ITEA 2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

CREATE

Deliverable 6.2: Evaluated Cross Domain Demonstrator

CREATE

Creating Evolution Capable Co-operating Applications in Industrial Automation

Project number: ITEA 2 ip10020

Edited by:

Contributors: Anastasios Martidis, Kostas Barounis, Ivan Tomasic, Peter Funk

Date: 01.12.2014

Document version no.: *0.4*

Revision History

Date	Version	Description	Author
24.10.2014	0.0	The template and contents for the deliverable	Anastasios Martidis (TIE)
13.11.2014	0.1	First version of NL contributions (sections 1,3,4)	Anastasios Martidis (TIE)
21.11.2014	0.2	First MDH contributions (sections 2, 5, 6, 7, 8, 9)	Ivan Tomasic (MDH)
28.11.2014	0.3	Final version of NL contributions (sections 5,6,9)	Anastasios Martidis (TIE) Kostas Barounis (TIE)
01.12.2014	0.4	Final polishing	Ivan Tomasic (MDH)

TABLE OF CONTENTS

Figures:

Tables:

1. Introduction – The CREATE Story

The CREATE project aimed to develop an innovative, fully decentralised software architecture for industrial automation systems which is based on modular and autonomously cooperating components called Smart Neighbourhood Modules (SNMs). These SNMs are composed of mechanical parts, and associated automation software, which are enhanced by artificial intelligence applications and visualization platforms for the inclusion of human workers. The innovations proposed by CREATE for the industrial automation, promised to result into increased control and monitoring, faster reconfigurations, improved quality assurance processes facilitated by decision support systems and increased interoperability. The CREATE project already developed, documented and presented prototypes that provide proof of concept (PoC) for the CREATE approach and its underlying architecture, in three national use cases in the domains of Flexible Material Flow, Industrial Metrology and Monitoring and Quality Assurance.

At the time of this deliverable, the CREATE project is close to its final review and completion. The involved partners have already provided multiple research and development contributions and PoCs for revolutionizing the established industrial automation procedures. Starting from the description of the use cases, the gathered functional and non-functional requirements and user stories, partners defined the CREATE architecture as has been documented in (two iterations in) deliverables D2.1 and D2.3. The CREATE partners, using the CREATE architecture as a blueprint, performed State-of-the-Art research to identify the best technological options for realizing the CREATE architecture, and for developing the demonstrators/prototypes for each national use case. The national demonstrators were as mentioned before developed, documented and presented in the Y2 review of the project.

During the process of researching and developing demonstrators for the CREATE national use cases, partners already produced significant results. The scientific research done under CREATE project was presented in a number of conferences and published in form of five conference papers. The research also resulted with four scientific journal publications so far. There are ongoing activities for future scientific journal publications which are to come after the project end date, and after all the patenting and other possibilities of protecting and exploring the project's inventions will have been thoroughly exhausted. On a high level overview of the results achieved in CREATE, partners have already developed 3 evaluated demonstrators, 12 new products (or new versions of products), published 9 academic papers, defined 4 new standards and provided 6 public deliverables to contribute in the innovation and enhancement of industrial automation.

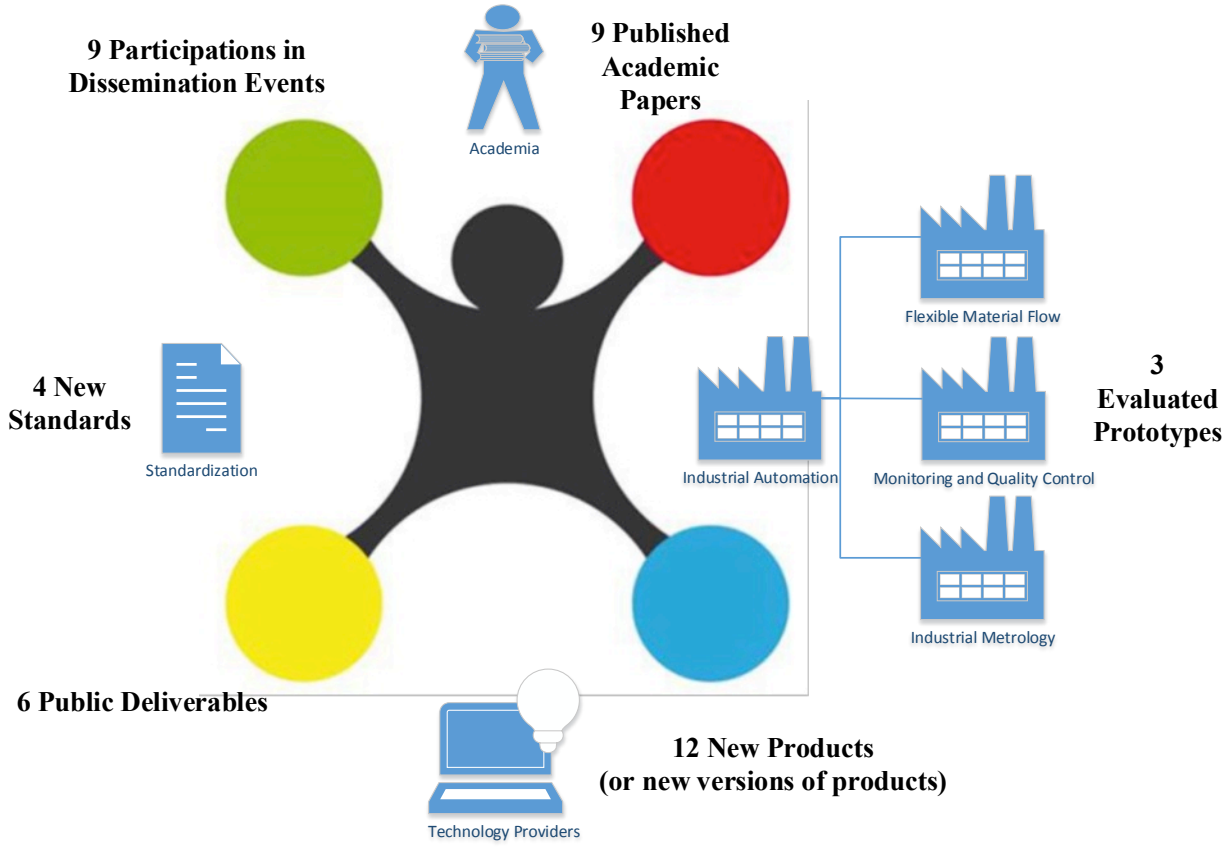

Figure 1: High level overview of the CREATE key achievements

The CREATE approach provides high interoperability and facilitates the integration of new components in a plug and play approach. In the CREATE methodology this is to come from innovation, mainly in decision support making, for the purpose of improving monitoring and quality control. To showcase this, the last year of CREATE project's life, partners worked focused on the definition and development of the Cross Domain Demonstrator (CDD) which is based on the Volvo C.C. use case and is composed of components from both the Dutch and the Swedish consortia (the ones still participating in the last year of the project). The CDD was defined, the technologies examined and selected and the development resulted the CDD prototype which is the final technological artefact in CREATE. The following section provides a detailed description of the Cross Domain Demonstrator use case.

2. Cross Domain Demonstrator use case

High quality demands a production process which can adapt and react to process variations. High quality also demands good control of production processes. If a production process is too complex to be modelled explicitly, it is advantageous to employ tools which can contribute with decision support to the operator, or be applied directly to the process, provided that an adequate automation level is implemented.

Due to the inevitable variations in inputs' characteristics and fluctuations in manufacturing process parameters, it is often necessary to adapt controllable process variables to the imposed changes. For a general model of a process/system please see Figure 2.

A process is controlled by adjusting controllable variables. There are always a number of uncontrollable variables that also influence a process. Relevant products' characteristics are preferably measured that can be used to assess the quality of the outputs. Depending on available measurement devices, their speed, and the costs of the measuring process, each of the outputs may be measured. Alternatively only a subset of outputs is measured (or some parts of them), which represents a sample to be latter statistically analyzed. The measured characteristics usually have associated tolerances.

Assembly processes in particular are manufacturing processes in which smaller parts are assembled into a new part, which is one final product (an assembly), or a sub-assembly that is to be assembled into a bigger assembly. The assembly processes can be organized in one or more serial workstations, in which parts are added in sequence by operators or robots. An assembly process composed of a number of serial work stations is called assembly line (or a multi station

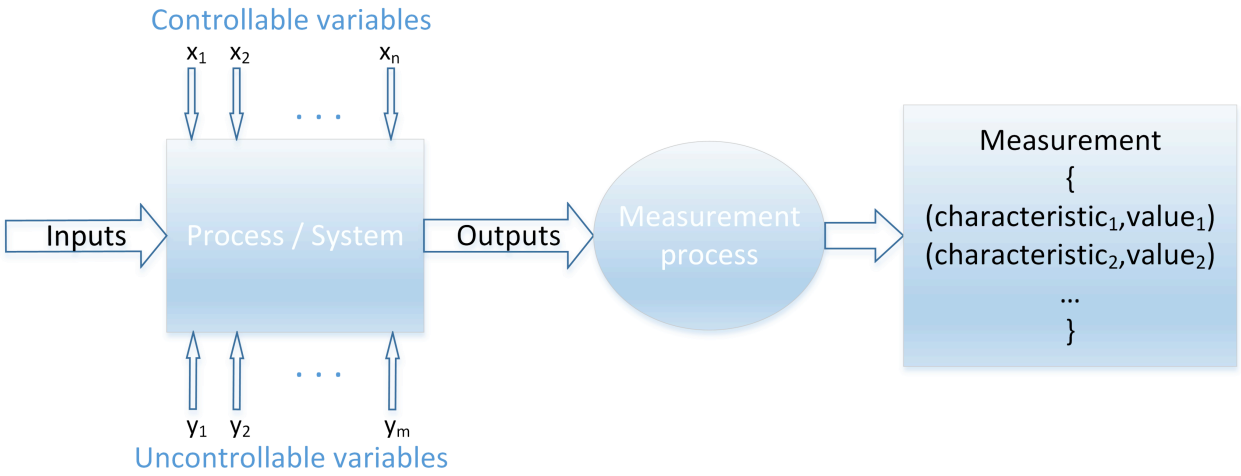

Figure 2: A general model of a process or a system

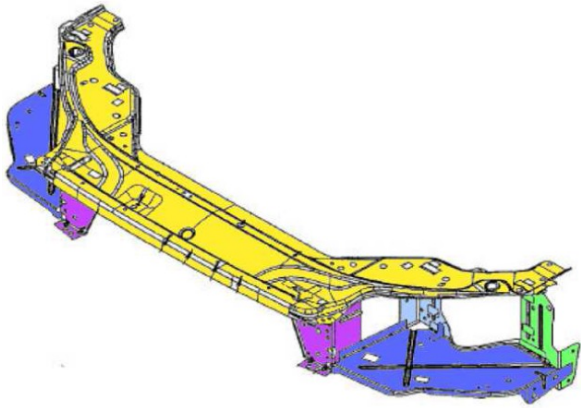

Figure 3: Breakdown of crossmember pieces.

assembly process). A typical example of an assembly line is the one that assembles cars or car components. The most influential uncontrollable variables in assembly process are environment temperature, variations in characteristics of ingoing parts, parts forming history [1], fixture design [2] variations caused by welding together the ingoing parts [3].

