQL). The above concepts are explained briefly in the following section, and compose the knowledge base from which device matchmaking algorithms and artificial intelligent applications are performed for the reconfiguration of the production line.

Ontologies derived from the field of philosophy. Generally, ontologies are used to formally describe concepts and elements as well as their relations (Gruber, 1995). In computer science, ontologies are used to formally describe real world objects, classes and the way they interact and are very domain specific. The main concepts that are comprised in ontologies are *individuals* which can be instances as well as objects; *classes* which group together objects based on common properties; *attributes* which depict the properties related to the individuals described; *relations* which depict the possible interactions between individuals and between classes. Based on different domains, several ontologies have been described like OWL for web ontologies.

[6] http://msdn.microsoft.com/en-us/library/dd456779.aspx

Ontologies are used extensively in automation systems and significant research has been performed on the subject.

Pandis et al (Ippokratis Pandis) stress the benefits of using semantic web technologies for achieving dynamic management of resources in infrastructures and services of ubiquitous computing. They developed a framework that is composed by sensors and perceptive interfaces for facilitating ubiquitous computing services with an emphasis on the role of knowledge bases for dynamic registration and invocation of resources. The sensors and actuators were controlled from ontology based mechanisms.

Christopoulou et al (Eleni Christopoulou) motivated by the belief that ontologies can contribute on key issues of ubiquitous computing environments such as knowledge representation, semantic interoperability and device discovery developed the GAS ontology. GAS ontology describes the semantics of concepts of an ubiquitous computing environment as well as their inter-relations. They aim to provide a common language for heterogeneous device communication that compose such environments and facilitate discovery device mechanisms.

Reinisch et al (Christian Reinisch) emphasize that in order to deploy automated systems we have to overcome the challenges of integrating heterogeneous system so that they combine their functionality. The way to achieve this requires a comprehensive communication between systems. They propose a generic application model to avoid the configuration effort that traditional integration approaches require such as use of gateways. They selected ontologies in order to provide seminal representation of knowledge, abstraction of the heterogeneous network infrastructure and automatic reasoning on the stored knowledge. Their representation implements single access point for configuration and maintenance tasks.

Alsafi et al (Yazen Alsafi) proposed a novel approach aiming to achieve fast reconfiguration of modular manufacturing systems performed by an ontology-based reconfiguration agent. The agent is able to reconfigure without human intervention based on ontological knowledge of the manufacturing environment. Their agent automates the reconfiguration process and uses inferencing of the manufacturing environment from the ontological knowledge in order to decide if an environment can support certain manufacturing requirements. Their approach uses agent architecture for the integration of the high level planning with the distributed low level control.

The Semantic Web: (W3C, W3C Semantic Web Activity, 2013)is a collaborative effort led by W3C with the participation of a large number of researchers and industrial partners. The main goal of the project is to provide a common framework so that data is more effectively shared and reused across different applications on the web. For this

reason, the two core pillars are the usage of *common formats* and *description languages*. The usage of common formats helps in the effort of integrating and combining data from diverse resources. Having common formats makes it easy to move information which is reusable and sharable. On the other hand, *description languages* aim to picture the relationship between real world objects and data. This languages need to be machine processable for automatic processing. However, they need also to produce an output which is human readable and meaningful. The effort has so far produced widely applied formats and languages like OWL, SPARQL and RDF.

The Resource Description Framework (RDF): resembles a set of specifications for data interchange on the Web (W3C, 2004). It defines a language for representing information and metadata about web resources (for example web pages). This language, however, easily adapts to other types of information like real world objects mapped in the virtual world in the context of IoT. The main aim of the language is to be application-processable avoiding loss of meaning. Existing web-technologies are used for this purpose like the Uniform Resource Identifier (URI) commonly used to identify web content. The language can be expressed by means of XML as well as using graphs.

The Web Ontology Language (OWL): is an ontology language based on RDF/XML documents used in the field of Semantic Web (W3C, 2012). The markup language is based on the first-order logic and is used to express meaning and semantic of objects using dictionaries and relations. OWL is not a single language, in fact several flavours exists which significantly differ from each other. The OWL specifications are the result of a specialized W3C working group which resulted in the production of a forma W3C recommendation.

A triple store is a type of database in which elements are stored in triples of strings (Rusher, 2003). The triple is composed by *subject-predicate-object* as is usually done in human languages. For example “foo is bar” can be a meaningful entity. These types of database are optimized to store and retrieve triples. The database structure is completely independent from the actual implementation. In fact, some implementations are based on existing systems like SQL RDBMS; others are completely new engines optimized to work with triples.

SPARQL is a query language which has been created precisely to find information from data stored in the RDF format (W3C, SPARQL Query Language for RDF, 2008). SPARQL is intended to efficiently query data expressed using directed, labelled graphs.

Its structure resembles in some way SQL, but clearly it is structured to better fit the RDF language.

Device Matchmaking. Based on ontologies, it is possible to automatically select sets of objects based on their description, properties and relations. Usually such a process is performed by an intermediate entity which is queried and which applies algorithms to dynamically retrieve the best fitting objects. This entity is usually called *matchmaker*. Besides, such a system can also be applied in the field of the IoT. In fact, devices are matched using ontologies and matchmacking algorithms based on ontologies. To obtain this goal, devices are mapped to the virtual world by means of services. Afterwards, services can be matched based on ontologies which enrich existing service descriptions (Studer, 2006), but also create more general ontologies (Lopez, 2007).

Service Registry. To collect all the information in a central repository based on which matchmaking and discovery are possible, the CREATE project is based on the usage of a service registry. Mostly, the state of the art implementations of ESBs have this component. However the inclusion of ontologies and semantics is almost completely discarded. The only available implementation is found in Apache jUDDI service registry where OWL-S is used. This system does not fit the CREATE project needs since the technology used is different in respect to what CREATE uses. Moreover, the lack of the usage of semantics for service registry functionalities in the industry suggests the need of additional research to fill the gap. The TSB will in this case provide the needed registry facility with the functionalities enriched by the application of ontologies. This part of the project is under development and will be the strong point of the next version of the CREATE demonstrator.

1.2.1.4 TSB on an integration layer

As described in the previous sections of the present report, the output of the CREATE project has high quality requirements. In fact, the product needs to be scalable but preserving the integrity of the data handled; the product must be reliable and available, hiding as much as possible failures and recovery processes. Finally, the portability requirement imposes the need to transform data in different formats to provide all the actors with reliable and readable data. To satisfy these requirements, the CREATE project will include the usage of the TIE SmartBridge service bus (TSB). SmartBridge is the business integration platform provided by TIE Kinetix. Based on its design and architecture, the TSB satisfies all the requirements that the CREATE project imposes.

Moreover, because of its frontend and configurability, TSB supports the communication between human actors as well as communication between human actors and devices or services.

Moreover, by enriching the information stored in the service bus with semantic descriptions, the system is able to backup human work which requires the progress of predefined structured workflows.

1.2.1.5 User-system communication via Cloud

An automated production line cannot disregard the human factor. There is a need for humans that interact with the automation software (e.g., to control or monitor production) as well as with the physical devices that are operating during production (e.g., in case of repairs etc.). The integration of manual work and the successful use of the devices in the virtual world via the Cloud (that is, use of the exposed web services) require a user interface by which, users interact. The user interface is considered in the architecture defined, as an important factor for the success of the CREATE platform.

The user interface is in the form of a portal (CREATE portal) which consumes the exposed services of the production line and the functionalities they provide as well as the re-configuration of production line features which are exposed as services too. The portal is built on top of Apache Rave.

Apache Rave is a mashup engine to aggregate widgets. It is a project based on several existing Apache projects, and aims to bring together a unified framework in which widgets can be set in place and easily managed. In fact, inter-widget communication, authentication, dynamic frontend engine and other features are provided to facilitate the creation of extensible and dynamic web interfaces. It complies to OpenSocial and W3C Widget standards. It provides context-aware personalization, collaboration and content integration capabilities.

1.2.2 Demonstrator description

Devices as web services in the demonstrator

The demonstrator for the GSC production line is composed at this stage from the GSC conveyor belt (actual device) and high accuracy Mathworks xPC[7] simulators, which act as simulators for the rest of the devices in the production line, namely *in/out-feeder*, *sensor* and *printer*. These modules have associated automation software developed on

[7] xPC Target is a real-time software environment from MathWorks and enables engineers to simulate and test models in real-time regarding physical hardware.

Mathworks xPC. Based on description of services available from the automation software, web service wrappers were developed using the WCF framework to expose the devices into the cloud.

The selected message exchange protocol of these services is REST, because REST:

- Is lighter than SOAP (limited bandwidth and resources)
- Is simpler and more flexible than SOAP
- Facilitates scalability
- It's generality of interfaces
- Allows independent deployment of components
- Facilitates intermediary components to reduce interaction latency, enforce security and encapsulate legacy systems
- Large vendors (Google, Yahoo, Amazon, Microsoft) are adapting REST architecture style.

The actual messages under REST architecture are in JSON format. The description of the web services will be implemented using WSDL 2.0 instead of WADL. WADL is REST compliant and so it could be used by our application but WSDL 2.0 is a W3C recommendation, has authentication features that WADL lacks, is significantly more popular than WADL and thus it is decided as better choice. WCF does not yet support neither WSDL 2.0 nor WADL (and previous WSDL versions cannot describe REST services) and efforts are allocated to tackle this challenge.

These services are currently used in a fixed "hardcoded" way, but a semantically based service registry is under research and development for the dynamic discovery and selection of services, which will be necessary when we have many devices and available services in the production line and select a subset of them. A graphical representation of the architecture implemented in the prototype is presented below (fig. 3).

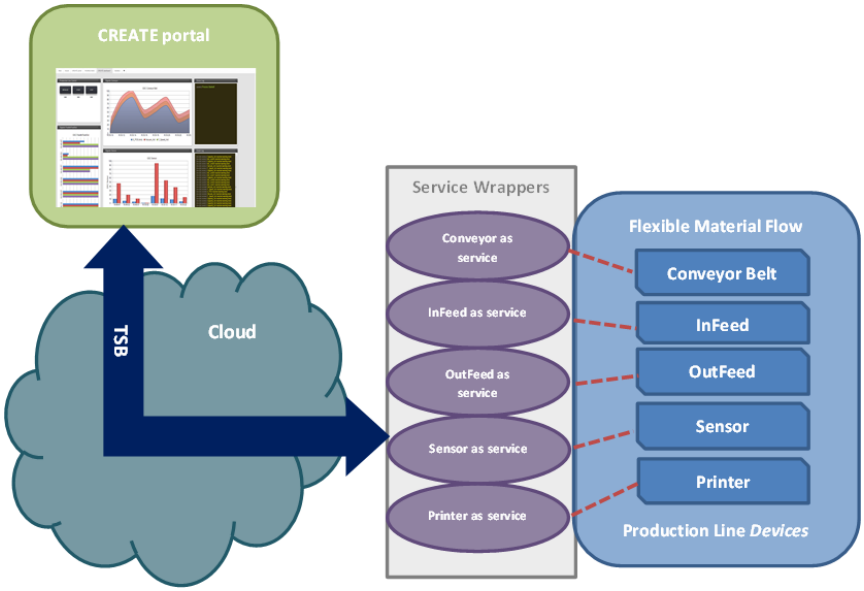

Figure 3: Devices as web services in the demonstrator

These exposed services allow the control of the behaviour, state and operation of the devices. Examples of the available operations via the Cloud to the exposed services include:

- To *Start*, *Stop* and *Initialize* devices individually or as a whole (the production line).
- *Get* parameters related to the device such as max allowed deviation of speed or operating range of the under-pressure.
- *Set* parameters such the above to the device.
- *Retrieve data* generated during production line (e.g., position of products, speed of conveyor belt as well as warning and error notifications) that allow the monitoring of the production.

