Accepted Manuscript

A model-based approach for design and verification of Industrial Internet of Things

Muthukumar N., Seshadhri Srinivasan, K. Ramkumar, Deepak Pal, Juri Vain, Srini Ramaswamy

PII: S0167-739X(18)32113-7
DOI: https://doi.org/10.1016/j.future.2018.12.012
Reference: FUTURE 4633

To appear in: *Future Generation Computer Systems*

Received date : 5 September 2018
Accepted date : 7 December 2018

Please cite this article as: Muthukumar N., S. Srinivasan, K. Ramkumar et al., A model-based approach for design and verification of Industrial Internet of Things, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2018.12.012

A Model-Based Approach for Design and Verification of Industrial Internet of Things

Abstract

This investigation presents an Industrial Internet of Things (IIoT) architecture and a Model-Based Engineering (MBE) approach for design, verification, and auto-code generation of control applications in process industries. The IIoT architecture describes the hardware components, communication modules, and software. It emerges as a major enabler for providing open connectivity to process industry which provides greater data-aggregation, visibility, availability, flexible control, and cloud-connectivity. The MBE approach is based on multiple views of the systems with each domain model describing a particular view. The multi-view modelling approach is used to perform design and verification of the IIoT enabled control in process industries. We show that such an integration of MBE, cloud-computing, and IIoT provides certain desirable features such as plug-and-play control and on-the fly verification which are lacking in the process industry. The proposed MBE approach and IIoT architecture are illustrated on the quadruple tank process, a benchmark problem in control. Our deployment results verify the benefits envisaged by IIoT, cloud, and MBE integration.

Keywords: Industrial Internet of Things (IIoT), Model-Based Engineering (MBE), Verification, process industries, IoT architecture.

1. Introduction

Combining the Internet of Things (IoT) with cloud intrinsic capabilities can transform the way industrial automation systems are designed, deployed and managed currently in process industries [1, 2, 3]. While the cloud offers capabilities such as virtualization, scalability, lifecycle management, and multi-tenancy, the IoT compliments it using its open connectivity and emergent computing environments (e.g., fog computing). In addition, the cloud offers attractive delivery models such as software-as-a-service, platform-as-a-service, and infrastructure-as-a-service with different deployment models. Consequently, many desirable features such as increased flexibility, adaptability, data-visualization, enterprise-wide communication, intelligence, and agility can be realized on low-power electronic devices. Besides, the cloud can host a variety of auxiliary services that can enhance the automation capabilities and promote smart manufacturing. Many recent investigations have stressed the need for transforming the cloud-IoT integration to deployment (see, [4, 5, 6] and references therein). However, industrial automation lacks engineering approaches and tools to accomplish this integration. Moreover, their deployment posses strict challenges due to hardware limitations of the IoT components such as real-time performance, reliability, and safety. Arguably, the IoT-based devices cannot fully substitute the legacy automation systems, but they can be deployed in tandem with them to perform specific/specialized tasks. This requires frameworks that consider both legacy and IoT devices in one framework.

Engineering industrial automation systems has been focus of many investigations and methods based on component-based [7], formal models [8], agent-based [9], service-oriented architecture (SoA) [10], design patterns [11] and Model-Based Engineering (MBE) [12] have been proposed. Vyatkin [13] provides a good review on these approaches. Notwithstanding these developments, the automation software complexity and the functionalities realized using them have grown steadily. This, the industries discern, will increase the design, validation and verification costs significantly. Moreover, design upgrades and post design validations are proving costlier. In this backdrop, the Model-Based Engineering (MBE), an approach using models to design software and perform component testing emerges as a promising solution. As they automate the design process through auto-code generation capabilities. Further, design validation can be performed early during the life-cycle. The use of MBE approach for code-generation in legacy industrial automation systems has been studied in [14]. Similarly, to handle the complexity of industrial automation system with entangled behaviours from various domains, artifacts, and interactions, multi-domain models have been studied in [15]. As for Industrial IoT, an UML (Unified Modelling Language) profile for IoT in manufacturing industry was presented in [16]. The use of semantic technologies adding meaning to machine-to-machine communication using ontologies of interlinked terms, concepts, relationships and entities was investigated in the context of IIoT in [17]. These investigations either model legacy systems or IoT systems without involving cloud features.

More recently, combining cloud-intrinsic features with IIoT for providing enterprise-wide connectivity has been studied in [18, 19]. The IMC-AESOP project [20] extended the engineering methods based on Object

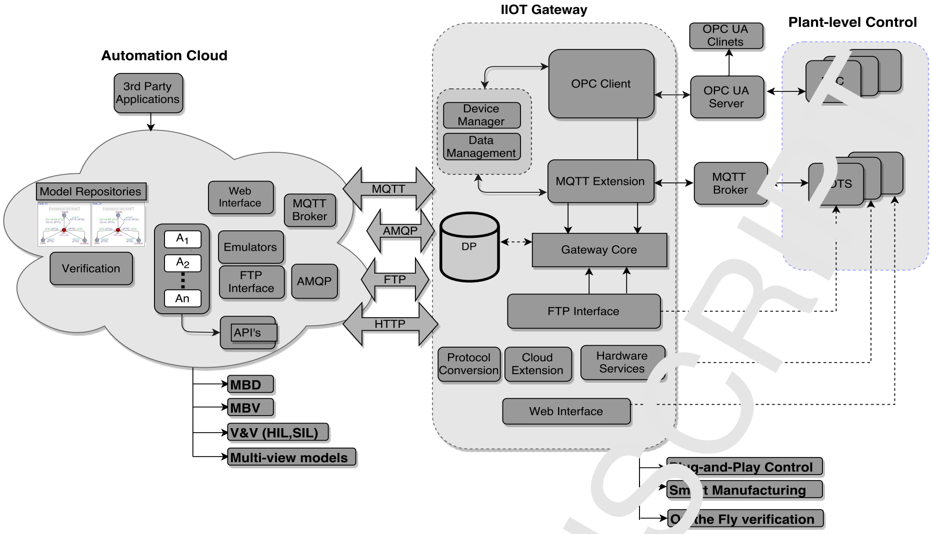

Figure 1: Proposed IIoT Architecture

Oriented and Aspect Oriented approach to industrial automation [21] using formal modelling extensions. Similarly, the use of agent-based approaches for cloud integrated IoT systems was studied in [22]. However, the use of MBE approach for cloud-based IIoT starting from the model to the deployment is currently not available to our best knowledge.

This investigation addresses this research gap by proposing a multi-view model of industrial automation and an MBE approach for design and verification of cloud-based IIoT implementations in process industries. The main contributions of this investigation are: *(i)* An IIoT architecture that promotes cloud-based engineering of the process control applications, *(ii)* Multi-view models for industrial automation systems in process industries that include various participating domains, artifacts, and interactions, *(iii)* A MBE approach for designing and verifying cloud-based IIoT, *(iv)* a workflow for performing Model-Based Design (MBD) and verification in emergent IIoT paradigm to realize sophisticated controllers, e.g., model predictive controller [23], *(v)* Present the advantages of the proposed architecture to perform plug-and-play control, on-the-fly verification, and smart manufacturing, and *(vi)* Demonstrate the MBE approach on a quadruple tank process applications.

