

Received December 4, 2019, accepted January 1, 2020, date of publication January 9, 2020, date of current version January 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2965259

A Model-Driven Mobile HMI Framework (MMHF) for Industrial Control Systems

IQRA QASIM^{ID}, MUHAMMAD WASEEM ANWAR^{ID}, FAROOQUE AZAM^{ID}, HANNY TUFAIL^{ID}, WASI HAIDER BUTT^{ID}, AND MUHAMMAD NOUMAN ZAFAR^{ID}

Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan

Corresponding author: Muhammad Waseem Anwar (waseemanwar@ceme.nust.edu.pk)

ABSTRACT With the advent of software technologies, over a period of time, the Industrial Control Systems (ICSs) have grown exponentially. Whereas, almost all ICSs comprise *Human Machine Interfaces* (HMIs), which are the key component for monitoring and controlling complex industrial systems. For decades, traditional HMIs with simple *User Interfaces* (UIs) remained operational to minimize the complexities and resulting operational costs. However, due to the emergence of smartphone technologies, the perception about user interfaces has been transformed significantly and users now demand same sort of experience with industrial HMIs, as well. There are few industrial solutions, like, ICONICS GraphWorX to support the development of mobile HMI screens. However, such proprietary solutions are quite expensive. Furthermore, the underlying development approaches and source codes are not accessible in public domain. On the other hand, the state-of-the-art approaches for the development of native mobile HMI screens are hard to find in the literature. Consequently, there is dire need of a cost-effective, easy to use, open source framework for the development of native mobile HMI screens. In order to achieve this goal, here we propose, a **Model-driven Mobile HMI Framework (MMHF)**. MMHF comprises, a Unified Modeling Language (UML) Profile for Mobile HMI (UMLPMH) for modeling of HMI screens. MMHF also includes, an open source *transformation engine* and a **Model Driven Mobile-based HMI Code Generator (MDMHCG)** to automatically transform UMLPMH models into target native mobile HMI implementations. Consequently, MMHF enables simpler way to design the HMI screens using UMLPMH and generates native *Mobile HMI Screen* implementations automatically using MDMHCG. The empirical evidence of MMHF is demonstrated through *three* (3) benchmark case studies, which prove that the MMHF is a feasible, cost effective and scalable solution to develop native HMI screens for wide-ranging ICSs.

INDEX TERMS Human machine interface, mobile HMI, model driven engineering, unified modeling language, industrial control system, industry automation, Internet of Things (IoT).

I. INTRODUCTION

Modern Industrial Control Systems (ICSs) are gaining complexities to match the rapid & high quality production needs of the industry [28]. The volume of data generated from industrial plants is also growing gradually. It is the responsibility of Human Machine Interface (HMI) to translate huge amount of complex process data into human readable format for decision making during plant processing. HMI acts as a visual component of Supervisory Control and Data Acquisition System (SCADA) [1]. User friendly and responsive HMIs result in cost effective and efficient monitoring system.

The associate editor coordinating the review of this manuscript and approving it for publication was Heng Zhang.

According to a report conducted at the end of 2015, the world-wide market of HMI is estimated to reach by US\$5,579.3 in 2019 at a compounded growth rate of 10.4% during a period of 2013-2019 [2]. These HMIs play a central role in monitoring and controlling industrial process from a computerized panel system.

For industrial machines to be integrated with HMIs, correct functioning of these devices with Programmable Logic Controllers (PLCs) is a prerequisite [3]. There are various input/output sensors connected to the industrial machines for monitoring temperature, pressure, weight, speed and feed rate etc. It is the PLC device that fetches data from these sensors and converts it to required format through ladder logic [29]. HMI then translates the communications from PLCs,

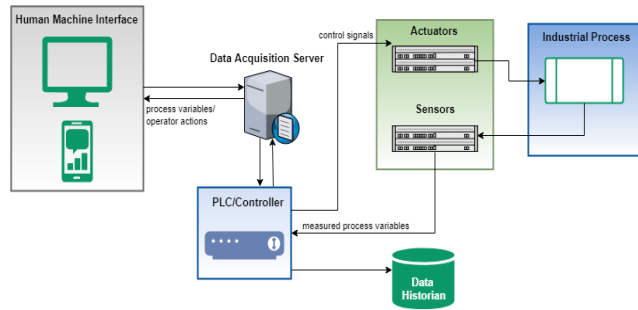


FIGURE 1. HMI based industrial control system.

sensors and other devices to the human readable format so that operators can visualize, monitor and control industrial processes as shown in **Figure 1**.

As users are now familiar with high quality smartphone UIs, they demand same level of usability for HMIs. Hence, due to the changing needs, traditional HMIs based on large systems are now transforming towards portable handheld mobile devices [4]. Traditional HMI systems usually comprise of huge fixed desktop panels. Moreover, web technologies are also being introduced in most of the industrial HMI systems [5], however, they introduce low performance and security related issues while configuring with real time data. Although numerous modern HMI systems available in market such as Simatic WinCC, Genesis 64, Wonderware Intouch, etc., to provide tool support for mobile HMI development [6], they are not freely available in market. Furthermore, the underlying development approaches and source code are not available publically. On the other hand, the state-of-the-art approaches for the development of pure native mobile HMI screens are hard to find in literature. Consequently, the need for the development of a complete, open source native mobile HMI approach for monitoring real time industrial process is highly desirable. The existing development methods for mobile HMI systems involve complex mechanisms, extensive low-level coding and require high level of expertise, lengthy development period and increased cost. This opens up doors for introducing model driven approach into native mobile HMI client development that can: 1) reduce the complexity of its development process [23], 2) provide a higher-level abstraction with fully automated end to end implementation, 3) provide a controlled development during early stages, 4) deliver an open source generic and real time solution which can provide a modern HMI for ICSs.

This article presents the cost effective, easy to use and open source framework for the development of native mobile HMI screens. Particularly, the contributions of this article are as follows:

- **Model-driven Mobile HMI Framework (MMHF)** is introduced to design native mobile HMI screens for industrial control systems. It provides a simple way for system integrators to design HMI screens and subsequently generate native mobile HMI screens implementations automatically.

- Firstly, a UML Profile for Mobile HMI (UMLPMH) is developed in MMHF to accomplish the modeling of HMI screens with simplicity. Particularly, it contains several HMI concepts for ICS, such as: widgets, real time data connectivity etc. This enables design of both simple, as well as, complex mobile HMI screens through UMLPMH.
- Secondly, an open source transformation tool is developed named as Model Driven Mobile-based HMI Code Generator (MDMHCG) for instantly transforming input UMLPMH models into target HMI android code. Particularly, transformation rules are developed for the conversion of UMLPMH concepts into low level android code. Subsequently, the implementation of rules is done through Model-to-Text (M2T) transformation approach by exploiting the features of Acceleo and Java services.
- Finally, the idea is validated through three case studies: 1) home automation, 2) sewage plant and 3) traffic controller.

This paper is organized as: Section **II** provides the detailed related work and industrial tools in the domain of HMI development. Section **III** covers the details of proposed UMLPMH. Section **IV** presents the detailed implementation regarding MDMHCG. Section **V** provides the validation of MMHF using three bench mark case studies. The case studies selected for validation purposes are from different domains and of different sizes to make sure that the proposed approach works on every case. Section **VI** contains a brief discussion on the MMHF as well as the limitations. Section **VII** concludes the research and recommends the future directions.

II. PRELIMINARIES

In this section, a brief overview of the approaches regarding *HMI*, *model driven HMI* and *mobile based HMI* are presented. Furthermore, the comprehensive analysis of renowned industrial automation tools in the context of HMI development is also performed. Finally, *research gap* is identified on the basis of both industry and state-of-the-art analysis.

A. LITERATURE REVIEW

1) HMI RELATED APPROACHES

Harpreet Singh Saini and R.D Daruwala [7] proposes an embedded processor along with Internet of Things (IoT) support for accessing real time values of industrial processes and subsequently display on HMI screen. This processor acts as an IoT server to acquire data from industrial processes and send it to local device HMI and remote IoT client HMI for display. These HMIs are developed in QT framework which is based on C++ language. It uses an open source ModbusPal emulator for fetching field devices data. In another study, Villani *et al.* [2] target the idea of design development of adaptive human machine interfaces for complex industrial control systems. The HMI adapts itself according to the physical and cognitive capabilities of human workers.

The proposed system first measures the workers' capabilities, then adapts the interface according to the measured abilities of user. Finally, the interface trains the unexperienced users about how to interact with the machines. It proposes the concept of a generic meta HMI that is customized according to a specific platform based on worker needs and requirements. In another study, Normanyo *et al.* [8] develops HMI screens using Siematic WinCC (a windows based software). It uses PLCSIM for simulation between PLC devices and HMI. The tag values are assigned to each HMI element using the WinCC software. Similarly, Jayanthi *et al.* [9] develop an HMI for process control systems using multiple software tools. WPLSoft is used to program PLC and is connected to HMI using a communication protocol (RS485). HMI is created in DOPSoft tool and connected to PC through eRemote. The same HMI can be connected to mobile device through TPLink for remote control of data. There is a non-conventional HMI based study [1] where a theoretical framework for 3D visualization of HMI driven by non-conventional display device (such as projector or table) with Natural User Interface (NUI) paradigm is proposed. 3D visualizations provide complete replica of a real factory that enhances processes monitoring and control. NUI provides an interactive user interaction in a more natural way. This method focuses on both interaction and visualizations of HMI. The proposed HMI system is based on a hierarchical structure in an agent oriented environment i.e. process control level, floor factor level and business level. The proposed method was used for implementing an experimental virtual reality process to combine 3D visualizations of HMI with NUI. Another research [10] introduces trust related factors into the HMI for automated driving vehicle systems. This framework provides guidelines for designers while developing HMI system for automated driving system.

2) MODEL BASED HMI APPROACHES

To cope with the inherent complexities of HMI development [38], model driven approaches are also proposed in this area. The motivation for adopting model based HMI development is based on cost effectiveness, shorter development time, reusability and improved quality [21].

Dorninger *et al.* [14] work on the model based reengineering of HMI systems for shifting existing systems to new technology. It reengineers HMI by performing static analysis of its available source code and extract the central information to develop implementation independent model. These implementation models serve as the starting point to automatically generate parts of HMI system based on new technology concepts. It uses Xpand code generation language based on Eclipse EMF models. In [15] a model driven prototyping tool is proposed for early checking of in-vehicles HMI design during requirements and design phase. It provides an open source software, based on image editor in Eclipse, relational database to store HMI information and web browser simulator for checking HMI design. It helps HMI designers to design new HMI requests and visualize them. In another study,

Martin *et al.* [6] use a model driven approach to develop a plant model from partially available data required for semi-automated generation of HMIs. Authors propose a meta-model for the description of production plant components. Based on proposed meta-model, a plant model is developed to process data that generates a concrete HMI model using a three step process. User can intervene at each step to influence HMI. This concrete HMI model is converted to executable web based HMI application. Ramaswamy *et al.* [16] propose a meta-model for the non-functional properties (NFPs) of the HMI (human, machine and their interactions). Functional requirements cannot be used alone to make runtime decisions so modeling NFPs are necessary in these type of architectures. The proposed solution reduces the troubleshooting time of faults and ensures safety of industrial workers.

