Virtual Commissioning: A Tool to Ensure Effective System Integration

Herman Vermaak[1]
[1] *RGEMS - Research Group in Evolvable and Manumation Systems, Department of Electrical, Electronics and Computer Engineering, Central University of Technology, Free State, Bloemfontein, South Africa*
hvermaak@cut.ac.za

Johan Niemann[1]
RGEMS – Research Group in Evolvable and Manumation Systems, Department of Electrical, Electronics and Computer Engineering, Central University of Technology, Free State, Bloemfontein, South Africa
Johan.Niemann@quintiles.com

***Abstract*—South African industry today need to utilise the available technology across disciplines to compete globally. One of the latest trends in the system integration field is to use virtual commissioning. Virtual commissioning allow the developer to validate the complete operation of new systems before anything materialise in the physical environment. It does not only allow operation verification but also informs the validation of the physical layout and architecture of the system in development. Virtual commissioning even allow verification of system code and rectification of design flaws. The paper will show with a suitable case study how it is possible to predict the functionality of a system with early verification of system code from your desktop. This will give companies the competitive advantage due to complete system verification and validation before any mayor capital layout.**

***Keywords*—Virtual Commissioning, Digital Factory, Smart-Devices**

I. Introduction

Virtual commissioning is the simulation of a system in a virtual environment and thus no need for the physical system during the development phase. Virtual commissioning also allows for easy reconfiguration of an existing system, where process, software or hardware changes can be made in the digital model of the system [1]. Virtualization will thus become the initial integral part of any new system development.

This system development is not limited to the hardware of a system but also the coding of these hardware devices. This allows for coding of the system before the physical hardware is installed and it give the developer the option to confirm correct operation of the code as well as the physical movement and feedback from the hardware. The aim of the project is to proof the advantages and use of virtual commissioning within the system integration environment.

II. Process to Virtual Commissioning

The process of virtual commissioning include the validation of the physical hardware and it is important to have knowledge on the behaviour and kinematics of these components. The process steps that will be explained in detail later in the paper, to achieve virtual commissioning must include the following but is not limited to these steps. First, all the characteristics of all the parts that will be used to develop the hardware need to be known including all sub-elements. This include the geometry and movements of the parts as well as any constraints [2], [3]. Then products need to be formed from these parts and sub-elements. Then smart-devices need to be developed using these products and applying logic behaviour to them. The development of the control logic is next followed by the supervisory interface.

The process from the initial identification of parts of the system to the virtual commissioning of the system will now be explained in detail making use of a case study in which "DELMIA" software was used to create a virtual environment to develop and validate a three axis assembly cell.

A. Smart-Devices

Any geometry which consists of multiple parts, or other products, is known as a product. A product is the root element of a hierarchy and contains multiple sub-elements to represent the branches of the hierarchy tree.

Fig.1 shows clearly that a collection of parts are grouped together to form an assembly [4]. Here, constraints and tasks can be allocated to the assembly, then be connected to internal device logic to form a smart-device. In a similar way, multiple smart-devices can be used together to form a work cells and ultimately an entire system [5]-[7].

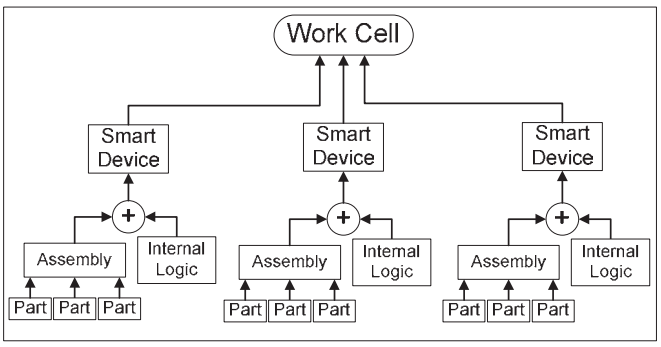

Figure 1: Hierarchy of a work cell [4]

B. Control and Device Logic

In Fig. 1 it is shown that the simplest component that can be used within a work cell is a part. Multiple parts or other products can be combined to get a product as mentioned earlier. A smart-device then get implemented using products (or assembly of products) and adding logic to it.

Smart-devices consist of geometry and internal logic (IL). The geometry of a smart-device consists of parts, assemblies and mechanisms (also called joints) [4]. There are two methods of attaining the geometry of parts and it is to download the parts from a vendors' website and alternatively, parts can be designed and created by using the suitable computer-aided design (CAD) software [8], [9]. This hierarchy of a work cell is shown in Fig. 1.