The paper is organized into six sections. Section II, presents the IIoT architecture and the MBE approach is discussed in Section III. The cloud enabled flexibilities are discussed in Section IV. Section V presents the deployment results of the IIoT. Conclusions and future course of investigation are discussed in Section VI.

2. Proposed IIoT Architecture

The architecture that enables MBE for cloud-based IIoT is shown in Fig. 1. It consists of three major blocks: plant-level automation, the IIoT gateway, and the automation cloud. The plant-level automation consists of conventional *Programmable Logic Controllers* (PLCs) and IoT based commercial-of-the-shelf (COTS) target platform. The PLC interfaces to the sensors using conventional industrial protocols (e.g., Modbus). While the COTS platform uses TCP based protocols such as Message Queue Telemetry Transport (MQTT) or Advanced Message Queuing Protocol (AMQP), wireless and other forms of dedicated communication (e.g., I2C) to interface the field devices. A gateway is used to communicate with COTS target platform and legacy protocols with an incompatible physical layer (e.g., Profibus PA). OPC UA is used for aggregating information from the conventional PLCs and field devices due to its prevalence in the automation industry. Further, its security and platform independence makes it a good choice for the IIoT.

The IIoT gateway has interfaces on one side to the plant-level automation, and on the other to the cloud. The IIoT core is the main component of the IIoT gateway that orchestrates different protocols, devices, applications and software routines. It collects data from OPC UA using a client and transfers to other devices using MQTT or AMQP extensions. The MQTT extensions, (i.e., services) are used to collect information from the MQTT broker (an entity that supplies information to all devices subscribing to it). The OPC UA client and MQTT extension perform both device and data management within the IIoT gateway. The FTP, web interfaces and web applications are used to communicate to plant-level devices and cloud. The IIoT gateway provides extensions for the cloud and hardware devices, data persistence (DP) for securing data delivery in events of communication failures, and a secured FTP for enhancing the application security. Here it should be clarified that the MQTT and AMQP are shown as communication links only for illustrative purposes of this investigation. The IIoT gateway can be used for other protocols as well with suitable modification.

The cloud-intrinsic features— DP, virtualization, communication interfaces, multi-tenancy, auxiliary application support and others are offered by the automation cloud. The cloud offers virtualization through model repositories and emulators. The model repositories consist of the processes and controller instances, topology, behaviourial models of the devices and all other aspects required for performing MBE. Employing the communication interfaces, the cloud talks to the IIoT gateway through the MQTT and AMQP. To compliment the IIoT gateway, the cloud has FTP, HTTP and external interfaces for enabling file transfer, web applications and using third-party applications. The IIoT architecture simplifies the communication between legacy devices and IoT devices in the plant-floor and enables open connectivity between plant-floor and cloud. Therefore, the architecture promotes the implementation of the cloud-based IIoT.

3. Model Driven Engineering for Industrial Internet of Things

With the emergence of IIoT, the heterogeneity and networking capability of the hardware, and the proportion of system functionality realized using software has increased stupendously leading to an increase in the design space. Coupled with these developments, market influences requiring smart and flexible manufacturing are obligating a more flexible automation that provides upgrades/modifications with minimum engineering effort. As stated earlier, the MBE approach is more suitable in such scenarios as it raises the abstraction levels and automates the labour-intensive and error-prone tasks in the design, e.g., code-development [13]. This not only brings down the design cost, but enhances reusability, efficient data exchange, and verifiability of the system. Above all, the MBE promotes MBD and Model-Based Verification (MBV). Using these methods the design, validation and verification can be automated to a greater extent even from the cloud. However, the model of the industrial automation system by itself is complex due to the interaction of multiple domains and heterogeneous entities. There is a lack of tools, formalisms and semantics capable of incorporating semantic relations among the disciplines. Developing a meta-model encapsulating all aspects of an industrial automation system is rather difficult. More recently, the use of multiple views for industrial automation systems have been investigated for industrial production units [24]. This investigation uses the multi-view modelling approach for performing MBE for cloud-based IIoT solutions.

3.1. Multi-View Model for IIoT

Multiple views model is an emerging concept for building complex systems wherein different stakeholder's viewpoints are captured as domain models or *views* [25]. The multi-view model of cloud based IIoT has different but entangled views—devices, architecture, information, software, control, domains, behaviour and others. These different viewpoints need to be considered simultaneously for engineering IIoT systems. Consequently, system integration emerges as a key challenge due to potential contradictions or overlapping information among the views. Therefore model transformations and mapping are required for engineering systems with multiple views. This investigation uses a meta-modelling approach for capturing the different views.

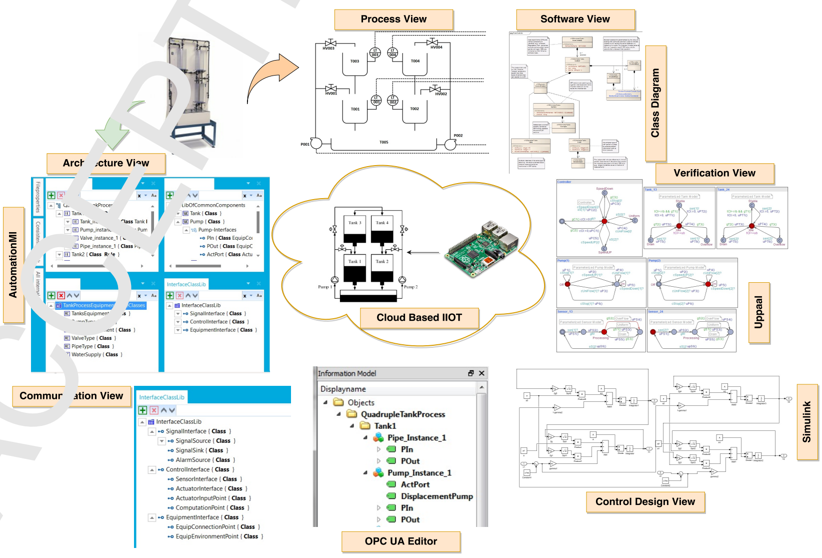

Figure 2: Multi-View model of IIoT

The multi-view model of the industrial automation system and the different tools for obtaining these views are shown in Fig. 2. To integrate these different views, this investigation uses the AutomationML (Automation Markup Language) [1] (AML) for providing the topology view and uses it as a meta-model of the IIoT based automation systems [26]. The process industries with its various process stations are modelled in the AML using suitable abstractions. The AML provides XML/CAEX (Computer Aided Engineering Exchange) formats for the topology view and in addition provides the communication view through the *InterfaceLibraryClass*, wherein additional interfaces specific to IIoT are defined. The process views are obtained from the P and ID diagram, the controller design is modelled in Simulink using state-space/transition formalisms, the OPC UA provides the information models, the software design is modelled using UML and behavioural models based on state-charts. In addition to these models, there can be domain views that capture the formalisms and artefacts of the different domains (electrical, mechanical) modelled using suitable software tools, e.g., Dymola. The multi-view model forms the basis on which the MBD and MBV are performed.