The adjustments of an assembly process are usually done when it is detected that a produced parts falls out of tolerances. It is usual in a car production line that the assembly process adjustments are done by experienced operators and engineers. Nevertheless this intervention does not come without a price in potentially slower production rates and increased possibility for imperfections in the final product characteristics. Additionally, not all of the technicians are able to correctly calibrate the manufacturing processes. Since the adjustments made are based on previous experience, i.e. on previous corrective actions and their outcome, the Case Based Reasoning (CBR) artificial intelligence methodology is imposed as a natural way of automating the adjustments. CBR provides adjustment actions based on the knowledge accumulated over time in a so-called case library, which is the counterpart of the knowledge accumulated by the technicians. A more detail description of CBR is given in section 5.1.

As a test bed for the CBR based approach for automating process adjustment, for the purpose of improving the quality of finished products, the Volvo Car Components (Volvo C.C.) manufacturing process for parts known as crossmembers is chosen. A crossmember is a metal part that goes inside the frontal part of the car and serves as a support and for housing other parts (Figure 3). The CBR based software system, collects the measurements of the finished parts geometry, together with the process adjustments done by the technicians, and builds the case library. By using the CBR, the software system is capable of suggesting corrective actions for cases in which a finished part falls out of tolerance, or even before a finished part falls out of tolerance.

2.1 Crossmember production line

The crossmember production cell is shown in Figure 4.

Figure 4: The crossmember robot production cell.

Crossmembers are formed by assembling smaller pieces in a symmetrical way. The blue, purple, green, light blue and yellow pieces shown in Figure 3 are placed in a dedicated workbench by a technician. In order to manipulate positions of the ingoing parts, a workbench in which the pieces are placed has placement pins which can be slightly displaced by adjusting the tie in knots that holds them to the workbench. The workbench holds the pieces together so that industrial robots can take part in the manufacturing process, welding the pieces together. A part of a workbench with an input part fixed, is shown in Figure 5.

Finished parts are measured on 290 predefined points by a Coordinate Measuring Machine (CMM). Each of these feature points have their associated tolerance levels predefined. If a measurement indicates that a part is out of specification, problem may arise when mounting that crossmember into a car and may generate problems such as in lamp fixation and similar. In those situations the controllable process variables need to be adjusted to correct the process.

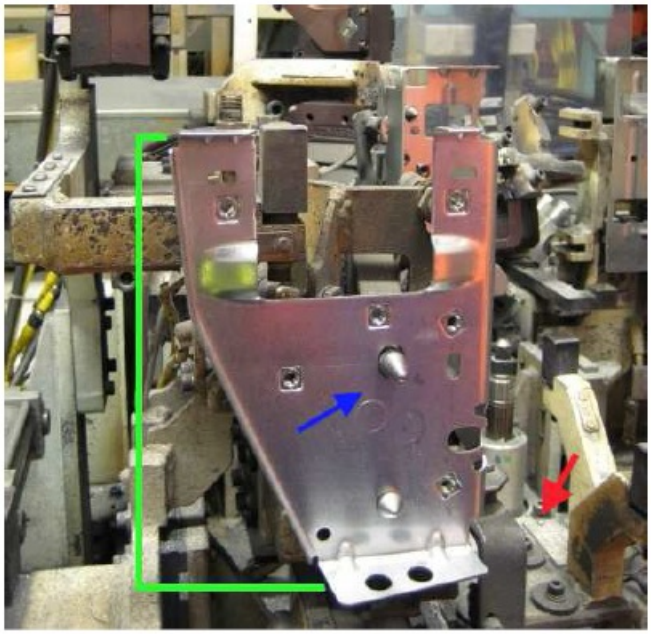

Figure 5: A part of a crossmember fixture (Metal Piece (Green), Adjustment Pin (Blue) and adjustment screws (Red arrow))

3. The CREATE Architecture

The defined CREATE architecture provides a high level blueprint for enhanced industrial automation systems. It groups components that are frequently found in industrial automation systems (e.g., devices, sensors, automation software, human operators) as well as innovative "entries" for such environments (e.g., Knowledge Bases and Artificial Intelligence applications) into conceptual building blocks and organizes them in a physically distributed but logically centralized, interoperable and flexible architecture. The conceptual building blocks of the CREATE architecture are:

- Physical Objects
- Automation Software
- Knowledge Bases
- Communication Platform
- Human Machine Interaction (aka Graphical User Interfaces)

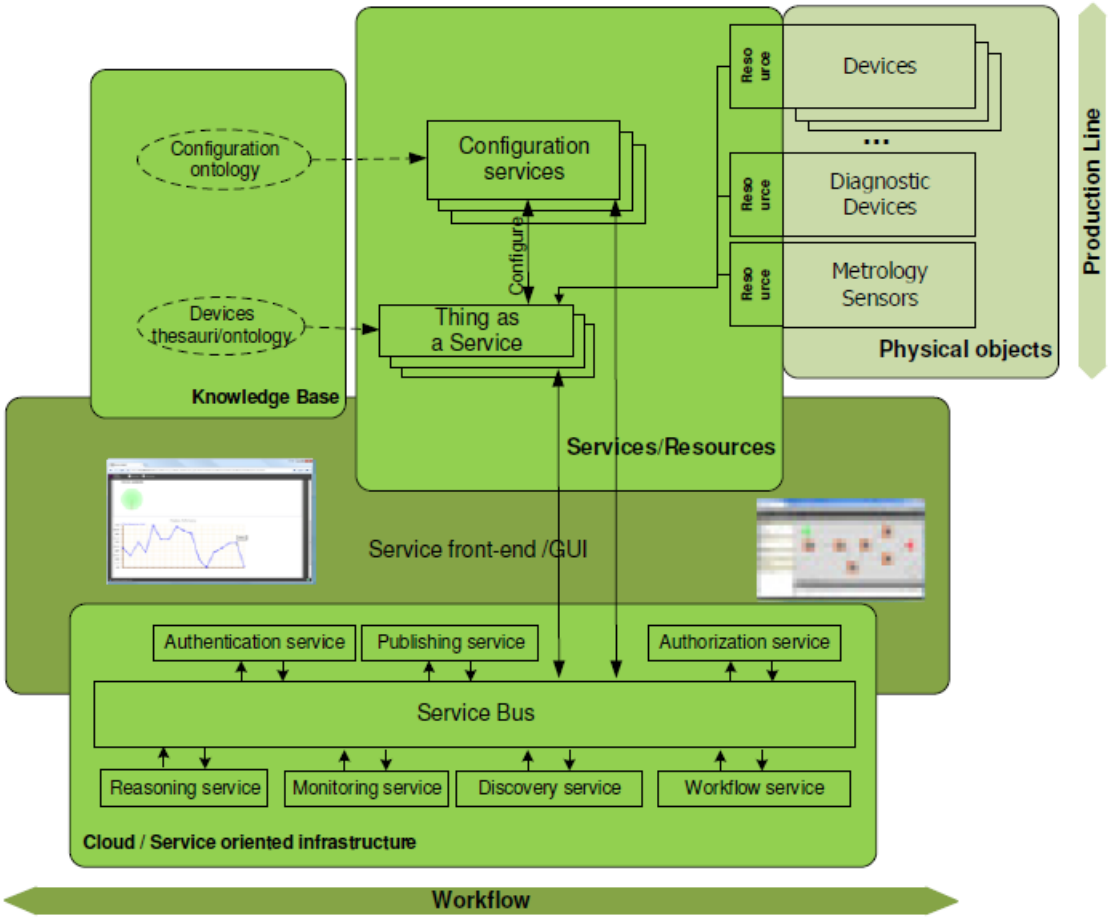

Figure 6: CREATE architecture overview

Physical Objects

The physical objects conceptual block in the CREATE architecture concerns tangible artifacts that are part of the production system such as sensors, hardware, mechanical parts, actual parts of the end product and similar. These physical objects are exposed into the virtual world in various forms, including the communication of their data during their real-life, real-time operation and their 3D virtual representation models. Most functionalities and applications developed in the CREATE approach are built on top of the physical devices and their associated automation software.

Automation Software

The CREATE architecture is heavily capitalizing on the 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 CREATE approach using a Service Oriented Architecture (SOA) allows the loose coupling of the different components and the portability in the use of the associated automation software of the devices. 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. Moreover, the devices must be able to get "instructions" from humans so that they control their actions and behavior which is also facilitated by automation software. Finally, the huge amount data generated during production require AI applications to extract knowledge that will contribute to the optimization of production.