Reconfiguration and device matchmaking

Additionally to the exposure of devices as web services, the devices are described as resources as well as with all associated actors (active or potential) in the production line. First, one ontology was developed using the OWL language that describes an explicit representation of entities and their relations in the domain of flexible material flow. The ontology includes classes and properties (object and data) that define relations between instances.

The CREATE ontology is constructed on 3 main pillars: components/devices, services and processes. These parameters act complementary to each other. More specific the 3 building blocks represent:

- *Structure/Devices*: information about the components of the system; the devices that cooperate in order for the system to function. Each component is described in detail regarding its physical and operational specifications.
- *Services*: This pillar mainly is going to serve in the discovery of services and includes information about the service that the device can perform, the endpoint where the service is available and the actual availability of the service. This component is still under research and development.
- Process: information about how the system works real time. Each component performs a specific task or is inactive. Moreover with the use of object properties the flow and sequence of actions performed by the device, and by cooperating with which devices, are explicitly stated.

The ontology was developed using the *protégé* [8]editor and an image of the image is presented below (fig. 4) demonstrating the current version of the ontology which is mainly developed on the Device building block, for achieving reconfiguration.

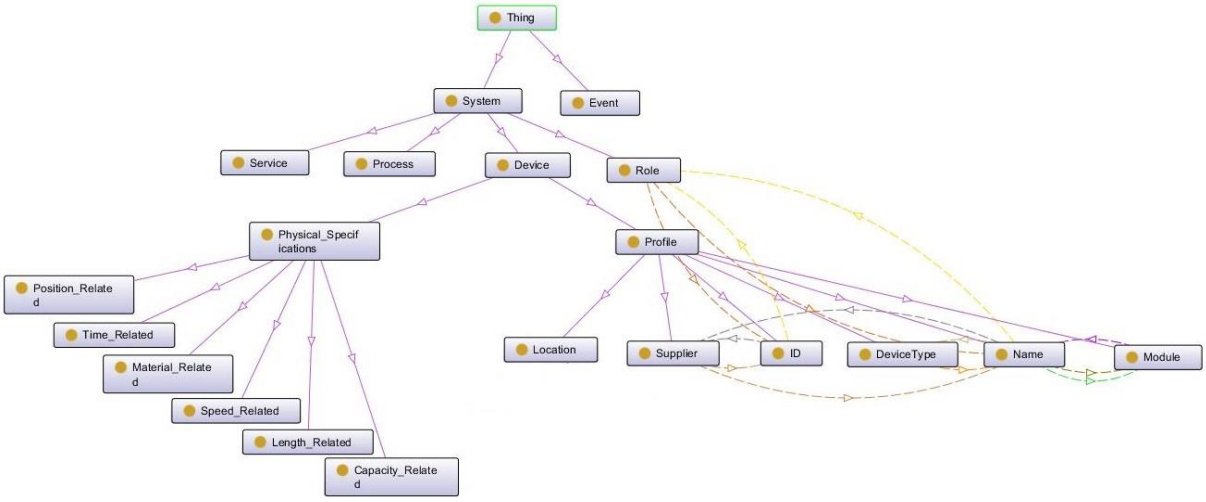

Figure 4: Automation systems ontology

The ontology is populated with information for components actively involved in the GSC production line as well as potential replacements/ substitutions. The data of the populated ontology is stored in triple store available from the dotNetRDF Project[9], which is an open source .Net library. On the information stored on the repository, device matchmaking applications are performed based on the user input and with use of the SPARQL query language.

In a simplistic way of defining matchmaking, it is the matching for a given input with appropriate outputs based on some rules. The device matchmaking is matchmaking with

[8] http://protege.stanford.edu/
[9] http://www.dotnetrdf.org/

devices as inputs. Currently, the device matchmaking application for reconfiguring the production line receives different input categories (such as devices, suppliers and device types) and retrieves available outputs that "match".

For example for a given device as input from the user (e.g., GSC D200 560-1500 which is a model of the GSC conveyor belt), the output of matching cooperating devices, would result by querying the triple store behind the GUI with the following query,

```
SELECT DISTINCT ?resultedDevices
WHERE{    {input device} <http://www.create.org/GSCLine.owl#isDeviceOfType> ?type.
          ?module <http://www.create.org/GSCLine.owl#ModuleOf> ?type.
          ?module <http://www.create.org/GSCLine.owl#cooperatesWith> ?cooperatingModules.
          ?cooperatingModules <http://www.create.org/GSCLine.owl#ModuleOf> ?cooperatingType.
          ?resultedDevices <http://www.create.org/GSCLine.owl#isDeviceOfType> ?cooperatingType.}
```

It is not the intention of this section to explain triples or how SPARQL works; the above query matches devices that could cooperate with the device that are is given as input, by fulfilling all statements, one after the other, for all possible results that meet the criteria. The sequence of statements for the above query,

- Gets the type of the input device
- Gets the module of this type of device
- Gets cooperating modules of this module
- Gets types of devices for the resulted cooperating modules
- Gets devices that are of the types retrieved in the above step

The results are devices that could cooperate with the input device based on whether the modules can cooperate (e.g., a conveyor belt with an in-feeder). The device matchmaking applications are being developed for improved matchmaking by adding ranking capabilities based for example on whether the devices are compatible on their physical specifications.

Using matchmaking features such as the above by providing inputs e.g. devices, device types and suppliers, users can benefit from automated retrieval of appropriate replacements or new entries on the production line, faster and considering many parameters/rules; it is depended on the ontology and the richness of the information that it holds and the strength of the artificial intelligence developed in the application. The device matchmaking features are exposed also as Web services.

To achieve the dynamic reconfiguration based on matchmaking the web services wrapping the real devices need to be registered in a service registry. Such capability will be integrated in the TSB service bus which will provide the necessary interface to store

information about each available service. TSB will also integrate the above described matchmaking system which will be used to describe workflows which integrated with the semantic information will let significantly support the human work facilitating reconfiguration and orchestration. The integration of the TSB service bus will be part of the next CREATE demonstrator.

CREATE portal

The demonstrator provides a user friendly GUI (CREATE portal) that can be used from any device that is equipped with a web browser such as tablets, smartphones and laptops/PCs. From this portal users can enjoy the functionalities provided by the CREATE approach. The design of the user interface is developed for intuitive and effective interaction with the users. The dashboard of the portal while operation is presented on the image below (fig. 5).

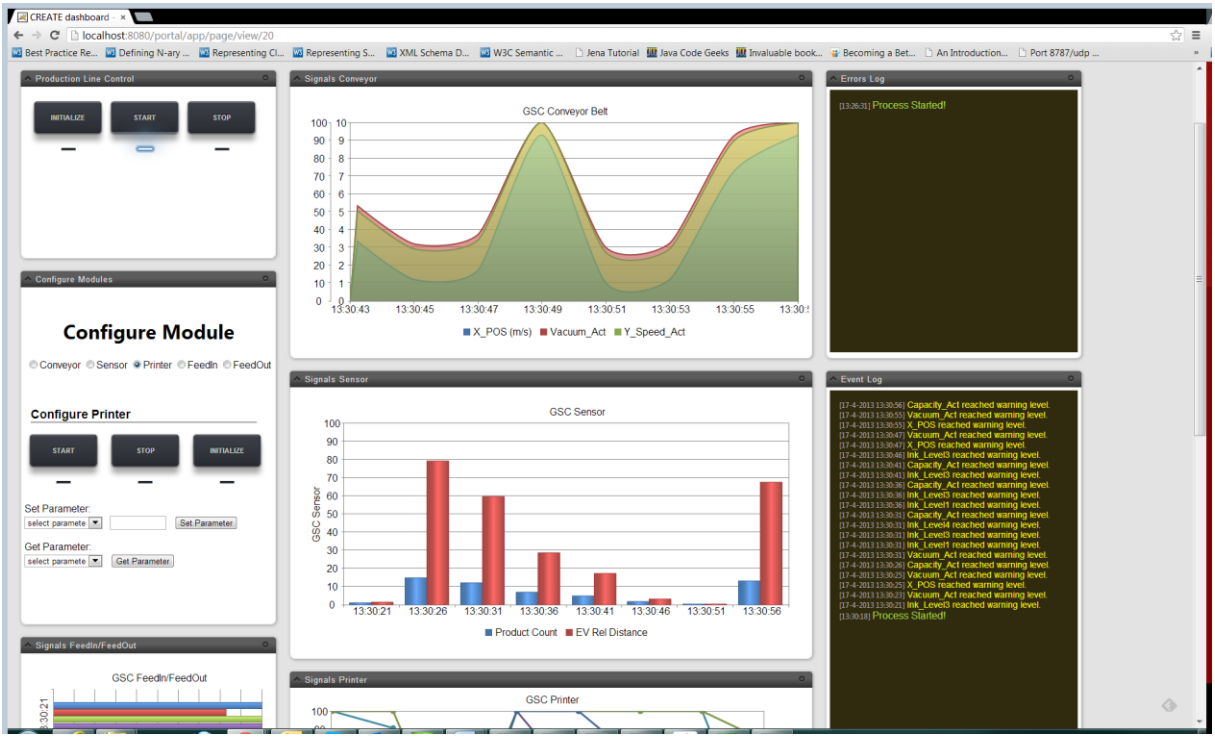

Figure 5: GUI of the CREATE portal while operation

The portal acts as an entrance point and as a visually rich interaction panel for consuming services available in the CREATE platform. The services include the control of devices as units and as a system as well as visualization of data generated during production in forms of charts. Moreover there are special widgets that present notifications about errors and warnings that occurred during production. The device

matchmaking features for re-configuration are also provided via a dedicated widget in the dashboard of CREATE portal. The design of the production line is also enriched with a graphical workflow designer that can be used on design time.