3.2. Workflow for Model-Based Design and Verification

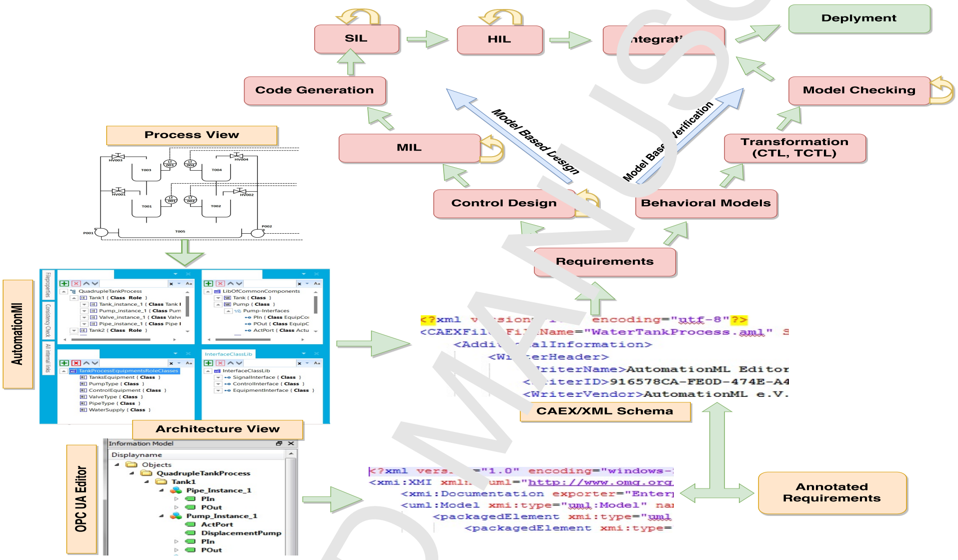

Figure 3: Workflow for MBE based Design of IIoT

The workflow for performing MBD and MBV from multi-view models is illustrated in Fig. 3. The P and ID's process view defined in the IEC 62424 standard is used as the starting point. It has three basic concepts: process control engineering request (PCE-R), process control engineering function (PCE-f), and process control loop (optional). The PCE-R defines the requirements of the process control equipment. The PCE-R collects all information about the functional requirements. PCE-R and its unique ID are important specifications for the requirements diagram.

The AML model uses the PCE-R to create a meta-model of the entire process that can be later used to map different models. The ability to produce neutral XML/CAEX schema makes AML a suitable tool for information exchange between engineering applications. The *InstanceHierarchy* represents the entire automation project and it has the child nodes called the *InternalElements* that hold the attributes of the different properties of the object and have objects that hold the attributes need to describe them. Here, process control loop implies the unitary process description, e.g., level control of the tank. Each object in the *InternalElements* is associated with a *RoleClassLibrary* that provides the functional view of the object, an useful aspect for semantic classification. In addition, there is the *SystemUnitClassLibrary* defining the specific aspects of the process control application, e.g., height of the specific tank.

The communication interfaces are modelled using the AML basic *InterfaceLibrary* which is extended using four additional classes for the IIoT applications: *IIoTEndPoint, CloudEndPoint, ProcessEndPoint*, and *LogicalConnectionEndPoint*. The *IIoTEndPoint* contains special plugs to model Ethernet-based connectivity of IIoT based TCP/IP, MQTT, AMQP, RS 232, Modbus, and other communication available with the COTS target platform and device. The *CloudEndpoint* defines the interfaces for cloud communication such as the FTP and HTTP services, AMQP and MQTT for data-transfer. The *ProcessEndPoint* defines the traditional connectivity

[1] https://www.automationml.org/

with non TCP/IP based protocols such as the Profibus using the gateway that delivers TCP/IP messages to the IIoT. The *LogicalConnectionEndpoints* model the communication between PLC and IIoT, master-slave, Bus, etc. In addition, we define *CommunicationRules* that enforce logically correct connectivity, e.g., MQTT to TCP/IP based devices. These interfaces model the physical interconnection of the components. The annotated AML CAEX schema thus generated represents the multi-view model of the automation systems. The COTS target platform is mapped as resources to specific process control instances using UML models. The multi-view models and the annotated requirements are the input to the model-based design and verification steps.

The CAEX schema is then annotated with additional user-defined requirements either using UML models or textual representations. The requirements are generated for both the design and verification. The MBD and MBV approach used to design and verify automation systems will be detailed in the next sections. Of particular interest to this investigation is the design of MPC from the requirements. As MPC is emerging as a workhorse for smart manufacturing and one of the sophisticated control algorithms executed in process industry.

3.3. Model Based Design for Cloud-Based IIoT

The PLCs are the processing units for performing control in process industries and they are programmed using IEC 61131 standard. The role of MBD approach for auto-generating code for IEC 61131 based process automation has been studied in literature [14]. The PLCs are bit inflexible due to real-time requirements and sophisticated controllers such as MPCs are usually implemented on dedicated hardware platforms. Even in literature, the MPC implementation on PLC is quite scarce. With the emergence of IIoT, these sophisticated controllers can be realized in IIoT hardware and executing conventional control in traditional PLC systems. In this scenario, the MBD approach should be able to automate the code generation of sophisticated control schemes such as MPC. Therefore, for the rest of the section, we focus on the auto-code generation for MPC, rather than executing simple logics or control actions such as Proportional Integral Derivative control. This brings down the cost and development time significantly.

The workflow for performing MBD-based design for cloud integrated IIoT is shown in Fig. 4 and it follows the V-model. It has four validation stages: Model-in-the-loop (MIL), software-in-the-loop (SIL), processor-in-the-loop (PIL) and Hardware-in-the-loop (HIL), before actual deployment. The real-time performance is validated through these different steps, an important requirement for process control applications. In the design flow requirements in the form of objective function (e.g., track a reference signal with minimum energy) and constraints (e.g., the maximum voltage of a pump) are fed as the requirements to the control design. The MPC parameters are computed based on the requirements using the design equations, (refer Appendix A for MPC models). Then both controller and process are simulated in a virtual environment to verify the control design, the procedure is called MIL. The model used is called platform independent model (PIM).

Following MIL, the target platform is identified, and the software code for the specific target is generated using an auto-coder. This model is called PDM, and the software emitted by the auto-coder is used to run the SIL, wherein the platform dependent software code and process models are simulated in virtual environment. This validation procedure tests the software code. The SIL code is ported to the target hardware, and tested on the virtual process with sensor and actuator models in the PIL validation, verifies the hardware capabilities, e.g., sampling time. Finally, the controller code working on the target hardware is interfaced to the sensors and it controls the virtual model of the process in the HIL. The validations are iterative procedures and design changes can be made based on the results. A controller design successfully validated in the four tests is deployed in the process industry with the control action performed by the target hardware. Two important observations here are:

1. There are not many auto-coders available for MPCs as they involve optimization solvers. These solvers face numerical accuracy, computational complexity and other numerical issues. This investigation used the jMPC[2], a MATLAB based toolbox for auto-code generation for MPCs.