Knowledge Bases

The CREATE approach introduces components that are based in artificial intelligence applications for decision support (and more) as implemented in the Monitoring and Quality Control use case with Case Based Reasoning (CBR) and the Flexible Material Flow use case (with the semi-automatic reconfiguration of the production line). The AI applications need data to operate on/with – data that are made available from what in CREATE architecture are labeled as knowledge bases. These knowledge bases can store data generated both during design time or real time.

Communication Platform

The demonstrators for the CREATE use cases are understandably incorporating limited set of components of the production system, since they serve as a proof of concept. However, real life industrial automation systems would involve an ever growing and changing number of components that need to work together and communicate with each other. The CREATE architecture provisioned for such a need and instructed the use of a communication platform that can integrate seamlessly all these components and facilitate their communication, no matter how heterogeneous they are. This can be accomplished by employing an Enterprise Service Bus (ESB). An ESB will facilitate a SOA approach and will integrate devices, legacy systems and other components and brokerage their communication. The communication platform should offer semantic interoperability and data transformation capabilities so that heterogeneous components in the system can still communicate. Communication is an important factor of the CREATE architecture and it's performed both vertically and horizontally across the system.

Human Machine Interaction

Industrial systems can be quite automated but at least for now the human factor is significant in production lines. The new automation features complement human work, helping humans to be more effective and productive. The CREATE architecture reinforces the inclusion of human

workers in the industrial automation systems by proposing the use of intuitive, friendly GUIs that are platform independent and highly portable so that users can access it from a variety of devices. The interaction with users should focus on simplicity and intuitive communication using graphs, charts and workflows that guide users and allow fast understanding of the data behind the visualizations.

The CREATE architecture was detailed in nearly all deliverables after its initial definition. The motivations for its design were explained, how it was implemented in the demonstrators was clarified and partners used every chance to provide guidelines to interested partners for implementing the CREATE architecture. The next section will once again argue the business value of implementing the CREATE architecture in industrial systems.

4. Business Value of the CREATE Architecture

The CREATE architecture facilitates functionalities and implementations which when realized translate in business value for industrial automation system. The main features based on the CREATE architecture are listed below.

***Control and Monitoring*:** The control and monitoring in industrial systems is critical for effective and efficient operation, and for fast responsiveness and the disaster prevention. Control and monitoring functionalities in traditional industrial systems and production lines are tightly coupled and usually require human operators to be physically close to the operating machines for their control and monitoring. The CREATE architecture allows the loose coupling of components, the high portability of control and monitoring systems and applications and even the automated response to the real-time information (e.g., system to automatically stop printers/production when their out of ink). Furthermore, the CREATE CDD showcases how even human-machine interaction can be automated. These benefits from CREATE result to more efficient production processes as well as more efficient use of the human resources.

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 CREATE architecture allows the integration of new components in a plug and play approach as well as the operation with the new integrated components fast with minimized costs and re-programming efforts. In the deliverable "D5.3 CREATE Interoperability" is presented in detail how through the CREATE project various

interoperability aspects are researched and enhanced. These aspects include *communication, functional, information schema* and *data format* interoperability.

Quality Assurance: Industrial systems and production involve a very important task which is the evaluation of the quality of the end products and the examination of whether they meet their pre-defined specifications and their aimed quality standards. This process is called quality assurance which is traditionally performed by human operators. 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 CREATE architecture proposes and facilitates such enhanced quality assurance in industrial automation systems.

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.

In the Cross Domain Demonstrator use case and prototype most of the above features are covered and specifically the production process optimization, the quality assurance and the interoperability aspect.

5. Technologies Used

5.1 Case Based Reasoning

In computer science Case Based Reasoning (CBR) is a method for implementing an artificial intelligence algorithm that has the ability to identify a solution to a problem, using previously solved problems as guidance. Even though the CBR systems have successfully been implemented before for improving some industrial processes, it is still a new and promising approach that has not yet been widely tested for its usefulness. CBR is a very promising methodology also for implementing quality assurance. It allows product measurements and its related adjustments to the production line to be stored as cases in a so called case library

(database with problem-solution pairs). The CBR system can learn and produce the most optimal response in order to adjust the system, and manufacture a quality part, which should reduce time and effort in solving problems in manufacturing and also reduce environmental impact by reducing resources and spill in manufacturing. The main strength lies in the fact that it enables directly reusing actual cases that have been solved in the past. The approach is especially suitable where simulation and modelling is to complex and adjustments cannot be calculated in real time. This is often the case in a real manufacturing environment where the relationship between the result and adjustment is so complex that it cannot be predicted; as too many factors influence the outcome, or the input-output relation is not known. Even skilled technicians learn over time how to adjust in order to get the desired result. Technicians' experience may have been acquired through costly mistakes whereas their memory is not as precise as a computer memory. CBR enables harvesting this experience and also transferring it between experts and to less experienced operators.

For the purpose of predicting geometrical quality and reducing variations in assembly processes, a number of different approaches have been suggested and used (e.g. [4-10]). Some of these approaches are oriented to identifying the root causes of the variations [11]. However, knowing the root cause does not necessarily provide guidelines for reducing the variations and/or offsets from nominal values of products' measured geometrical characteristics (GCs).

Even though CBR systems have been successfully implemented before for improving some industrial and engineering processes (e.g. [12-15]), even in combination with methods such as artificial neural networks [16], CBR can still be considered as a new technique, which is not yet widely tested for its usefulness in industrial processes. CBR enables harvesting experience obtained during production and also transferring it between experts [17]. Synergies between CBR and data mining methods can potentially improve and boost the performance of CBR systems. Our implementation of a synergy between CBR and regression analysis is presented in next section (5.1.2).

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 if it has a solution from a similar problem already available. The second assumption is that similar problems tend to happen multiple times [18].

A CBR system maps a specific problem to a specific solution. If a new problem arises and a known solution is not known a priori, the CBR system proposes a solution based on the similarity that this new problem has to other problems that have been solved before. If the proposed solution proves to be efficient, the solution is then stored in the case library and the system has successfully learned to solve a new problem. The CBR system has following main

stages for solving a problem that has been issued [18]:

- Find the nearest problem,
- Retrieve the solution for the most similar problem from the case library,
- Reuse the information from said solution adjusting it as necessary - adaptation (possibly by a pre coded algorithm),
- Revise or check the proposed solution,
- Retain the solution for future use if proved successful.

In classical CBR theory, there are two spaces defined for the CBR system (see Figure 7): the "problem space" and the "solution space" [19]. Since the system needs to compute a solution according to similar experiences, a mathematical approach to detect the most similar (or nearest) case in the problem space has to be implemented. Usually a different similarity measurement is used for a different implementations of CBR since the similarity measure used is very much dependent on a particular process for which the CBR is used.

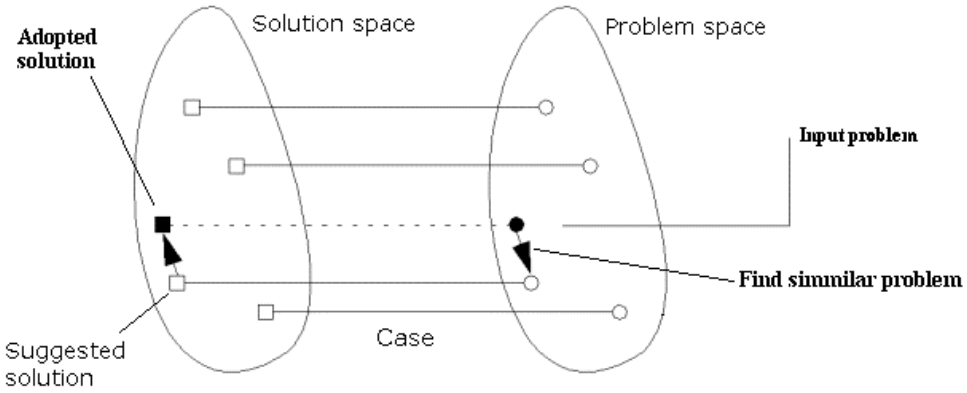

Figure 7: Problem and solution space in CBR

5.1.1 CBR Knowledge Model

CBR is a knowledge-based system. Knowledge-based systems are a sub-class of intelligent systems that are designed by having a knowledge base in an independent module. Knowledge can either be represented explicitly or be hidden in an algorithm. In any case the knowledge is presented in some formulation. The formulation is stored in so called knowledge container. There are four main knowledge containers in CBR:

- **The vocabulary container** - determines what one can discuss explicitly, i.e. by which terms

the system being explored can be described with. There are generally an infinite number of terms that that can describe a certain object but only a few are relevant for a specific task.

- **The similarity container** - consists of all knowledge needed to determine what makes a case similar to another. There are multiple ways to calculate similarity: the use of simple similarities where the values are either equal or not, the use of weights to represent relative importance of the attributes, the use of fuzzy algorithms that consider all attributes and their importance at once.
- **The case base container (case library)** - contains stored cases. The cases can be collected from the past, generated, or be artificial (these are usually used for testing purposes). This is the main knowledge contained in CBR.
- **The Adaptation Container** – the most important knowledge in adaption container is definition of rules on how to adapt cases stored in the case library to a new case. In other words, the adaptation container contains information on how to modify a solution.

In the following chapter specifics will be presented of the knowledge containers used for the CBR system applied to crossmember production.

5.1.1.1 The CBR knowledge model in the Cross-Domain Demonstrator

The vocabulary container main elements for CBR applied to crossmember production are:

- Pin – an adjustable point in fixture holding the crossmember input parts,
- Pin direction – Pin can be adjusted in one or more possible Cartesian space directions (X, Y, Z),
- Feature – measurable geometrical characteristic of an assembled part,
- Feature direction – feature can be measured in one or more Cartesian space directions (X, Y, Z), or normal (perpendicular) to the assembled part surface (usually denoted with NOR).
- Tolerances – each feature has a tolerance interval defined for each direction. If a measured value for a feature in a certain direction falls out of tolerance interval, that generally means that an adjustment is needed to correct the process.