3) MOBILE BASED HMI APPROACHES

Smartphones and mobile devices can provide a portable remote access of production plants to the operating engineers [20]. They can remotely silence an alarm or send commands to the industrial process [24]. With the increasing trend of mobile devices in daily lives [22], same sort of UIs is now demanded in industrial control systems as well. Several mobile based HMI solutions have been presented in the academia. For example, Jain *et al.* [4] propose the idea of reproducing existing HMI screens on a mobile device. Remote mobile connects with HMI panel through a wireless connection with a server component. Proposed method divides HMI system into dynamic and static parts. Static parts do not change over a period of time and are loaded once e.g. labels, buttons etc. Dynamic parts are continuously sent to the mobile client by server component as they continuously change e.g. filling of tank bar, movement of dial meter etc. This approach allows minimum data to be stored on mobile device so reduces threats of device thefts. It also reduces development costs due to reusability of HMIs. Lojka *et al.* [5] use web technologies in designing and developing HMI systems. They develop JavaScript framework for simplifying HMI development. The created HMIs do not communicate directly with PLC devices instead it reads/writes data to SQL database through mobile service feature of Microsoft Azure. Communication server is used to fetch data from PLCs and update database accordingly. Oscar Neira [19] proposes a method based on Adaptive HMI Engine for generating HMIs at runtime taking into consideration user needs and requirements to improve user-system interaction. This HMI engine is based on two components: HMI Definer is based on web application (running at remote server) that allows to adapt runtime data to the interfaces. It creates design and layout of HMI based on user tasks, user profile and characteristics of the device. This layout specification is based on JSON messages that are transformed to web HMI elements by HMI Builder. HMI builder then displays final interface in a web browser of a mobile device. Willocx *et al.* [20] propose a method for handling security requirements of mobile access to critical processes of ICSs. It uses VPN network for

TABLE 1. Evaluation of HMI features in leading industry automation tools.

	Genesis 64	SIMATIC WinCC	AggreGate	OAS
HMI Module	GraphWorX™64	Graphics System	HMI Builder	OAS Web HMI
Main Screen Dynamics	<ul style="list-style-type: none"> Color dynamics Pick Actions Dynamic Time date object Data entry object Hide, Rotation, Size, Location dynamics 	<ul style="list-style-type: none"> Process picture dynamics (tanks, dials, knobs etc.) Color Dynamics Bar Dynamics Trend View Guage Dynamics 	<ul style="list-style-type: none"> Colors, strokes, fonts and other attributes dynamics Zooming, moving and rotating image parts dynamics (tank and knob levels) 	<ul style="list-style-type: none"> Radial gauges and interactive switches dynamics Real time historical Trend dynamics
Visualization Dimension	2D,3D	2D, 3D	2D	2D, 3D
Errors	Diagnostic Windows	Errors Logs	Popup messages, SMS and Email notifications	SMS, Email or voice message
Screen Refresh Rate	Defined by user	Defined by user	Defined by user through job scheduler	N/A
Scripting Language	SupportWorX, VB	VB	Java, Python	VB, C#
Proprietary	Yes	Yes	Yes	Yes
Licensing	Yes, with 30 days trial period	Yes, with 21 day trial period	Yes, with 30 days trial period	Yes, with 30 days trial period

secure transfer of data from ICS to mobile devices. pfSense (an open source firewall/router) is used on ICS network to allow only traffic from VPN network. Access to VPN network is based on user and device authentication (through OpenVPN app on the device). For establishing a VPN tunnel connection, the credentials of both VPN endpoint and mobile device are verified along with user login details. MobileIron MDM (Mobile Management System) is used to enroll trusted mobile devices in account. Only devices that are enrolled in MDM account can successfully establish a VPN connection. Upon successful connection, devices can get access to the ICS network through VPN.

B. INDUSTRY PERSPECTIVE

There are multiple industry automation tools available in the market for developing HMI/SCADA systems. The four (4) most widely used HMI/SCADA tools are described in **Table 1**.

ICONICS GENESIS 64 suite provides a deployment platform for integrating all available plants and business data into a real time distributed platform. It is a complete software solution for all HMI and SCADA applications. It is used for a variety of applications such as HMI/SCADA, Building Automation, Plant Data Historian and more. It provides GraphWorX™64, an HMI software package, to design highly interactive and animated graphics for process control systems. It is a canvas where one can display real time process data through any OPC UA compliant data source.

GraphWorX™64 allows you to create own symbols (HMI objects) or use from Symbols Library available in Genesis 64. It provides static as well as dynamic objects to create animated displays. GraphWorX™64 symbols library provides basic shapes such as line, circle, ellipse, square, rectangle, arc, polyline and polygon. Other than basic symbols it allows you to add sliders, dials, gauges, indicators, buttons, lights, clock and other real time data based objects.

SIMATIC WinCC is an innovative visualization system with high performance functions to monitor and control industrial processes data [30]. It provides a high performance HMI system for use with Windows Platform [42]. The Graphics system of WinCC provides an editor to draw HMI screens and handles dynamic values on interface [44]. It handles display of static (buttons, texts etc.) as well as dynamic objects (e.g. modifying bar length). This editor provides objects palette that includes standard objects (line, polygon, ellipse, rectangle, text), smart objects (bar, input/output field, status display), windows objects (slider, button, checkbox), tube objects (tube bend, polygon tube, T-piece) and controls (such as ActiveX controls). One can also import graphics to the editor. The dynamic objects of the screen are connected to process tag values. The input/output field property for each object allows to add dynamics e.g. tag values can be set for tank object to show change in its water level.

AggreGate SCADA/HMI is a complex system that offers advanced data acquisition and processing capabilities, intended to facilitate supervisory control and monitoring for a wide range of sectors, such as process control, industrial automation, telemetry and machine-to-machine communications. The built-in HMI Builder assists in drawing and animating any HMIs [35] containing both simple components (buttons, captions, text fields, lists, etc.) and complex ones (tables, multi-layer panes, tabbed panes, charts, geographical maps, dynamic SVG images, video windows, etc.).

Open Automation Software (OAS) develops and markets IoT, SCADA, HMI, and database application software for device to device and device to human interface [36]. The OAS Web HMI product allows to create user interfaces to display and interact with real time data, enabling a whole range of applications on any device with a web browser.

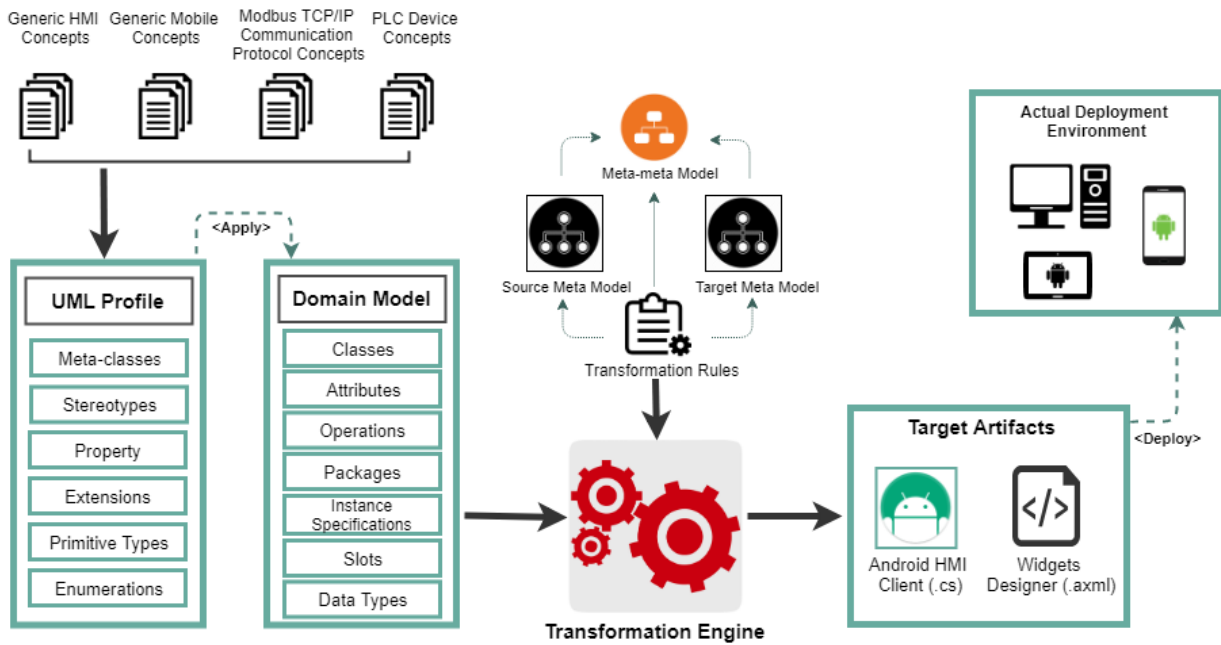


FIGURE 2. Overview of model-driven mobile HMI framework (MMHF).

Each tool supports its own HMI module to develop HMI system, different types of HMI screen dynamics based on real time values and supports visualization dimensions such as 2D or 3D graphics. Although these tools provide powerful graphical libraries, they are not freely available in market that makes the development process costly. Also, each of these tools support a specific scripting language that arises language barrier for users as they must learn specific language for customizing tool. Furthermore, the information regarding the underlying development technologies is not provided by any tool vendor, therefore, extension of aforementioned industrial tools is not possible. This makes the usage of these readily available tools, less applicable in vast scenarios.

C. RESEARCH GAPS AND PROPOSED SOLUTION

Traditional HMIs were based on fixed PC panels but now with emerging trends of smartphones and advancements in technology, users demand highly interactive HMIs similar to their mobile device applications. For this purpose, multiple solutions exist in academic research. After a detailed analysis of the existing researches for HMI, we concluded that current solutions have following issues: -

- Monopoly of expensive proprietary tools that are not freely available and demand high costs for native mobile HMI widgets access.
- Lack of high-level solution to reduce complexities involved in mobile HMI development.
- Web based HMI solutions compromised with low performance and security related issues.
- Lack of open source tool for native mobile HMI development.

In order to solve the identified gaps (after a detailed literature review of HMI development approaches and industrial tools),

we have proposed a solution based on a complete, open source mobile HMI application (Android) for monitoring real time industrial process data. We have integrated a *model driven approach* into *mobile HMI client* that reduces the complexity of its development process, provides a higher level abstraction with fully automated end to end implementation, a controlled development during early stages, as well as, an open source generic and real time solution which can provide a modern touch-based HMI for industrial control systems.

The proposed approach results in following advantages:

- Solution with simple development process
- Cost-effective open source solution
- Ready to use android HMI application code
- Reusability of the developed HMI models
- Reduced development time

III. PROPOSED METHODOLOGY

HMIs have emerged as a critical industrial component used to control, monitor and execute complex industrial processes.. These HMIs require real time industrial data through sensors by using communication protocols. Developing such systems that involve real time field data with complex industrial processes is: *time consuming, costly* and require *special skills / knowledge*. To provide a generic, automated, easy-to-use, fast and high level mobile HMI development a Model-driven Mobile HMI Framework (MMHF) is proposed. **Figure 2** provides an overview of the proposed framework. In order to provide a complete mobile HMI application, multiple concepts like UML Profile and Model to Text (M2T) transformation concepts are involved in the proposed solution.