An assembly is a collection of parts that are linked together by means of specified constraints as shown in Figure 2. Assemblies need to contain at least one fixed part and can have a number of various moving parts. It is important to understand that the assemblies need to be constructed in a way to ensure the virtual part matched the physical part in all detail. The construction of these assemblies will now be explained.

Fig. 2 is explaining the process visually and it start from number 1 to number 4 in the figure. In number 1 the parts are moved into the working environment. This is the parts that are either downloaded from manufacturer or were design by the developer. The parts then need to be assigned constraints as is shown in numbers 2 and 3 of Fig. 2. Constraints include alignment, orientation of parts and surface constraints. The next step after the constraints were allocated is to update the geometry forcing the parts to move into the intended positions as is shown in number 4 of Fig. 2. Kinematical commands can be linked to the assembly and physical limits be specified that include the direction of movement, speed and the acceleration of the assembly. Through this process a mechanism (or joint as stated earlier) was created.

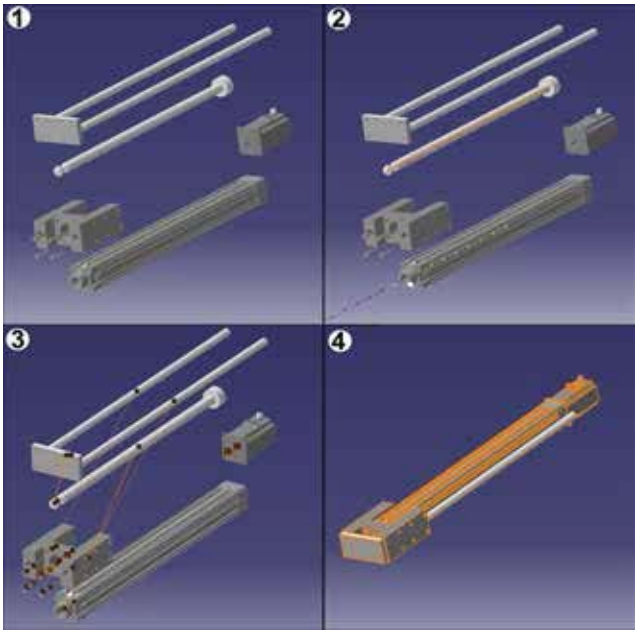

Figure 2: Assembly of parts [4]

It is possible to give these mechanism behaviour through linking tasks and operation to it. The definition of an operation is that it allows the movement of a mechanism or several mechanisms and the definition of a task is to execute a series of operations and functions. Operations allows the developer to test certain functionality needed from the mechanism. If not possible then the mechanism can be re-design to provide the needed functionality. The tasks allow the developer to create the sequence of possible activities and to teach it to the mechanism. The mechanism need to be moved to certain positions that are then recorded in a table to be saved as part of the mechanism design. All these operations and tasks are then combined to create a smart-device.

A smart-device is thus controlled by the operations of the device which in turn is controlled by the recorded tasks that will be executed sequentially. Through this process the smart-devices needed to develop the system are developed with the correct characteristics and specifications. Part of this smart-device development will be to ensure that the operation of the devices are within the working environment where the systems needs to fit in.

It is also possible to combine a number of smart-devices to form a more complex assembly or system [1]. Smart-devices can be linked in a parent-child configuration that will enable the smart-devices to move in relation to each other. Fig. 3 is showing a number of smart-devices that are imported into the working environment (number 1), they are then aligned into their positions (number 2). In number 3 in Fig.3 the smart-devices are linked in the above-mentioned parent-child configuration.