By the term “current case” it is meant a new measurement for a finished part containing at least one feature whose measured value falls out of tolerance interval. Additionally, a case is defined as ‘‘unfinished’’ if it lacks information about adjustments and/or outcome measurements, and it is defined as ‘‘finished’’ if it includes all the parts (the precise parts are defined below).

The case base container, the heart of the CBR system, is consisted of the stored finished cases each represented by:

- measurement of an assembled part that had at least one feature measurement out of its associated tolerance interval,
- adjustment,
- measurement after adjustment.

The nearest case retrieval is done by calculating distances between the current measurement (problem case) and the cases stored in the case library. This can be done by using various similarity measures [19]. For the CDD the root mean square distance (RMSD) is used since it directly reflect the differences in geometrical characteristics. Given a new problem case A, the distance to a target case T is calculated as shown in the following equation:

$$Dist_{A,T} = \sqrt{\frac{\sum_{k=1}^{n}(T_k - A_k)^2}{n}}, \quad (1)$$

where k is a k-th feature in a certain direction and n is the total number of measured features. Additionally, the normalized root-mean-square deviation distance (NRMSD) has been used in CDD. It is the RMSD divided by the range of measurement values. RMSD and NRMSD form the similarity container for our CBR system used in CDD.

By using the similarity container the prototype CBR system compares the current problem case (measurement of an assembled part with at least one feature being out of its tolerance level) to the measurements before adjustment, for all the cases stored in the case base container. The system then presents the nearest cases and more importantly their adjustments to a user (technician) in an adequate and user-friendly way. (This is functionality of the GUI which is described in more detail in (chapter 6)). The technician then decides which one of the presented adjustments to apply, or forms a combination of the presented adjustments. Consequently our CBR system implemented in the CDD does not have an algorithmic adaptation container but the user is the one doing the adaption of the presented potential solutions.

5.1.2 Synergies between Case-Based Reasoning and Regression Analysis in Assembly Processes

If we denote the vector of assembled part's measured GCs (i.e. features) as $F = [F_1, \dots, F_n]$, the vector of controllable variables as $C = [C_1, \dots, C_m]$ and the vector of uncontrollable variables as $U = [U_1, \dots, U_o]$ (here n, m, o are the total number of GCs, controllable and uncontrollable variables respectably). Then the measured GCs are generally a function of controllable variables

and a function of uncontrollable variables:

$$F = f(C) + \varepsilon, \qquad \varepsilon = g(U) + E, \tag{2}$$

where term ε contains the influence of uncontrollable variables $g(U)$ and the random error E.

The general task given to regression analysis here is to find the function f that is most adequate for a specific process, and to estimate its parameters. For each of the measured GCs F_i we can assume that the function f_i is of the same type, since in an assembly process normally the same physical principle governs the forming of every output GC. Still the parameters of each f_i will be different because the influence of each input variable is different for each of the GCs. For the parameters estimation the least squares method is applied on the input and output data obtained from an assembly process.

One of the methods for evaluating a regression model is the analyses of residual errors (i.e. differences between the predicted and observed values). In regression analysis normally the significance of a residual error increases linearly with the error value, but for the assembly process the residual error can be considered significant only if it causes a predicted GC value to fall out of its tolerance level, in a case when corresponding observed value is inside tolerance level, and vice versa.

One of the possibilities of how to use the regression model is to use it for a verification of adjustments suggested by a CBR system. The adjustments are fed to the model and the model output is examined to determine if the adjustments provide desired outcomes (Figure 8).

Furthermore the regression analyses provides t and F statistics for each of the input variables in the model [20]. Only the input variables with satisfactory associated p-values for the t-statistics are considered to be able to influence the output GC value. The corresponding F-statistics can be used for assessing the statistical significance of each input variable, since F-statistics shows how much of the total variance in one output variable is explained by a particular input variable.

List of input variables that can influence a particular output variable, and the strengths of influences described by the F-statistics, is a valuable knowledge that can be used to improve the performance of a CBR system. One obvious application of that knowledge is for the adaptation of an adjustment. It can be used to ensure that the adjustment contains only input variables that can influence GCs being out of their tolerance levels.

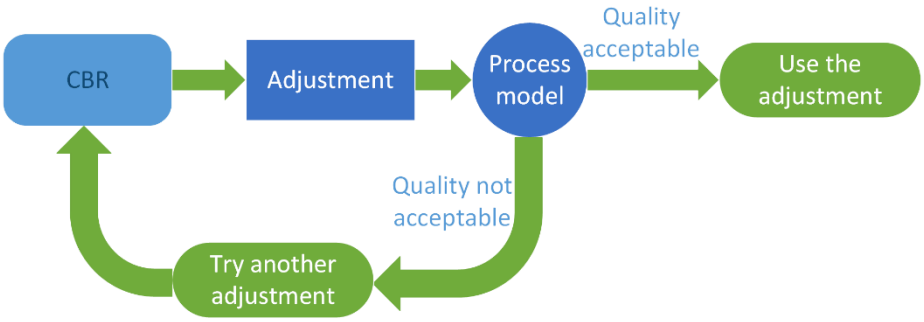

Figure 8: Verifying CBR output by using process model in a closed loop.

We have tested the collaboration between CBR and regression on a real case obtained from the crossmember production line. The case was randomly selected among the set of available cases. For that purpose we have presupposed a model type between inputs and outputs of the assembly process examined. The hypothesized model was a liner model for which it was allowed to have also squared input variables. It can be concluded that the model performs well for the particular case presented. For details please see publication [21].

5.2 Service Oriented Architecture

Service Oriented Architecture (SOA) is an architecture for building business applications as a set of loosely coupled components in order to deliver a well-defined level of service by linking together business approaches. Using this architecture pattern, independent systems and applications can communicate with each other by exposing and using services. Some of the benefits of SOA pattern are explained below:

a. **Standardized Service Contract:** Services within the same service inventory are in compliance with the same contract design standards

b. **Service Loose Coupling:** Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment

c. **Service Abstraction:** Service contracts only contain essential information and information about services is limited to what is published in service contracts

d. **Service Reusability:** Services contain and express agnostic logic and can be positioned as reusable enterprise resources.

All functions or services are defined using a description language and have able to invoke interfaces that are called to perform business processes. Each interaction is independent of each and every other interaction and the interconnected protocols of the communicating devices.

Interfaces can be platform independent and a client from any device using any operating system in any language can use the published services.

5.3 Web Services

Web services can be regarded as an implementation of the Service Oriented Architecture that uses a number of emerging standards like WSDL, UDDI and SOAP. The Web service specifications are completely independent of programming language, operating system and hardware. Through a web service an application can respond to requests using a protocol built on top of the HTTP. The calling machine might be a PC while the machine hosting the service could be a mainframe. Web services provide an interface layer that shields the caller from having to know any of the technical details of the implementation. The main technology that Web Services rely on is XML. XML is a convenient way to exchange any kind of information as any data like numbers, integers, floating point and names can be represented as strings. By using strings rather than bit streams, XML achieves platform neutrality although the cost of parsing overhead can be considerable. Three of the building blocks of Web services are described below:

- **Universal Description, Discovery and Integration (UDDI):** A directory service where businesses can register and search for Web services
- **Web Services Description Language (WSDL):** A language for describing web services. It specifies the location of the service and the operations (or methods) the service exposes.
- **Simple Object Access Protocol (SOAP):** SOAP is an XML based protocol for accessing web services. SOAP uses HTTP and it is being used in order to define the general structure of messages. The specific structure of the SOAP body is defined in the WSDL file.

5.4 REST

Representational State Transfer is an architectural style that is used as an alternative to SOAP for accessing web services. In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URI) also known as links on the web. RESTful applications use HTTP requests to post data (create/update), read data (make queries) and delete data. These operations can be translated to the following HTTP methods:

- **GET** as a read operation
- **PUT** as a write/modify operation
- **POST** as a create/new operation
- **DELETE** as a delete operation

As its predecessors it is also platform independent, language independent, standards-based and can also be used in the presence of firewalls.

5.5 HTML5, CSS & AngularJS

HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and *AngularJS* are all technologies regarded as front-end technologies. HTML and CSS are both combined for the presentation or the look & feel of a web page on a user's web browser. AngularJS is regarded as a Javascript technology running on the browser's javascript interpreter thus making the web page more dynamic and interactive. Some details follow for these technologies below:

HyperText Markup Language (HTML) is used for creating and visually representing a webpage. It consists of HTML elements where tags enclosed in angle brackets are being used. As the Web browser parses the html file it displays the content of this file according to the HTML elements being used. HTML5 is an extensible form of HTML (also known as HTML 4) called Extensible Hypertext Markup Language (XHTML). HTML5 has made web pages far more dynamic and has also solved compatibility problems that affected HTML4 and the different browsers involved. Furthermore HTML5 is designed to support for multimedia on mobile devices.

Cascading Style Sheets (CSS) is a style sheet language used for writing formatting rules. These rules tell a web browser how webpage content should look like in terms of layout and style. CSS3 is the latest evolution of the CSS language. CSS3 brings some more novelties as well as new layouts extending existing CSS capabilities.

AngularJS is a Model View Controller (MVC) open-source framework that defines various concepts in order to build Single Page Applications (SPA). Some of the very well-known features of AngularJS include expressions, directives, data binding, filters, views, modules, scope, controllers and dependency injection.

5.6 Enterprise Service Bus

An *Enterprise Service Bus (ESB)* is an architecture which consists of rules and principles for integrating numerous applications together over a bus-like infrastructure. The main concept of the ESB architecture is that different applications can be integrated by putting a communication bus between them and then enable each application to talk to the bus. The main advantage of ESB is that systems are now decoupled from each other as they are able to communicate without and dependency or knowledge of other systems on the bus.

Some of the implementation features of ESB are the following:

- The bus concept is usually implemented using a messaging server like JMS or AMQP
- Data being exchanged usually has an XML format
- An adapter between the application and the bus is used for marshaling data between the two parties

Some of the disadvantages of the ESB architecture can be increased overhead and slower communication speed.

