Deliverable 2.4: The CREATE Method

CREATE

Creating Evolution Capable Co-operating Applications in Industrial Automation

Project number: ITEA 2 ip10020

Edited by:

Contributors:

Date: 30.05.2014

Document version no.: *5*

Revision History

Date	Version	Description	Author
05.05.2014	0	Template and contents of the deliverable	Anastasios Martidis (TIE)
21.05.2014	1	Initial MDH contributions	Ivan Tomasic (MDH)
23.05.2014	2	Initial TIE contributions	Anastasios Martidis (TIE)
27.05.2014	3	Final TIE contributions	Anastasios Martidis (TIE)
28.05.2014	4	Final MDH contributions	Ivan Tomasic and Peter Funk (MDH)
30.05.2014	5	Final deliverable	Ivan Tomasic (MDH)

Table of Contents

List of Figures

1. Why the CREATE method?

1.1 Automation in Industrial Applications

Automation processes have constantly affected (in fact improved) the various procedures in industries and their production lines. Industries from early on strived to automate their processes. The first automation attempts can be considered to take place in 1500s-1600s with the use of water power for metalworking. The modern version of industrial automation concerns integrated manufacturing systems, intelligent and sensor based machines, artificial neural networks, virtual environments and high-speed information systems. The main benefits of industrial automation include:

- *Cost and waste reduction*
- *Increased productivity*
- *Decreased production life-cycle and* consistency
- *Increased quality control*
- *Increased availability and reliability*
- *Increased safety.*

The application of automation however is dependent in the technological landscape; the available technologies that enable the automation of systems and processes. In the following section, the current technological enablers will be described.

1.2 Technological Enablers

During the last years the technological landscape is arguably moving towards the realization of the "Internet of Things". The Internet of Things (IoT) concerns a world where all physical objects, living beings and in general everything tangible in the natural environment is available for communication through the Internet and uniquely identified. In essence in IoT sensors, actuators and wireless technologies come together to make physical objects interconnected through information networks. Entities in the natural environment always produced vast amounts of data, but these data were not captured. Now, physical objects are getting seamlessly integrated and interconnected, so we can capture the generated data, store them, use them and make sense of them. This realization of the Internet of Things is enabled by technological changes, with the main of them being:

- IP addresses, the introduction of IPV6 which uses 128 bit addresses provides $3.4x10^{38}$ unique addresses to cover the needs (for now)
- Network bandwidth has increased dramatically with 4G motivated by the extensive of mobile devices such as smartphones and tablets
- Cloud data storage decreased the costs of storing data
- Emergence of standardized ultra-low power wireless technologies
- Integrated precision-analog capabilities.

Moreover, another transition is taking place, which concerns the type of information captured.

The information captured and considered of interest used to be mostly data generated and entered by humans such as ideas, business processes, banking and market transactions and continuing to social content. But with the highly increasing number of objects being interconnected and autonomously gathering data, the nature of the captured data is changing which will probably require different algorithms and types of interest. With the development of adequate artificial intelligence applications and Machine to Machine (M2M) communication, the objects can collect data autonomously as well as act autonomously based on the inputs they receive.

1.3 Underlying principles and objectives of the CREATE Method

The above technological enablers and the realization of the IoT, provide the necessary infrastructure to design and realize an architecture for industrial automation systems that are driven by the below objectives:

Control and Monitoring: Industrial systems and production lines especially during operation (production) require close monitoring and responsiveness to unexpected events. Additionally, human operators in traditional systems must be physically close to control machines (e.g., starting, stopping, initializing, etc.). Current production systems however, need virtualized and portable way of performing these operations in order to loosen the (physical) coupling between humans and machines as well as to be able to automate production procedures. The CREATE architecture took into consideration this drive in its design.

Interoperability and Flexibility: Traditional production lines when setup, are very demanding regarding future changes. Demanding in this context concerns the cost, time and re-programming to change components, insert new components in the system or perform even small adjustments that are very commonly required during everyday production processes. This approach is certainly not optimal and industrial environments that use so inflexible production lines are not able to be competitive in current market needs. The architecture of industrial automation systems requires abilities to integrate new components in a plug and play approach as well as operate with the new integrated components fast with minimized costs and re-programming efforts. The CREATE method was designed to cover this requirement.

Quality Assurance: Traditionally during production the evaluation of whether artifacts are meeting the defined specifications and are reaching the aimed quality were performed by human operators measuring and examining the hardware (artifacts). Due to the fact that this is an evaluation based on human assessment, it is an error prone and time consuming procedure. This process is desired to be faster and less error prone. This can be accomplished using sensors and scanners that virtualize hardware components in CAD models which then can be compared to the defined specifications and be evaluated on their quality. The technology for this is ready and is used in the CREATE architecture.

Production Processes Optimization: Traditionally any decisions about production processes,

planning and optimizing was based on human input. This came usually from individuals with extensive experience and domain specific knowledge but this meant a strong dependency of production lines and processes to specific people. Current technologies, and especially the advances in the domain of artificial intelligence applications, can and are improving these aspects and minimize such dependencies. The CREATE architecture capitalizes on artificial intelligence applications, to improve production processes for example using recommendation systems for reconfiguration of production lines, or for receiving adjustments suggestions for defective parts in an assembly line.

1.4 Intelligent systems in CREATE Method

For the improvement of existing industrial systems (as discussed in the previous section), the integration and communication between a number of research and technology areas is required. The most important among these are:

- Process control systems,
- Sensor development and fusion,
- Robotics,
- Fault diagnosis,
- Linear and non-linear optimization,
- Data mining, descriptive analysis and visualization,
- Machine intelligence.

These areas incorporate what is called intelligent industrial systems, which are characterized by their ability to learn from accumulated sensor data, and to handle the statistical uncertainty inherent to the data. The most important consequence of learning is that intelligent industrial systems are able to adapt to new operation conditions and/or requirements. It is often the case in industrial systems that mathematical models of process are not adequate (do not reproduce the process behavior precisely) or are too complex for real-life usage. The complexity can be especially problematic for real time application, in which mathematical models need to be evaluated quickly, i.e. in real time. On the other hand, the handling of uncertainty is particularly important in cases for which the mathematical models are not applicable. In these cases usually some stochastic models are employed, but we can also witness the usage of artificial neural networks and fuzzy models.

Incorporating intelligence in industrial systems can help to increase productivity, cut-off production costs, and to improve working conditions and safety in industrial environments [1]. To achieve this we can witness intensive development of control methods (and related models) for industrial systems and robots. Furthermore, for the detections and prevention of critical situations in industrial work-cells and production plants, fault detection and isolation methods are researched and developed. Additionally, optimization methods are developed and applied that aim at a more efficient operation of industrial installations, together with machine

intelligence methods aiming at reducing human intervention in industrial systems operation.