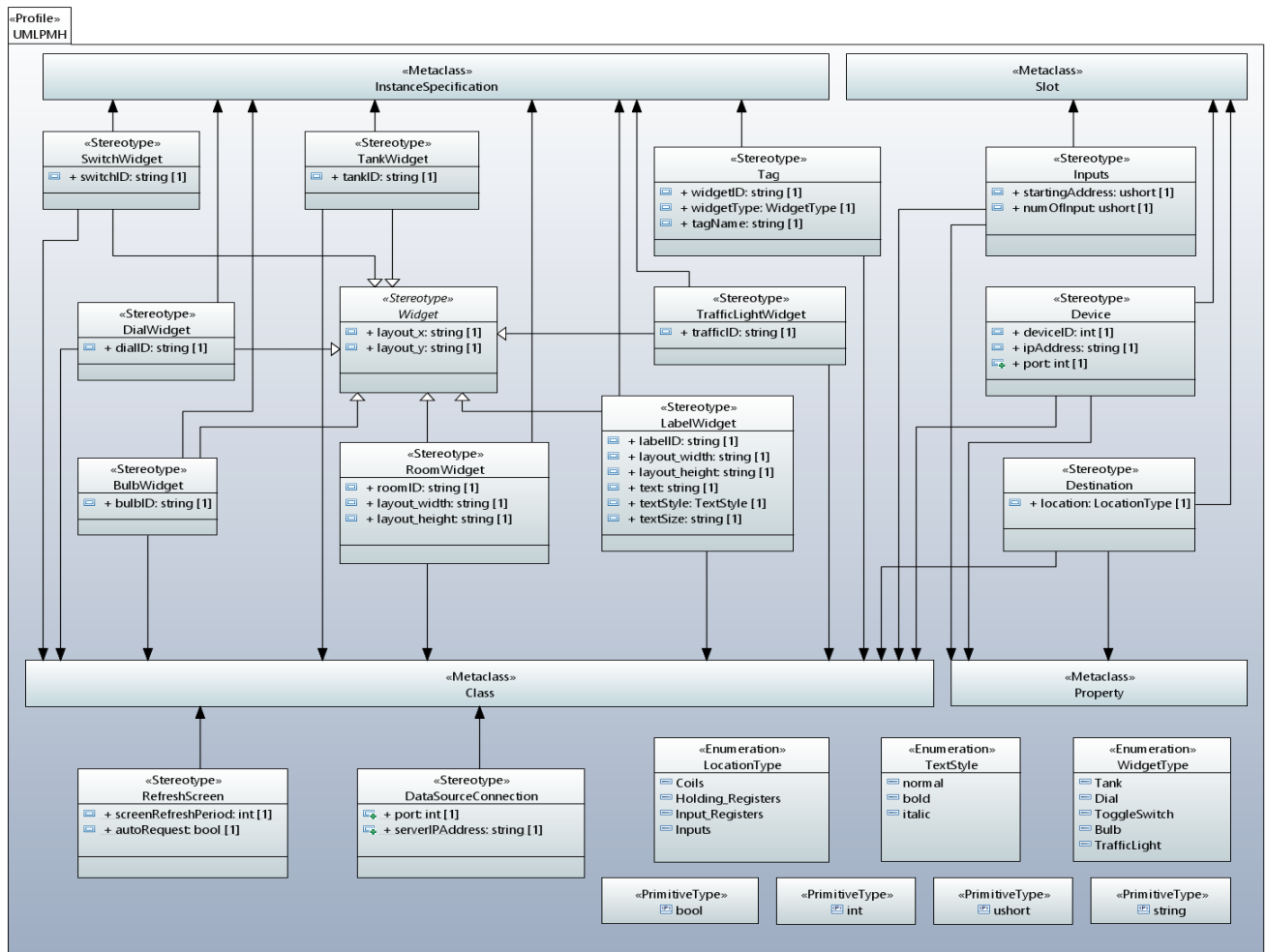


FIGURE 3. UML profile for mobile HMI (UMLPMH).

The MMHF is based on model driven engineering which includes the UML Profile Diagram to extend and enforce the meta-level concepts [31] in modeling the requirements of mobile HMI system. Proposed UML Profile only adds new domain specific concepts to the existing metamodels [39] and do not modify or create new metamodels. Model to Text transformation technique is utilized to take domain model as an input model to generate complete HMI mobile android code and Widgets Designer (.axml) code for the application. These files are then needed to be deployed on an actual environment to provide a fully functional HMI mobile application based on real time PLC device, where, data is fetched using an open source Data Acquisition (DA) server. The proposed framework helps non expert user of the domain to add widgets of their choices, set their properties and assign PLC device addresses without knowing the complex code and technical details. It finally generates a full fledged mobile HMI deployable code for their application.

A. UML PROFILE FOR MOBILE HMI (UMLPMH)

UML Profile for Mobile HMI (UMLPMH), provides a generic extension mechanism that allows to extend and customize meta-model (i.e. UML) using stereotypes [43]

by introducing domain specific concepts of mobile HMI for industrial control systems. Stereotype is a profile class defines, how an existing metaclass can be extended as a part of profile [32]. UMLPMH is developed in papyrus modelling editor tool. Proposed profile contains multiple stereotypes that help to introduce mobile HMI related domain concepts into the UML modeling as shown in **Figure 3**. These stereotypes extend metaclasses of Class, Instance Specification, Property and Slot. UMLPMH stereotypes can be categorized based on following system components: -

- HMI Widgets based Concepts
- PLC Device related Concepts
- Data Source related Concepts

1) HMI WIDGETS BASED CONCEPTS

HMI widget related concepts help to define HMI widgets and assign them to real time PLC devices data using an open source Data Acquisition (DA) server. It contains following stereotypes i.e. Widget, TankWidget, DialWidget, TrafficLightWidget, SwitchWidget, BulbWidget, RoomWidget, LabelWidget, Tag and RefreshScreen. These stereotypes, as represented in **Figure 3**, are required to model both simple as well as complex widgets.

«Widget»

Widget stereotype contains common attributes required by all widgets i.e. attributes for specifying their x and y positions on mobile interface. These common attributes are based on layout_x property specifying distance of current view from x-axis in dp unit (Density-independent pixels) and layout_y property specifying distance of current view from y-axis in dp. Widget stereotype is an abstract stereotype and is not extended from any metaclass. It only provides common properties for all types of widgets.

«TankWidget»

Tank is a type of HMI widget that displays filling of a tank based on real time data obtained from PLC devices. TankWidget stereotype facilitates to model tank objects and assign them unique identification defined in its attributes as tankID. TankWidget stereotype is extended from metaclasses Class and Instance Specification that enables it to be applied on multiple instances of tank widget in domain model. TankWidget is derived from another abstract stereotype named as Widget and thus inherits its properties as well.

«DialWidget»

Dial is a type of HMI widget that provides visual representation of a current value of a dial based on real time data. DialWidget stereotype enables to model dial objects and assign them unique identification number defined in its attributes as dialID. DialWidget stereotype is extended from metaclasses Class and Instance Specification that enables it to be applied on multiple instances of dial widget in domain model. DialWidget is derived from another abstract stereotype named as Widget and thus inherits its properties as well.

«BulbWidget»

Bulb is a type of HMI widget that displays on and off states of a bulb based on real time data. BulbWidget stereotype enables to model bulb objects and assign them unique identification number defined in its attributes as bulbID. BulbWidget stereotype is extended from metaclasses Class and Instance Specification that enables it to be applied on multiple instances of bulb. BulbWidget is derived from another abstract stereotype named as Widget and thus inherits its properties as well.

«TrafficLightWidget»

Traffic Light is a type of HMI widget that displays different states of a traffic light i.e. signal on, signal off and signal going to turn off based on real time data. TrafficLightWidget stereotype enables to model traffic light objects and assign them unique identification number defined in its attributes as trafficID. TrafficLightWidget stereotype is extended from metaclasses Class and Instance Specification that enables it to be applied on multiple instances of traffic light. TrafficLightWidget is derived from another abstract stereotype named as Widget and thus inherits its properties as well.

«SwitchWidget»

Toggle Switch is a type of HMI widget that displays on and off states of a toggle switch based on real time data. SwitchWidget stereotype enables to model toggle switch

objects and assign them unique identification number defined in its attributes as switchID. SwitchWidget stereotype is extended from metaclasses Class and Instance Specification that enables it to be applied on multiple instances of switch. SwitchWidget is derived from another abstract stereotype named as Widget and thus inherits its properties as well.

«RoomWidget»

Room is a type of HMI widget that allows to model rectangular areas for different purposes like control rooms, bed rooms etc. RoomWidget stereotype enables to model room objects for human machine interface. Unique identification number is assigned to this widget based on its attribute named as roomID. It allows settings for height and width of the RoomWidget in dp unit based on its owned attributes named as layout_width and layout_height respectively. RoomWidget stereotype is extended from metaclasses Class and Instance Specification that enables it to be applied on multiple instances of room. RoomWidget is derived from another abstract stereotype named as Widget and thus inherits its properties as well.

«LabelWidget»

Label is a type of widget that displays specified texts or labels on a mobile HMI screen. LabelWidget stereotype enables to model label objects for human machine interface. LabelWidget stereotype is extended from metaclasses Class and Instance Specification and has multiple attributes i.e. labelID, layout_width, layout_height, text, textSize and textStyle. labelID assigns unique identification to label widget. layout_width and layout_height set width and height of a label in dp units. Text attribute allows to enter text to be displayed on HMI screen. textStyle attribute helps to select style of a text as bold, normal or italic defined in the enumeration TextStyle. textSize sets the size of a text specified in dp unit.

«Tag»

Tag is basically a connection between HMI screen widget and the PLC device. In UMLPMH, mobile HMI acts as a client that requests DA Server for real time data from target PLC's. For that purpose, it is required to specify PLC device address for each widget to assign real time device data to the widget. Tag stereotype allows to specify PLC device address for each widget. Tag stereotype extends metaclasses Class and Instance Specification. It consists of attributes: widgetID that specifies id of a widget for which device data is requested, widgetType that specifies type of widget, i.e. tank, dial, bulb, toggle switch or traffic light defined in enumeration WidgetType, for which device data is requested and tagName that specifies name of a tag containing PLC device address.

«RefreshScreen»

Stereotype RefreshScreen is used to define time constraints on data requests from Modbus DA Server for HMI widgets. screenRefreshPeriod attribute allows to refresh a screen in loop after a specific time period. This time period is defined in seconds or minutes. With each screen refresh, request for device data is resent to server for each widget. autoRequest defines the condition if request to all devices has to be

generated in a loop or just once. RefreshScreen stereotype is extended from metaclass Class.

2) PLC DEVICE RELATED CONCEPTS

Device related concepts help to specify device address for an HMI widget to visualize and monitor real time devices data on HMI screens. Device address is based on IP address of a device, port number, destination or memory location of data, starting address and range of data. Device related concepts are managed through Device, Destination and Inputs stereotypes as shown in **Figure 3**.

«Device»

Device stereotype is extended from metaclasses Class, Slot and Property. Device stereotype represents the physical existence of the real time data based on PLC device. Each PLC device communicates and acquires data from a specific type of industrial machine or process so a unique identification number is assigned to each device as deviceID. PLC devices communicate with DA server through TCP/IP communication protocol and for that purpose each device has its own location identification address over the internet defined as ipAddress. To communicate with PLC devices a physical endpoint is defined and named as port.

«Destination»

The stereotype Destination represents the memory location of targeted PLC device and extended from both metaclasses of Class, Slot and Property. These memory locations are also known as registers or relay outputs which are used to read and write data on device memory. Four types of memory locations are present in a PLC device which are defined in the enumeration LocationType i.e. Coils, Inputs, Holding Registers and Input Registers.

«Inputs»

Stereotype Inputs represents the device memory physical address from where real time data is acquired. The attributes defined in this stereotype are; startingAddress that represents the starting address of a memory location for data and numOfInput attribute that specifies the number of consecutive locations from where data is acquired. Inputs stereotype is extended from metaclasses Class, Slot and Property.

In domain Model of mobile HMI, three stereotypes i.e. Device, Destination and Inputs are applied on an attribute deviceAddress of a class Tag to define a complete device address for a widget.