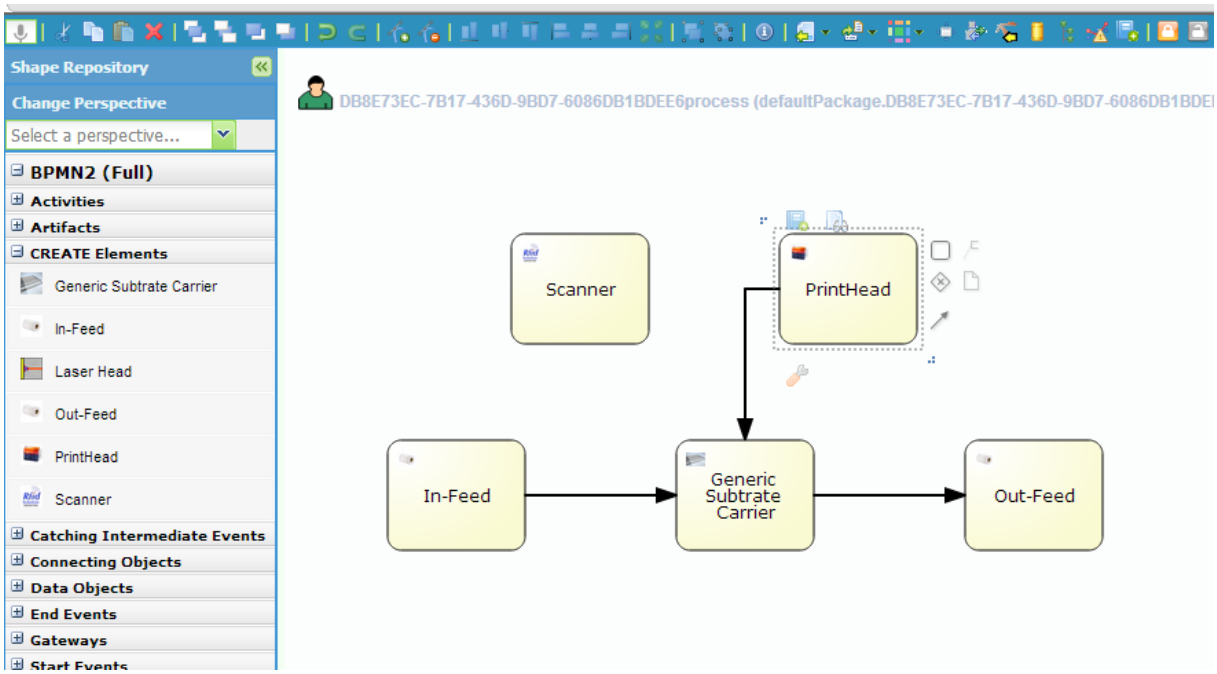

Figure 6: Workflow designer

1.3. Test and results

1.3.1 Tests and evaluation

For the evaluation of the demonstrator, numerous tests were conducted, and the targeted results were mainly qualitative in nature. The tests concerned the use of the production line and its modules, exposed as web services and as resources for functionalities such as control of the production line, the transmission of data generated during production and use of device matchmaking features for reconfiguration purposes. Furthermore, the GUI of the CREATE portal and the portability of the access were evaluated. The test were performed by employees of the two Dutch partners (TIE, CCM), in diverse locations. The first location was CCM HQ, where the access to the portal was performed via Intranet and the second was from TIE HQ with the use of a VPN connection between TIE and CCM firewalls. The infrastructural set-up of the tests, are depicted below (fig. 7).

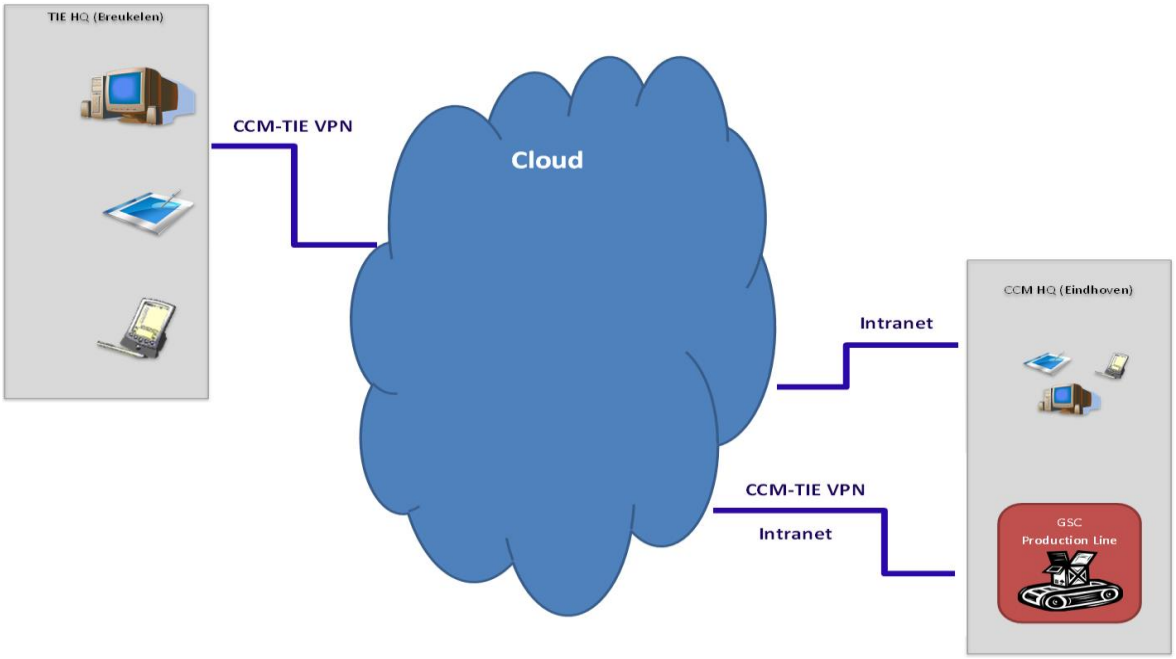

Figure 7: Test bed of the evaluation for the demonstrator

The reason for the use of VPN connection is that TSB is not yet fully integrated in the demonstrator so that it would provide security (amongst its others capabilities) to the communication between web services and CREATE portal.

The users that were involved in the tests had different backgrounds (users with knowledge of the domain, engineers, actual users of final product, and users unrelated to the project completely). The factors by which the demonstrator was evaluated are detailed below.

1. *Actual control of devices as units and of production line as a system from the portal.*

For the control of the production line and each device as a module, there were two possible outcomes, *Success* and *Fail*. It was examined for every command communicated to the devices via CREATE portal, from all users that took place in the tests, if it performed successfully or not.

2. *Accuracy of production line data transmitted and displayed.*

The data generated during production, transmitted from the devices and visualized in forms of charts and in error/ warning notification panels in the CREATE portal, were examined to verify that the information depicted in the virtual world was correspondent to the real life production.

3. *Speed of re-configuration using the features of the portal.*

The reconfiguration capabilities were tested by users that are responsible for such tasks, and will use these functionalities, such as system operators. They evaluated by stating if they agree or disagree with the statement "*The reconfiguration feature of CREATE platform adds value in the reconfiguration process*", selecting one of the following choices: a) Strongly Agree, b) Agree, c) Not sure, d) Disagree, e) Strongly Disagree.

4. *Ease of use of the GUI of the portal.*

The GUI of the CREATE portal was tested by users with diverse profiles. They had an introduction on the context of CREATE project and presented with the CREATE portal to operate the production line. Users evaluated based on the statement "*The GUI of the CREATE portal is intuitive and easy to use*" and were given the following choices: a) Strongly Agree, b) Agree, c) Not sure, d) Disagree, e) Strongly Disagree.

5. *Access of portal from several devices (tablets, smartphones, laptops, PDAs).*

Users that participated on the tests accessed CREATE portal from various devices, and specifically, smartphones, tablets, laptops and PCs. The evaluation was revolved on whether the control of the production line was achieved through all the above devices that are equipped with a web browser and the ease of use of the portal from each device. So the evaluation concerned the *Fail* or *Success* of the operations and whether users agree or disagree with the statement "*The GUI of the CREATE portal is intuitive and easy to use*", as above for each device (tablets, smartphones etc.).

1.3.2 Results

The internal evaluation of the demonstrator provided positive feedback. The control of the production line and the individual devices via the Cloud behaved as was supposed to, in all tests performed. All possible operations for all modules were performed and examined, and each time the outcome of the operation was *Success*. Every time a command was communicated through the portal, the device(s) performed as the users intended.

The data generated during production were successfully broadcasted through the web services and the charts were depicting accurate information as was examined in cooperation of users, observers of the operation and personnel overviewing the physical devices.

Potential future users of the CREATE platform and specifically users interested in the reconfiguration features (e.g., system operators) answered that either strongly agree or agree with the statement that "The CREATE platform adds value to the reconfiguration

process". Users provided feedback, pointing that the results of the matchmaking applications can save time by filtering a vast amount of candidates in the production line and presenting suitable matches/substitutes (depended of the input of the users to the reconfiguration feature).

The user interface of the portal received very positive comments with most users that participated in the tests reporting that they strongly agree with the statement that "The GUI of CREATE portal is intuitive and easy to use". The users involved in the test ranged from potential future users to unrelated "control group" users (e.g., non-technical TIE employees) but all of them, after given a context of the application and presented with the portal, were able to operate the production line through it.

The portability of the platform, meaning the availability of accessing the virtual production line using devices equipped with web browser was also very successful. Tested by tablets, smartphones, laptops/PCs, the production line was also controlled, monitored and reconfigured successfully. The users however indicated that the monitoring of the production line was easier through bigger devices compared to small ones (smartphones) because, devices with bigger screens provided visualization of information from more modules without scrolling up or down.

4. Conclusions

After the several tests performed on the first version of the demonstrator and the interpretation of the results, conclusions can be drawn. Valuable feedback has been provided and the next steps have been analysed and scheduled. The CREATE approach was implemented successfully in the flexible material flow domain, on the first version of the demonstrator. The production line and its modules are exposed as web services and can be *controlled* and *monitored* in a very *portable* way from devices that are equipped with a web browser (tablets, smartphones, laptops/PCs etc.). In the future more devices will be exposed as services and the *dynamic* selection of the available services/modules will be possible. Moreover, the components of the production line are described as *resources* and the reconfiguration of the production line can be significantly faster because of the semantically enriched information schema containing active and potential components as well as the device matchmaking features and the artificial intelligence applications. The application will be extended for ordering the necessary replacements as resulted from the search from within the device matchmaking features of the portal. The GUI of the CREATE portal has been proven to be very intuitive to a range of users with different background but efforts will be put in improving it. In next steps TSB will be fully employed between the CREATE portal and the available services of the production line, to achieve communication that is secure, reliable and able to facilitate heterogeneous components. The first version of the

prototype is considered to reached its objectives and the first results meet the expectations of the market analysis performed for the CREATE approach. Next steps will be focused on a more generic approach towards the cross domain demonstrator.

5. Definition of Abbreviations and Terms

Abbreviation / Term	Definition
ESB	Enterprise Service Bus
GUI	Graphical User Interface
GSC	Generic Substrate Carrier
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
JSON	JavaScript Object Notation
OWL	Web Ontology Language
RDF	Resource Description Network
REST	Representational State Transfer
UDDI	Universal Description Discovery and Integration
SME	Small and Medium size Enterprises
SNM	Smart Neighbourhood Modules
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and Query Language
TaaS	Things as a Service
TSB	TIE Smart Bridge
VPN	Virtual Private Network
WADL	Web Application Description Language
WCF	Windows Communication Foundation
WS	Web Services
WSDL	Web Service Description Language
XML	Extensive Markup Language

6. Bibliography

Moreira, et al., 2008. Luciana Moreira, De Souza, Patrik Spiess, Dominique Guinard, Moritz Köhler, Stamatis Karnouskos, and Domnic Savio. 2008. SOCRADES: a web service based shop floor integration infrastructure. In Proceedings of the 1st international conference on The internet of things (IOT'08), Christian Floerkemeier, Sanjay E. Sarma, Marc Langheinrich, Friedemann Mattern, and Elgar Fleisch (Eds.). Springer-Verlag, Berlin, Heidelberg, 50-67.