2. Proposed Architecture

The key pre-requirement for enabling industrial automation systems enhancement in the CREATE approach is the communication. Components that are part of the automated systems "exist" in different levels of the automation pyramid (Sensor, Device, Control and Enterprise level) as shown in Figure 1.

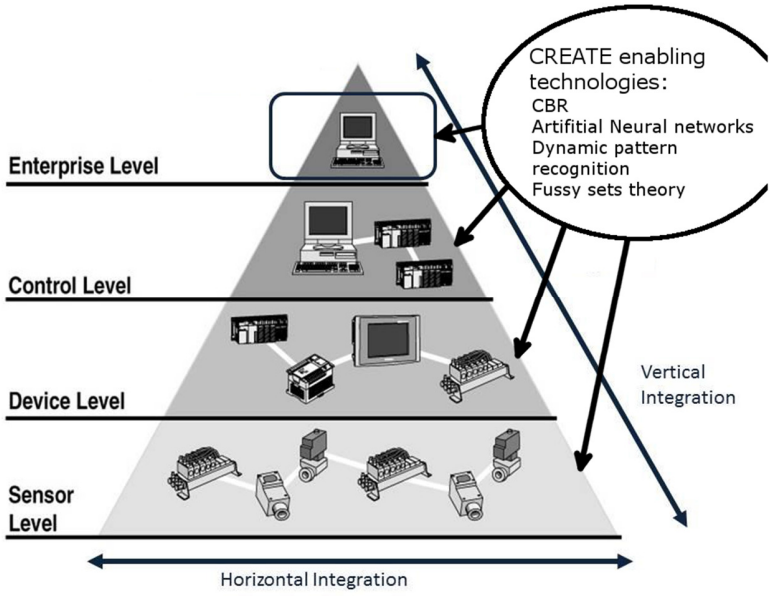

Figure 1. Automation pyramid and CREATE methodology.[1]

Communication has to take place in both dimensions of the automated production line ecosystem (vertically and horizontally). Systems that communicate both horizontally and vertically are highly flexible, automated and effective. Vertical integration however is fragmented, and each module operates as a standalone module, possibly having predefined, fixed interfaces to other modules of the same level of the automation pyramid, or with the modules in the neighboring levels. Even though the modules are usually closed proprietary systems obtained from different vendors, they have to be integrated and exposed together virtually, so that they can be controlled and monitored, and communicate the data generated during production. In this way the CREATE SoA methods and approaches, together with CREATE enabling technologies, can be applied on all the levels of the automation pyramid (Figure 1). With the use of CREATE method and technologies such as case-based reasoning and data mining technologies, important insights for

[1] Adapted from source: http://www.al-pcs.com/Products/images/Pyramid.jpg

optimizing the production processes can be made. Horizontal integration is necessary for modules to communicate with each other and to adjust their behavior according to information they receive from other modules with no need for human intervention.

The CREATE method and its defined architecture for vertical and horizontal communication as well as AI applications on the Enterprise Level, is based on five main building components which are going to be described briefly below:

Automation Software

The CREATE method and its architecture is heavily based on automation software from the lower levels such as embedded systems for control and communication of the devices to higher levels such as web services for communication and AI applications for knowledge extraction from data generated during production line. The automation software is important on many aspects; first it is required for devices to be able to communicate with each other and act based on the input they get from other devices (horizontal integration). Moreover, the devices must be able to get "instructions" from humans so that they control their actions and behaviour which is also facilitated by automation software (vertical integration). Finally, the huge amount data generated during production require AI applications to extract knowledge that will contribute to the optimization of production.

Physical Devices

The physical devices which in this context concern the hardware involved in the production lines are the basis of the systems which adopt the CREATE method. These components are exposed into the virtual world and communicate information during their real-life, real-time operation. All functionalities and applications developed in the CREATE approach are built on top of the physical devices and their associated automation software.

Knowledge bases

Systems that follow the CREATE method maintain a data repository for knowledge management. The information generated during production line and information related with production line on design time is communicated and stored in repositories and AI applications such as CBR and device matchmaking algorithms are applied for optimization of production processes, reduced times of production line reconfiguration and decision making support.

Communication platform

The CREATE method advices the use of a communication platform where devices, legacy systems and other repositories can communicate with each other. The platform has to be able to provide semantic interoperability and data transformation capabilities so that heterogeneous components in the system can still communicate. Communication is an important factor of the

CREATE method and is performed both vertically and horizontally across the systems that adopt the CREATE architecture.

Human Machine Interaction

The CREATE method proposes the presence of an easy to use, friendly GUI (CREATE portal) which is platform independent and highly portable so that users can access it from a variety of devices. The interaction with users has to focus on simplicity and intuitive communication using graphs for representation of production data communicated from the devices and interactive forms and buttons for humans to control, monitor and receive recommendations and decision support during production.

These building blocks are "put together" by the CREATE method to define an architecture which is illustrated in the figure below:

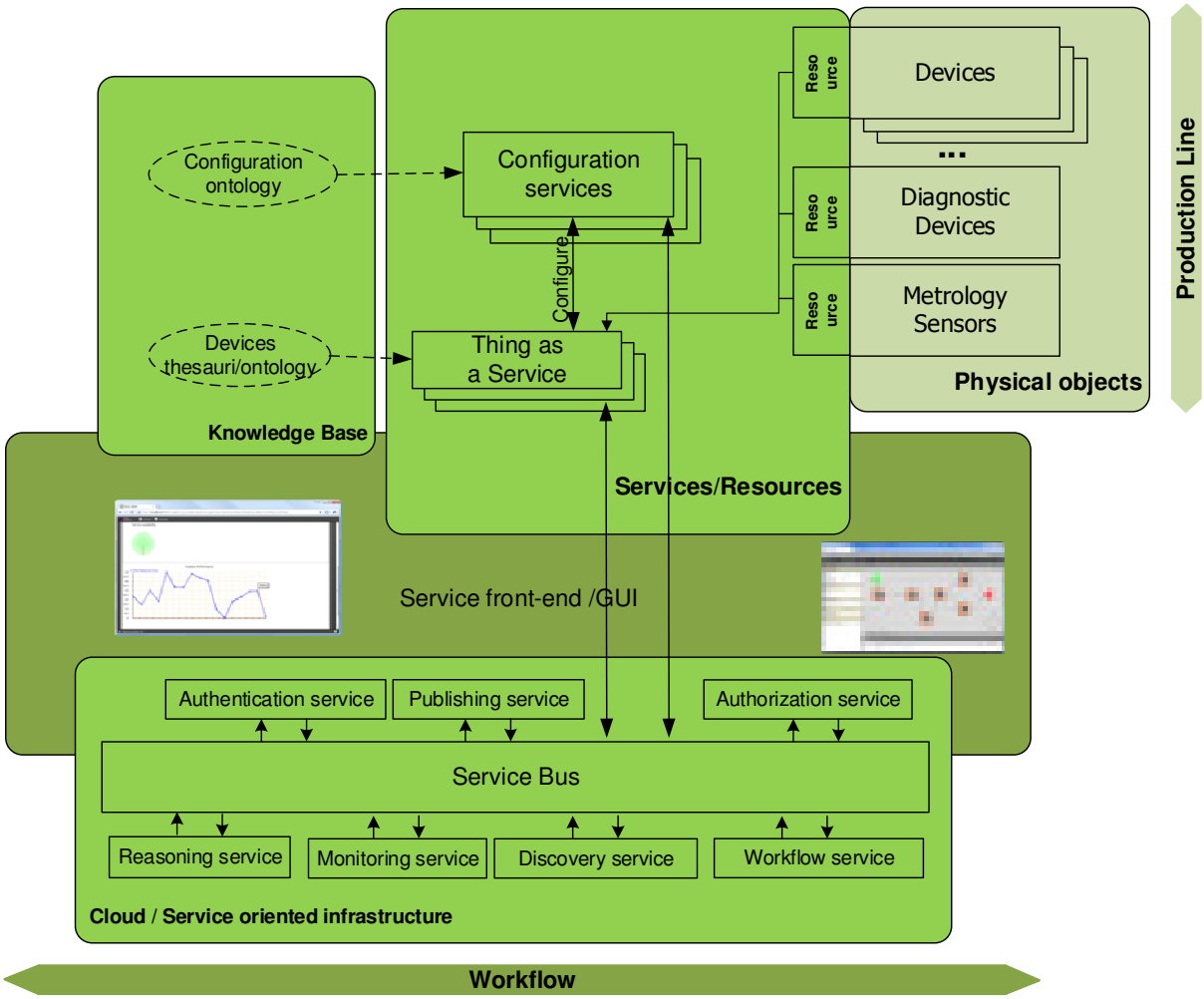

Figure 2: CREATE method and its defined architecture

3. Technologies Description

The CREATE project uses the most relevant State Of the Art (SoA) methods and modules in an innovative way with an aim of improving industrial automation and information systems. To be able to optimally explore the available SoA modules and methods, during the CREATE project a number of other technologies were recognized are incorporated in the CRETE methodology. These technologies are the driving force of the CREATE methodology and are listed and concisely presented here.

3.1 Artificial Neural networks

One of the most important characteristics of the intelligent industrial systems is that they are able to learn from accumulated sensor data. It is often desirable to form model of a system, based on the acquired data and leaned information. In cases when it is too complicated to form a mathematical model, or mathematical models are not precise enough, it is possible to use one of the "black box" modeling techniques one of which, and probably the most popular, is the artificial neural networks.

Artificial Neural Networks (ANNs) is a computational tool that imitates the interconnection of neurons in the nervous systems of living organisms. ANNs are network systems constructed from atomic components known as neurons. Neurons receive a number of input signals, apply a predefined function on the inputs (so called transfer function), and produces a resultant signal that is transmitted further to other neurons. Neurons in ANNs are organized in layers (Figure 3). In a so called feed-forward ANNs, if a neuron is in one of the intermediate layers (hidden layers), its output is transmitted to other neurons, but if a neuron is in the final level (output layer), then its output is an output from that ANN. Each layer represents a non-linear combination of non-linear functions from the previous layer. Modern industrial systems are regularly non-linear. This represents no problem for ANNs because it is proven that ANNs are universal function approximators (Kolmogorov theorem) [2]. Given the right number of neurons with continuous and differential transfer function, and one hidden layer it can approximate any continuous function. With two hidden layers it can approximate any computable function [2]. The downside of using ANNs is that network

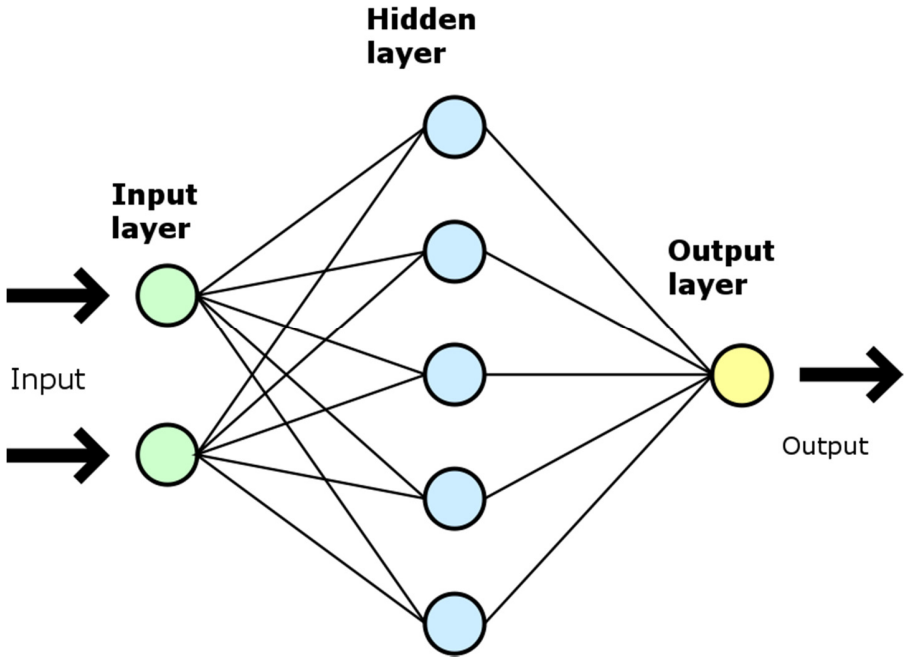

Figure 3. Feed-forward ANN – general structure (circles represent neurons)[2]

representation is black box: the properties of model cannot be analyzed, but there are a lot of systems where understanding of the system is not so important. The limitation of ANNs.is that there is no general prescription on how to construct an ANN for a specific problem and how to train it. Still ANNs are simple but powerful technique for process modeling, especially in situations when it is hard to find some other ways to model a process.