2. Combining the virtualization and multi-tenancy capabilities of cloud, when emulators of the specific hardware are in the cloud, then MBD can be performed from the cloud and the solution can be deployed in process industries.

3.4. Model Based Verification of IIoT

To perform model-based verification, the formal requirement specifications generated by AML (XML/CAEX schema) are mapped to abstract behavioral models (networks of timed automata). The automata model of the system under verification describes how the system is required to behave. The model, built in a suitable machine interpretable formalism is fed to model checker which verifies the model w.r.t properties of the specification. There are multiple different formalisms used for building formal requirement models. Our choice is Uppaal

[2]http://www.i2c2.aut.ac.nz/Resources/Software/jMPCToolbox.html

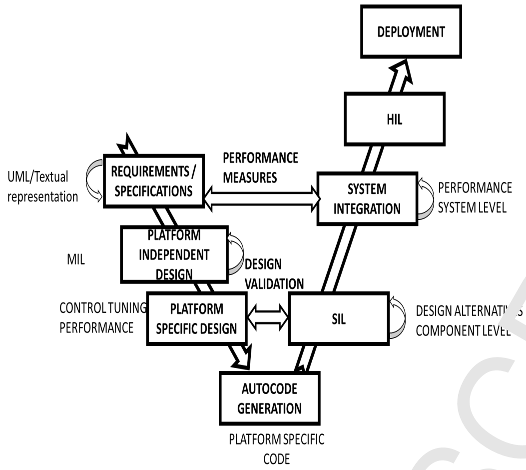

Figure 4: MBD Workflow

timed automata (UTA) [27] because the formalism is designed to express the timed behavior of state transition systems and it has been previously been applied successfully to verify industrial automation systems in [28].

In the second step, the model templates for the component models are defined. The component models are modelled using UTA templates. The timed-action pattern shown in Fig. 5 is used to model the requirement specification following [29]. These are called the action patterns and timing wrapper have been presented in [28] for industrial automation systems. The timing patterns are interlaced with the component models of the process industry, e.g., pump. The component models with their timing interfaces for a quadruple tank process is illustrated in Fig. 6. A detailed discussion on the models is presented in the results section.

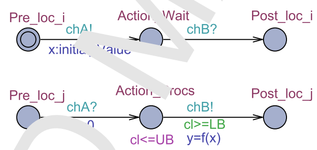

Figure 5: A Synchronous-Parallel Composition of Time Action Pattern cf[29]

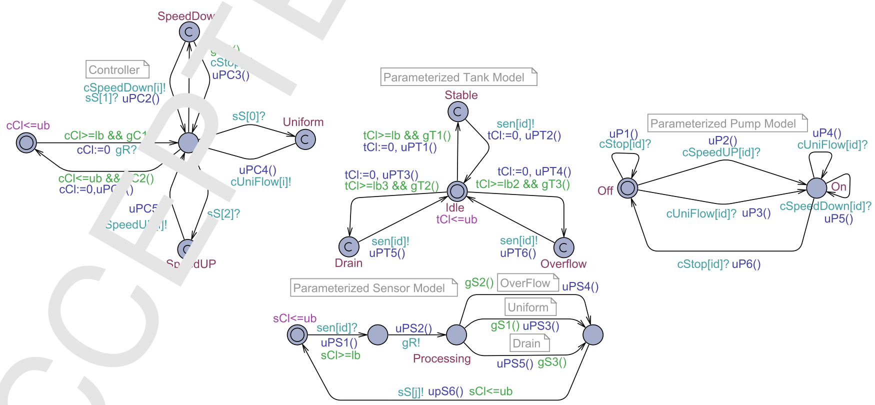

Figure 6: Parameterized Models of Quadruple WaterTank Process

In the third step, the model checker is used to verify the formal model w.r.t to a requirement specifications (properties). Like the model, the properties are expressed in a formal well-defined logics such as subset of CTL (computation tree logic) as in [29]. The CTL offers several temporal operators to express the requirements as CTL formulae can be classified by properties they express as reachability, safety and liveness, detailed analysis of these properties are provided in [29].

4. Cloud-Intrinsic Features for Enabling Flexibility in IIoT

This section highlights the opportunities in enhancing the performance of industrial automation by combining cloud capabilities with IIoT. In particular, three cases are considered: *(i)* plug-and-play control, *(ii)* smart manufacturing, and *(iii)* on-the fly verification.

4.1. Plug-and-Play Control

Vast control designs in industries are monolithic, i.e., entire control system needs to be changed, when a sub-system or hardware modifications are performed. As flexibility is emerging as a key requirement, control objectives of the plant change with time or even within production processes. In such scenarios, it is desirable to change control laws without diminishing existing controllers. The IIoT provides a way to flexibly change control algorithms using cloud services. The workflow for performing plug-and-play control is shown in Fig. 7. To guarantee security of the applications, the file transfers for the plug-and-play control happens using secure FTP, while for less important actions using FTP. In these scenarios, the user informs the cloud through a web

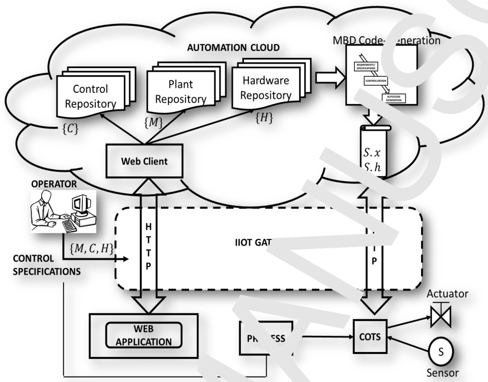

Figure 7: MBD Based Plug-and-Play Control for IIoT

interface about the changes. This is transmitted using HTTP interface of the IIoT gateway to the cloud. The cloud's HTTP interface receives this request. There are three components in the request, process, controller and hardware specification. The process model informs which of the process loop requires an upgrade, the controller instance/requirements, and the target hardware. The cloud then instantiate the virtual environment to obtain the process model, requirements for the specific controller, and controller design in PIM. It generates the PSM based on the controller specified by the user and ports it into the emulator and validates the design. Once the validation tests are successful, the control code is transmitted via IIoT gateway's FTP interface to the COTS embedded controller of the controlled process. Now, the control code is deployed on the hardware.