5.7 Microsoft Azure

Microsoft Azure is a cloud computing product created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed datacenters. It provides both Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) services and supports many different programming languages, tools and frameworks.

Here is a list of Azure's most prominent features (Nov. 2014):

- IaaS:
  - Virtual machines
  - Storage, backup, and recovery
  - Big data compute
- PaaS:
  - Websites
  - Mobile services
  - Cloud services
  - SQL databases
  - Media services
  - Integration services (Service bus)
- Application as a Service:
  - Microsoft Visual Studio Online
  - HDInsight
  - BizTalk.

For the purpose of implementing the CDD we have used the following Azure services:

- Azure Storage: for maintaining the websites and other non-relational data,
- Websites: for hosting the CDD server side which is a Windows Communication Foundation (WCF) service encapsulated in a web site, and for hosting the GUI website
- SQL Database: for storing all the necessary relational data like CBR database, GUI

resources etc.

Azure Storage is scalable and highly available storage for non-relational data. It can be used by any kind of application as a data source, but it also provides the storage foundation for Azure Virtual Machines.

Microsoft Azure Web Sites is a PaaS that allows publishing web applications running on multiple frameworks and written in different programming languages and in different frameworks (including Microsoft proprietary frameworks and other frameworks as well). For the developers in Microsoft Visual Studio it is possible to publish (i.e. make publicly available) their developed websites, literally by a single click.

SQL Database is Azure service offering relational-data-storage capabilities (similar to Amazon Relational Database Service). AzureSQL Database allows users to make relational queries against stored data, which can either be structured or semi-structured, or even unstructured documents. SQL Database enables querying data, search, and data analysis and data synchronization. In comparison to Microsoft SQL Server edition it has some limitations[22].

6. CREATE Cross Domain Demonstrator Description

The general software and network architecture of the CDD is shown on Figure 9. From the functional perspective CDD's functionality can be expressed as having server and client side.

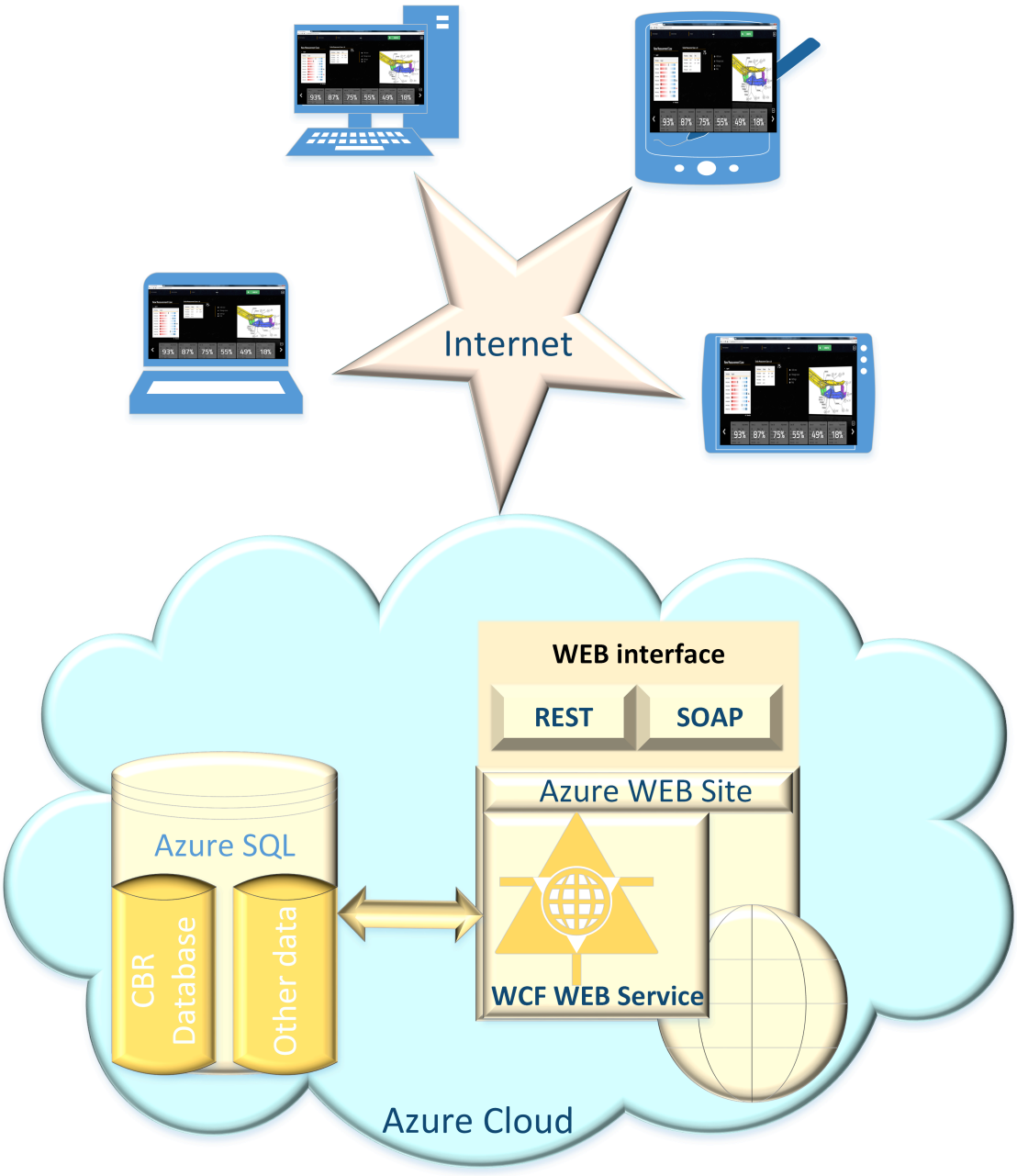

Figure 9: General CDD software and networking architecture.

The server side is entirely hosted inside Microsoft Azure Cloud which enables its high availability. The availability is important especially for industrial software applications since the industrial computer networks infrastructures are normally highly protected cable-based networks with restricted connections with outside world, i.e. the Internet. At the same time there are usually Wi-Fi networks inside factories with Internet connections. The CDD client side, i.e. the Graphical User Interface (GUI), as a WEB application can potentially run on any device that supports Internet browsing (even though it is currently optimized only for personal computers), and can use WiFi to connect to the Internet, i.e. to the Azure Cloud. Since the sever side is available through the Internet, the client users can even use the public wireless internet infrastructure (3G) or in fact any internet connection. This enables CDD client application to be used even from outside the factory. On the other hand, if the company using the CDD based industrial software would prefer to keep it all inside premises, all the CDD components can be installed and used from the on-premises servers since they are hosted inside widespread and open technologies: relational database and Web server.

The server side is composed of one relational database which can be seen as having two separate sections, and one web site. The database contains on one side the tables that host the CBR database, and on the other side all the other data needed for the software functionalities like for example: user logins, usage information, logs, etc. The database contains also different views, triggers and stored procedures which are used to implement CBR logic as well as implement complex business rules in the data.

The WEB site hosts a Windows Communication Foundation (WCF) web service. WCF is a tool often used to implement and deploy a service-oriented architecture (SOA). It is designed using SOA principles to support distributed computing where services have remote consumers. Clients can consume multiple services; services can be consumed by multiple clients. Services are loosely coupled to each other.

Our WCF web service has a WSDL interface (Web Services Description Language) that any WCF client can use to consume the service, regardless of which platform the service is hosted on. Additionally it exposes REST and SOAP endpoints which can be accessed by any client over the internet. The service uses Web Services Security (WS-Security) to assure integrity, confidentiality and restrict access to the service by identification. The service main function is to, in collaboration with the database, implement and expose all the necessary functionality for the GUI and potentially other clients.

Hosting the relational database and the web service inside a Cloud has two important additional benefits:

- the data consistency, reliability and fault-tolerance,
- improved reliability and accessibility.

The above benefits are achieved by geo-replicating the data. The data is replicated at three different locations in the world. Additionally by using the cloud scalability becomes natural since the storage or computing resources used can be configured to even automatically adjust to the current needs of the application, or to the number of current clients to the service. One more advantage is that the CDD's Cloud hosted service can be used from diverse geographical locations, and from different factories.

The GUI of the CDD serves as an interface for interacting with the automation system as provides these global functionalities:

- import new measurements that represent a new case,
- identify the features (i.e. geometrical characteristics) that are out of tolerance,
- receive adjustment suggestions through cases similar to the current case,
- presentation of the adjustments performed in those cases.
- browsing CBR database

These functionalities are all visualized with pictures presenting the features that are out of tolerance, and other functionalities to facilitate the operators compare the points that are out of tolerance between similar cases. The adjustments are also presented so that the operator can have immediate perception where each adjustment point is positioned in the actual car part.

Operators can save the new measurement and the adjustments they made as a new case to increase the performance of the CBR system. Moreover users can browse the library of cases and visualize the information in them. These are the main functionalities. Below are provided a series of figures presenting the look and feel of the CDD GUI.

On figure 11 is presented a graphical way of comparing the imported measurement to similar cases. In this view the values that are out of tolerances in the new measurement (on the left) are graphically (by using colored slide-bars) compared to the values out of tolerance in the most similar cases. Current view compares the new measurement with a case that is 88% similar.

Figure 11. shows an alternative presentation of comparison between the current case and the same one of the similar cases. In this view the actual measured values are shown. The user is able to interactively compare current case to previous cases by hovering over the values presented.