The process of using ANNs for modelling processes is presented in Figure 4. Model building is done by adjusting ANN parameters (i.e. the parameters of neuron transfer functions) by applying a learning algorithm. The learning typically happens during so called training phase. Once the network has been trained, it enters a production phase in which it produces results independently. Networks which are able to continuously learn even during production are known as dynamical ANNs.

[2] Adopted from http://commons.wikimedia.org/wiki/File:Neural_network.svg

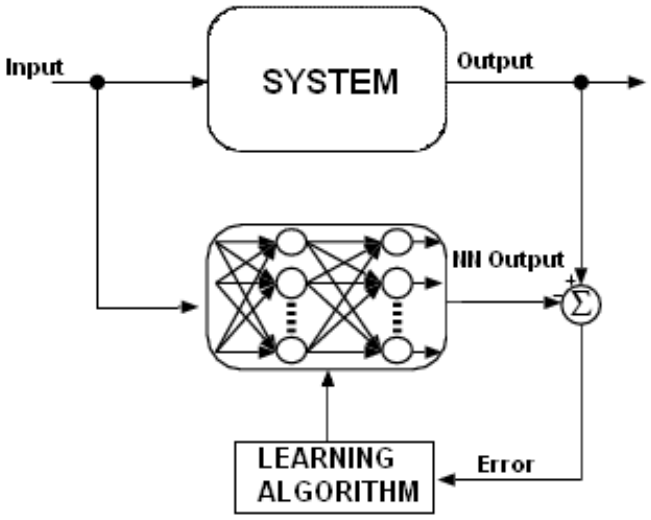

Figure 4. Neural network system modeling.

3.2 Case Based Reasoning (CBR)

A Case Based Reasoning (CBR) system is an artificial intelligence implementation in which the system solves a problem basing its output in the experience collected from solving similar problems in the past, inspired by the doctrine that similar situations lead to similar outputs. The prerequisite for the CBR systems is to have a knowledge database big enough to encompass a considerable amount of solutions, and therefore be able to solve all the problems that may arise on a given task.

CBR systems are based on two simple assumptions. The first is that similar problems have similar solutions; therefore a system that wants to solve a new problem will be in a much better starting point using the solution from a similar problem. The second assumption is that similar problems tend to happen many times; having a solution for them already computed is efficient and advantageous.

It is evident that for a concrete CBR system to function effectively, it must contain at least some previous knowledge of problems and their respective solutions. This collection of problems and solutions is called the case database. As the case database grows the CBR systems is said to be learning, which is one of the most important requirements for the intelligent industrial system (see chapter 1.). In addition to modeling knowledge by using the case based database, the CBR system can adopt to a specific problem, or to some new requirements, by adopting other CBR system parameters, like for example similarity metrics of vocabulary containers (see [3] for details).

In addition to providing a solutions and/or advices to correcting a specific industrial process, CBR can be used on upper levels of the automation pyramid (Figure 1). When used on the upper levels CBR takes as input, the outputs of the systems used for direct process control. This data can be used to build a case based database and therefore learn, or it can be used to build a model of a Model Based Reasoning system, that can be used in parallel to CBR, to verify and confirm the output from a CBR system [4].

On the highest, enterprise level, the main requirement for making correct decisions is the data accuracy [5]. Often the case is however that the required data is not completely available, or that it is not 100% reliable. In these cases the decisions are made more pragmatically. CBR can be helpful here to learn on the previous decisions and provide reliable suggestions for the future decision making.

3.3 Petri Net Models

A Petri net (also known as a place/transition net or P/T net) is one of several mathematical modeling languages that may be used for describing distributed systems. Petri nets are graphically represented by nodes of two types:

- Bars which represent transitions (i.e. events),
- Circles which represent places (i.e. conditions),

together with arrows (called arcs in Petri net terminology) which describe which places are pre-conditions and/or post-conditions for a specific transitions (see Figure 5 and Figure 6 for an example of a Petri net). Arcs run from a place to a transition or vice versa, never between places or between transitions. Places in a Petri net may contain a discrete number of tokens which are graphically usually presented with marks inside circles representing places. Current distribution of tokens over the places represents a configuration of the net called a marking. A transition of a Petri net may fire if it is enabled, i.e. if there are sufficient tokens in all of its input places. When the transition fires, it consumes the required input tokens, and creates tokens in its output places. The number of input tokens consumed is defined for each arc and usually written above or below each arc.

Figure 5 shows a simple Petri net: it contains two places and one transitions. The arch connect the elements and specify how many tokens are transferred from each place in a transition. For example in a transition always three tokens in total (2+1) are consumed from place p_1. Figure 6 shows the net from Figure 5 after a transition. It may be seen that three tokens have been consumed from place p_1 and transferred to transition t. From transition t only two of them are forwarded to place p_2.

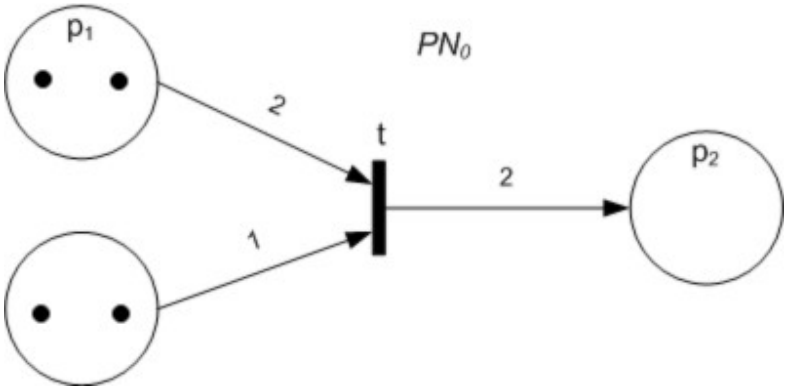

Figure 5. A Petri net in zero marking before a transition[3]

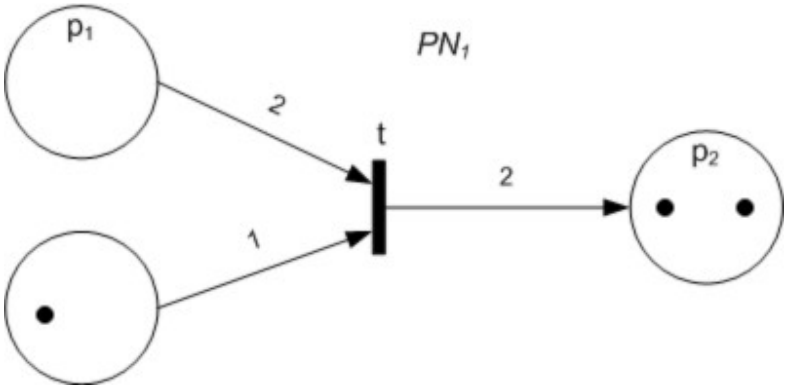

Figure 6. A Petri net from Figure 5 after a transition (marking 1)[4]

Petri nets offer a graphical notation for stepwise processes that include choice, iteration, and concurrent execution. In comparison to other similar approaches like UML activity diagrams (see [6] for details), Petri nets have an exact mathematical formalization, with a well-developed mathematical theory for process analysis. This makes them a very robust and reliable tool that can be used in intelligent industrial systems.