3) DATA SOURCE RELATED CONCEPTS

Data source related concepts help to define connection mechanism to DA Server for request and fetch real time device data through PLC. It is based on DataSourceConnection stereotype, as shown in **Figure 3**, to specify IP address and port for a connection to DA Server.

«DataSourceConnection»

DataSourceConnection stereotype helps to define concepts required for connection with DA server to request devices data and transfer the response to the HMI widgets. DataSourceConnection stereotype extends metaclass Class.

It consists of two attributes i.e. serverIPAddress that defines the location address of the server over TCP network and port attribute that defines the endpoint of the communication for a server.

4) DATA TYPES AND ENUMERATIONS

It is required to define certain data types and enumerations to achieve the mobile HMI modeling through UMLPMH. Therefore, three (3) enumerations (i.e. LocationType, TextStyle and WidgetType) are defined as shown in **Figure 3**. Particularly, enumeration LocationType represents four types of Modbus memory registers used for HMI widgets i.e. Coils that are based on binary on/off outputs, Inputs are based on read only binary inputs, Holding_Registers are based on analog parameters that can be changed and Input_Registers are based on read only analog inputs. TextStyle enumeration defines different text styles for widget labels i.e. bold, normal and italic. Enumeration WidgetType deals with types of HMI widgets i.e. Tank, Dial, ToggleSwitch, Bulb and TrafficLight.

We have extended metaclasses Class, Slot, Instance specification and Property to define fourteen stereotypes in UMLPMH. Out of these stereotypes, ten stereotypes are proposed for HMI widgets, three stereotypes are proposed for defining PLC settings for widgets and one stereotype is proposed for connection with DA server. The practical demonstration of these stereotypes is given in Section V where three case studies (i.e. Home Automation, Water Treatment Plant and Traffic Light Control System) are modelled using UMLPMH.

IV. IMPLEMENTATION

This section provides the implementation details of the proposed transformation engine, Model Driven Mobile-based HMI Code Generator (MDMHCG), which transforms a UML Class Diagram (.uml) modelled through UMLPMH into android code for HMI application. Section A deals with the architecture of the proposed transformation engine and Section B provides transformation rules.

A. TRANSFORMATION ENGINE ARCHITECTURE

The architecture of our transformation engine is presented in **Figure 5**. MDMHCG takes input models and transforms them into HMI android code using transformation rules. It is implemented in JAVA and Acceleo Tool (for writing transformation rules). MDMHCG is based on Model to Text Language (MTL) standard. The transformation Engine MDMHCG, is based on two main components i.e. Tool User Interface and Code Generator. Detailed description of these components is as follows:

Tool User Interface: Main interface of MDMHCG is presented in **Figure 4**. It provides two main functionalities i.e. System Modeler and Transformation Engine. System Modeler provides a facility to open Papyrus tool to model mobile HMI applications. Transformation Engine option allows to provide input model and transform it into HMI android code through transformation engine interface as

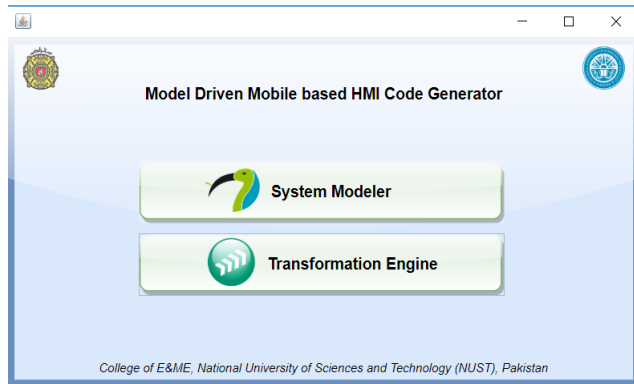


FIGURE 4. Main interface of model driven mobile based HMI code generator (MDMHCG).

shown in **Figure 10**. This transformation interface comprises of: Input Model allows to browse through and select available .uml files in system. Destination Folder allows to set path for generated output files. Generate button performs the action of automatically generating mobile HMI code files from input model. Status shows current state of tool. Reset button allows to enter new input model and destination folder. Open folder directs to the folder where output files were generated.

User interface of MDMHCG is developed using four main java classes i.e. Launcher, MainScreen, TextRefiner and WinMain. MainScreen serves as the main executor of transformation engine. It provides a graphical user interface (GUI) including buttons and input fields. Launch and WinMain are the java based controller classes that implement these functionalities. Text Refiner does the string manipulation in a required format for further use.

Code Generator: The specified input models are fed into code generator that processes these models into deployable android HMI code using transformation rules. Code generator is based on two main components i.e. Generate (Generate.java) and Template (generate.mtl) files to implement transformation rules. The main component is Template file that is based on multiple sub templates. It fetches the input models and passes them to its respective sub templates. Each sub template applies its transformation rules on each UML model element to generate the output. The output artifacts are based on deployable android HMI code and a widgets designer (.axml) code.

It is important to note that the source code of transformation engine (MDMCG) is publically available [45] for evaluation. Furthermore, UMLPMH profile, sample case studies along with domain model and comprehensive user manual can also be found at [45].

B. TRANSFORMATION RULES

Model transformation is the process of automatic generation of output artifacts (model, code, text, documentation etc.) from an input source model based on transformation rules [33]. Transformation rules are the set of formal definitions [34] that define how one or more constructs in source

model language map to one or more constructs in target model language [18]. While developing transformation rules, main focus is to reduce overall efforts and information loses during transformation process [17].

1) TRANSFORMATION RULES FOR CODE GENERATION














The process of generating code or text artifacts through Model to Text (M2T) transformation is carried out in transformation engine [37]. The transformation rules required to generate low level mobile HMI android code from UMLPMH models are given in **Table 2**.

Model Artifacts are based on elements of Unified Modelling Language (UML) for modeling mobile HMI system. Code Artifacts define android code elements that are converted during transformation from UML model elements. Mapping maps UML model elements to their respective code elements. Package is an element of UML that groups together other packageable elements and provides a hierarchical view of the model. During transformation process, the Name of a Package is mapped to the name of a folder containing code files. Model Class describes a static structure of a system in UML. The Name of a Model Class is mapped to the name of class in code by prefixing keyword class. Visibility constrains the usage of a named element by allowing four types of accessibility options i.e. public, private, protected and package.

The Visibility of a Model Class is mapped to the access specifier of code class by prefixing it to the class keyword. Model class Attributes, Attributes' Visibility and Applied Stereotypes' Property are mapped to the class owned attributes and visibility of attributes in code. Association provides a semantic relationship between classifiers in UML and its Member Ends are mapped to the instances of the Owned Class as attributes of an Owning Class in code artifact. Member Ends Visibility and Multiplicity are mapped to the access specifiers and cardinality of the instances of the Owned Class respectively. Cardinality '1' is mapped to single instance and cardinality '0..*' is mapped to List<Type> instance.

Operations define behavioral feature of a Model Class in UML and its Name is mapped to the method name of a class in code. Name of Owned Parameters is mapped to the name of parameters of class and direction of Owned Parameters is mapped to the type of parameters i.e. in, out, inout and return type of method. Opaque Behavior is based on implementation specific semantics in UML like a block of code. Description of Opaque Behavior is mapped to the body of method or method definition in code. Enumeration is a user defined list of named elements in UML. The Name of Enumeration is mapped to the name of Enum in code by prefixing enum keyword to the name. Enumeration Literals are mapped to named values of Enum in code. Widgets define five types of HMI widgets (*Tank, Dial, Traffic Light, Bulb and Switch*) for modelling mobile HMI system. Name of Instance Specification of an HMI widget is mapped to the id of ImageView tag in code artifact. Classifier Name of Instance Specification is mapped to the widget type in code.

TABLE 2. Transformation rules for translating UMLPMH models into mobile HMI android code.

Model Artifacts		Code Artifacts (Xamarin Android)	Mapping
Package			
Package Name		Folder	Package Name → Folder Name
Model Class			
Class Name		Class	Model Class—Name → class Name
Visibility			Model Class—Visibility → Access specifiers class
Attribute			Model Class—Attribute → Owned Attributes
Attribute Visibility			Model Class—Attribute Visibility → Owned Attribute Access Specifiers
Applied Stereotype			Model Class—Applied Stereotype—Property → Owned Attributes
Association			
Member Ends		Class Instance	Association—Member ends → Owning Class/ Owned Class.
Member Ends Visibility		public/ private/ protected	Association—Member ends visibility → Owned Instance Access specifier.
Member Ends Multiplicity		List<Type> Instance Name OR Single Instance	Association—Member ends Multiplicity/ Cardinality → List<Type> or Single Instance.
Operation			
Operation Name		Method	Operation—Name → Method Name.
Owned Parameter		Method Definition	Operation—Owned Parameter → Method Parameter (in, out), Return Type of Method (return).
Opaque Behavior			Opaque Behavior—Description → Body of Method.
Enumeration			
Enumeration Name		enum Enumeration	Enumeration—Name → enum Name
Owned Literal		Enumeration Literal	Enumeration—Owned Literal → Enum Values.
Widget			
Instance Specification		<ImageView>	Instance Specification—Name → ImageView ID
Instance Specification Classifier Name		Widget Type	Instance Specification—Classifier Name → Widget Type (Tank, Dial, Bulb, Traffic Light, Toggle Switch)
Instance specification Applied Stereotype		Owned Attributes	Instance Specification—Applied Stereotype—Property → Owned Attributes of widget instance

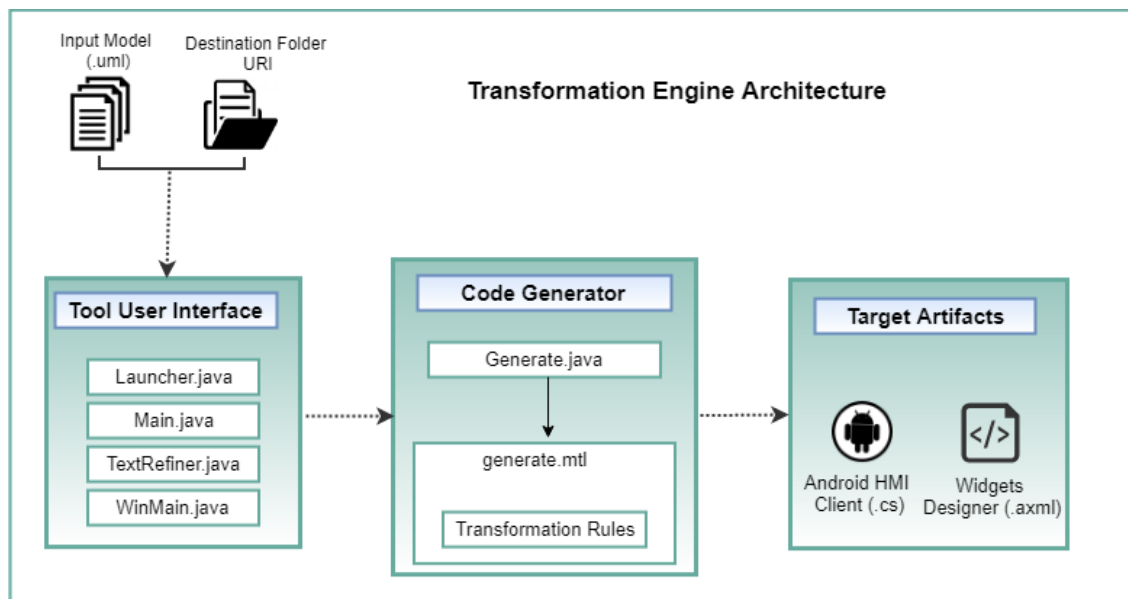


FIGURE 5. Architecture of transformation engine.