(s.f.). Obtenido de http://www.wisegeek.com/what-is-web-services-for-devices.htm

SOAP 1.2. (17 de December de 2001). Obtenido de W3C: www.w3.org/TR/2001/WD-soap12-part0-20011217/

UDDI. (2004). Obtenido de UDDI v3.0.2: http://uddi.org/pubs/uddi_v3.htm

Web Service Definition. (11 de February de 2004). Obtenido de W3C: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice

WSDL. (26 de June de 2007). Obtenido de W3C: http://www.w3.org/TR/2007/REC-wsdl20-adjuncts-20070626/

That 'Internet of Things' Thing. (22 de June de 2009). Obtenido de RFID Journal: http://www.rfidjournal.com/articles/view?4986

WADL. (31 de August de 2009). Obtenido de W3C: http://www.w3.org/Submission/wadl/

Christian Reinisch, W. G. (s.f.). Integration of Heterogeneous Building Automation Systems using Ontologies.

Eleni Christopoulou, A. K. (s.f.). GAS Ontology: an ontology for collaboration among ubiquitous computing devices.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 907-928.

Ippokratis Pandis, J. S. (s.f.). An Ontology-based Framework for Dynamic Resource Management in Ubiquitous Computing Environments.

json. (s.f.). Obtenido de json: www.json.org

Lopez, U. A. (2007). A Semantic Matching Algorithm for Discovery in UDDI. *Proceedings of the International Conference on Semantic Computing*, 751-758.

Rusher, J. (13 de 11 de 2003). *Triple Store*.

Studer, S. A. (2006). Automatic Matchmaking of Web Services. *IEEE 4th International Conference on Web Services*, 45–54.

W3C. (10 de 02 de 2004). *RDF Primer*. Obtenido de RDF Primer W3C Recommendation 10 February 2004.

W3C. (15 de 01 de 2008). *SPARQL Query Language for RDF*. Obtenido de W3C Recommendation: http://www.w3.org/TR/rdf-sparql-query/

W3C. (11 de 12 de 2012). *OWL 2 Web Ontology Language*. Obtenido de http://www.w3.org/TR/owl2-overview/

W3C. (29 de 03 de 2013). *W3C Semantic Web Activity*. Obtenido de http://www.w3.org/2001/sw/

Yazen Alsafi, V. V. (s.f.). Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing.

2. Industrial Metrology

2.1. Introduction

This demonstrator aims to show the benefits and advantages acquired by the production lines when the CREATE methodology is implemented. In this case we have implemented the architecture and tools than have been developed in the previous WPs, such as the integrator and communication modules, the metrology robots and optical sensor and all the data processing algorithms.

In the following figure it is shown the scheme of the demonstrator, it is based on the real EPC's production line for camshafts manufacturing.

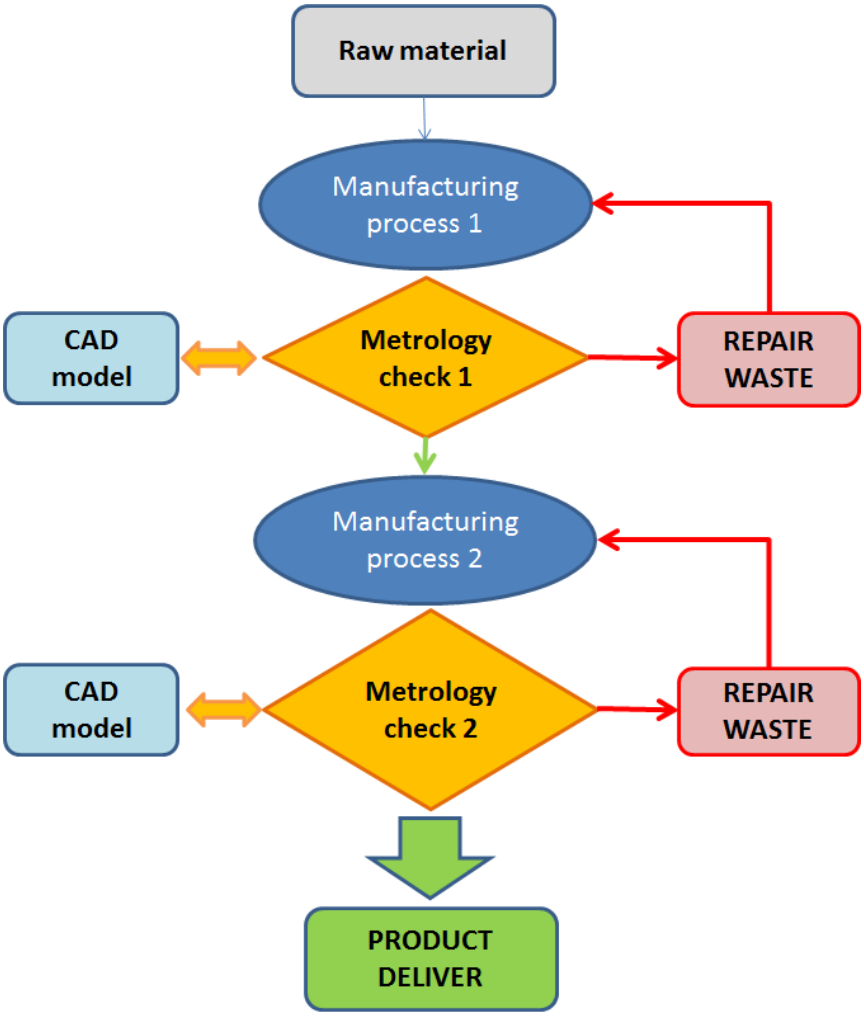

Figure 8: Industrial metrology demonstrator scheme

As it can be seen, we have established two metrology controls in order to check in real time the quality of the pieces after two of the main manufacturing processes. In both cases the data will be acquired by an optical sensor, in the first case mounted in an articulated robot and in a Cartesian machine in the second.

After each control the virtual part will be compared to the CAD model to determine if the manufactured piece fulfils the dimensional tolerances required by the company. More details about the measuring plans are explained in section 2.2.3. From these results the system will take a decision about the next processing step:

1. If dimensional measure is below tolerance, the piece can follow in the manufacturing line
2. If dimensional measure is above tolerance, the piece will be defined as defective and may be sent for repairing processes or eliminate it for material recycling.

In both cases the decision will be made automatically by the system as a function of the tolerances defined by the company and the results calculated by the metrology SW.

When the manufactured piece reaches the final step of the manufacturing line, we can be sure that this piece is dimensionally correct, with no deviations and fulfilling all the requirements. In this way, the productivity will be increased by reducing corrective actions and avoiding defective product sales.

2.2. Brief tool description and integration

2.2.1. Tool integrator

The industrial metrology demonstrator has implemented the CREATE Integrator Module, CMI, to allow the integration of the different SNMs. As explained in the D3.1 this module guaranties the communication and interoperability of the system depending on the SNM configuration required by the company.

For this case, the demonstrator will be form by HW and SW SNMs. In the HW part, it has been selected a set up for optimizing the data acquisition for dimensional quality control: an articulated robot, a CMM and one optical sensor; this equipment is directly integrated in this platform and the communication between them will be supported by it. It this way the trazability of the information is ensured and also the data can be easily store independently of the source. Concerning the SW modules, this demonstrator has implemented the metrology software required for measuring plans and geometry extraction and the module for communication with the equipment, iCMeCom (see D3.1).

This represents just an example of a real configuration for an industrial case. Depending on the requirements of the company for the dimensional quality control more tools can be added or removed from the platform. In this way the production line can be reconfigured as a function of the market and product needs. Moreover, other solutions such as language translator and monitoring control (some of the solutions developed in the other demostrators) may be inserted in the CMI, so the companies can implement new configurations with both existing and new solutions acquired.

2.2.2. Industrial metrology set up

The metrology set up needed to implement the dimensional quality control industrial environments required hardware (robots and sensors) and software (controls, measuring plans and geometry extraction) components.

Coordinate Measuring Machine

This sort of machine has up to three linear orthogonal axes. This robot configuration is widely used in metrology because it is easy to calibrate and it brings a superior accuracy than articulated robots, with a relation of 1/8. Its workspace may be a line, rectangle or cubic volume depending on the number of axes. Due to the configuration and dimensional properties of the camshafts, we have selected a CMM with just one axis, then the movement head orients the optical sensor to acquire the point clouds.

Figure 9: Coordinate Measuring Machine for large camshafts

As mentioned above, to change the orientation of the sensor, a rotating head of up to tree axes can be added. In metrology applications, this is an indexed head, which can take only a limited number of fixed orientations, for an increased repeatability. The rotation of this head is considered as a configuration change and it is never performed during the sensor operation. Note that if the rotating head were able to perform a

continuous movement of its axes, synchronized with the linear axes, the resulting robot could not be considered as a Cartesian one.

Articulated robot

The components of its structure are linked with rotary joints. The movement of these joints allows the tip to reach all the points in its workspace. The volume and shape of the workspace depends on the number of joints and on the assembly. Some tip positions may be reached with several combinations of joint positions.

Figure 10: Articulated robot for in-line scanning

In this demonstrator we are using this robot to acquire point clouds after the first machining process, so just a critical zone of the camshafts is controlled.

Optical sensor

The optical sensor used in this demonstrator is based on laser triangulation. The laser optics generated an intensity constant line on the piece surface to be controlled. The deformations of the line over the piece are reflected in the CCD camera. From the known working distance and the angle between laser and camera, it is possible to calculare the coordinates of the surface points.

Main advantages of this type of technology are: piece is not contacted, high amount of information acquired, high velocity and very high fidelity.

The model used in this demonstrator has the following technical properties:

- Speed scan: 60.000 scans per second
- Working distance: 100 mm
- Vision field: 40 mm
- Laser class II
- Precision: 10 µm

Figure 11: Optical sensor for pointcloud acquisition

Communication protocol

The communication protocol developed for this case is based on a modular software structure:

In order to maintain the integrity and flexibility of the solution, the control of the device has not been modified. This control talks to the driver, API or physical controller of the device and performs the needed processing and data formatting to expose the resource in a standard way to the rest of the system, through a server. This configuration allows installation and removing the robots and sensors without affecting the rest of the system.

As each HW component has its own reference system and acquire different coordinates, it is required another module that manages the synchronized acquisition, assembling the data from robots and sensor, taking care of some synchronization issues. For example, the acquisition must be started in the device that receives the trigger before than in the device that issues the trigger.

So, the communication protocol can be built upon an ideal model of the system, regardless of the peculiarities of the hardware.

This scheme has two flows; on one hand all the commands are originated in the custom application, and are sent to the module which is responsible for the functionality: movement commands are sent to the RobCom Server; synchronized acquisitions are ordered to the Data Assembler, while non-synchronized acquisitions may be directly ordered to the SenCom Server.

On the other hand, information flow follows the inverse path: from the data sources, it is sent to the custom application. Synchronized information has to be sent to the Data Assembler, in order to be combined before being sent to the application.