If there isn't an execution policy defined for a specific Petri net, its execution is nondeterministic. This is because all the transitions are enabled all the time and any one of them can fire given that enough tokens are provided for it. This makes the firing nondeterministic, and multiple tokens may be present anywhere in the net. Petri nets are therefore well suited for modeling the concurrent behavior of distributed systems.

[3] Adopted from http://en.wikipedia.org/wiki/Petri_net
[4] 'Adopted from http://en.wikipedia.org/wiki/Petri_net

3.4 Fussy sets theory

As it was stated in chapter 1, management of uncertainty is one of the most important characteristics of intelligent industrial systems. Fuzzy set theory (FST) is one of the most popular approaches to modeling uncertainty in many application areas, and is therefore widely used and applied in industrial systems.

Fuzzy sets are by definition sets whose elements have degrees of membership [7]. In classical set theory, the membership of elements in a set is assessed in binary terms: an element either belongs or does not belong to the set. On the other hand, fuzzy set theory permits the gradual assessment of the membership of elements in a set; this is described with the aid of a membership function valued in the real unit interval [0, 1]. Fuzzy sets generalize classical sets, since the indicator functions of classical sets are special cases of the membership functions of fuzzy sets, if the latter only take values 0 or 1.

Decision making is happening all the time on different levels of automation. It can be done by applying objective rules and models but when these approaches don't provide a satisfactory results, a more pragmatic approach toward decision-making can be applied that can also take into account human subjectivity. The uncertainty of human behavior can be modeled and simulated by fuzzy decision-making. Fuzzy systems are suitable for uncertain or approximate reasoning, especially for the systems for which it is difficult or even impossible to develop mathematical models [8]. Fuzzy logic based on fuzzy sets, enables decision-making with incomplete or uncertain information or with data that is not precise but only estimated. This is because fuzzy set theory naturally incorporates imprecision, uncertainty and subjectivity into decision model formulation, and to solution process obtained by the model simulation. For all its characteristics fuzzy set theory is a valuable tool to assist decision making and process improvements on different levels of automation pyramid when the available data possesses intrinsic uncertainty, whether the data has been obtained directly from sensors, or it is produced by other element of the control system.

3.5 Dynamic Pattern Recognition

Pattern recognition is a branch of artificial intelligence that focuses on the recognition of patterns and regularities in data. The patterns are either learned from labeled data (supervised learning), or when no labeled data is available, other algorithms can be used to discover previously unknown patterns (unsupervised learning). The most important characteristic of pattern recognition algorithms is that they provide a more or less reliable recognition for all possible inputs, or perform most likely matching of the inputs to the predefined patterns (or classes of patterns). This characteristics separates pattern recognition from pattern matching algorithms, which try to exactly match the inputs to pre-existing patterns or pattern classes. More importantly this characteristic of performing most likely matching makes patter recognition more

suitable for handling data uncertainty (see chapter 1).

One of the applications of pattern recognition methods is in diagnosing and monitoring of systems [9]. By identifying patters in the available data a conclusion (diagnosis) about the system state is derived. The patterns, describing the system functioning, can be static or dynamic. A static pattern is represented by a point in a feature space whereas a dynamic pattern is represented by a multidimensional trajectory in a feature space that incorporates also the time dimension. Pattern classes can also be static or dynamic. Static classes are represented by restricted areas formed by similar static patterns. If the data is non-stationary classes become dynamic, i.e. they change in time. This requires an adaptive classifier that assigns pattern to classes with a mechanism that adjusts over the time.

The data non-stationarity here means the underlying data probability distribution is changing over time. This can be caused by a change in operating conditions, by a fault, or by significant changes in the monitored process dynamics.

3.6 Service Oriented Architecture (SOA)

Businesses and organizations change and evolve trying to respond in their environment and adapt and work with newer, better and more efficient technologies. Existing systems however, cannot simply be replaced. Service Oriented Architecture (SOA) provides to existing systems the flexibility and agility to respond to a business environment that is always changing. With SOA the capabilities of systems are exposed as services and only the service provider is concerned with the implementation. Furthermore the services are loosely coupled through a standard interface, promoting reuse and offering possibilities of combining services into new composite applications. Due to the fact that existing assets are reused efficiency is increased and application development costs are reduced, as well as enabling return of investment (ROA). With SOA businesses become more agile responding to market and competitive dynamics. SOA is based on the exchange of messages between services and for that reason the best infrastructure for allowing communication of services in a SOA approach would involve an ESB (see the next section for a description of Enterprise Service Bus (ESB)).

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. SOA facilitates the development of simpler systems, reducing the complexity compared to other approaches of integration such as on a solution-to-solution basis. Moreover, the maintenance

costs are lowered while the architectural flexibility is enhanced by using other services just by their interface. The integration costs of SOA applications are also reduced because of web service standards for message exchange (e.g., SOAP) and service descriptions (e.g., WSDL) and service discoverability (e.g., UDDI).

3.7 Enterprise Service Bus (ESB)

An Enterprise Service Bus (ESB) is a flexible connectivity infrastructure for integrating applications and services. ESBs serve as the backbone for connecting and integrating applications and services of businesses. ESBs improve the agility of businesses making them more adaptive because are designed to accommodate changes easily and with low costs. ESB connects various partners and makes it easy to add new ones. Moreover an ESB can improve the ability of efficient execution for a business and contribute to meet its goals via business rules that can be enforced across the breadth of the enterprise. ESBs can expand business intelligence, since as the messaging facilitator has full view of business activity and business information can be accessed analysed in real time. ESBs act as a single point of access for a variety of clients and services that are "attached" to the services bus, coordinate and manage distributed transactions of cooperating services and assure the security of the transactions. ESBs also act as a gateway to the world for services, devices and systems. ESBs are providing the infrastructure on top beneficiary and innovative technologies/architectures such as SOA.

3.8 Web Services

Web Services[10] concern software systems that are designed to support machine-to-machine interaction over a network. They are based on the technologies used for web navigation and communication such as HTTP, XML and so on. Their interfaces are described in a machine-understandable format. This allows the dynamic adaptation to an interface provided and using it to request the Web service to perform operations. Web services are based on these standardized formats for message formatting, processing and data communication:

3.8.1 SOAP

The Simple Object Access Protocol [11] concerns a protocol that specifies rules for exchanging structured data across the network. It uses XML to prescribe 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.