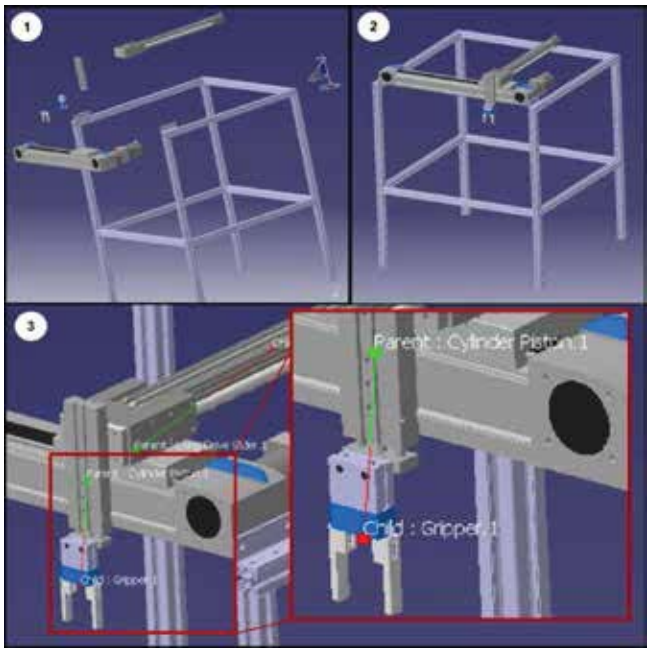

Figure 3: A completed smart-device [4]

Control logic need to be added to ensure that the developed smart-devices follows the operation and movement of the physical devices. This will also allow the developer to create a digital factory in which the smart-devices will

cooperate with each other to produce the needed functionality for the system under development. This added control are separated into control logic and device logic. Fig. 4 also shows that the control logic can be further divided into internal and external control logic. The internal logic represent a standalone control module that can be incorporated into any device. The external logic refers to any external controller (processor) linked to the system. This external controller can be a number of different controllers or processors and will be used in the physical developed system.

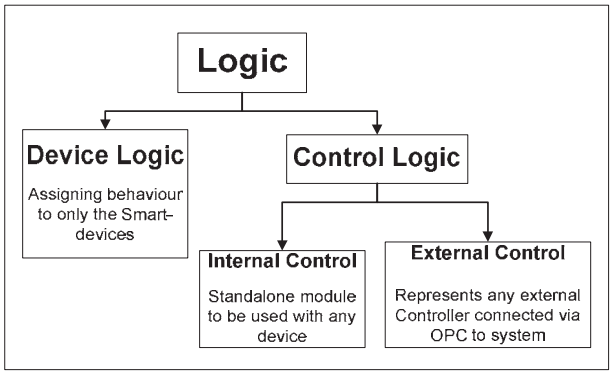

Figure 4: Device and control logic

The device logic, also called internal logic (IL), assigns unique behaviour to the device that was designed according to what was described earlier. The device actions are controlled via the inputs and outputs of each device through the IL. This process allows each device to have its own distinct behaviour as indicated in Fig.5.

The opposite is true for the control logic. The control logic is a standalone supervisory control option that allows the developer to control a number of smart-devices within the digital factory. In short, the internal control allows the virtual controller to be linked to the devices to be able to emulate the operation of the physical controller that will be used in the system.

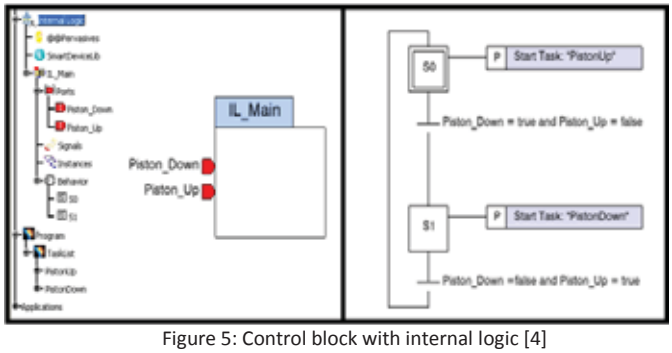

Figure 5: Control block with internal logic [4]

With the external control the developer is able to link the real controller to the created smart-devices. This enables the developer to verify the code of the real controller. In the case study used in this paper, the communication between the real controller and the smart-devices within the virtual environment was done making use of OLE for Process Control (OPC) protocol and an OPC server. The reason for using OPC is that all major manufacturers like Siemens, Allen Bradley and National Instruments are having OPC drivers available. This have the advantage that no manufacturer specific software except the OPC drivers is needed to communicate with these equipment.

III. Implementation and Validation

The process to obtain virtual commissioning can better be explained by referring to Fig. 6. The development steps towards virtual commissioning are summarised in Fig. 6. The initial step is to ensure all parts needed for the system are designed or obtained from the manufacturers. Then all combinations of assemblies and smart-devices need to be created. The devices need to provide all operations and tasks to ensure all behaviours of the system are available. The following step is the validation that all mechanical operation is able by combining the device or making complex assemblies of devices.