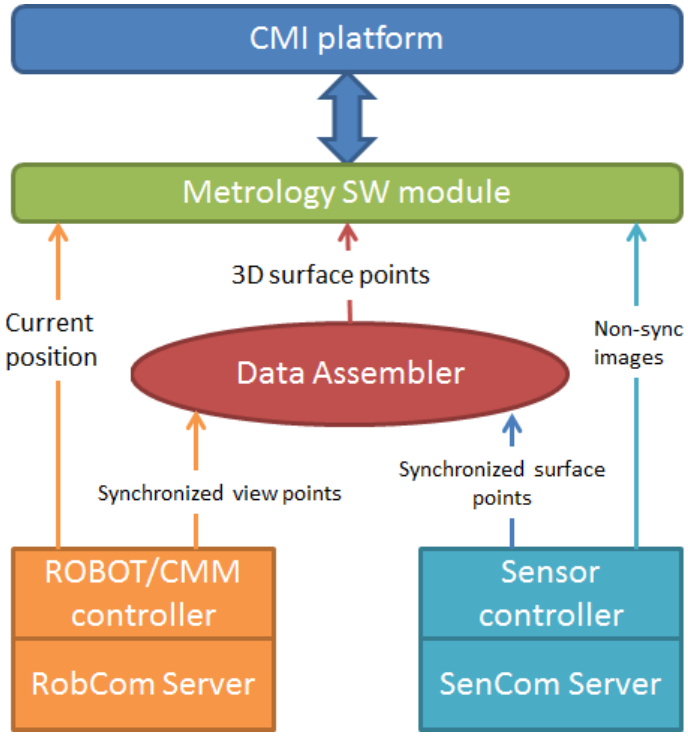

Figure 12: Communication architecture and information flow

2.2.3 Trajectories and measuring plan

The basis for the real time decision support is the acquisition of real data in real time. As mention above, the best technology to achieve this requirement is the optical sensors. This technology allows the generation of virtual parts that are the digital replica of the physical ones.

To generate the virtual part it is necessary that the sensor scans all piece surfaces. This issue may imply some time-consuming if the trajectories and movements are not optimized.

The metrology module enables a user-friendly interface to develop the robot and sensor movements so the time is minimized. Next figure shows the visual aspect of one of the programmes.

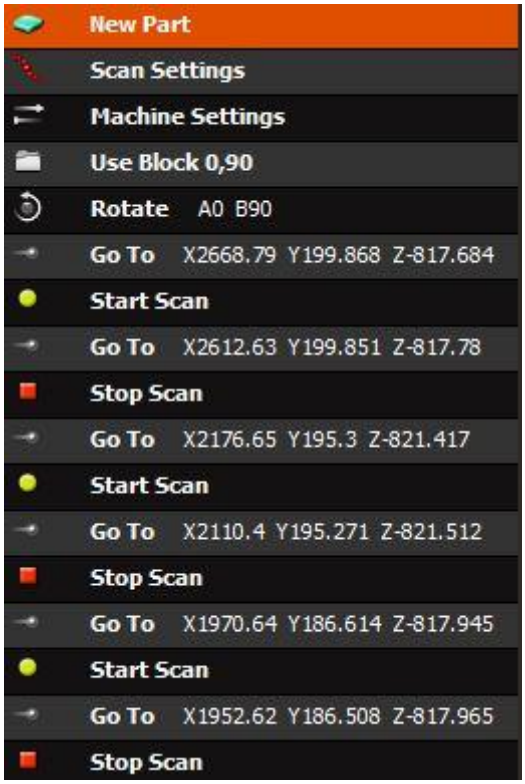

Figure 13: Movements and trajectories for scanning

The main functions needed to generate the trajectories are: Start, stop scanning, go to and rotate. Then, the lower layer implements automatically the complete program and the communication with the robots and sensor.

Once the sensor has scanned all the surfaces, the user starts measuring the geometries by using the pointcloud and the corresponding virtual part. A measuring plan is the programme used for planning of the features and geometries to be controlled in each piece. Depending on the shape, size and geometry of the manufactured product the user can be customized the distances, angles, areas and geometries.

This module allows the definition and saving of new custom geometries depending on the piece shape. For example, in the case of camshafts, the geometry key (“chaveta”) has been created as it represents a critical feature. Once the measuring plan is defined, the user can enter the upper and lower tolerances than can be assumed by the manufacturing processes. Then, the report shows the result both in quantitative and qualitative way by a colour code. As it can be seen in the figure below, all the distances and geometries agree with the dimensional requirements but one of them, a height, is out of tolerances and a red signal appears.

Figure 14: Measuring plan and geometries to be controlled

2.2.4 Decision making support

In line implementation of non-contact dimensional metrology set up enable the possibility of make decision based on real and real time data.

To optimize these capacities to metrology set ups has been implemented in order to control different features depending on the production process phase.

Point clouds and virtual parts are generated in real time, so immediately the sensor ends the scanning the metrology SW module starts processing and generating results.

In the first metrology check, by using the articulated robot just the critical features are controlled. In this case, the key of the camshaft is controlled after the first machining process; this is a critical geometry related to the coupled of the piece with the engine, so any minimal deviation generates no-acceptance by the customer due to high risks of engine breaks.

Next figure shows the colour mapping of the key and the results from the analysis of the geometry. The dimensions and shape of the outer surface are within the tolerances but the diameter of the key is out of it, so a red signal appears in the result table.

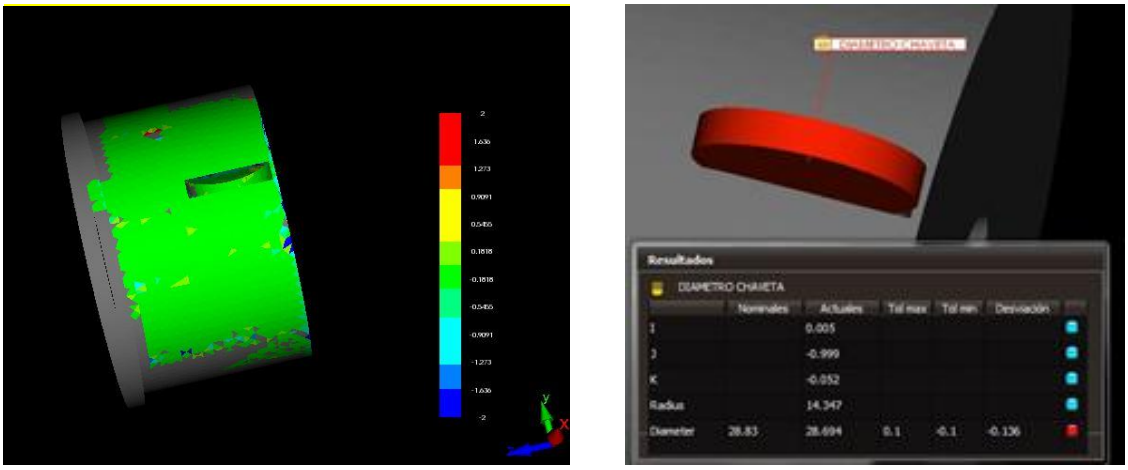

Figure 15: Key analysis in the first metrology check

With this result the system automatically decided to take out that piece from the production line, avoiding over cost and saving energy. Afterwards, depending on the defect and the deviation, this piece will be remanufactured or neglected.

In case the first metrology check does not detect any deviation, the piece is machined in the next process. At the end of the line, the final manufactured camshaft is scanned in a CMM in order to generate the virtual part to be used for real time analysis and other measurements in the future. This is a complete measure of the piece, so many distances and geometries are controlled in order to ensure that the piece fulfil all the dimensional requirements.

The first result is a colour mapping of the piece in order to get a visual result. Moreover, a report with all the quantitative results and the differences with respect to the nominal value are shown (Section 2.2.4).

Finally the virtual part, the measuring plan and the results are stored in the server for future modifications and reviews.

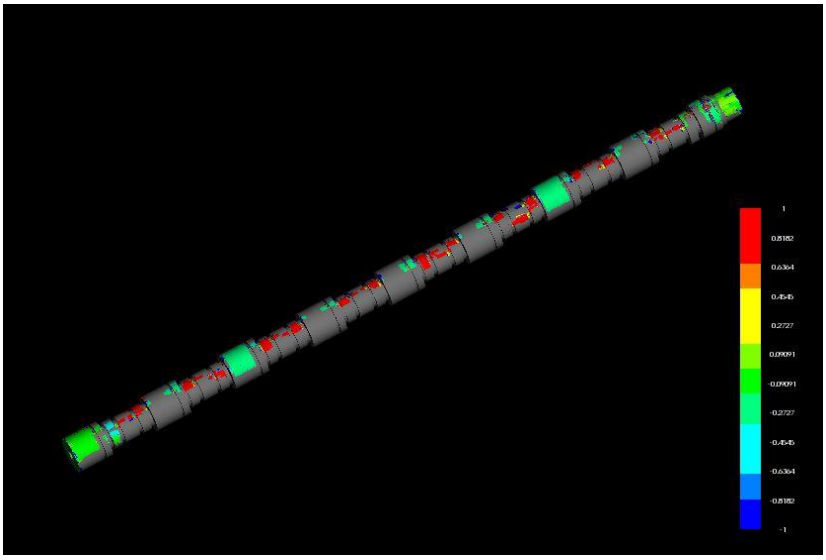

Figure 16: Colour mapping of manufactured camshafts

2.2.5 Data storage

One of the main advantages of this demonstrator is the implementation of the virtual metrology methodology in industrial plants. VP allows the generation of measurements and final information by measuring and analysing the virtual parts. So, the system has to implement an automatic storage of the VP and the information generated, in order to be accessible and modified in the future.

Each part that is scanned and controlled will be saved as an independent project. New and existing measuring plans can applied to one project or to a list of project with the aim of measuring a new feature or controlling the dimensional evolution of some critical geometry.

For example, one digital camshaft part is form by more than ten millions of points, which is saved in files of more than 200 MB. For that reason, the vast amount of data and information has to be compressed and saved to avoid overload of the computer and reduce storage requirements.

For this purpose, CBT, TRIMEK and DATAPIXEL with the support of INNOVALIA have been working of algorithms for filtering the raw data and optimize the segmentation process for geometry extraction. In that way, the size of the files is reduced and the resource consumption lower.

2.3. Test and results

In this section some of the main test and measurements are shown for the dimensional quality control of camshafts. The implementation of this equipment is not in-line at this stage of the project, but all the tools and solutions developed so far have been completely tested.

Nest figure shows the point cloud of a camshaft, it represents the type of data that will be acquired at the end of the line, when all the manufacturing processes have passed the intermediate quality controls.

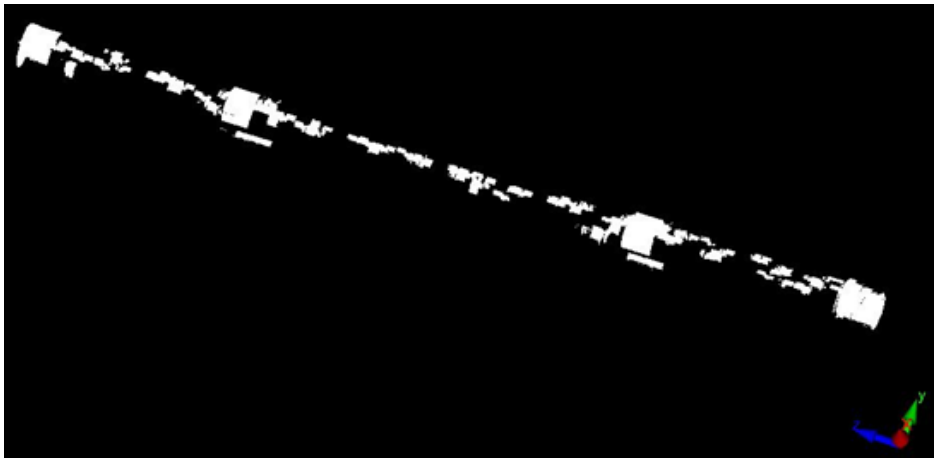

Figure 17: Camshaft pointcloud

This representation provides no information to the company; it is just a set of points in the space. However, it helps to ensure that the integration of the robots and sensors in the platform has been done properly. User has been able to apply the trajectory plan and acquire information from the sensor.