3.8.2 REST

Representational State Transfer (REST) [12], 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.

3.8.3 WSDL

The Web Services Description Language [13] is an XML format that is used to describe in detail properties of an exposed web services. The main properties that can be described are the endpoint 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.

3.8.4 WADL

The Web Application Description Language [14], is an XML format that as WSDL is used to describe services. WADL is also machine-understandable. The main difference with WSDL is that WADL is fully REST compliant. In comparison to WSDL, WADL is much less used in real applications.

3.8.5 UDDI

The Universal Description, Discovery and Integration [15], was conceived as a XML based standard for companies to publicly provide services. It can provide information related to the company-provider, information about services endpoints, 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.

3.8.6 JSON

JavaScript Object Notation (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.

4. Proof of concept in the CREATE project use cases

4.1 Use case description

At this stage of the CREATE project the CREATE Method has been implemented in real life scenarios and applications. CREATE partners provided proof of concept through the use of prototypes in 3 different use cases, namely Flexible Material Flow, Industrial Metrology and Monitoring and Quality Control.

The *Flexible Material Flow* use case showed how devices, sensors and other components of a production line can be exposed to the virtual world, exposed as services and be controlled and monitored through any computing device equipped with a web browser. Moreover it showed the use of artificial intelligence applications for reconfiguring production lines based on parameters such as their physical dimensions, software, communication protocols, etc.

The *Industrial Metrology* use case showed how the CREATE Method adds value in production processes by transferring real life objects into the virtual world using metrology components which scan and provides 3D representations of hardware components. This virtualization of the hardware components (tangible objects) is then used for comparing the virtual model of the scanned hardware to the specifications of the production. This verification of the quality of the production processes can happen multiple times in the production life-cycle (meaning, in each step of the production till the final product) ensuring that any possible flaws can be detected and be fixed in early stages saving costs, money and increasing efficiency.

The *Monitoring and Quality Control* use case concerns the repairing part of identified flaws in hardware components in the production lifecycle. The Monitoring and Quality Control use case is based on artificial intelligence applications and specifically Case Based Reasoning (CBR). Operators in the assembly line are evaluating hardware components and for those that are considered flawed, the operator has to make adjustments in order to reach specifications defined for the part. The CBR system acts as a recommendation engine that proposes adjustments to repair the defective hardware.

As a reader can deduce from the above descriptions, the 3 selected use cases complement each

other and cover the complete life cycle of a production, from setting up and controlling the production line to evaluating the quality of the hardware artifacts to repairing them – all enhanced by the use virtualization of the processes and control from computing devices with high portability opportunities. The prototypes for these use cases have already been documented in detail (D4.1 Evaluated Demonstrators deliverable) as well as presented in the yearly review (October 2013). The final year of the CREATE project is focused on a cross domain demonstrator of the above use cases.

The Cross Domain Demonstrator (CDD) concerns the most important stakeholder of the CREATE project's results, the Volvo Car Components. Here the Flexible Material Flow prototype and the Monitoring and Quality Control prototype come together to enhance the assembly line in the Volvo manufacturing cell (as documented in D6.1 Cross Domain Demonstrator Design and Specification deliverable). These components, due to the flexible and interoperable nature of the CREATE method, come together easily and compose a new system which is presenting the CREATE architecture and method benefits.

4.2 Cross Domain Demonstrator Prototype Description

The prototype for the Cross Domain Demonstrator (CDD) is realized for a Volvo manufacturing cell that produces the gore part of a car. The advances in the gore part assembly line are focused on correcting positioning of input parts. During assembly process, adjustments are necessary to compensate for the variation of the ingoing parts in order to reach the specification demands on the final sub assembly, the gore.

In the CDD prototype the measurements of the gore of the car (also named the crossmember,) together with other the parts and entities involved in the production, are "transferred" in the virtual world of CREATE. Human operators and technicians through the CREATE portal - a user friendly GUI - receive decision support for adjustments necessary to apply to the assembly process, if a part has one or more measurements out of the acceptable limits. The decision support is achieved with use of a CBR system, which identifies similar cases and the adjustments they received to be in the specification limits. New cases from the manufacturing cell are also added to the case library of the CBR system, to improve its performance in future suggestions.

This workflow and the communication of the various components is achieved through an ESB (specifically TIE SmartBridge) which acts as the backbone that integrates the CREATE portal and the CBR system as well as the sensors and diagnostic devices from inside the Volvo manufacturing cell.

4.3 Technologies used

In this chapter we will briefly present the most important technologies currently used for building the CREATE demonstrator prototype.

4.3.1 *SQL Server Relation Database Management system*

For the CBR database the SQL Relation Database Management System (RDBMS) has been chosen. Its primary function is to store and retrieve data associated with cases of the CBR system, and to serve the request coming from other parts of the prototype crossdomain demonstrator.

There is a number of different editions of SQL Server aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. Since the ultimate goal for the crossdomain demonstrator is to be hosted inside a Cloud, the SQL Server Azure addition was chosen, since it is the cloud-based version of SQL Server, presented as software as a service on Azure Services Platform (see section 4.3.3).

4.3.2 *Windows Communication Foundation*

For building and hosting CREATE prototype demonstrator server side logic, the Window Communication Foundation (WCF) technology has been chosen. WCF is a runtime and a set of application programming interface (APIs) in the .NET Framework, for building connected, service-oriented applications (SOA). The ".NET" framework is Microsoft and possibly world's most widely used software development framework.

WCF is designed using service-oriented architecture (SOA) principles to support distributed computing. WCF services typically provide a so called WSDL interface described by the "Web Services Description Language" that any WCF client can use to consume the service, regardless of which platform the service is hosted on. WCF implements many advanced Web Services (WS) standards such as: WS-Addressing, WS-ReliableMessaging, WS-Security, RSS Syndication Services, WS-Discovery. WCF services also support routing and Representational State Transfer REST services development (see [16] for details).

WCF includes the following set of features:

- Service Orientation
- Interoperability
- Multiple Message Patterns
- Service Metadata
- Data Contracts
- Security
- Multiple Transports and Encodings
- Reliable and Queued Messages
- Durable Messages
- Transactions
- AJAX and REST Support
- Extensibility

4.3.3 Azure Cloud Platform

Microsoft Azure is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services. It provides both Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) services, and supports many different development languages, tools and frameworks, including both Microsoft-specific and third-party software and systems. Azure is an open and flexible cloud platform that enables building, deploying and managing applications across a global network of datacenters. It has a built in network load balancing and provides resiliency to hardware failure. Additionally it is a very reliable platform that enables hosting of highly available applications which makes it a very good option for hosting CREATE crossdomain demonstrator applications.