4.2. Smart Manufacturing using Cloud's Auxiliary Services

Computing power of IoT devices restricts their applications to perform computationally intensive task and is a major hindrance in the deployment of IIoT. Typically in smart manufacturing, data-mining models are used for creating knowledge from raw-data, both intrinsic and extrinsic to process industries. For example, forecasts on energy prices can be obtained using the data mining model and then integrated with optimization routines to perform smart manufacturing. Such data-mining models requires large memory for storing data and execution. They can also be available as third party applications as APIs. Exploiting the cloud features, the data-mining algorithms can be implemented in the cloud and knowledge aggregated can be transferred to the process controller using IIoT gateway using the MQTT and AMQP interfaces. Such aggregated knowledge can be embedded in the MPC controller for making knowledge based and optimization driven decisions.

4.3. On-the Fly Verification in the cloud

When hardware like sensor or actuators are updated, generally the control loop's timing performance is changed and the controller implementation needs to be modified as sampling and quantization levels have changed. If the devices match, such a scenario may not arise. But, the problem is faced with most legacy automation systems. When a sensor or actuator different from the one used is changed, the performance of the IIoT has to be verified. In our IIoT framework, the model templates (behaviour models) of the different components and their timing interfaces are available in the cloud's model repository. In case, a particular specification is unavailable for a model template, it is obtained from the field using a web application or FTP. These model templates are then composed and the MBV workflow is implemented from bottom to check whether the timing or safety requirements are met. This allows dynamic configuration of components in IIoT.

5. Results

5.1. Case Study: Quadruple Tank Process

To illustrate the MBE approach for cloud-based IIoT, this investigation uses the quadruple tank process (QTP), a benchmark control problem in process control. The schematic of the QTP and its prototype used for deployment of IIoT is shown in Fig. 8. The QTP consists of four uniform sized cylindrical tanks with cross-sectional area A and outlet cross-sectional area a. In addition, there are two identical pumps namely Pump 1 and Pump 2. Four valves namely, HV 1, HV 2, HV 3 and HV 4 are provided to regulate the inlet liquid flow to the tanks. The objective of the low-level control is to maintain the liquid level in Tank 1 (h_1) and Tank 2 (h_2) at pre-defined value called the reference by varying the flow rate ([illegible]) of Pump 1 and Pump 2, by adjusting their supply voltage V_1 and V_2, respectively. The equations modelling the dynamics of QTP are given in Appendix B. To illustrate the use of the proposed approach, three use-cases are presented here: *(i)* Model-

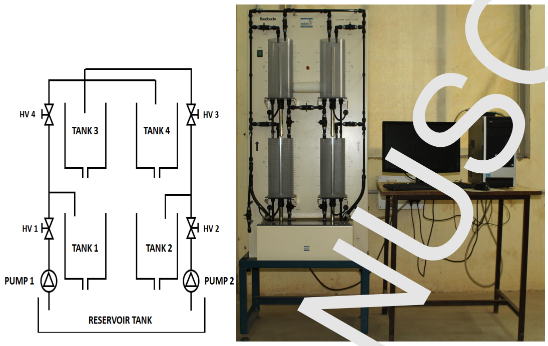

Figure 8: Schematic and the Process Station of the Quadruple Tank Process

Based Design, *(ii)* plug-and-play control, and *(iii)* Model-Based Verification. The Raspberry PI 3 was chosen as the target hardware for our experiments.

5.2. Use-case: 1 Model Based Deign

The MBD approach was used to design four different MPCs $\mathcal{M}_1 - \mathcal{M}_4$ described in Appendix A. The MBD workflow shown in Fig. was used to generate the auto-code using jMPC toolbox for the target embedded platform, Raspberry PI 3 in our case. The requirements were generated from the multi-view model of the QTP generated as shown in Fig. 2. The requirements are: offset-free tracking, faster response time (rise time) and settling time for the levels in the tank, i.e., h_1 and h_2. With MBD, the four MPC models $\mathcal{M}_1 - \mathcal{M}_4$ were studied. Our results showed that $\mathcal{M}_1$ met the design requirements and it was validated using MIL, HIL, and real-time deployment. The controller was deployed on the target hardware and it was used to control the process. The results of MIL, HIL and real-time deployment are shown in Fig. 9. While the MIL and HIL validated the results, small pulsations in the output are seen due to sensor noise from the environment that impacts the process performance. The other MPCs $\mathcal{M}_2$- $\mathcal{M}_4$ did not meet the requirements that were identified either during MIL, SIL or HIL. This results shows the ability of MBD approach to generate auto-code from requirements for even sophisticated controller such as MPC and to detect design issues early during the design phase, eliminating costly design upgrades later. It should be pointed here that using the emulator stack, the MBD approach can be performed in the cloud as well, thereby enabling cloud-based engineering of the solution.

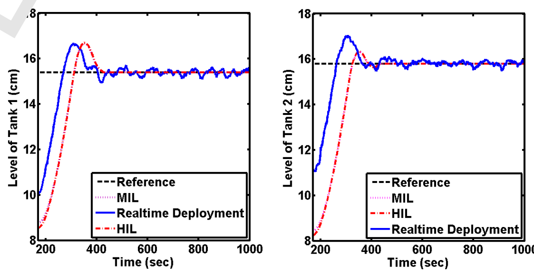

Figure 9: MIL, HIL and SIL Validation for M1

Table 1: Execution time of auto-generated code and CVXGEN code for the MPC models for 25 iterations

MPC	Auto code (sec)	CVXGEN code (sec)
M1	24.8	198.77
M2	25	162.308
M3	24.9	113.24
M4	25	144.55

5.3. Use-case:2 Plug-and-Play Control

The user sends information on the requirements, hardware controller, and the process to the cloud. The cloud's virtualization and persistence services are used to generate the model templates for the process and controller as a PIM and the auto-code is emitted from the PSM for the target hardware. This is followed by SIL and the virtualization ability of the cloud is used to instantiate an emulator to perform the HIL. On successful validation of the requirements, the control code file is transferred using secure FTP interfaces of the cloud and IIoT gateway to the specific target platform. The plug-and-play deployment of the QTP is shown in Fig. 8 and the results obtained are shown in Fig. 10. One can verify that the plug-and-play control is performed and the requirements are met by the deployment. A slight pulsations are seen in the levels due to sensor's inertia and noise.

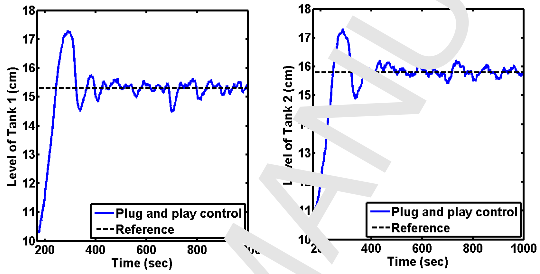

Figure 10: Plug-and-Play Control in IIoT

The computation time the plug-and-play control for the MPC models $\mathcal{M}_1 - \mathcal{M}_4$ is compared with the code generated by the auto-coder CVXGEN[3] for the target hardware within the process station (without file transfers). The computation times for 25 iterations of these codes are shown in Table 1. It can be seen that the auto-generated code for the target platform is lesser than CVXGEN code directly ported to the target hardware. This is due to run-time compilation that happens with the Python code as against compiled execution of the auto-generated code. This results demonstrates the plug-and-play capabilities introduced due to cloud's capabilities.