In order to generate valuable information and check if the data has been collected in the good way, the metrology module has to process this information and apply the measuring plan defined at the same time than the product design. For example, if the comparison between the CAD model and the virtual part (Figure 9) shows no relation, it means that the data acquired by the system were wrong. The normal situation is that some locations and feature can differ from the theoretical model, not all the piece.

With the aim of exposing the potential of the metrology module, customized geometries have been generated for this particular case: key and cam; moreover some other common geometry such as height, diameter and parallelism are also shown.

Param	Nominal	Tol max	Tol min	Actuales	Desviación
ANCHURA CHAVETA					
dY	6.340	0.023	-0.023	6.375	0.035
SIMETRIA					
dY	0.000	0.100	-0.100	0.062	0.062
Distancia 75					
dZ	53.670	0.250	-0.250	53.599	-0.071
ALT CHAV					
dX	33.820	0.050	-0.050	33.771	-0.049
PARALE					
D	0.000	0.050	0.000	0.034	0.034

Figure 18: Measurement of the cam

Param	Nominal	Tol max	Tol min	Actuales	Desviación
Leva ADM4					
Angle	15.430	1.000	-1.000	14.756	-0.674
Leva ESC6					
Angle	0.640	1.000	-1.000	0.113	-0.527

Figure 19: Measurements of the key

Param	Nominal	Tol max	Tol min	Actuales	Desviación
DIAM AGUJERO ESPIGA					
Diameter	6.500	0.500	-0.500	6.523	0.023
ALTURA ESPIGA					
X	-18.500	0.250	-0.250	-18.736	-0.236
PERP					
D	0.000	0.020	0.000	0.008	0.008

Figure 20: Measurement of spike

In these tables the titles are geometry / nominal value / max tolerance / min tolerance / real value / deviation. Then the colour code display in a visual way if the measurements fulfil with the tolerances. Note that for each geometry an own tolerance can be defined, depending on the manufacturing capabilities and customer requirements.

These measurements show that just the width of a key is out of tolerances, so this piece will be taken out of the production line and evaluated for repair action and remanufacturing.

2.4. Conclusions

The first version of the industrial metrology demonstrator has fulfilled the objective of testing and showing the integration of the solutions and its capabilities for being integrated in a production line.

The CMI platform has allow the integration of two different robots and one optical sensor for acquiring the product information. The communication between the controls and the platform has managed the data for generating good pointclouds. Then, the metrology module has received this set of data to apply the measuring plan.

Afterwards, the results automatically generate a report with the main critical features to be controlled in quantitative and qualitative way. The system exploits the visual report for a quick detection of defects by the humans and also generates alerts when a measurement is out of tolerances for automatically takes the decision of retiring the manufactured piece of the line.

This demonstrator shows some results that are advances beyond the state of the art. For example, the CMI allows the integration and communication of different robots and sensors, enabling the communication and management of data and operation from a single platform. Moreover, the metrology module applies automatically the measuring plan depending of the type of piece that is manufactured, generating a report that is also customized. Finally, the interaction between the metrology module and the production line allows the generation of automatic decision, such as retire the piece from the line when a defect is detected.

The following steps of this demonstrator will be the integration of all this solutions in the EPC production line, so the demonstrator can be evaluated in real working conditions. For that purpose, some work concerning the scanning speed and data processing has to be done before the implementation of that second version.

3. Monitoring and Quality Control

3.1. Introduction

The basic idea behind Case-Based Reasoning (CBR) is that similar problems have similar solution, so the notion of similarity is right in the core of CBR. CBR uses experience to solve problems using a cognitive process similar to how humans reason. If we face a problem, we most often solve it by applying a solution from a similar situation from the past.

Case-Based Reasoning in general is not considered to be a set of algorithms or methods but a knowledge management methodology. Thus, CBR is usually defined as a system acting according to the CBR cycle. A CBR system has at least four processing steps: Retrieval, Reuse, Revise and Retain. Given a new case problem, the set of cases most relevant or similar are retrieved. Then, the retrieved solution(s) are applied to the new problem. Thereafter, the outcome of applying the adapted solution from the previous step is evaluated.

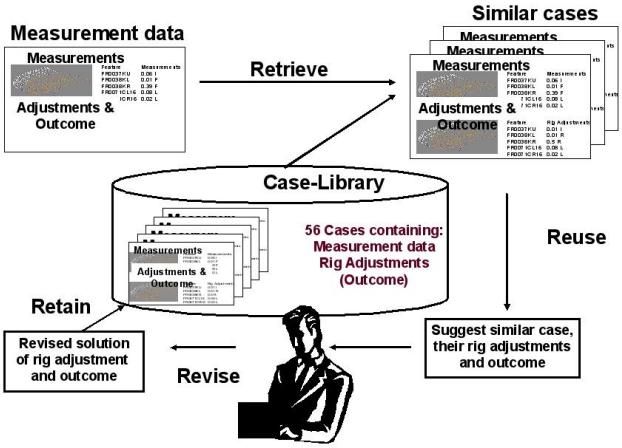

Figure 21: CBR applied to geometric production measurements (Volvo CE)

Technicians in manufacturing industry estimate that a cost reduction of more than 30% in corrective and preventive maintenance is achievable through the use of better tools for monitoring and decision support. The Worldwide market in automation equipment for monitoring and controlling is evaluated with €188B [Report EC, 2008].

Overall production equipment effectiveness is in Swedish industry estimated to ~60% and has an improvement potential of more than 20% (NUTEK 2006/2009[10], Ahlmann 2002[11]). Ahlmann assigns 6,7 billion € per year of the improvement potential to factors such as quality drawbacks, delivery delay, production losses, reduced scrap, reduced running in costs.

With the *CREATE* approach including monitoring, quality control and diagnostics a large part of this potential is achievable since the core problem of a dynamic production system easily can be reconfigurable to meet real time production requirement with aspect to software and hardware.

In order to optimise the product life cycle it is necessary to monitor the production process and the use of the product, especially when the life cycle of a product is in focus. The collected data enables both continuous quality improvement as well as continuous improvement of the life cycle cost in terms of resources, environmental, product life span and maintenance. Today much of the experience in how to diagnose and carry out maintenance is manual labour by technicians and experts acquiring their skill over many years. This makes the process sensitive to change of staff and knowledge transfer is difficult.

In the area of monitoring, quality control and diagnostics and innovative approach making manual monitoring, quality control and diagnosis easily interchangeable with automated equipment is desirable.

On the Swedish monitoring and quality control demonstrator, Mälardalen University is developing their learning algorithms for diagnostics and prognostics. Data has been collected at Volvo CC/CE from their manufacturing and aftermarket, used by Mälardalen University to develop the algorithms with real data.

Also the hardware and software module is under development at SEMA-TEC and will contain a web server so it can communicate with the diagnostic/prognostic module at Mälardalen University and connect to different sensors and actuators. Furthermore, SEMA-TEC and TIE are cooperating for applying the web services for third parties.

The algorithms for monitoring and quality control are developed at Mälardalen University; collecting data from sensors is on-going in manufacturing at Volvo CE/CC and developing the hardware module with operating software to connect sensors, actuators and ability to communicate.

[10] NUTEK Report on "Productivity Potential Assessment", Swedish Agency for Economic and Regional Growth – Tillväxtverket

[11] Ahlmann, H. (2002): "From Traditional Practice to the New Understanding: The Significance of the Life Cycle Profit Concept in Management of Industrial Enterprises". International Foundation for Research in Maintenance: Maintenance Management & Modelling Conference, Växjö, Sweden.

In Sweden the monitoring and quality control module for manufacturing is being developed for sound and dimensional measurements. SEMA-TEC is currently developing the necessary hardware and software module and Mälardalen University is developing the diagnosis and prognosis algorithms and module able to learn and improve performance. Volvo is providing classified data from their manufacturing sites enabling Mälardalen University to develop their algorithms with real data used for evaluation.

The hardware and software module developed for sensor measurements is made by SEMA-TEC for the CREATE project will result in a flexible hardware unit with integrated intelligence, multiple reconfigurable inputs and outputs for virtually any sensor signals, versatile communication protocols and TCP/IP communication. The flexible software solutions and communication abilities ensures its ability to operate as a node within a networked solution as well as an autonomous unit performing all required tasks. The digital and analogue outputs offers the possibility to automate and perform corrective actions, i.e. any deviations measured by sensors, assessed by the algorithms, resulting in a physical corrective action, such as activating an actuator, adjusting a control signal, etc. can be performed directly by the unit. Case-based reasoning algorithms are currently developed by Mälardalen University that may be integrated in the unit's software system, or, the measured data can be uploaded to an external service/database and analysed by an external application. The unit can also perform corrective actions downloaded to the unit and physically executed.

Volvo CE/CC are collecting case-data used by Mälardalen University to develop their diagnostic and prognostic algorithms. The data is from a number of different sources in production (both sound and dimensional data) in the production and aftermarket enabling the development of more generic algorithms. One PhD student is working on the development of the algorithms and framework stationed both at Volvo CE and Mälardalen University.

SEMA-TEC and Mälardalen University in communication with TIE have looked at examples for web services with 3rd party interest in the area of monitoring, quality control and diagnostics with remote access to sensors, data, history etc. By enabling modules to request web services to perform different tasks, e.g. enabling different providers to provide services of analysing and classifying sensor data and delivering back results to the monitoring and quality control module, making the final decision on corrective action or adjustments of the manufacturing process.

3.2. Brief tool description and integration

Mälardalen University has developed a framework with a number of algorithms that can be used to quickly build a decision support tool in monitoring and quality control. By both handling dimensional data and sound data shows that the tool is generic and can be adapted to different manufacturing situations.

To give a picture of the CDSS we give the Volvo Car case as an example:

Decision support module for assembly fixture adjustment

The gore consists of nine ingoing parts which each one contributes with its own variation. The position of the ingoing parts is fixed in an assembly fixture. The internal position of the parts can be adjusted by adding shims at the position of the reference- and support points.

The position of the gore in the car can be seen in figure 3

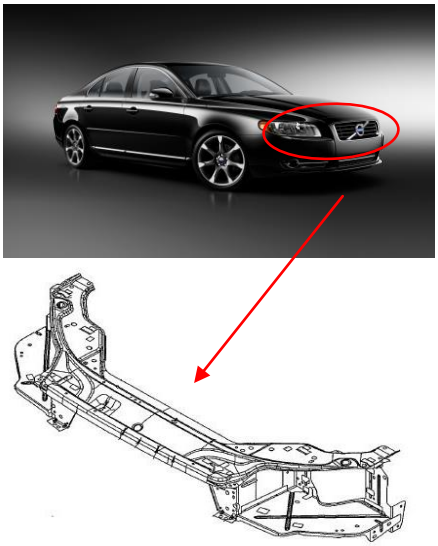

Figure 16: Position of the gore in the car

In the next figure the assembly cell is shown. The position of the ingoing parts is fixed in an assembly fixture. The internal position of the parts can be adjusted by adding shims at the position of the reference- and support points. Adjustments are used for compensate for the variation of the ingoing parts in order to reach the specification demands on the final sub assembly, the gore. The geometry of the part is controlled by measurement of predefined points, spread over the geometry to reflect the process stability.