Currently Azure Cloud Platform has data centers in USA, two in each of Europe, Asia, Japan, and one in Brazil. The global network of datacenters can be used by Azure to maintain the availability in a cost-effective manner, but also to achieve data redundancy and security by storing the data on different locations. Azure can be used to scale out available on-premises resources but Azure services can also be easily integrated with existing on- premises infrastructure.

4.3.4 Transport Layer Security

For securing the communications between the client and the server side of the CREATE demonstrator prototype, the Transport Layer Security (TLS) is used. TLS is a cryptographic protocol used for securing Internet communication. It uses use X.509 certificates and hence asymmetric cryptography (involving public and private keys) to assure the authenticity in communication, and to exchange a symmetric key. This symmetric key is then used to encrypt data flowing between the parties. This allows for data confidentiality and integrity.

As a consequence of using X.509 certificates, certificate authorities and a public key infrastructure are in general necessary to verify the relation between a certificate and its owner, as well as to generate, sign, and administer the validity of certificates. For the CREATE prototype however the self-signed certificate is used. A self-signed certificate is an identity certificate that is signed by the same entity whose identity it certifies. In technical terms a self-signed certificate is one signed with its own private key. This means that there is no higher authority signing the certificate and therefore guaranteeing for the certificate owner authenticity. Since the server side of the prototype system is to be used by only one client application, which trust the server side self-signed certificate, there is no need for obtaining a certificate signed by a certification authority.

Only recently (4 Apr 2014) the famous TLS Heartbleed security bug was publicly disclosed. At that time about 17 percent (around half a million) of the Internet's secure web servers, certified by trusted authorities, were believed to be vulnerable to the attack due to the Heartbleed bug. The

patch for TLS was quickly developed. TLS it is still the most important standard for securing Internet communication.

4.3.5 HTML5

The fifth revision of the HTML mark-up language that is used for presenting content in the World Wide Web. HTML5 adds many new syntactic features that optimize websites‘ content for multimedia, graphics, semantics, mathematical formulas and more.

4.3.6 CSS3

The third revision of CSS style sheet language that is used for formatting the design of documents written in mark-up languages. CSS allows the separation of the document‘s content and the document‘s presentation including elements as the layout, fonts and colors.

4.3.7 JavaScript

A dynamic computer language which is used extensively in applications for client-side programming. JavaScript is a multi-paradigm language supporting object-oriented, imperative and functional programming.

4.3.8 AngularJS

An open-source JavaScript framework developed by Google. It enhances single page applications and follows the Model – View – Controller design pattern, making development and testing easier.

4.4 Cross Domain Demonstrator Prototype Evaluation

To test the prototype system (described in section 4.2) a subset of solved cases is selected and fed to the CBR part of the prototype system. These cases are used to simulate new cases but since they are solved they come with the technicians’ adjustments that successfully corrected the process. These adjustments are compared to the ones suggested by the CBR system. Note that for the testing purposes the cases selected are complete so that their adjustments can be compared to the system suggested adjustments. In the operational scenario only problem cases (without known adjustments) will be presented to the system.

The precise testing procedure is as follows:

1. Fourteen cases are chosen randomly from the CBR case library containing in total 127 cases.
2. For each of the fourteen cases:
    a. The current case is excluded from the CBR case library,
    b. The adjustment done for the current case is extracted and stored,
    c. Similarity is calculated between the current case and all the other cases in the database,
    d. The cases are sorted by the calculated similarity,

e. Adjustment for the most similar case is extracted and stored ,
f. The current case is returned to the CBR case library so that it can be compared to the other cases,

3. The known adjustments (step b) are aligned to the adjustments done in the most similar cases (step d) and exported (saved).

This procedure results with a table comparing the done adjustments with the suggested adjustments (Table 1).

Table 1 Comparison between the adjustments done and the suggested adjustments.*

Case num.	Known adjustment				Adjustment for the most similar case			Similarity between current and the most similar case		
	Pin	Direc-tion	Initial shim	Change in shim	Pin	Direc-tion	Initial shim	Change in shim	RMSD	NRMSD (%)
1	19	X	5,3	1,5	19	X	5,3	2,1	0.31	91.15
2	19	X	5,5	-1,3	19	X	5,5	-0,9	0.17	95.03
3	22	X	5	-1,2	22	X	5	-1,6	0.14	94.65
4	**25**	**X**	6	0,8	**22**	**X**	5	-1,6	0.26	90.87
5	**25**	**X**	6	-1,4	**19**	**X**	5,5	0,5	0.08	85.88
6	25	Y	4,2	-1	25	X	6	-0,4	0.06	97.50
7	27	X	3,8	0,4	27	X	3,8	-0,8	0.12	95.27
8	**27**	**X**	3,8	-0,4	**39**	**Z**	5,6	-0,4	0.10	95.80
9	27	Y	4	0,8	27	Y	4	-0,4	0.23	92.47
10	27	Y	4	-0,8	27	Y	4	-1,2	0.46	90.22
11	**37**	**Z**	6,2	-1,4	**25**	**X**	6	0,4	0.13	95.28
12	37	Z	4,8	-1,4	37	Z	4,8	-2	0.22	96.18
13	39	Z	5,4	1,4	39	Z	5,6	0,4	0.08	97.17
14	39	Z	5,6	-1,4	39	Z	5,6	-2	0.15	95.36

*The shaded rows indicate the cases in which the suggested adjustment pin and direction are not the same as the done adjustments (the Pin names and directions that differ are bolded). Note that pins are adjusted in directions of Cartesian coordinate system (X, Y, Z). RMSD is the root-mean-square-deviation and NRMSD is the normalised RMSD.

It can be noticed in Table 1 that there are four cases (out of fourteen) for which the suggested adjustments are not on the same pins as the adjustments done. Still the similarity between each of these four cases and their suggested cases are all higher than 85%. Even though this result may be unexpected, or seem incorrect, it is quite possible that the suggested adjustments are suitable for these cases, as they can be better than the expected ones, i.e. the ones that were done by technicians.

In all the other cases the pins and their associated directions coincide in the suggested adjustments and the adjustments done. The comparison between the changes in shim values show that in general the suggested shim changes are quite close to the changes done. As the number of the stored cases will increase (in the production system) it can be expected that this precision in the suggestions will increase as well.