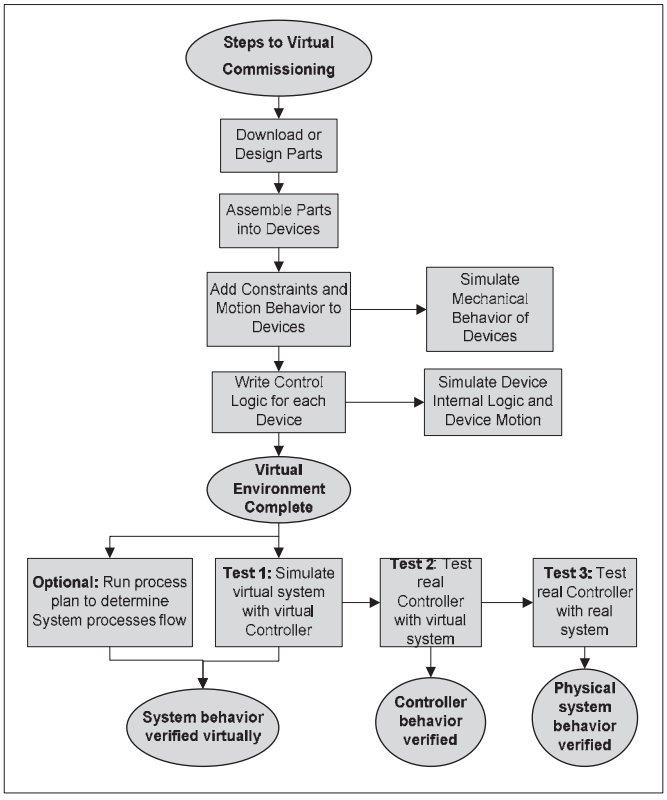

Figure 6: Steps to virtual commissioning

The final steps is to develop the logic for the virtual environment. The logic is an indication of the intelligence incorporated in each device. Now the geometry of all devices need to be validated to ensure similar operation and movement limitations as the physical components. Now all smart-devices are incorporated into the virtual environment to establish the digital factory. The environment is now set up for virtual commissioning of the developed system [5], [10].

A number of test procedures was developed within the case study environment to assess the ability of virtual commissioning to aid to the validation of a system and the verification of its controller code.

The first test procedure is aimed to validate the virtual system (digital factory developed) along with the logic through a simulation exercise. This is very important as it will indicate how the virtual system will operate with the current code in the controller. This will however only be able to validate the actual behaviour (smart-device operations and logic) of the virtual system.

The next test procedure will verify the correct operation of the current code in the real (physical) controller. The communication between the virtual system and the real controller will be done by making use of OPC and the OPC server. This communication process was explained earlier. The simulations will be carried out until all operations and functionality of the real controller code is verified successfully. This process will form part of the re-design if the code in the real controller need modification to be verified successfully. This will conclude the process of virtual commissioning.

The final test procedure is now to connect the physical system to the verified controller and test all operations and functionality of the real system. It is now important to compare the operation of the system with the predictions made in the virtual commissioning phase. This process will also be an indication then if the virtual commissioning process was successful in the development of the system. The evaluation of the virtual commissioning to successfully predicted and validate the operation of the physical system is now complete [11]-[14].

The next section will explain and show how this test procedures were carried out in the case study. It will indicate how the simulation were set up and executed. It is important to note that the real controller in the case study is called a Programmable Logic controller (PLC) and the three-axis hardware a Cartesian Robot (CR).

A. Validation Environment Setup

To be able to execute the test procedures the system validation architecture as shown in Fig. 7 need to be combined. The setup consist of the real controller called the "System PLC" in Fig. 7 linked to the OPC server (this is actually a normal Personal Computer (PC)) and linked to the PC running the emulation software. The system PLC is linked to the real CR and the PC with the emulation software is linked to the virtual CR. The OPC server and PC running the emulation software can be set up on the same PC to minimise the cost. The server and PLC are connected via an Ethernet connection to establish bi-directional communication. This enable the server to provide the system with all the needed information as well as the input/output (IO) status from the PLC controller.

The virtual CR need to be developed as was explain earlier including the logic and all behaviour functionality which need to be virtually validated. The physical CR need to be built and all functionality tested in comparison with operation and limitations used when developing the virtual CR. The control and feedback from the physical CR will be via the PLC that controls and monitor the CR's operation.