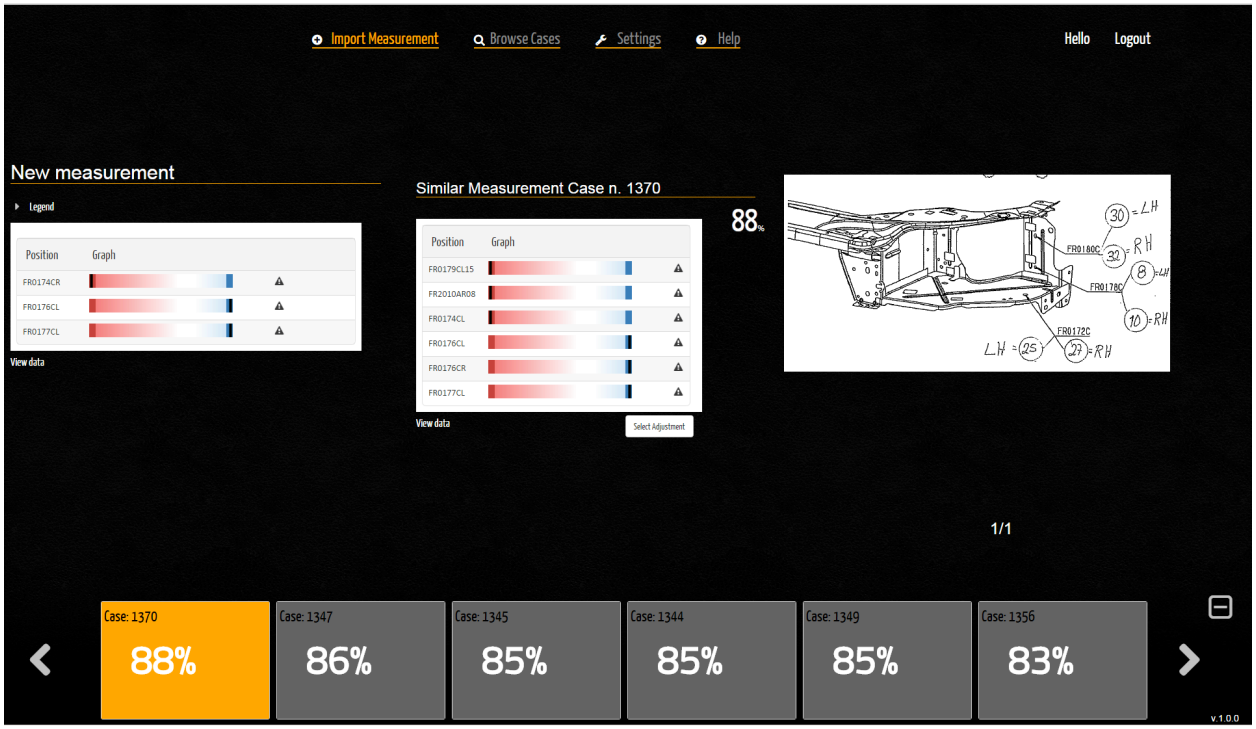

Figure 10: Comparing the imported measurement to similar cases

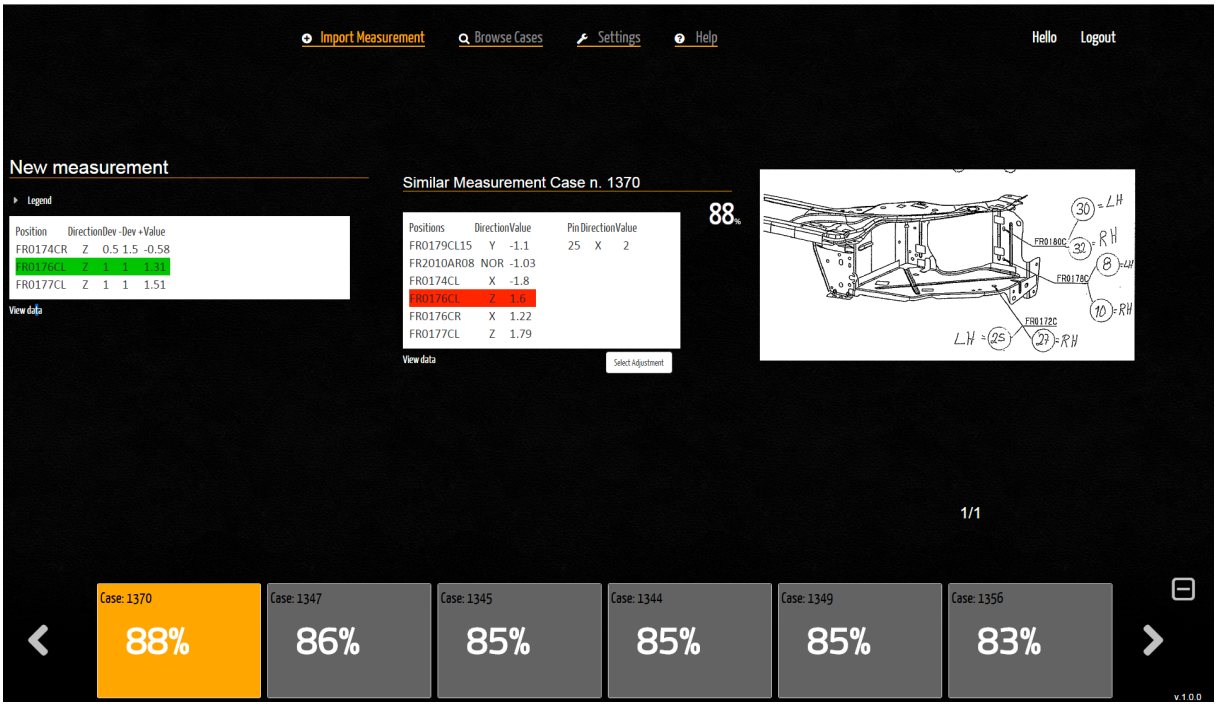

Figure 11: Adjustments of the similar cases as recommendation for the case at hand

Furthermore the operator can edit the adjustments suggested (adaptation), as illustrated in figure below:

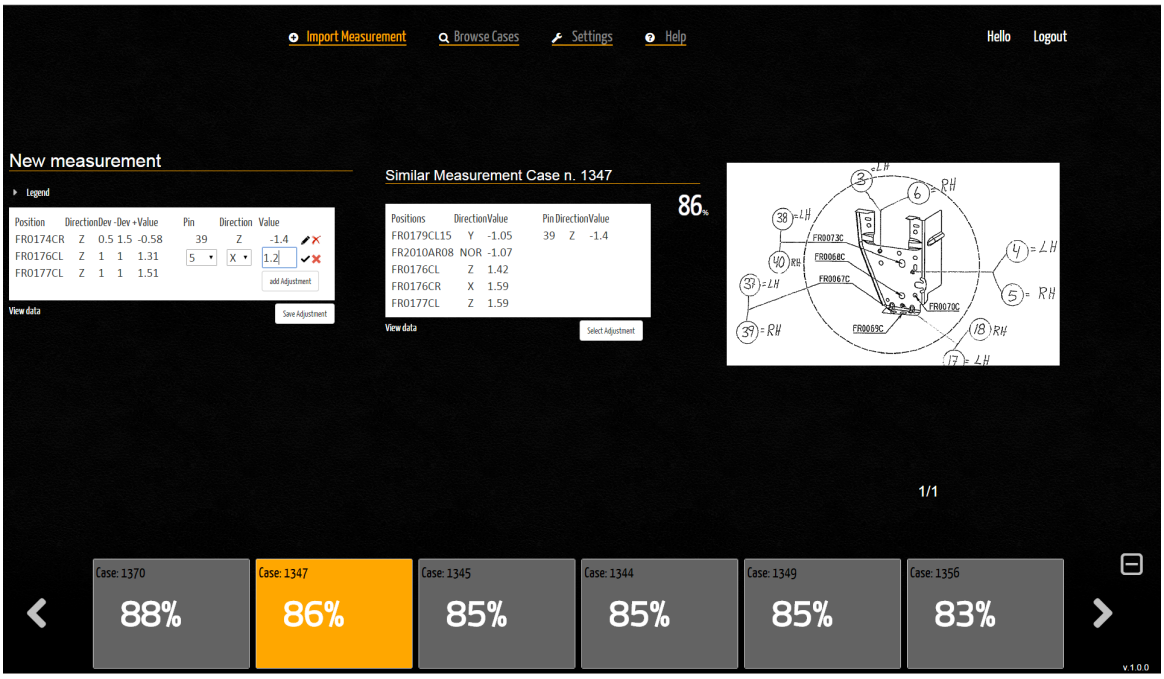

Figure 12: The operator can enter the adjustments made in the new case

And save the adjustments applied together with the imported measurement to the case library.

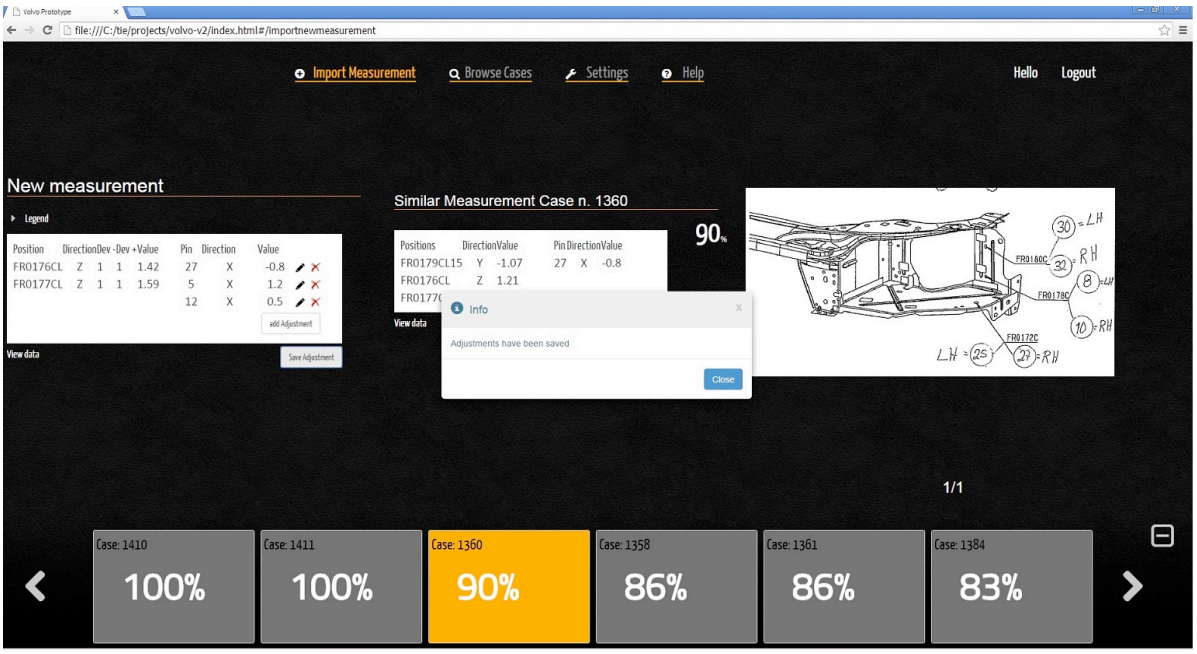

Figure 13: Save new measurement and adjustments as a new case to the library

Cases in the library can be browsed by various parameters as showed in the figure below