For the needs of the presented test only the most similar cases (to the current case) have been used. In the envisaged operation system the technician will be presented with a number of the most similar cases (e.g. five) together with their adjustments, from which he can choose or adopt them for the current case.

5. Motivation and Guidelines for Industrial Automation

The results of the CREATE project are already providing value for the defined use cases for which partners developed prototypes and applied the CREATE method. However, partners in the CREATE project are striving to bring innovation to the complete industrial automation domain. The use cases and success stories are the proof of concept which CREATE partners aspire to motivate industry to adopt the CREATE method. With this aspiration, CREATE partners consider valuable to detail which business objectives of industries can be reached by the adoption of the CREATE method:

- Increased control and monitoring capability over hardware devices of their production lines in a high portable manner
- High quality assurance competences by use of CAD models to provide virtualized version of hardware parts to compare them with the desired specifications
- Decrease time, costs, errors and reprogramming efforts during (re)configurations of production lines
- Improve production processes and decrease individual expertise dependency for the production with the use of artificial intelligence applications that act as recommendation systems

On the technological aspect CREATE targets industries that need to:

- Integrate new components in a plug and play approach with minimized changes to their systems, making their systems highly interoperable and flexible
- Maintain their legacy systems and keep them on the operations while adopting and follow newer and more effective architectures and technologies
- Improve the human machine interactions between operators and their systems to improve their efficiency and performance based on human-centered design.

The CREATE project has showed through its success stories in the Industrial Metrology, Flexible Material Flow and Monitoring and Quality Control that its approach (a.k.a. CREATE Method) can lead them and bring them closer to these objectives.

Moreover, through CREATE project the current cutting edge technologies to implement the CREATE Method have been implemented, tested and evaluated in the use case prototypes as well as documented in deliverables, this one included (see sections 2 & 3). The potential adopters of the CREATE method are free to use the described technologies and tools, however they can (and actually are encouraged to) try, select and use the technologies and tools of their choosing to implement the CREATE Method (or architecture). One of the strongest points of CREATE Method is that it defines design principles, architectural designs and design patterns without being coupled/dependent in specific technologies, tools or frameworks.

6. Conclusion

This deliverable is the final deliverable in a series of four deliverables of work-package two in the CREATE project. The series starts with the first version of the CREATE architecture, then follows a deliverable describing the current SoA in the CREATE related industrial, technological and research fields. After that follows the deliverable describing second version of the CREATE architecture.

In this final deliverable of the architectural work-package we have presented the final and completely formed CREATE architecture and methodology. Firstly the CREATE method enablers and underlying principals and objectives were introduced together with the intelligent systems and methods used and embodied in the CREATE methodology. Then, in chapter 2, the CREATE method and its definitive architecture was described. The CREATE architecture is based on five main building components: Automation Software, Physical Devices, Communication platform, Human Machine Interaction.

The intelligence, as one of the most prominent characteristics of advanced industrial systems, is manifested by their ability to learn from accumulated sensor data, and to handle the statistical uncertainty inherent to the data. The CREATE method promotes and stimulates intelligence for existing industrial systems by incorporating a number of SoA technologies and research fields in the CRETE methodology: artificial neural networks, case based reasoning, fussy set theory, Petri net models etc. These technologies, described in chapter 3, are the driving force of the CREATE methodology.

At this stage of the CREATE project, the CREATE method has been implemented in real life scenarios and applications. CREATE partners provided proof of concept through the use of prototypes in three different use cases, namely Flexible Material Flow, Industrial Metrology and Monitoring and Quality Control. The knowledge and experience gained from building these use cases is planned to be incorporated in the cross-domain demonstrator (CDD) until the end of the project. The CDD is currently in his prototype version. This deliverable describes the prototype CDD technologies used and presents one evaluation of the prototype system. The evaluation confirms the usefulness of the CRETATE architecture and demonstrates the strengths of the CERATE methodology.

The deliverable ends with CREATE developed and introduced methods and guidelines for industrial automation, with which the CREATE method provides added value for the parties involved in the development of the three project prototypes (Flexible Material Flow, Industrial Metrology, and Monitoring and Quality Control), but also strives to bring innovation to the complete industrial automation domain.

7. Bibliography

[1] G. Rigatos, *Intelligent Industrial Systems: Modeling, Automation and Adaptive Behavior*, first ed.: Information Science Reference, 2010.

[2] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, p.^pp. 327: Addison-Wesley, 1991.

[3] M. M. Richter, and R. Weber, "Basic CBR Elements," *Case-Based Reasoning: A Textbook*, pp. 17-41: Springer, 2013.

[4] N. Xiong, P. Funk, and O. T., "Case Based Reasoning Supports Fault Diagnosis Using Sensor Information," in The 2nd International Workshop and Congress on eMaintenance, Luleå, 2012.

[5] P. Johnson, R. Lagerström, M. Ekstedt, and M. Österlind, "Data accuracy," *IT management with Enterprise Architecture* 5, 2014.

[6] t. f. e. Wikipedia. "Activity diagram," May, 2014; http://en.wikipedia.org/wiki/Activity_diagram.

[7] t. f. e. Wikipedia. "Fuzzy set," 2014; http://en.wikipedia.org/wiki/Fuzzy_set.

[8] C. Kahraman, M. Gülbay, and Ö. Kabak, "Applications of Fuzzy Sets in Industrial Engineering: A Topical Classification," *Fuzzy Applications in Industrial Engineering*, Studies in Fuzziness and Soft Computing C. Kahraman, ed., pp. 1-55: Springer Berlin Heidelberg, 2006.

[9] M. S. M. Laurent Hartert, Patrice Billaudel, "Monitoring of Non Stationary Systems Using Dynamic Pattern Recognition," *Intelligent Industrial Systems: Modeling, Automation and Adaptive Behavior*, G. Rigatos, ed., p. 36: Information Science Reference, 2010.

[10] "Web Service Definition," *http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice*, 2004.

[11] "SOAP 1.2. ," *www.w3.org/TR/2001/WDsoap12-part0-20011217/*, 2001.

[12] R. T. Fielding, and R. N. Taylor, "Principled Design of the Modern Web Architecture," *ACM Transactions on Internet Technology,* vol. 2, no. 2, pp. 115-150, 2002.

[13] "WSDL," *http://www.w3.org/TR/2007/RECwsdl20-adjuncts-20070626/*, 2007.

[14] "WADL," *http://www.w3.org/Submission/wadl/*, 2009.

[15] "UDDI," *http://uddi.org/pubs/uddi_v3.htm*, 2004.

[16] t. f. e. Wikipedia. "Representational state transfer," 2014; Representational state transfer.