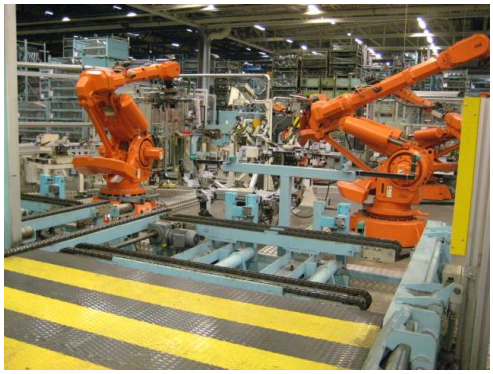

Figure 22: Assembly cell

Cases in the case library are built on a generic manufacturing structure, measurements adjustments and outcome. Over time more and more cases are collated in the production improving performance and accuracy of the system:

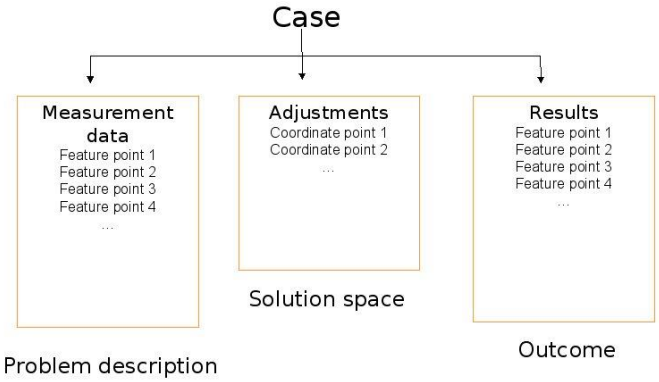

Figure 23: Case representation of geometric production measurements

The decision support module will be able to give advices on corrective and preventive actions and enable transfer between manual monitoring and control and fully automated control, diagnosis and correction (communicating with following smart neighbourhood module (SNM) that is able to perform corrective actions. By building up and sharing experience amongst the monitoring modules, which perform similar tasks, the modules will learn and improve their capabilities. The measurements and decision will follow each manufactured part and can be used for corrective actions, classification or as a reference value later on in the product life cycle.

New cases in the case library contain valuable experiences for future adjustments and improvements. Ideally this will prevent mistakes from occurring more than once as the experience will propagate out to all the relevant modules and enable the production system to learn and improve. According to technicians and engineers, reoccurring

mistakes constitutes a large part of production costs. Monitoring and diagnosis over complex production processes can be further automated than with traditional approaches based on modelling, simulation or statistics. The CREATE approach enables decisions and decision support on at least the same level, or better, than experts. One of the advantages is that the case based approach is able to suggest solutions prior to sufficient data have been collected enabling quantitative methods for decision support. It will also become better than individual experts once the number of cases grow in the case library capturing the collective experience and knowledge. The monitored data and decision connected to a specific manufactured part enables feedback to previous steps in the manufacturing and design process. For example if certain problems occur more or less frequently after a design alteration, the design department will immediately be notified about it and necessary actions can be taken to adjust the design to improve the product or production process.

The methods developed have been applied both for dimensional and sound data from manufacturing and show ability to identify similar cases used for decision support. Running the algorithms as a web service on a server gives enough processing power to make fast and reliable solutions. It also keeps the processing power on the SNM connected to the sensors and eventual actuators low. If the bandwidth is not large enough then the SNM with the sensors may need to carry out sensor signal abstraction (in collaboration with SEMA-TEC).

3.3. Test and results

The data collection and algorithms have been tested at Mälardalen University and perform well, currently we are working on a scientific publication aiming at a journal publication describing parts of the algorithms and the results.

Both Volvo Car and Volvo Construction Equipment have collected and classified sensor data for the project (geometrical and sound). The data has been used by Mälardalen University to develop the algorithms and methods further.

SEMA-TEC is developing the hardware module connecting the sensors and actuators with the web service. The hardware will be completed before summer. The hardware is first critical when doing the cross platform demonstrator since it is the link between the production and external SMNs and web services. By using PC computers we have been able to validate and test the monitoring, quality control and diagnosis case. As soon as the hardware is completed we will start testing the SNM for monitoring as a web service.

3.4. Conclusions

We have shown that the Case-Based Reasoning methodology is a potent solution for monitoring, quality control, diagnosis and corrective actions in manufacturing. The algorithms are able to learn and improve their performance. By using real sensor from production in the demonstrator empirical results show that the proposed approach also works in a practical context. All partners have delivered and participated fully and we are in the most active phase in the project. Mälardalen university has secured funding for one more project member (faculty funding) since the project is seen as a strategic investment giving the university credit for their applied research and producing value for industry and SME companies.

4. Conclusions

The CREATE project is at its most fruitful phase, with first versions of the three different demonstrators developed for all defined use cases proving the added value of the CREATE approach in different domains.

This version one of the demonstrator exposes the initial results of the CREATE methodology. The first aspect to consider is that, in the whole document, in all the cases the same architecture has been assumed, so that, although each use case has developed its own solutions to meet the industrial cases, it will be quite easy to integrate and implement them in the final cross domain demonstrator.

It is also shown that each demonstrator is in a different level of development, but they are advanced enough to clarify the CREATE methodology and the sort of information and result generated from its implementation.

For the second version of these demonstrators, expected by 28/02/2014, more advanced implementation and real working conditions test and results will be shown to validate the efficiency and benefits provided by CREATE. Also an integration of solutions between the demonstrator is been taken into account, so the interoperability of the solutions may be shown clearly.

After the first evaluations and feedback, the partners will work even more closely, focusing on the cross domain demonstrator. Next steps have already been decided, and actions have been planned towards this objective.

To sum up, at this stage of the project the technical state of the demonstrators shows that the CREATE architecture and the related technologies can provide valuable benefits in the companies independently of the industrial sector, which is one of the key objectives of the project.

Annex

A. Flexible Material Flow

This section aims to present some parts of the software developed for the exposure of the devices as web services, as well as resources. The web services (both service wrappers and services that expose the device matchmaking features) were developed in Microsoft Visual C# 2010 Express (.NET Framework 4) using the Windows Communication Framework (WCF). The ontology developed to host information associated with the production line and the population of the ontology was implemented using the protégé tool. This information is necessary for the device matchmaking applications for reconfiguration. The information is stored in semantic repositories (triple stores). The services related to the reconfiguration use an open source .NET library (dotNetRDF) for its semantic capabilities (use of triple stores, SPARQL etc.).

A.1 Devices as web services implemented via service wrappers

Each module is exposed as web service by means of web service wrappers that make the operations of the automation software that controls the device, available to the cloud. We will present the approach of doing that, by presenting certain sections of the service wrapper solution for the Generic Substrate Carrier (GSC) module.

The solution includes a project that contains the DataContracts, so that service providers and clients have an agreement of the data to be used in their transactions such as enums with the parameters, signals etc. for each module and a class with the parameters of the module which hold their values. The DataContracts solutions contains service reference to System.ServiceModel and System.Runtime.Serialization.

The GSCService project contains the service contracts, the implementation of the services and operations, as well as hosting the services and exposing them for potential clients. The project is saved as .NET 4 and not as client applications (from project's properties) and has references to System.ServiceModel, System.ServiceModel, System.ServiceModel.Web , System.Runtime.Serialization and the DataContract project created earlier. The methods that are described in the description of services for GSC are defined as operation contracts in the service contract. Each module has different service contracts, because the input parameters for the methods for each contract differ (the methods have different enums as input for each module). In the operations defined we also annotate the restful method (WebGet), a template for the uri for accessing the

services as well as the response format (json). A sample of the contracts is presented below:

```
using System;
using System.Collections.Generic;
using System.ServiceModel;
using System.ServiceModel.Description;
using System.ServiceModel.Web;
using System.Text;
using System.Runtime.Serialization;
using DataContracts;

namespace GSCConveyor_ServicesWrapper
{
    [ServiceContract]
    public interface IGSCConveyor
    {
        [OperationContract]
        [WebGet(UriTemplate = "GetState", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        eState_Generic Get_State();

        [OperationContract]
        [WebGet(UriTemplate = "GetNotifications", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> Get_Notifications();

        [OperationContract]
        [WebGet(UriTemplate = "GetErrors", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> Get_Errors();

        [OperationContract]
        [WebGet(UriTemplate = "InitializeModule", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        int Initialize();

        [OperationContract]
        [WebGet(UriTemplate = "StartModule", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        int Start();

        [OperationContract]
        [WebGet(UriTemplate = "StopModule", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        int Stop();

        [OperationContract]
        [WebGet(UriTemplate = "Identification", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        string Get_Identification();

        [OperationContract]
        [WebGet(UriTemplate = "ModuleType", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        string Get_ModuleType();

        [OperationContract]
        [WebGet(UriTemplate = "getParam?par={eParam}", ResponseFormat =
WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped)]
        double Get_Param(ParametersGSCConveyor eParam);
```

Figure 24: Web service contract for the conveyor

These services and operation contracts are implemented by the GSCConveyor:

```
namespace GSCConveyor_ServicesWrapper
{
    public class GSCConveyor : IGSCConveyor
    {
        /// <summary>
        /// Method that requests the state of the module.
        /// </summary>
        /// <returns>An enumerable with the state of the module.</returns>
        public eState_Generic Get_State()
        {
            eState_Generic state = Conveyor.ModuleState;
            return state;
        }

        /// <summary>
        /// Methods that requests notifications/warnings associated to the module.
        /// </summary>
        /// <returns>A list of strings with the warnings added since last call of
method.</returns>
        public List<string> Get_Notifications()
        {
            Conveyor.wrnCounter = Conveyor.tempWarnings.Count + Conveyor.wrnCounter;
            Conveyor.tempWarnings.Clear();
            for (int i = Conveyor.wrnCounter; i < Conveyor.Notifications.Count; i++)
            {
                Conveyor.tempWarnings.Add(Conveyor.Notifications[i]);
            }
            return Conveyor.tempWarnings;
        }

        /// <summary>
        /// Method that requests errors associated to the module.
        /// </summary>
        /// <returns>A list of strings with the errors added since the last call of the
method.</returns>
        public List<string> Get_Errors()
        {
            Conveyor.errCounter = Conveyor.tempErrors.Count + Conveyor.errCounter;
            Conveyor.tempErrors.Clear();
            for (int i = Conveyor.errCounter; i < Conveyor.Errors.Count; i++)
            {
                Conveyor.tempErrors.Add(Conveyor.Errors[i]);
            }
            return Conveyor.tempErrors;
        }

        /// <summary>
        /// Sends command to initialize module by setting it's state to READY.
        /// </summary>
        /// <returns>An integer that indicates success or fail of the operation.</returns>
        public int Initialize()
        {
            if (Conveyor.ModuleState == eState_Generic.UNKNOWN || Conveyor.ModuleState ==
eState_Generic.ERROR)
            {
                Conveyor.ModuleState = eState_Generic.READY;
                Console.WriteLine("Attempted to Initialize, returned 0");
                return 0;
```