5.4. Use-case:3 Model Based Verification

The UTA model for the quadruple water-tank process (QTP) is composed of automata of water tanks, sensors, pumps and controller are shown in Fig. 6. The model-templates using action model patterns and composition operators, that are used to construct the formal model of timing variations, and timing-wrapper is used in case of periodic operations. The composed model of the QTP with its component and timing interfaces is shown in Fig. 11

5.4.1. Verification of Requirement Specifications

This investigation verifies QTP performance in two modes: minimum and non-minimum phase. In minimum phase mode, the level of *Tank 1* depends on the flow from *Pump1* and that of *Tank 2* is influenced by *Pump2* and this is a stable operation mode. While in non-minimum phase, the level of *Tank 1* depends on the flow from *Pump2* and that of *Tank 2* depends on *Pump1* leading to an unstable mode. To facilitate verification, the requirements specifications is mapped to the formal specifications of the QTP (for notations please refer the Nomenclature section).

The level of the tank *w_Lev* and the additional parameter *TOver* are used to denote overflowing of the tank *w_Lev* >= *TOver*, in such situations the pumps are slowed down. Including these new parameters, the UTA model is redefined for verifying the following properties:

[3] https://cvxgen.com/

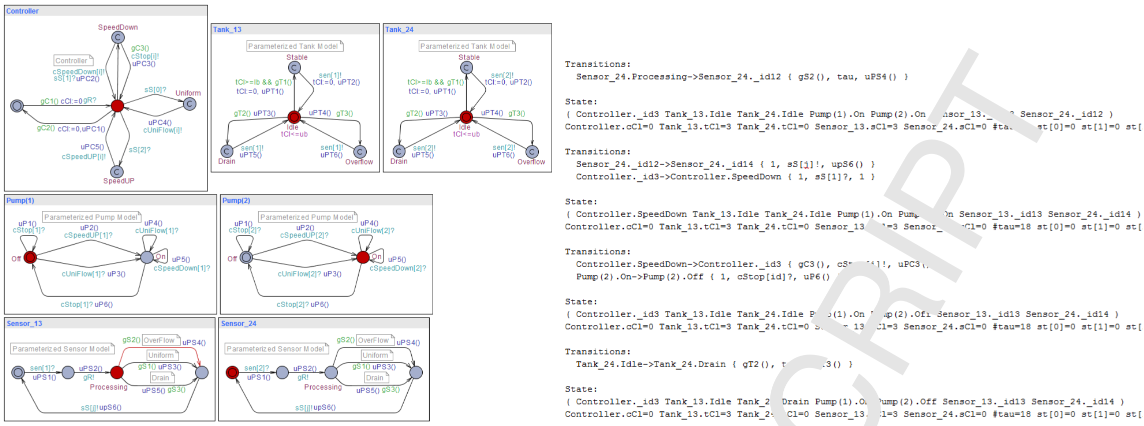

Figure 11: Simulation and Generated Traces for above properties

- *(a) Deadlock Property*, we prove at first that there is not blocking states in the system. It is proved by running the model checking query

$$A[]\ not\ deadlock$$

- Verifying Minimum phase model of the QTP.
  Property 1: The reachability properties need to be verified for showing that the reaction time requirements are met. First we show that the both pumps supply sufficient water flow to tanks i.e.,

$$A <> Tank\_13.w\_Lev == TLev\_Max\ \&\&\ G\_Clock <= Ub$$

  The query proves that the water tanks filling time from level 0 to *w_Lev* should not be exceed time bound *Ub*.

  Property 2: The property expresses that whenever the water level in particular tank reaches to *TOver* level, sensor measure the level and pass the signal to the controller, which issues the control signal *cStop* to the particular pump.

$$E <> Tank\_24.w\_Lev >= TOver\ imply\ (Tank\_24.Overflow\ \&\&\ p\_run[2] == 0\ \&\&\ pCl <= Ub1)$$

- Verification of non-minimum phase model of operation *Property 1:* Similarly as above, we prove that the controller issue the control signal to *Pump2* to maintain water level in *Tank13* as the requirement of Non-Minimum Phase Mode of Operation.

$$E <> Tank\_13.w\_Lev >= TOver\ imply\ (Controller.SpeedUP\ \&\&\ Pump(2).Off)$$

  The query proves that the upon receiving signal from *sensor[1]* at *Tank13* (overflowing) the control issue a signal to *Pump2* to *Stop* or *speedDown* the water supply in *Tank3*.

The model checker generates the witness or counterexample depending upon if property is satisfied by the model. The automatic generation of witness and counterexample is considered as the key advantage of model checking which provides useful source of diagnostic information and a basis for automated test generation. The Fig. 11 represents the simulation layout and generated traces for particular property. By using the model templates in the cloud, the MBV can be done on-the fly as illustrated in the example.
Comments: During the deployment of the MPCs in IoT devices, there were few issues that surfaced. First, the speed of the control algorithm depended on the target code language. For example, a C-code performed better than a run-time compiler language such as Python. Second, the latencies in the sensors and computations were not significant with on board communications, but were significant in IP based communication. However, they were not at a level to destabilize the operations for the process application chosen. Third, the IoT controllers and sensors the effect of timing imperfections and noise created pulsations in the output. Fourth, there were some MPC implementations that could not be validated in the HIL, but they passed the other validation tests. Fifth, the real time performance of the target platform is greatly influenced by the amount of TCP based communication used. Sixth, the cloud based communications and field level TCP communications generate only the same amount of latencies, this is partially due to the high computing power of the server. Finally, the cloud services communicating through the TCP based protocols have the same computation burden as any TCP device.

6. Conclusions

This investigation presented an IIoT architecture, a model-based engineering approach (MBE), and workflows for implementing cloud-based IIoT. The IIoT architecture combined the open connectivity with cloud-intrinsic features. To perform model based engineering, a multi-view model of the industrial automation capturing various aspects was proposed. A meta-model of the automation system integrating these different views was generated using AutomationML. This meta-model provided the basis for performing Model Based Engineering. Further, it generated the requirements for the design and verification. The Model Based Design (MBD) approach was used to design MPC, a sophisticated controller that repeatedly solves an optimization routine, for the target platform. The MBD approach generated the auto-code for the MPC and also validated the design through MIL, SIL and HIL during its workflow. The behaviour models from the requirements were used to perform model based verification. The Uppaal Timed Automata (UTA) models with action patterns of timing behaviour were composed to verify the timing performance to guarantee timing. Consequently, reducing the engineering efforts of cloud-based IIoT significantly. Insights into performing MBD and MBV from cloud was also provided. Next, the additional benefits provided by cloud-based IIoT was discussed with features such as plug-and-play control, smart manufacturing, and on-the-fly verification. The proposed IIoT architecture, MBE approach, and workflow were demonstrated on a QTP, a benchmark problem in process control. Our results showed the benefits of the combining cloud and IoT, and MBE as an approach for realizing it. Studying deployment of cloud-based IIoT for providing enterprise wide connectivity and performing plant wide optimization are future course of this investigation.