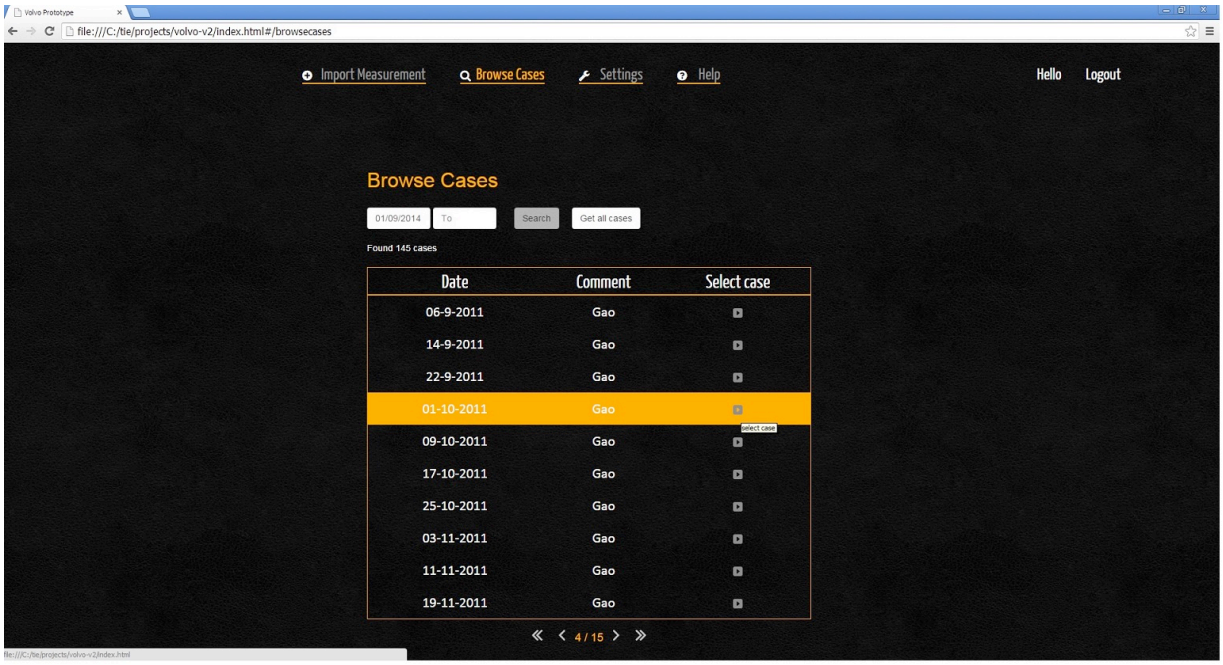

Figure 14: Browsing cases in the library

and their details can be viewed as depicted in the following picture:

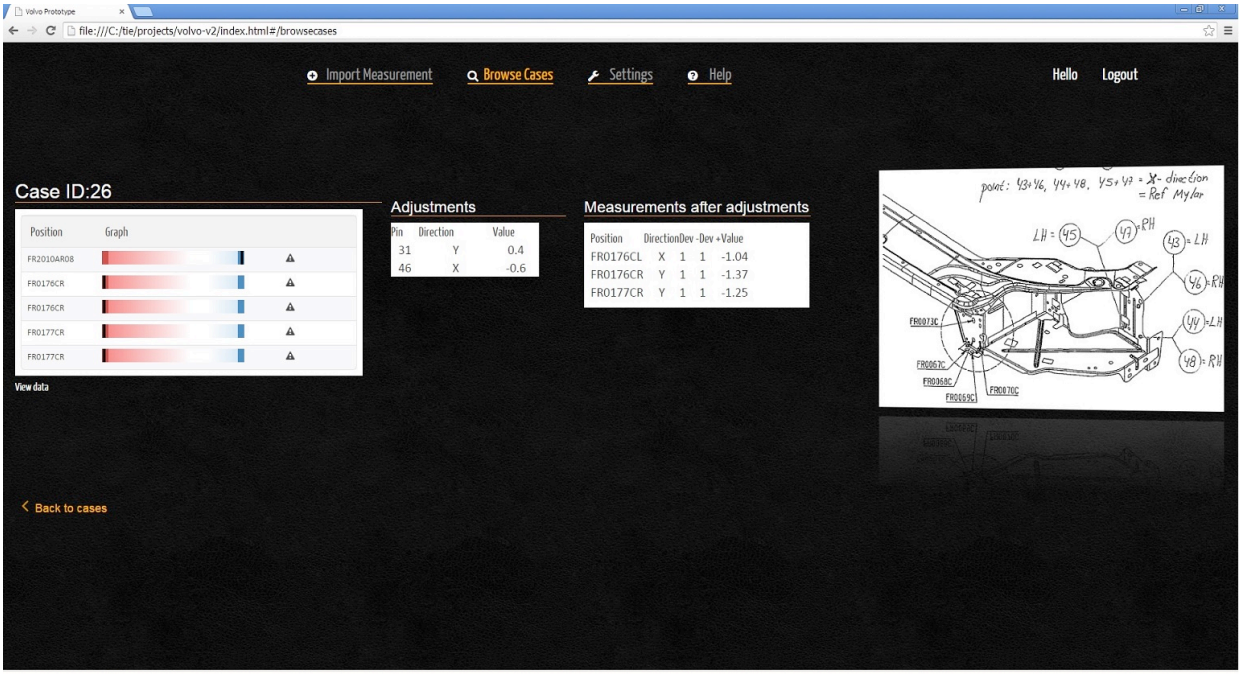

Figure 15: Overview of the selected case from the library

7. CDD Evaluation and Testing

To evaluate the performance of the CDD's CBR system a subset of solved cases is selected and fed to the CBR system. These cases are used to simulate "current cases" (for the definition of the "current case" please see section 5.1.1.1). The solved cases 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 normal everyday operation of the system (i.e. not for testing) only problem cases (without known adjustments) will be presented to the system through the GUI presented in previous chapter.

The precise testing procedure is as follows:

1. Fourteen cases are chosen randomly from the CBR case library containing in total 127 cases at the time of testing.
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).It can be noticed in Table 1 that there are five cases (out of fourteen) for which the suggested adjustments are not on the same pins, and/or directions, 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.

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	6.8	-1.5	19	X	7.4	-2.1	0.31	91.15
2	19	X	4.2	1.3	19	X	4.6	0.9	0.17	95.03
3	22	X	3.8	1.2	22	X	3.4	1.6	0.14	94.65
4	**25**	X	6.8	-0.8	**22**	X	3.4	1.6	0.26	90.87
5	**25**	X	4.6	1.4	**19**	X	6	-0.5	0.08	85.88
6	25	**Y**	3.2	1	25	**X**	5.6	0.4	0.06	97.50
7	27	X	4.2	-0.4	27	X	3	0.8	0.12	95.27
8	**27**	**X**	3.4	0.4	**39**	**Z**	5.2	0.4	0.10	95.80
9	27	Y	4.8	-0.8	27	Y	3.6	0.4	0.23	92.47
10	27	Y	3.2	0.8	27	Y	2.8	1.2	0.46	90.22
11	**37**	**Z**	4.8	1.4	**25**	**X**	6.4	-0.4	0.13	95.28
12	37	Z	3.4	1.4	37	Z	2.8	2	0.22	96.18
13	39	Z	6.8	-1.4	39	Z	6	-0.4	0.08	97.17
14	39	Z	4.2	1.4	39	Z	3.6	2	0.15	95.36

*The shaded rows indicate the cases in which the suggested adjustment pins and/or directions are not the same as in the done adjustments (the Pin names and/or directions that differ are bolded).

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 testing purposes only the most similar cases (to the current case) have been used. In CDD's GUI however a technician is presented with a number of the most similar cases together with their adjustments, from which he or she can choose or adopt them for the current case.

8. CDD Results and Perspectives

As a test bed for the CBR based approach for automating process adjustments, the Volvo Car manufacturing process for parts known as crossmembers was chosen. A number of crossmember measurements was collected, together with the available adjustments done, which enabled us to create 127 CBR cases. Dedicated relational database model was designed to host the cases. CBR logic and methods were implemented inside the relational database by using database objects: stored procedures, functions, and views.

For the purpose of exposing the CBR logic and the available knowledge stored in the CBR database, a web service was developed. Together with the CBR database, web service was hosted in the Microsoft Azure Cloud. The service exposes SOAP and REST endpoints for communicating with clients, i.e. the GUI.

The CBR system was evaluated on the available set of cases. The presented results of the testing show that CBR performs very well, especially if one considers that the case base database that the system is currently equipped with is quite small. It contains only 127 cases. In a production environment as the number of the stored cases will grow, the usefulness of the CBR system is expected also to grow since its knowledge base will be bigger. The results of the evaluation support the conclusion that CBR is performing well for the particular assembly process, but also suggest that CBR is a valuable and promising methodology that can be used for improving other manufacturing processes.