Figure 25: Implementation of service and operation contracts

In the implementation of the service contract we allow clients to act on the state of GSC with operations such as Start(), Stop() and Initialize(). Also clients can set parameters for the module (for those that is allowed) with Set_Param(), get parameter values (Get_Param()) and signals (Get_Signal()). They access the services using Uris composed of the endpoint exposing the services and the template for each method. Examples of implementation of operation contract defined are presented below:

```
/// <summary>
 /// Method that initializes the module, by setting its state to 'READY'.
 /// </summary>
 /// <returns>Integer. If the operation was succesful 0, else 1.</returns>
 public int Initialize()
 {
     if (_ModuleState == eState_Generic.UNKNOWN || _ModuleState ==
         eState_Generic.ERROR)
     {
         _ModuleState = eState_Generic.READY;
         return 0;
     }
     else
     {
         Console.WriteLine("Error: Module has to be in state 'UNKNOWN' or 'ERROR'
         in order to Initialize()");
         return 1;
     }
 }
```

Figure 26: Implementation of the Initialize operation

```
public double Get_Signal(SignalsGSCConveyor eSignal)
        {
            double signal = Double.NaN;
            switch (eSignal)
            {
                case SignalsGSCConveyor.X_POS:
                    if (Conveyor.X_POS >= Conveyor.X_Err)
                    {
                        Conveyor.Errors.Add("["+DateTime.Now+"] X_POS reached error
level.");
                        Console.WriteLine("X_POS reached error level.");
                        Stop();
                    }
                    else
                    {
                        signal = Conveyor.X_POS;
                        if (Conveyor.X_POS >= Conveyor.X_Wrn)
                        {
                            Conveyor.Notifications.Add("[" + DateTime.Now + "] X_POS
reached warning level.");
                            Console.WriteLine("X_POS reached warning level.");
                        }
                    }
                    break;
                case SignalsGSCConveyor.Y_Speed_Act:
                    if (Conveyor.Y_Speed_Act >= Conveyor.Y_Speed_Err)
                    {
                        Conveyor.Errors.Add("[" + DateTime.Now + "] Y_Speed_Act reached
error level.");
                        Console.WriteLine("Y_Speed_Act reached error level.");
                        Stop();
                    }
                    else
                    {
                        signal = Conveyor.Y_Speed_Act;
                        if (Conveyor.Y_Speed_Act >= Conveyor.Y_Speed_Wrn)
                        {
                            Conveyor.Notifications.Add("[" + DateTime.Now + "] Y_Speed_Act
reached warning level.");
                            Console.WriteLine("Y_Speed_Act reached warning level.");
                        }
                    }
                    break;
                case SignalsGSCConveyor.Vacuum_Act:
                    if (Conveyor.Vacuum_Act >= Conveyor.Vacuum_Err_Max ||
Conveyor.Vacuum_Act<=Conveyor.Vacuum_Err_Min)
                    {
                        Conveyor.Errors.Add("[" + DateTime.Now + "] Vacuum_Act reached
error level.");
                        Console.WriteLine("Vacuum_Act reached error level.");
                        Stop();
                    }
                    else
                    {
                        signal = Conveyor.Vacuum_Act;
                        if (Conveyor.Vacuum_Act >= Conveyor.Vacuum_Wrn_Max ||
Conveyor.Vacuum_Act <= Conveyor.Vacuum_Wrn_Min)
                        {
```

Figure 27: Implementation of Get_Signal() operation contract in GSCConveyor

The services can be configured and exposed. The exposure can be achieved either programmatically or in a configuration file. We selected the second option because we can add/remove service endpoints, behaviours and bindings easily with no need to intervene to the code. Hosting and configuring are presented below:

```
<?xml version="1.0"?>
<configuration>
  <system.webServer>
    <modules runAllManagedModulesForAllRequests="true"/>
  </system.webServer>
  <system.serviceModel>
    <services>
      <service name="GSCConveyor_ServicesWrapper.GSCConveyor">
        <endpoint address="http://localhost:8000/GSCConveyor"
              binding="webHttpBinding"
              bindingConfiguration="webHttpBindingWithJsonP"
              contract="GSCConveyor_ServicesWrapper.IGSCConveyor"/>
      </service>
    </services>
    <behaviors>
      <endpointBehaviors>
        <behavior>
          <webHttp />
        </behavior>
      </endpointBehaviors>
    </behaviors>
    <bindings>
      <webHttpBinding>
        <binding name="webHttpBindingWithJsonP" crossDomainScriptAccessEnabled="true" />
      </webHttpBinding>
    </bindings>
  </system.serviceModel>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
  </startup>
</configuration>
```

Figure 28: Configure service endpoints

We expose the services defined in the service contracts and we specify the address from where the services can be consumed and since we want to expose the services through rest, we use the webHttpBinding. Furthermore we specify the contract that the service implements. The clients will be hosted in web applications hosted in apache rave and in order to deal with cross domain issues (e.g., we would have problem to call from a javascript function working in a domain, a different domain) we configure the webHttpBinding to allow cross domain scripts and we use jsonp to communicate with client. All these can be seen in the configuration file above.

A.2 Devices described as resources and exposed as services

The ontology (information schema) was developed and populated using the protégé tool, in OWL and using the turtle [12]syntax. Some "chunks" of this populated ontology is presented below.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix owl: <http://www.w3.org/2002/07/owl#> .

@prefix : <http://www.create.org/GSCLine.owl#> .

@prefix xml: <http://www.w3.org/XML/1998/namespace> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@base <http://www.create.org/GSCLine.owl> .
```

Figure 29: definition of prefixes

[12] http://en.wikipedia.org/wiki/Turtle_(syntax)

```
#################################################################
#    Object Properties
#################################################################

###  http://www.create.org/GSCLine.owl#ModuleOf

:ModuleOf rdf:type owl:ObjectProperty ;

rdfs:domain :Module ;

rdfs:range :Name ;

owl:inverseOf :isModule .


###  http://www.create.org/GSCLine.owl#cooperatesWith

:cooperatesWith rdf:type owl:ObjectProperty ,

owl:SymmetricProperty ;

rdfs:range :Module ;

rdfs:domain :Name .


###  http://www.create.org/GSCLine.owl#isDeviceOfType

:isDeviceOfType rdf:type owl:ObjectProperty ;

rdfs:range :DeviceType ;

rdfs:domain :Name .


###  http://www.create.org/GSCLine.owl#isModule

:isModule rdf:type owl:ObjectProperty ;

rdfs:range :Module ;

rdfs:domain :Name .
```

Figure 30: Definition of Object properties

```
#################################################################
#    Individuals
#################################################################

###  http://www.create.org/GSCLine.owl#24-PIN_Printer

:24-PIN_Printer rdf:type :DeviceType ,

owl:NamedIndividual .

###  http://www.create.org/GSCLine.owl#496-F

:496-F rdf:type :Name ,

owl:NamedIndividual ;

:isDeviceOfType :Friction_Sheet_Feeder ;

:suppliedBy :Kirk-Rudy .

###  http://www.create.org/GSCLine.owl#9-PIN_Printer

:9-PIN_Printer rdf:type :DeviceType ,

owl:NamedIndividual .

###  http://www.create.org/GSCLine.owl#ASGCO

:ASGCO rdf:type :Supplier ,

owl:NamedIndividual .

###  http://www.create.org/GSCLine.owl#Atlantic_Roll_102

:Atlantic_Roll_102 rdf:type :Name ,

owl:NamedIndividual ;

:suppliedBy :Lasermax ;

:isDeviceOfType :Roll_Winder_Feeder .
```

Figure 31: Individuals populating the ontology

This turtle file containing the information, which is the components of the production line described as resources, is loaded in a triple store using the dotNetRDF library. Operation and device matchmaking methods are developed and performed on the data in the repository and are exposed as web services.

```
namespace DeviceMatchmakingService
{
    [ServiceContract]
    public interface IMatchmaker
    {
        [OperationContract]
        [WebGet(UriTemplate = "getAllDevices", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> getAllDevices();

        [OperationContract]
        [WebGet(UriTemplate = "getAllSuppliers", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> getAllSuppliers();

        [OperationContract]
        [WebGet(UriTemplate = "getAllDeviceTypes", ResponseFormat = WebMessageFormat.Json,
BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> getAllDeviceTypes();

        [OperationContract]
        [WebGet(UriTemplate = "getCooperatingDeviceTypes?dev={device}", ResponseFormat =
WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> getCooperatingDeviceTypes(string device);

        [OperationContract]
        [WebGet(UriTemplate = "getTypeOfDevice?dev={device}", ResponseFormat =
WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> getTypeOfDevice(string device);

        [OperationContract]
        [WebGet(UriTemplate = "getSuppliersOfDevice?dev={device}", ResponseFormat =
WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> getSuppliersOfDevice(string device);

        [OperationContract]
        [WebGet(UriTemplate = "getDevicesOfType?type={type}", ResponseFormat =
WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> getDevicesOfType(string type);

        [OperationContract]
        [WebGet(UriTemplate = "getSuppliersOfType?type={type}", ResponseFormat =
WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> getSuppliersOfType(string type);

        [OperationContract]
        [WebGet(UriTemplate = "getCooperatingDeviceOfType?type={type}", ResponseFormat =
WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> getCooperatingDeviceOfType(string type);

        [OperationContract]
        [WebGet(UriTemplate = "getSupplierDevices?sup={supplier}", ResponseFormat =
WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped)]
        List<string> getSupplierDevices(string supplier);

        [OperationContract]
        [WebGet(UriTemplate = "getSupplierDeviceTypes?sup={supplier}", ResponseFormat =
WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped)]
```

Figure 32: Service contract of the device matchmaking service

The implementation and the configuration of the service was achieved in the same manner as demonstrated in the previous section of the Annex.

```
namespace DeviceMatchmakingService
{
    public class Matchmaker : IMatchmaker
    {
        // The base Uri of the ontology stored in the store where we
query.
        string ontoBaseUri = "http://www.create.org/GSCLine.owl#";

        // Retrieves devices. The results are used in the autocompletion.
        public List<string> getAllDevices()
        {
            List<string> allDevices = new List<string>();

            // Query to retireve all devices.
            SparqlQueryParser queryParser = new SparqlQueryParser();
            SparqlQuery query = queryParser.ParseFromString("SELECT
DISTINCT ?result " +
                                                            "WHERE
{GRAPH ?g { ?result <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.create.org/GSCLine.owl#Name>.} }");

            Object results = Program.store.ExecuteQuery(query);

            //Print out the Results
            if (results is SparqlResultSet)
            {
                // Print the results of the query.
                SparqlResultSet resultsSet = (SparqlResultSet)results;
                foreach (SparqlResult result in resultsSet)
                {
                    int i = result.ToString().IndexOf('#');
                    Console.WriteLine(result.ToString().Remove(0, i + 1))
                    try
                    {
                        allDevices.Add(result.ToString().Remove(0, i +
1).Replace('_', ' '));
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine("Exception: " + e.Message);
                    }
                }
            }
            return allDevices;
        }

        // Retrieves suppliers. The results are used for the
autocompletion.
        public List<string> getAllSuppliers()
        {
            List<string> allSuppliers = new List<string>();

            // Let's see a model with what types it cooperates
            SparqlQueryParser queryParser = new SparqlQueryParser();
            SparqlQuery query = queryParser.ParseFromString("SELECT
DISTINCT ?result " +
```

Figure 33: Implementation of the service contracts and operations necessary for the device matchmaking service.