Similarly Applied Stereotype's Property of Instance Specification is mapped to the HMI instance's owned attributes in code artifact. In MMHF, five different types of widgets have been introduced, i.e., *traffic light, bulb, toggle switch, dial & tank widgets*. In this regard, complex transformation rules haach widget to successfully generate corresponding android code in order to achieve proper real time visualization as per the values of sensors. In this regard, it is only required to include the desired widget in HMI model and set the given properties accordingly. The developed transformation rules generate accurate corresponding android code, e.g., *data connectivity, positioning* of widget in screen etc.

V. VALIDATION

This section deals with the validation of the proposed framework with the help of three case studies i.e. *Home Automation, Sewage Treatment Plant* and *Traffic Control System* case studies, which prove the applicability and usefulness of the proposed framework.

A. EXPERIMENTAL SETUP

We have used open source Modbus based Data Acquisition (DA) sever, which is developed in our institution, for the validation. To develop realistic environment, *five (5)* Fatek PLC's [13] are configured with 1000 tags. Fatek PLC's provide standard support for Modbus protocol, therefore, the employed DA server is able to perform both read and write operations on PLC's. Particularly, we configured all four types of tags (i.e. *coils, discrete inputs, holding and input registers*) in order to meet the requirements of all case studies. It is important to note that the actual sensors are not attached with the PLC's due to limited availability of resources. However, proper read and write operations are performed on PLC's through DA server in order to visualize the effects of data change on generated mobile HMI screens. We have

designed and transformed several mobile HMI screens for first two case studies (i.e. Home Automation and Sewage Plant). *Here, we are only providing the design, transformation and deployment details of one HMI screen for each case study*. In this regard, the UMLPMH profile, sample case studies along with *domain model, user manual* and *source code* of transformation engine (MDMCG) can be found at [45] for further evaluation.

B. HOME AUTOMATION CASE STUDY

1) REQUIREMENTS

With the advancement of technology, home automation has grown rapidly in the past few years [41]. It gives access to control devices in his home from a mobile device, tablet or a PC, anywhere in the world [25]. Home automation systems include nearly everything e.g. *fans, lights, switch buttons, heating and cooling systems etc*. Home Automation is a step towards IoT [26], where everything has an assigned IP address and can be controlled and monitored remotely.

In this section, a home automation system is designed to validate our proposed framework. This system involves *TV lounge, bedroom* and *kitchen* with various type of HMI widgets i.e. *bulbs, buttons, AC temperature sensors* and *electric power consumption sensors*. The MMHF allows to model home automation HMI widgets, assign them PLC device values and automatically generate android code. Subsequently, the generated code can be deployed on target mobile devices for monitoring home devices. Particularly, the Fatek PLCs are configured with several tags to provide connectivity with different sensors e.g. *lights, switch buttons, fans, AC temperature sensors, etc*. The generated mobile HMI client is able to request the data of 512 Input Output (IO) tags of PLCs which are configured on different IP addresses. The distribution of tags are as follows:

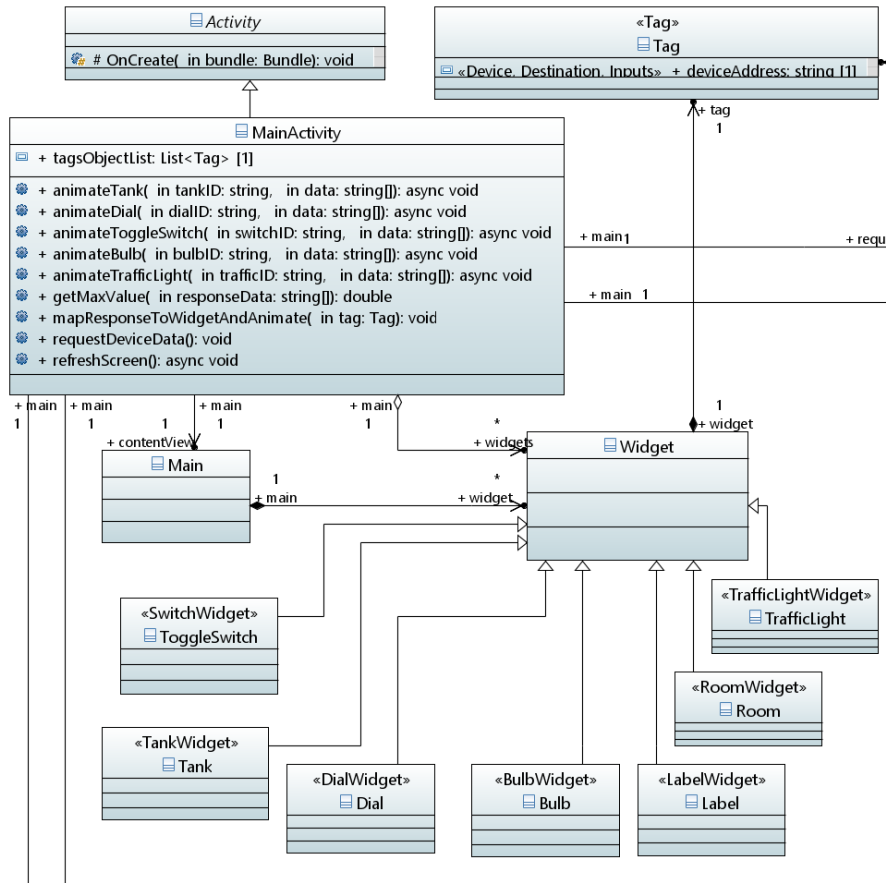


FIGURE 6. Domain model classes for HMI widgets.

- 250 IO tags belong to Bulbs/Lights.
- 62 IO tags belong to different types of switches e.g. AC, Heater etc.
- 150 IO tags belong to Power Consumption Sensors.
- 50 IO tags belong to Temperature Sensors.

2) MODELING

Complete domain model of a mobile HMI screen for Home Automation case study is shown in **Figure 6** and **Figure 7**. This model contains two types of main concepts i.e. HMI Widgets and Data Source Connection concepts. The model of HMI screen composes multiple classes along with necessary applied *stereotypes*, *attributes* and *methods* for proper execution of the system. HMI widget concepts are based on *twelve* classes; Activity and Main Activity classes handle main operations of the system. They define working / visualization of widgets based on acquired real time device data. Main class is used to define layout and properties of HMI widgets. Tag class is used to assign PLC address and address space to each widget based on which request is sent to DA server. Widget, ToggleSwitch, Tank, Dial, Bulb, Label, Room and TrafficLight classes are used to define the HMI widgets. Data source connection concepts of model are based on *ten* classes; RequestHandler and ConnectionHandler are used to initiate device requests to DA server and receive response from server back. IPValidator class performs validation of IP addresses

of devices. Device, Input and Destination classes are used for storing and transmitting device requests. ResponseData class stores and transmits response from server. Constant class defines necessary static attributes of the model. Utility and StateObject are *support classes* which provide support like converting data in proper format for transmission.

Figure 8 shows *instance model* for home automation system. Multiple instances of HMI widgets have been modelled and have assigned them tag values based on device address properties. These instances comprise of widgets like *bulb*, *toggle button* and *dial* along with tag class instances for specifying device addresses. **Figure 9** shows how values are assigned to these instances in papyrus class diagram.

It is important to note that we developed several mobile HMI screens for home automation case study. Here, we are including the details for *one screen* only. In this regard, MMHF is publically available [45] for further evaluation.

3) CODE GENERATION

The domain model of Home Automation system is given as an input model to MDMHCG as shown in **Figure 10**. It then applies transformation rules and algorithms to transform input domain model (.uml file extension) into deployable android source code for HMI screen as shown in **Figure 11**.

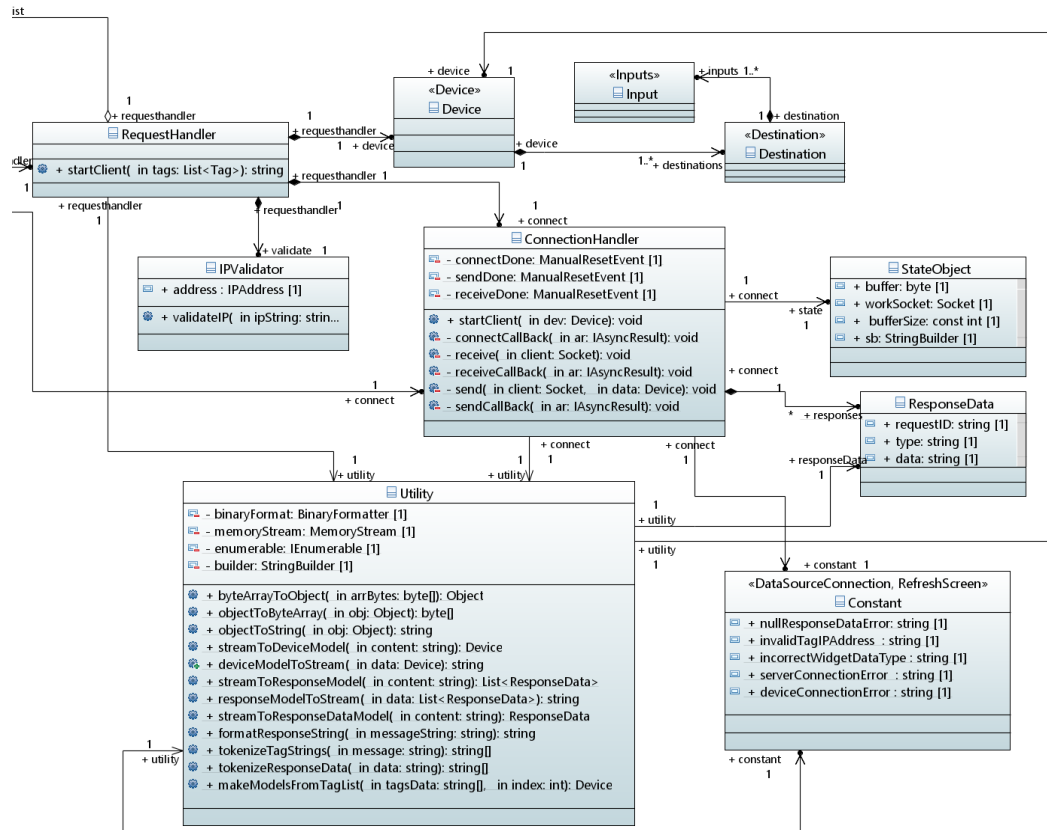


FIGURE 7. Domain model classes for data source connection concepts.

The generated outputs are based on following files and folders:

- 1) HMIcode folder contains ‘.cs’ files for the development of android HMI application.
- 2) WidgetsDesigner folder contains a ‘.axml’ file for the properties and layout of HMI widgets.

4) DEPLOYMENT

For verification of generated android code, compilation and execution is necessary. For this purpose, we have used *Visual Studio 2015* for compilation and execution of code. We created new *Xamarin Android project* [11] in visual studio and pasted our generated code files into an empty project as shown in **Figure 13**. After the copying files of generated code in visual studio, it first compiles code by checking for the syntax errors. Upon successful compilation, generated code files are then executed to display the output of mobile HMI screen through real time values of PLCs. *It is important to note that we have performed several read / write operations on tags through DA sever in order to confirm the proper visualization effects of mobile HMI screen.* Furthermore, write operations are also successfully performed through mobile HMI screen. In this regard, **Figure 12** (a) represents a running Home Automation HMI system on Android Emulator in processing state to fetch devices data through DA server. **Figure 12** (b) shows HMI widgets successfully displaying devices data requested through server.