Appendix 1

MPC Optimization Models

In the MBD workflow, the objective function and constraints of the MPC denote the requirements of the control algorithm. The investigation considers four different MPC models, they are:

$$
\begin{aligned}
\mathcal{M}_1 : \underset{U}{\text{minimize}}\ \text{J} &= (Y - Y_r)^T Q(Y - Y_r) + \Delta U^T R \Delta U \\
\text{Subject to:}\ \ \mathcal{C} & \\
\mathcal{M}_2 : \underset{U}{\text{minimize}}\ \text{J} &= (Y - Y_r)^T Q(Y - Y_r) + \\
& \qquad (U - u_d)^T R(U - u_d) \\
\text{Subject to:}\ \ \mathcal{C} & \\
\mathcal{M}_3 : \underset{U}{\text{minimize}}\ \text{J} &= (Y - Y_r)^T Q(Y - Y_r) + U^T R U \\
\text{Subject to:}\ \ \mathcal{C} & \\
\mathcal{M}_4 : \underset{U}{\text{minimize}}\ \text{J} &= (Y - Y_r)^T Q(Y - Y_r) + \Delta U^T R \Delta U \\
\text{Subject to:}\ \ \mathcal{C} &
\end{aligned} \tag{1}
$$

where the constraints $\mathcal{C}$ is given by

$$
\begin{aligned}
x(k+1) &= Ax(k) + Bu(k) + \hat{d}(k), \quad \forall k = 1, ..N_p \\
y(k) &= Cx(k) \ \ \forall k = 1, ..N_p \\
u_{min} &\leq u(k) \leq u_{max} \quad \forall k = 1, ..N_p \\
\Delta u_{min} &\leq \Delta u(k) \leq \Delta u_{max} \quad \forall k = 1, ..N_p \\
y_{min} &\leq y(k) \leq y_{max} \quad \forall k = 1, ..N_p
\end{aligned}
$$

These constraints model the physical and operating constraints of the MPC. They capture the system dynamics, constraints on the control input, change in control input and output, respectively.

Quadruple Tank Process Dynamics

The dynamics of the quadruple process is given by

$$
\begin{aligned}
\dot{h}_1(t) &= \frac{1}{A}(a\sqrt{2gh_3} + \gamma_1 f_1 - a\sqrt{2gh_1}) \\
\dot{h}_2(t) &= \frac{1}{A}(a\sqrt{2gh_4} + \gamma_2 f_2 - a\sqrt{2gh_2})
\end{aligned}
$$

$$
\begin{aligned}
\dot{h}_3(t) &= \frac{1}{A}((1-\gamma_2)f_2 - a\sqrt{2gh_3}) \\
\dot{h}_4(t) &= \frac{1}{A}((1-\gamma_1)f_2 - a\sqrt{2gh_4})
\end{aligned} \tag{2}
$$

References

[1] C. Wang, Z. Bi, L. Da Xu, Iot and cloud computing in automation of assembly modeling systems, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1426–1434.

[2] A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of cloud computing and internet of things: a survey, Future Generation Computer Systems 56 (2016) 684–700.

[3] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future generation computer systems 29 (7) (2013) 1645–1660.

[4] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi, S. Yin, Industrial cyberphysical systems: A backbone of the fourth industrial revolution, IEEE Industrial Electronics Magazine 11 (1) (2017) 6–16.

[5] W. He, L. Da Xu, Integration of distributed enterprise applications: A survey, IEEE Transactions on Industrial Informatics 10 (1) (2014) 35–42.

[6] L. Da Xu, W. He, S. Li, Internet of things in industries: A survey, IEEE Transactions on industrial informatics 10 (4) (2014) 2233–2243.

[7] I. Calvo, F. Pérez, I. Etxeberria, G. Morán, Control communications with dds using iec61499 service interface function blocks, in: Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on, IEEE, 2010, pp. 1–4.

[8] H.-M. Hanisch, Closed-loop modeling and related problems of embedded control systems in engineering, in: International Workshop on Abstract State Machines, Springer, 2004, pp. 6–19.

[9] S. Theiss, V. Vasyutynskyy, K. Kabitzsch, Software agents in industry: A customized framework in theory and praxis, IEEE Transactions on Industrial Informatics 5 (2) (2009) 147–156.

[10] F. Jammes, H. Smit, Service-oriented paradigms in industrial automation, IEEE Transactions on Industrial Informatics 1 (1) (2005) 62–70.

[11] V. N. Dubinin, V. Vyatkin, Semantics-robust design patterns for iec 61499, IEEE Transactions on Industrial Informatics 8 (2) (2012) 279–290.

[12] K. Thramboulidis, Model-integrated mechatronics-toward a new paradigm in the development of manufacturing systems, IEEE Transactions on Industrial Informatics 1 (1) (2005) 54–61.

[13] V. Vyatkin, Software engineering in industrial automation: State of the-art review, IEEE Transactions on Industrial Informatics 9 (3) (2013) 1234–1249.

[14] M. Obermeier, S. Braun, B. Vogel-Heuser, A model-driven approach on object-oriented plc programming for manufacturing systems with regard to usability, IEEE Transactions on Industrial Informatics 11 (3) (2015) 790–800.

[15] B. Vogel-Heuser, D. Schütz, T. Frank, C. Legat, Model-driven engineering of manufacturing automation software projects–a sysml-based approach, Mechatronics 24 (7) (2014) 883–897.

[16] K. Thramboulidis, F. Christoulakis, Uml4iot—a uml-based approach to exploit iot in cyber-physical manufacturing systems, Computers in Industry 82 (2016) 259–272.

[17] S. Mayer, J. Hodges, D. Yu, M. Kritzler, F. Michahelles, An open semantic framework for the industrial internet of things, IEEE Intelligent Systems 32 (1) (2017) 96–103.

[18] F. Tao, Y. Cheng, L. Da Xu, L. Zhang, B. H. Li, Cciot-cmfg: cloud computing and internet of things-based cloud manufacturing service system, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1435–1442.

[19] Z. Bi, L. Da Xu, C. Wang, Internet of things for enterprise systems of modern manufacturing, IEEE Transactions on industrial informatics 10 (2) (2014) 1537–1546.

[20] A. W. Colombo, T. Bangemann, S. Karnouskos, Imc-aesop outcomes: Paving the way to collaborative manufacturing systems, in: Industrial Informatics (INDIN), 2014 12th IEEE International Conference on, IEEE, 2014, pp. 255–260.

[21] A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, J. L. Lastra, et al., Industrial cloud-based cyber-physical systems, The IMC-AESOP Approach.

[22] A. W. Colombo, S. Karnouskos, J. M. Mendes, P. Leitão, Industrial agents in the era of service oriented architectures and cloud based industrial infrastructures, Industrial Agents: Emerging Applications of Software Agents in Industry (2015) 67–87.