At the same time GUI for the CBR system was developed by using State-of-the-art software development tools. The GUI communicates with the CBR service through the REST protocol and by using JavaScript Object Notation (JSON) open standard format. The GUI enables a user of the CDD to access all the underlying functionality in a user friendly way.

The testing procedure presented takes into account only the most similar cases. In production environment the technician will be presented with a number of the similar cases, and select one of the adjustments, or adapt and produce a new adjustment, which is potentially better than just applying the adjustment associated with one of the most similar cases. This is where the importance of the GUI design becomes prominent. Since our prototype CBR system does not have an algorithmic adaptation knowledge container, the user is the one doing the adaption of the presented potential solutions. To facilitate and leverage the adaption of the presented corrective actions (i.e. adjustments) it is very important how the previous cases, together with their adjustments, are presented to the user. The efficiency of cases' graphical presentation influences directly the person doing the adaptation of the presented adjustments. The CDD GUI implements an innovative and efficient way of presenting cases in which a user can interactively graphically

compare a new case with previous cases. When the user confirms an adjustment by using the CDD's GUI, the new adjustment is stored in the database as a part of a new case. In this way the system has gained experience, and improved its overall performance (learning).

In addition to providing new knowledge to the system, the adjustments done by using the CBR are stored in the CBR database together with the adaptation actions that users did on the suggested adjustments. The stored adaptations can be used as a knowledge source for developing algorithmic adaptation system in the future work on the corossmember production CBR system. The algorithmic adaptation system can potentially be of and additional value to the CDD user since it would be able to suggest adjustments that are more than replicas of the adjustments done in the history, but their combinations done by a predefined algorithm. This algorithm is expected to imitate the adaptation that operators (i.e. users of the CDD) will be doing while the CDD system will be used in the production environment. For this to be possible the system needs to know what adaptations have been done before a current case.

The additional testing of the CBR system are also possible by introducing additional new measurements, possibly obtained by in-line measuring system. They can potentially be used to further fine tune the CDD system in terms of similarity metrics used for comparing cases in the CBR sub-system. After some time of being used in production environment, the CDD's GUI can also be fine-tuned to the specific environment by using the feedback form the GUI's users.

Furthermore, we have presented concepts and methods for corporation between CBR and regression analysis for the purpose of potentially pushing even further the performance of our CBR system implemented in the CDD. Those concepts have been so far applied only on one case (see section 5.1.2). We plan to conduct an extensive evaluation of the presented methodology by applying it on all available cases in CDD database. If the concepts proves to be useful, the plan is to integrate it to the CDD with the already prepared methodology described in section 5.1.2.

Even though the CDD is implemented for a specific assembly process it is evident that it can be applied to any assembly process directly with possibly only some minor changes, provided that the assembled parts can be measured. The fact that the CDD models human interactions with an industrial process, makes it generally applicable almost everywhere in today's industry. The purpose of modelling human-machine interaction is to provide support in decision making, or even to completely automate the process of decision making that humans do before interacting with a process. Even the synergies between regression analysis and CBR (described in section 5.1.2) are generally applicable provided that industrial system can be modelled as a black-box having inputs and outputs. The CDD makes use of the stored human-machine interactions data to provide suggestions for a correct human action when it is needed. The CDD is a PoC of how this human-machine data can be used and how useful it can be.

9. Conclusions

The Cross Domain Demonstrator was developed to showcase the CREATE approach as the final PoC for the CREATE methodology. It is applied on a specific assembly process in the Volvo C.C. manufacturing line, but as it was describe in the previous chapter, it can be directly used for any assembly process, provided that the adequate level of automation is reached. Even more the CREATE method the CDD is using is generally applicable almost anywhere in today's industry. Moreover, the CDD is an attempt to show that legacy systems and new components can be integrated in a plug and play approach with the CREATE method to form new innovative systems that can utilize the latest technologies with low costs and reprogramming efforts.

It can be generally said that most of the industrial process today rely more or less on human interactions. All modern industrial plants measure all that can even potentially be of interest in industrial processes (e.g. speeds, forces, temperatures, technological product characteristics, etc.). Human interactions with a process are however rarely stored, and no more than their outcomes. The CREATE approach besides integrating seamlessly human workers to the automation systems and processes, also reinforces the use of the human interactions in the processes. The CDD showcases how such data can be stored and used to construct a suggestion system to the humans interacting with the process. This CDD suggestion system bases its quality on the previous interactions human-machine, and grows better in time as more and more data about these interactions is stored. Depending on a productions system, i.e. its complexity and possibility of being modelled, a suggestion system always has a potential of becoming completely automatic, if in time suggestions it provides to operators are not altered by the operators before being applied. The CDD aims to prove a concept, and shows importance of storing and reusing the data coming from human-machine interactions. This is to improve the today's industry since such data is rarely used or even stored.

The CDD is the final PoC for the CREATE approach and this document served both as a detailed description of its specific use case and its implementation, but also attempted to indicate once again the underlying principles of the CREATE method and the enhancements they bring to industrial automation.

10. References

[1] A. Govik, R. Moshfegh, and L. Nilsson, "The effects of forming history on sheet metal assembly," *International Journal of Material Forming,* vol. 7, no. 3, pp. 305-316, 2014/09/01, 2014.

[2] J. A. Camelio, S. J. Hu, and D. Ceglarek, "Impact of fixture design on sheet metal assembly variation," *Journal of Manufacturing Systems,* vol. 23, no. 3, pp. 182-193, 2004.

[3] K. Wärmefjord, R. Söderberg, and L. Lindkvist, "Variation Simulation of Spot Welding Sequence for Sheet Metal Assemblies," in Proceedings of NordDesign2010, the 8th International Conference on Methods and Tools for Product and Production Development, Gothenburg, Sweden, 2010, pp. 519-528.

[4] S. Du, J. Lv, and L. Xi, "A robust approach for root causes identification in machining processes using hybrid learning algorithm and engineering knowledge," *Journal of Intelligent Manufacturing,* vol. 23, pp. 1833–1847, 2012.

[5] J. Jin, and J. Shi, "State Space Modeling of Sheet Metal Assembly for Dimensional Control," *Journal of Manufacturing Science and Engineering,* vol. 121, no. 4, pp. 756-762, 1999.

[6] Y. G. Liao, "Optimal design of weld pattern in sheet metal assembly based on a genetic algorithm," *The International Journal of Advanced Manufacturing Technology,* vol. 26, no. 5-6, pp. 512-516, 2005/09/01, 2005.

[7] L. M. G. Guzman, P. C. Hammet, and G. D. Herrin, "Understanding multivariate components – sheet metal assembly relationships using dimensional slow build studies," vol. 42, no. 13, pp. 2651–2666, 2004.

[8] S. Dahlström, L. Lindkvist, and R. Söderberg, "Practical Implications in Tolerance Analysis of Sheet Metal Assemblies: Experiences from an Automotive Application," *Models for Computer Aided Tolerancing in Design and Manufacturing*, J. Davidson, ed., pp. 311-320: Springer Netherlands, 2007.

[9] M. Hu, Z. Lin, X. Lai, and J. Ni, "Simulation and analysis of assembly processes considering compliant, non-ideal parts and tooling variations," *International Journal of Machine Tools and Manufacture,* vol. 41, no. 15, pp. 2233-2243, 2001.

[10] I. Watson, "Case-based reasoning is a methodology not a technology," *Knowledge-Based Systems,* vol. 12, no. 5–6, pp. 303-308, 1999.

[11] S. Du, J. Lv, and L. Xi, "A robust approach for root causes identification in machining processes using hybrid learning algorithm and engineering knowledge," *Journal of Intelligent Manufacturing,* vol. 23, no. 5, pp. 1833-1847, 2012/10/01, 2012.

[12] B. Mark, "Case-Based Reasoning for Autoclave Management," in Proceedings of the Case-Based Reasoning Workshop 1989.

[13] S. Shokouhi, P. Skalle, and A. Aamodt, "An overview of case-based reasoning applications in drilling engineering," *Artificial Intelligence Review,* vol. 41, no. 3, pp. 317-329, 2014/03/01, 2014.

[14] G. Chen, J. Zhou, W. Cai, X. Lai, Z. Lin, and R. Menassa, "A framework for an automotive body assembly process design system," *Computer-Aided Design,* vol. 38, no.

5, pp. 531-539, 2006.

[15] E. Olsson, P. Funk, and N. Xiong, "Fault diagnosis in industry using sensor readings and case-based reasoning," *J. Intell. Fuzzy Syst.,* vol. 15, no. 1, pp. 41-46, 2004.

[16] J. Kim, E. Sim, and S. Jung, "An Automated Blowing Control System Using the Hybrid Concept of Case Based Reasoning and Neural Networks in Steel Industry," *Advances in Neural Networks – ISNN 2005*, Lecture Notes in Computer Science J. Wang, X.-F. Liao and Z. Yi, eds., pp. 807-812: Springer Berlin Heidelberg, 2005.

[17] E. Olsson, P. Funk, and A. Andersson, "Case-based reasoning applied to geometric measurements for decision support in manufacturing," *International Journal of System Assurance Engineering and Management,* vol. 4, no. 3, pp. 223-230, 2013/09/01, 2013.

[18] A. Aamodt, and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches," *AI Commun.,* vol. 7, no. 1, pp. 39-59, 1994.

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

[20] N. R. Draper, and H. Smith, "Fitting a straight line with least squares," *Applied Regression Analysis*, pp. 15-46: Wiley, 1998.

[21] I. Tomasic, and P. Funk, "Potential synergies between case-based reasoning and regression analysis in assembly processes," in Workshop on Synergies between CBR and Data Mining at 22nd International Conference on Case-Based Reasoning, 2014.

[22] Microsoft. "Comparison of SQL Server with Azure SQL Database," Nov, 2014; http://social.technet.microsoft.com/wiki/contents/articles/996.comparison-of-sql-server-with-azure-sql-database.aspx.