It is important to note that the UMLPMH profile, sample case studies along with domain model, user manual and source code of transformation engine (MDMCG) can be found at [45] for further evaluation.

C. SEWAGE TREATMENT PLANT CASE STUDY

1) REQUIREMENTS

Automation processes are being introduced into Sewage Treatment Plants to increase reliability and safety of the process and reduce development and maintenance costs [27]. These automation systems control almost everything from electricity supply to the processes to the safe discharge of waste water from plants. The MMHF helps to develop mobile HMI applications for waste water sewage treatment plants to monitor treatment processes remotely. In this case study, several mobile HMI screens are developed through MMHF. Here, an example mobile HMI screen is demonstrated for control room. Particularly, the Fatek PLCs are configured with several tags to provide connectivity with different sensors e.g. control switches, PH value sensor, solid flow meter, ferric chloride and decantation tanks etc. Generated mobile HMI client is able to request the data of 800 Input Output (IO) tags of PLCs which are configured on different IP addresses. The distribution of tags are as follows: -

- 250 IO tags belong to temperature sensors.
- 100 IO tags belong to liquid level sensors.
- 150 IO tags belong to PH value meters.

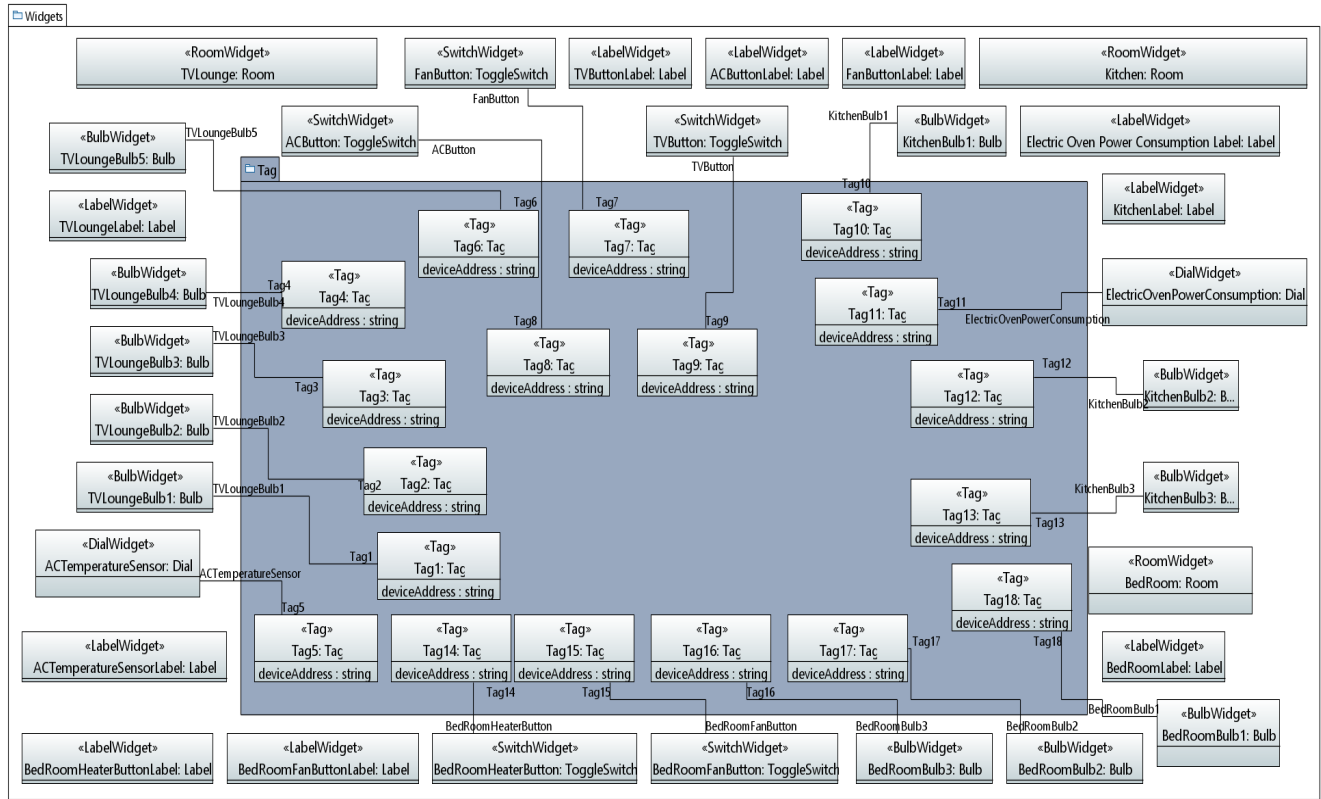


FIGURE 8. Instance model of home automation system containing widgets and devices information.

- 80, 90, 60 and 70 IO tags belong to Solid flow meters, Pressure transducers, Point level switches and Speed sensors respectively.

2) MODELING

As we have discussed earlier, the MMHF provides a generic method to develop mobile HMI screens. Therefore, the domain model that we have discussed in previous case study, can be implemented here as well. However, it is required to design instance specification model for each HMI screen separately as per requirements. The instance specification model for control room of sewage treatment plant is given in Figure 14.

3) CODE GENERATION AND DEPLOYMENT

The model file (.uml extension) is given as an input to the MDMHCG. It transforms the input model, by applying transformation rules and algorithms and meta-level concept, to the android source code for mobile HMI screens. The generated files are based on two folders HMIcode and WidgetsDesigner as shown previously in Figure 11 for home automation system.

For the verification of generated android code, we have paste it to Visual Studio 2015, Xamarin Android component as already explained for home automation case study in Figure 13. Finally, the execution of generated mobile HMI screen is performed as shown in Figure 15, where Figure 15 (a) represents a running Sewage Treatment Plant system on Android Emulator in processing state to

fetches devices data through DA server. Figure 15 (b) shows HMI widgets successfully displaying devices data requested through server.

D. TRAFFIC LIGHT CONTROL SYSTEM

Traffic light control systems are widely used to monitor and control the flow of automobiles through the junction of many roads. They aim to realize smooth motion of cars in the transportation routes. HMIs are being introduced into the traffic light control systems that allows operators to remotely monitor and control all activities of a traffic light system [40]. In this case study, a traffic signal HMI widget is used at the junction of four roads to control traffic remotely. The MMHF helps to develop mobile based HMI system for traffic control system that comprises of three main components i.e. PLC, DA server and HMI mobile client. PLC devices acquire real time data for states of signal lights, countdown timers, density of vehicles at different lanes etc. DA server communicates with PLC to provide real time data requested by HMI mobile client. The instance specification model for traffic light control system is shown in Figure 16. Subsequently, the functional HMI screens are shown in Figure 17 and Figure 18.

VI. DISCUSSION AND LIMITATIONS

This article presents a Model-driven Mobile HMI Framework (MMHF) to develop the native mobile HMI screens for industrial control systems. This is the first real

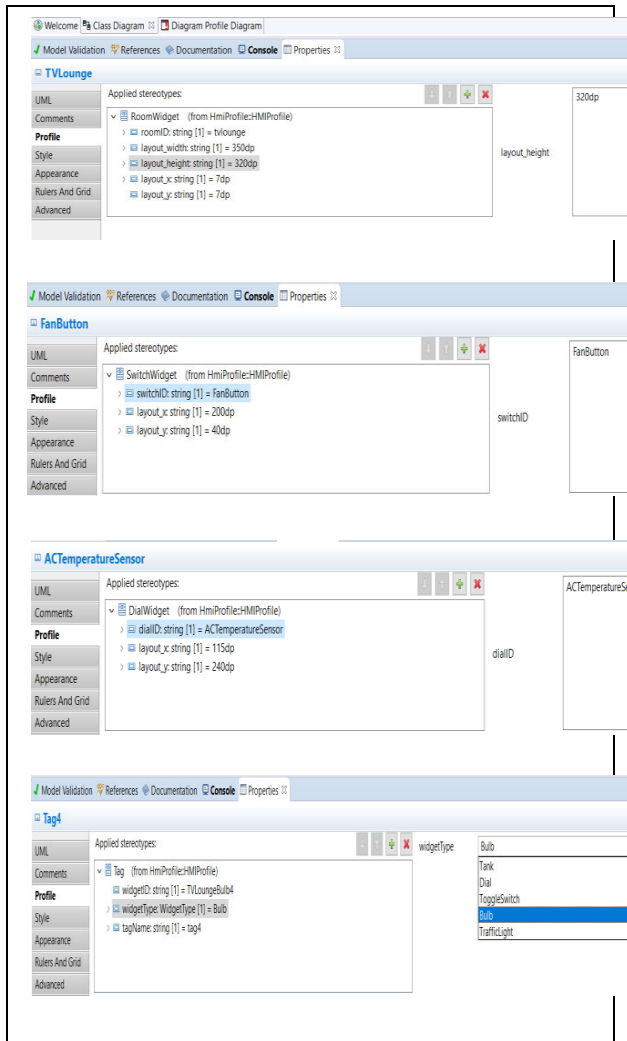


FIGURE 9. Assigning values to the instances of home automation system.

step towards the cost effective and scalable mobile HMI solution for industry automation because:

1) Existing industrial tools like Genesis64 are highly expensive and usually provided with pay per tag policy. On the other hand, MMHF is publically available free of cost. Furthermore, it is capable of designing various HMI screens for different industrial control systems. Therefore, MMHF is significantly cost effective solution as compared to proprietary tools. As far as state-of-the-art is concerned, it is hard to find wide-ranging open source framework particularly for the development of native mobile HMI screens.

2) Industrial tools do not provide any information about the underlying development techniques and tools. Consequently, the extension of industrial tools in academia is not possible. On the other hand, MMHF underlying development approaches and supporting tools are explicitly stated in this article. Furthermore, it is publically available [45]. Consequently, the researchers and practitioners of the domain can easily extend the current features of MMHF. Particularly, it opens the door for academia to propose different industry

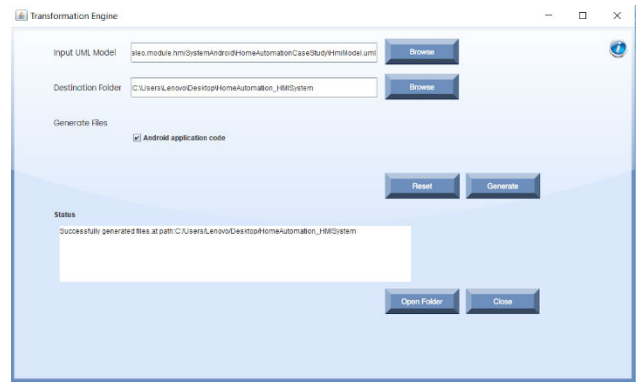


FIGURE 10. Transformation of UMLPMH Models into mobile HMI android code for home automation system.

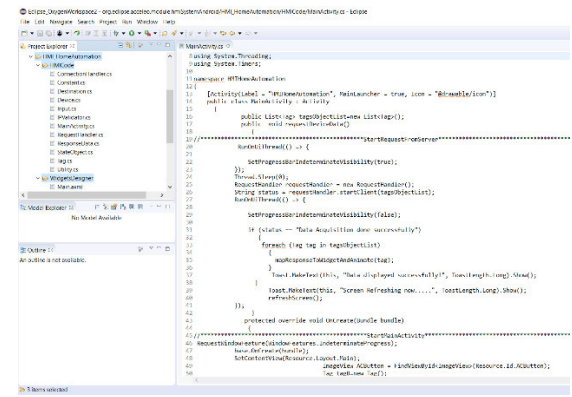


FIGURE 11. Generated code files for home automation system.

automation techniques for HMI on the basis of MMHF. Therefore, the proposed framework in this article is highly scalable.