[23] A. Maffei, S. Srinivasan, P. Castillejo, J.-F. Martinez, L. Iannelli, E. Bjerkan, L. Glielmo, A semantic middleware supported receding horizon optimal power flow in energy grids, IEEE Transactions on Industrial Informatics.

[24] K. Thramboulidis, et al., The 3+1 sysml view-model in model integrated mechatronics, Journal of Software Engineering and Applications 3 (02) (2010) 109.

[25] J. Reineke, S. Tripakis, Basic problems in multi-view modeling, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2014, pp. 217–232.

[26] B. Brandenbourger, M. Vathoopan, A. Zoitl, Engineering of automation systems using a metamodel implemented in automationml, in: Industrial Informatics (INDIN), 2016 IEEE 14th International Conference on, IEEE, 2016, pp. 363–370.

[27] S. Balasubramaniyan, S. Srinivasan, F. Buonopane, B. Subathra, J. Vain, S. Ramaswamy, Design and verification of cyber-physical systems using truetime, evolutionary optimization and uppaal, Microprocessors and Microsystems 42 (2016) 37–48.

[28] S. Srinivasan, F. Buonopane, J. Vain, S. Ramaswamy, Model checking response times in networked automation systems using jitter bounds, Computers in Industry 74 (2015) 186–200.

[29] S. Srinivasan, F. Buonopane, S. Ramaswamy, J. Vain, Verifying response times in networked automation systems using jitter bounds, in: Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on, IEEE, 2014, pp. 47–50.

Muthukumar Natarajan is currently pursuing his PhD in SASTRA University from 2014. His research interests are broader topics of optimization driven control, the Internet of Things (IoT), and process control. He is a recipient of the SASTRA research scholarship for pursuing PhD and had a brief stint with industry before embarking on the current position. He has 8 journal papers and 10 conference papers published through his research work.

Seshadhri Sriniavsan obtained his PhD from National Institute of Technology-Tiruchirappalli in 2010. He is currently working with Berkeley Education Alliance for Research in Singapore (BEARS), Singapore working with Prof. Kameshwar Poolla. He is working on SinBerBEST2 and GBIC projects. Before taking up this position, he was a researcher at GRACE, Italy, Technical University of Munich, Germany, Kalasalingam Academy of Research and Education, and Center for Excellence in Nonlinear Systems. He worked in European projects eGotham, I3RES, and Complex networked control systems. In 2011, he worked as an Assoc. Scientist with ABB Global Industries and Services Pvt. Ltd., Indian Corporate Research Center, India. He was involved with the IEEE CSS Standardization committee in 2016 and was awarded the IEEE CSS Outreach fund in 2017. In addition, he has teaching stints at Kalasalingam Academy of Research and Education. His publications include around 50 journals, reputed conferences, 4 books, and has patented 3 technologies.

Deepak Pal is currently pursuing his PhD in the department of software science, Tallinn University of technology, Tallinn, Estonia. Currently, he is studying and implementing the problem of generating tests for real-time distributed systems by exploiting model-based techniques and has publication in reputed conferences. His research interests are software engineering, software testing, model-based testing, model checking, theorem proving, program verification, modelling, real-time distributed systems, cyber-physical systems.

Kannan Ramkumar was born in Madurai, India, in 1975. He received the B. Tech degree in Instrumentation and control Engineering from Madurai Kamaraj University in 1997, subsequently he did his M. Tech from Regional Engineering college, Trichy in 2000 and obtained his Ph.D. degree in Control Engineering from SASTRA Deemed University, India in 2010.Since 1998, he has been with the Department of Electronics and Instrumentation, SASTRA Deemed University, where he was an Assistant Professor, became an Associate Professor in 2011, and a Professor in 2018.His current research interests include Mobile Robotics, Estimation and Control theory and Electrical drive systems. He has received funding from DRDO, an Indian government defense agency, for investigating the localization and mapping problem in Mobile robot. He was the Chair Professor for the Wipro Mission 10X, a pedagogy skills improvement initiative in SASTRA University for the year 2011-12. He is the recipient of the prestigious “Innovative Practitioner” a teaching excellence award from Wipro for the year 2011.Currently he is solving Machine learning, Estimation and Control problems related to applications like Electric vehicle, Mobile robots and Process control. He is also heading CALIBRE (Control Artificial intelligence, Biomedical and Robotics Engineering), a research group.

Jüri Vain graduated in System Engineering from Tallinn Polytechnic Institute, Estonia in 1979. He received his PhD in computer science from the Institute of Cybernetics at Estonian Academy of Sciences in 1987. Currently, he is Professor of Computer Science at the Department of Software Science, Tallinn University of Technology. His research interests include formal methods, model-based testing, cyber physical systems, human-computer interaction, autonomous robotics, and artificial intelligence. He has been leading researcher in several international projects under EU framework programmes and the coordinator of Estonian-Japan medical robotics collaboration programme. He has published over 150 scientific articles including journal papers, book chapters and conference papers.

Srini Ramaswamy currently serves as a Global R&D Project Manager for Engineering in the Power Generation Business Unit. Earlier he was the *global lead for Software Tools Development and Services* for the Software Development Improvement Program (SDIP), *headed the Industrial Software Systems research* group at its India Corporate Research Center (CRC), and headed the *Tools and Support Services* group for its India Development Center (IDC) in Bangalore, India. On the academic front, he serves as a visiting professor at the University of Arkansas at Little Rock and as the Associate Director for the Australia-India Center for Automation Software Engineering, a $3M three-way Industry – University – Government partnership focused on software systems research for the Engineering and Automation sectors. He is on the Executive Committee of the Computing Accreditation Commission of ABET, the IEEE-CS representative member of the board at CSAB, a senior member of the ACM, and a senior member of the IEEE.

Muthukumar. N

Electric Vehicle Engineering and Robotics (EVER) Lab,

SASTRA Deemed to be University,

Thanjavur, India – 613401

Email: muthukumar.n@sastra.ac.in

Seshadhri Srinivasan
Berkeley Education Alliance for Research in Singapore,
Singapore – 138602
Email: seshadhri.srinivasan@bears-berkeley.sg

K. Ramkumar

Electric Vehicle Engineering and Robotics (EVER) Lab,

SASTRA Deemed to be University,

Thanjavur, India – 613401

Email: ramkumar@eie.sastra.edu

Deepak Pal
Department of Computer Science,
Tallinn University of Technology,
Tallinn, Estonia
Email: Deepak.pal@ttu.ee

Juri Vain
Department of Computer Science,
Tallinn University of Technology,
Tallinn, Estonia
Email: juri.vain@ttu.ee

Srini Ramaswamy
ABB Inc.
USA
Email: srini@ieee.org

Highlights

- Presents a Model-Based Engineering Approach for Industrial Internet of Things in Process Industries.
- Shows that combining the Internet of Things, Cloud-Services, and MBE desirable control features such as plug-and-play control, on-the-fly verification can be achieved.
- An IIoT architecture for process industry is proposed.
- Workflow for model-based design.
- Demonstrates the MBE approach in a process industry.