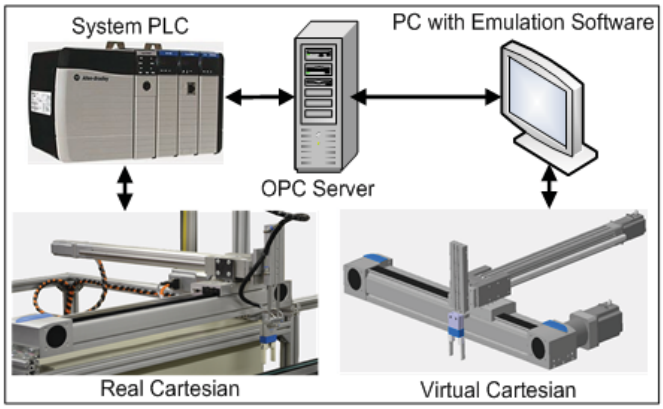

Figure 7: System validation architecture [4]

B. Environment Conditions for Validation Process

In the previous paragraph all needed component for the validation environment were explained. The conditions under which the validation of the case study will take place will now be explained.

The most important condition is that the virtual system need to be tested and simulated to get perfect operation and behaviour. This include the need to record all limitations and constraints from the different incorporated developed smart-devices. This will now ensure that a validated virtual smart work cell is available. The same need to be done with the physical system. Numerous control sequences need to be executed by the PLC to validate that the operation of the physical CR is matched to the virtual CR and vice versa. The validation environment now has a matched virtual and physical system.

The simulations will now be executed in two processes. The first process is running the simulations with the virtual CR connected via OPC to the real PLC controller. The second process is when the physical CR is linked to the virtual controller. In the first process the I/O of the PLC are monitored to ensure proper operation from the virtual CR in relation with the I/O signals. Then the physical CR will be monitored to ensure correct operation in relation to the virtual controller. The setup to establish these conditions are shown in Fig. 8.

The mapping is showing the connection of the PLC I/O to the virtual system exactly the same as the physical wiring of the real CR to the PLC. When all this is done the complete environment set up will be compiled and used to run continues the process. All possible operations and functionality need to be noted before the compiled system is executed. During the running of the compiled environment all operation and functionality identified before was monitored and validated. No faults were noted and the environment was verified as successful. This was successful and the developed system operated as to the initial specifications given at start of development process. The next

steps was to disconnect all connections to the virtual systems and connect the physical CR direct to the PLC and start using the physical PLC controlled CR.

If for some reason all operations and functionality was not verified then the process need to go back to phase that can correct the fault either the creating of the smart-devices or the adding of the logic.

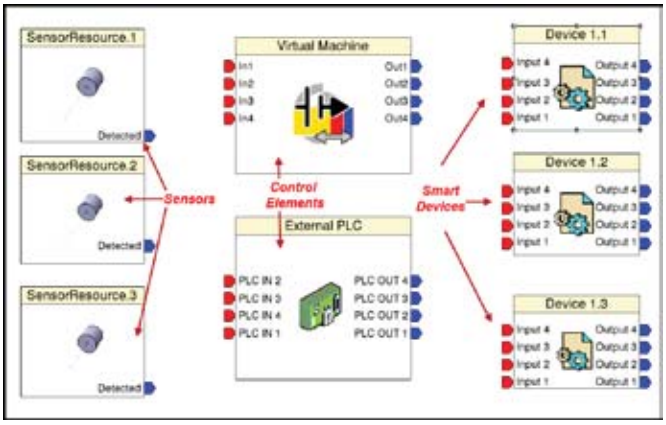

Figure 8: Mapping devices to controller [4]

IV. Validation Results Discussion

The results obtained from the case study will now be discussed and solution implemented explained. The first part of the validation provided a problem with the control of the one axis of the physical CR. The axis did move with inconsistent speed for some reason. It was investigated and it was found that the physical axis did move slower than the virtual designed axis. This had the result that the virtual controller did assume that the physical axis did reach the destination before it actually did. The limitations of the specific smart-devices (created for the axis) was adjusted to accommodate the slower movement of the axis. An extra sensor was also added to indicate once the axis did reach the end stops to ensure no damage to axis when moving to the ends of its reach. This then did go through the validation process and proof to be successful in solving the problem. The rest of the system either virtual or physical produce no other errors.

V. Conclusion

The results obtained from the validation of the case study proof that virtual commissioning is possible and that it also contribute to the optimization of a system cycle times. The optimization aspects was not part of the initial specifications of the system. However during the simulation of the physical system when controlled by the virtual controller it was realised that the operation of the physical system can be optimised by optimising the operation and tasks of the smart-device.