It can be argued that the development of HMI screens in MMHF is a bit complex because it requires to understand the UML modeling concepts for the development of HMI screens. We admit that MMHF is not currently providing the drag and drop facilities for HMI screen development as provided by industrial tools. Actually, MMHF is a solid foundation and several functionalities can be incorporated in it due to its highly scalable features. For example, the drag and drop features can be included in the MMHF with minimal efforts by exploiting the features of Sirius framework.¹ Particularly, the concepts of UMLPMH can be exported to Sirius framework to develop a full tool for design mobile HMI screens with sophisticated drag and drop facilities. Similarly, it can be argued that MMHF is not currently providing the advanced HMI development features like dynamics (e.g. Flash etc.), 3D support etc. Therefore, it is difficult to develop HMI screens through MMHF in real industries. We admit that such advanced features are missing in MMHF, however, we provide the proof-of-concept by incorporating few important widgets and proper visualization is achieved in real time as per configured settings and according to the

¹ <https://www.eclipse.org/sirius/>

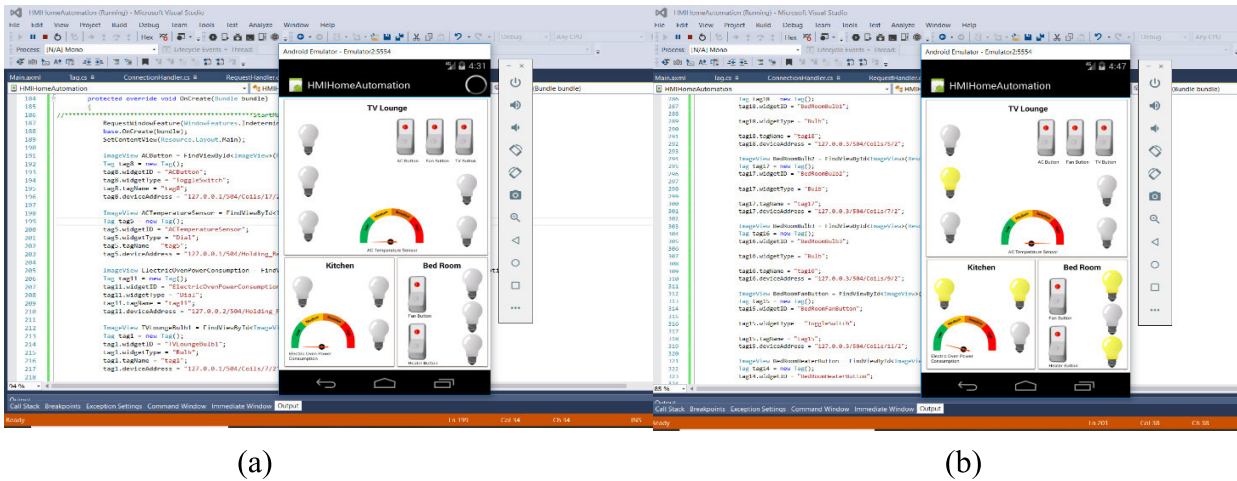


FIGURE 12. Home automation system’s HMI connecting with DA server and displaying real time state of home system through widgets in (a) and (b) respectively.

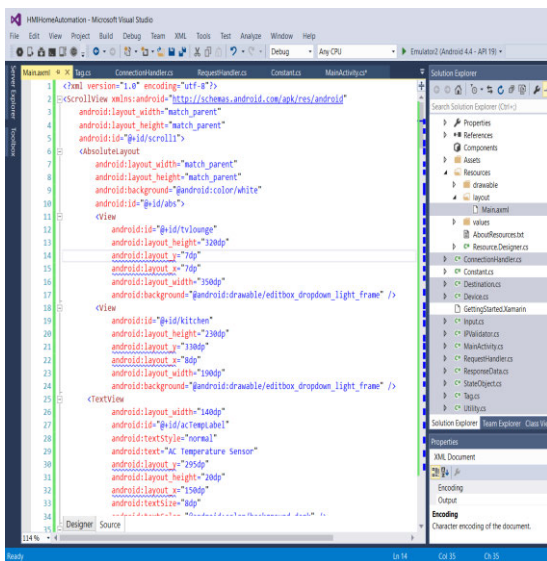


FIGURE 13. Deployed code for mobile HMI of home automation system.

values of sensors. Therefore, more widgets and dynamic can be included in the MMHF in similar manner.

Another important aspect of MMHF is the connectivity support for data acquisition sever. Particularly, we use open source *Modbus based* data acquisition sever, which is developed in our institution, for the validation of MMHF. However, we provide complete connectivity mechanism in MMHF to support the data acquisition from standard technologies like OPC UA. Particularly, we provide different connectivity features like: *number of devices* and *corresponding IP address*, etc. in UMLPMH. Furthermore, the generated native HMI client *sends and receives response* from DA server through *value, quality and timestamp* attributes as defined in the standard OPC DA server specifications,² e.g., 192 value of quality belongs to *Good* etc.

Therefore, native mobile HMI clients of MMHF can easily communicate with *OPC based data acquisition server*. In fact, we successfully tested the connectivity of native HMI clients with classic Matrikon OPC server³ (Modbus) for evaluation purposes.

It can be argued that the validation of MMHF is questionable as it is not yet applied in any real industry. Basically, MMHF is solely developed in academia without any funding or industry support. Therefore, we have certain limitations of resources like sensors, real devices etc. However, we have tried our level best to develop a real industry environment as much as possible for the validation of MMHF. For example, we have used *real PLC’s* available in the institution’s control lab. Moreover, we have properly configure more than *1000 IO tags* in PLC’s in order to perform realistic Read and / or Write operations through mobile HMI screens. Furthermore, the correctness of visualizations on data change is comprehensively evaluated in the generated mobile HMI screens. Particularly, we have tested generated mobile HMI screens with 1000 IO tags without any lag. Therefore, MMHF is fully capable to be applied in most of the real industries without any issues for the development of mobile HMI screens. In fact, we are currently contacting several industries in this regard.

During the investigation of existing solutions for mobile HMI development, we found few small applications for the android based HMI screens on Google Play store. Particularly, Virtuino Modbus [12] and HMI Modbus applications are providing few realistic features for android HMI screens. However, the underlying development mechanism and source code is not given, therefore, scalability is an important issue for such applications. Furthermore, proper working of such applications is highly questionable because we encountered a significant lag on HMI screens after 300 tags while evaluating Virtuino Modbus [12] application. On the other hand, we have

² <https://opcfoundation.org/about/opc-technologies/opc-classic/>

³ <https://www.matrikonopc.com/downloads/types/drivers/index.aspx>

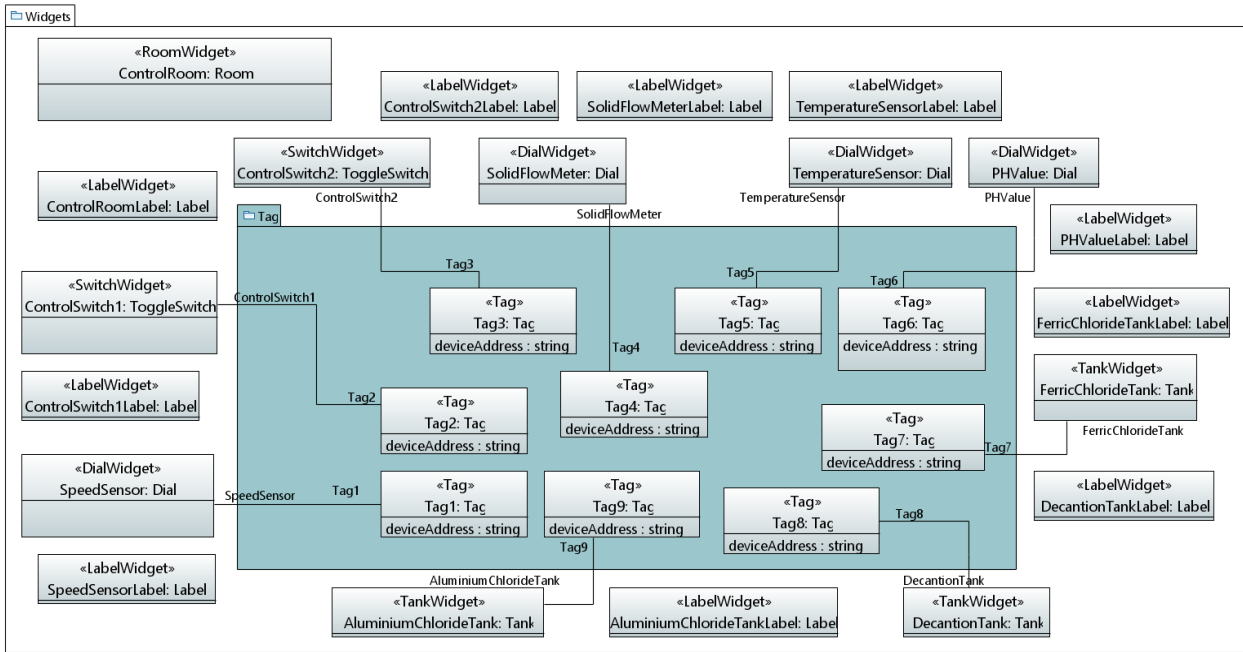
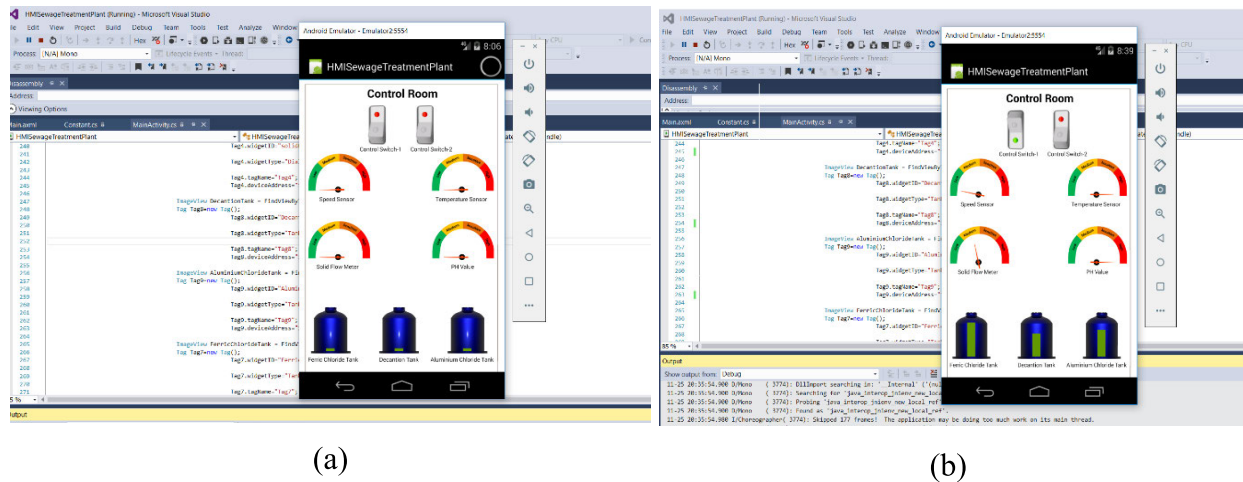


FIGURE 14. Instance model of water treatment plant containing widgets and devices information.



(a)

(b)

FIGURE 15. Water treatment system’s HMI connecting with DA server and displaying real time state of system through HMI widgets in (a) and (b) respectively.