The case study used in this paper showed the advantages of using virtual commissioning includes but are not limited to effective planning, no capital layout before system is verified and limit the risk of damage to real equipment before process and movement of equipment were validated. It also highlighted the added advantage of optimising the system processes and cycle time through virtual commissioning.

Virtual commissioning gives the developer the opportunity to see the operation and functionality of the system and be able to modify or change anything without any capital cost. This will assist the system developer to show a demonstration of the proposed system to the client before starting with the physical system and allow the client to propose changes if the systems operation and functionality is not what the client expected or need.

Virtual commissioning give system developers the opportunity to do proper planning, system integration and code verification before having any physical equipment. Then the developer can validate the physical equipment and verify the code of the controller before connecting the controller to the physical hardware. Virtual commissioning allow the developer to determine if it is actually profitable to build a system—thus build it right the first time or not at all [4]. This will enable system builders and integrators too shorten ramp-up time for system development and be competitive in the global market.

Further development in the virtual commissioning field can look at the automated generation of system controller code from the virtual controller. This will need the code to be in line with a specific standard for coding as is used in the PLC programming environment where the IEC 61131 standard is applicable.

There are other software packages on the market that are also able in doing virtual commissioning but the aim of the project and this paper, as stated earlier, was not to promote “DELMIA” software but to proof the advantages and use of virtual commissioning as stated in the introduction.

Acknowledgments

The authors would like to thank the Central University of Technology Free State and AMTS under the project 07-11-P funding agreement; for the opportunity to conduct research on this topic.

References

[1] H. Krause, "Virtual commissioning of a large LNG plant with the DCS 800XA by ABB," in 6th EUROSIM Congress on Modelling and Simulation, Ljubljana, Slovénie, 2007.

[2] H. Carlsson, B. Svensson, F. Danielsson, and B. Lennartson, "Methods for reliable simulation-based PLC code verification," Industrial Informatics, IEEE Transactions on, vol. 8, pp. 267-278, 2012.

[3] Kepware Technologies - OPC Servers / Communications for Automation, Available Online: http://www.kepware.com, last accessed in August 2013.

[4] H. Vermaak and J. Niemann, "Validating a Reconfigurable Assembly System utilizing Virtual commissioning," in proceeding of the Pattern Recognition Association of South Africa and the Robotics and Mechatronics International Conference, 2015, Port Elizabeth. South Africa: ISBN: 978-1-4673-7449-1, pp 258 - 262

[5] J. Niemann, "Development of a Reconfigurable Assembly System with Enhanced Control Capabilities and Virtual Commissioning,"

Master Dissertation, Faculty of Engineering and Information Technology, Central University of Technology, Free State, 2013.
[6] 3DS - Dassault Systemes, Available Online: http://www.3ds.com/products/delmia, last accessed in August 2013
[7] S. C. Park, "A methodology for creating a virtual model for a flexible manufacturing system," *Computers in Industry,* vol. 56, pp. 734-746, 2005.
[8] O. Salamon and A. Heidari, "Virtual commissioning of an existing manufacturing cell at Volvo Car Corporation using DELMIA V6," Master Thesis, Department of Signals and Systems, Chalmers University of Technology, Sweden, 2012.
[9] CATIA Online Documentation, Available Online: http://catiadoc.free.fr/online/CATIAfr_C2/prtugCATIAfrs.htm, last accessed in August 2013.
[10] D. Cachapa, A. Colombo, M. Feike, and A. Bepperling, "An approach for integrating real and virtual production automation devices applying the service-oriented architecture paradigm," in Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, 2007, pp. 309-314.
[11] Z. Liu, C. Diedrich, and N. Suchold, *Virtual Commissioning of Automated Systems*: INTECH Open Access Publisher, 2012.
[12] X. Kong, B. Ahmad, R. Harrison, Y. Park, and L. J. Lee, "Direct deployment of component-based automation systems," in Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on, 2012, pp. 1-4.
[13] C. G. Lee and S. C. Park, "Survey on the virtual commissioning of manufacturing systems," Journal of Computational Design and Engineering, vol. 1, pp. 213-222, 2014.
[14] P. Hoffmann, R. Schumann, T. M. Maksoud, and G. C. Premier, "Virtual Commissioning Of Manufacturing Systems A Review And New Approaches For Simplification," in ECMS, 2010, pp. 175-181.