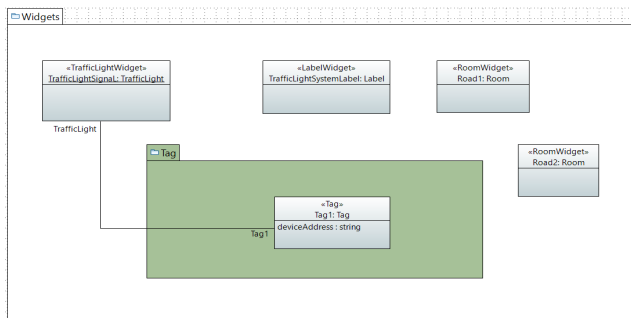


FIGURE 16. Instance model of traffic light control system containing widgets and devices information.

successfully tested MMHF mobile HMI client on 1000 tags without any lag.

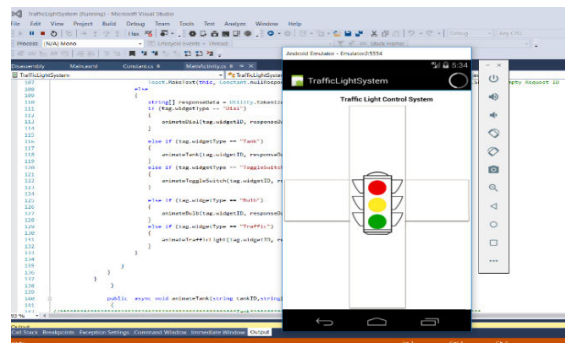


FIGURE 17. Traffic light control system’s HMI connecting with DA server for devices data.

To summarize, MMHF is a feasible, cost effective and scalable solution for the development of native HMI screens.

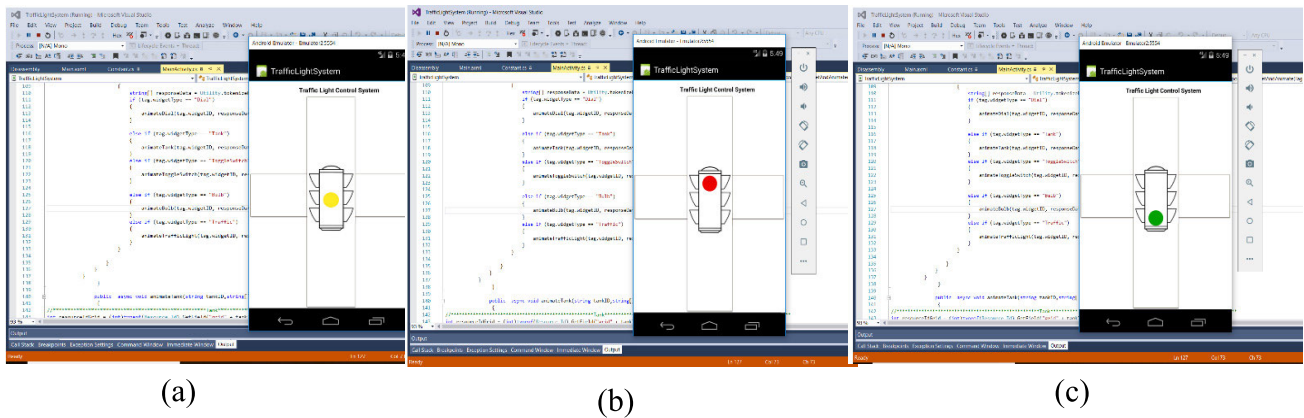


FIGURE 18. Traffic light control system's HMI displaying "Check", "Stop and "Start" states of traffic signal in (a), (b) and (c) respectively.

Particularly, it provides certain benefits to the researchers and practitioners of the domain, such as: -

- **Ready to use solution:** MMHF provides a model template to add widgets and set their properties accordingly for designing desired HMI screen. Subsequently, the design of HMI screen represented through model can be automatically transformed to low level android implementations with a click of button. Finally, the generated android code for HMI screens can be directly deployed to the target mobile devices in order to visualize and control the industrial processes.
- **Reduced complexity:** MMHF provides abstract modelling environment for HMI designing, instead of directly coding the native HMI using extensive low-level programming. This significantly reduces development complexity in terms of time and cost.
- **Scalability:** MMHF is highly scalable as both UMLPMH and MDMHCG are fairly extendable. For example, it is straightforward to include more industrial widgets in UMLPMH for designing complex HMI screens. Similarly, MDMHCG is publically available and existing transformation rules can be upgraded with simplicity in order to generate native HMI screens with advanced visualization features.

VII. CONCLUSION AND FUTURE WORK

This article presents a **Model-driven Mobile HMI Framework (MMHF)** for the development of native mobile HMI screens for (ICSs). Particularly, the design of HMI screens can be accomplished through a UML Profile for Mobile HMI (UMLPMH) which is developed as a part of MMHF. Furthermore, an open source transformation engine, Model Driven Mobile-based HMI Code Generator (MDMHCG), is implemented in MMHF to automatically transform UMLPMH models into target native mobile HMI implementations. The viability of MMHF is demonstrated through three bench mark industrial case studies. The experimental results prove that the MMHF is a feasible, cost effective and scalable solution for developing pure native HMI screens in wide-ranging ICSs.

Unlike industrial HMI tools, the underlying development methodologies of MMHF are explicitly given in this article. Furthermore, MMHF is available publically for further enhancements. Consequently, it provides an opportunity for further future work for academia to work on different HMI approaches based on this novel idea of MMHF. For example, the concepts of UMLPMH can be extended to incorporate more widgets and dynamic in MMHF in order to support the design of complex HMI screens. Moreover, the concepts of UMLPMH can be exported to Sirius framework to develop a full designing tool with sophisticated drag and drop facilities. Furthermore, the extension of MDMHCG is fairly possible to generate both desktop and web HMI screens implementations along with android code. Finally, to summarize, as a future work, several extensions of MMHF are possible as per the demanding requirements of HMI screens for modern ICSs.

REFERENCES

- [1] T. Skripcak, P. Tanuska, U. Konrad, and N. Schmeisser, "Toward non-conventional human-machine interfaces for supervisory plant process monitoring," *IEEE Trans. Human-Mach. Syst.*, vol. 43, no. 5, pp. 437–450, Sep. 2013.
- [2] V. Villani, L. Sabatini, J. N. Czerniak, A. Mertens, B. Vogel-Heuser, and C. Fantuzzi, "Towards modern inclusive factories: A methodology for the development of smart adaptive human-machine interfaces," 2017, *arXiv:1706.08467*. [Online]. Available: <https://arxiv.org/abs/1706.08467>
- [3] M. W. Anwar and F. Azam, "Proposing a novel architecture of script component to incorporate the scripting language support in SCADA systems," in *Computer Information Systems and Industrial Management (Lecture Notes in Computer Science)*, vol. 8838, K. Saeed and V. Snášel, Eds. Berlin, Germany: Springer, 2014.
- [4] M. Jain and S. Tolety, "An optimized design approach for extending HMI systems with mobile devices," in *Proc. Companion 36th Int. Conf. Softw. Eng.*, 2014.
- [5] T. Lojka, P. Satala, J. Mocnej, and I. Zolotova, "Web technologies in industry HMI," in *Proc. IEEE 19th Int. Conf. Intell. Eng. Syst. (INES)*, Sep. 2015, pp. 103–106.
- [6] C. Martin, M. Freund, A. Braune, R.-E. Ebert, M. Plebow, S. Severin, and O. Stern, "Integrated design of human-machine interfaces for production plants," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Automat. (ETFA)*, Sep. 2015, pp. 1–6.
- [7] H. S. Saini and R. Daruwala, "Human machine interface in Internet of Things system," in *Proc. Int. Conf. Comput. Commun. Control Automat. (ICCUBEA)*, Aug. 2016, pp. 1–4.

- [8] E. Normanyo, F. Husinu, and O. R. Agyare, "Developing a human machine interface (HMI) for industrial automated systems using siemens simatic WinCC flexible advanced software," *J. Emerg. Trends Comput. Inf. Sci.*, vol. 5, no. 2, pp. 134–144, 2014.
- [9] G. Jayanthi, S. Arunachalam, K. Praveen, and K. P. S. Unni, "Cost effective SCADA for remote monitoring and control for effective process automation using HMI," in *Proc. Int. Conf. Power, Energy, Control Transmiss. Syst. (ICPECTS)*, Feb. 2018, pp. 342–346.
- [10] F. Ekman, M. Johansson, and J. Sochor, "Creating appropriate trust in automated vehicle systems: A framework for HMI design," *IEEE Trans. Human-Mach. Syst.*, vol. 48, no. 1, pp. 95–101, Feb. 2018.
- [11] *Xamarin Android*. Accessed: Oct. 2019. [Online]. Available: <https://docs.microsoft.com/en-us/xamarin/android/>
- [12] *Virtuino Modbus*. Accessed: Oct. 2019. [Online]. Available: <https://virtuino.com/index.php/virtuino-modbus>
- [13] *Fatek PLC*. Accessed: Oct. 2019. [Online]. Available: <http://www.fatek.com/en/prod.php?catId=1>
- [14] B. Dorninger, W. Beer, M. Moser, R. Zeilinger, and A. Kern, "Automated reengineering of industrial HMI screens by static analysis," in *Proc. IEEE Emerg. Technol. Factory Automat. (ETFA)*, Sep. 2014.
- [15] Y. Atarashi, M. Morita, and N. Koga, "HMI development of derived products by model-driven prototyping tool for in-vehicle system," in *Proc. 5th IIAI Int. Congr. Adv. Appl. Inform. (IIAI-AAI)*, Jul. 2016, pp. 1072–1077.
- [16] A. Ramaswamy, B. Monsuez, and A. Tapus, "Model driven software development for human-machine interaction systems," in *Proc. ACM/IEEE Int. Conf. Hum.-Robot Interact. (HRI)*, 2014, pp. 270–271.
- [17] M. W. Anwar, M. Rashid, F. Azam, M. Kashif, and W. H. Butt, "A model-driven framework for design and verification of embedded systems through SystemVerilog," *Des. Automat. Embedded Syst.*, vol. 23, nos. 3–4, pp. 179–223, Dec. 2019, doi: 10.1007/s10617-019-09229-y.
- [18] K. M. Aziz, "Evaluating model transformation technologies—an exploratory case study," B.S. thesis, Dept. Comput. Sci. Eng., Univ. Gothenburg, Gothenburg, Sweden, 2011. Accessed: Jan. 2020. [Online]. Available: <https://gupea.ub.gu.se/handle/2077/27847>
- [19] O. Neira, A. N. Lee, J. L. M. Lastra, and R. S. Camp, "A builder for adaptable human machine interfaces for mobile devices," in *Proc. 11th IEEE Int. Conf. Ind. Inform. (INDIN)*, Jul. 2013, pp. 750–755.
- [20] M. Willocx, J. Vossaert, V. Raes, and V. Naessens, "Using android devices as mobile extensible HMIs," in *Proc. 5th Int. Conf. Internet Things, Syst., Manage. Secur.*, Oct. 2018.
- [21] D. Akdur, V. Garousi, and O. Demirors, "A survey on modeling and model-driven engineering practices in the embedded software industry," *J. Syst. Archit.*, vol. 91, pp. 62–82, Nov. 2018.
- [22] I. Qasim, F. Azam, M. W. Anwar, H. Tufail, and T. Qasim, "Mobile user interface development techniques: A systematic literature review," in *Proc. IEEE 9th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Nov. 2018.
- [23] H. Tufail, F. Azam, M. W. Anwar, and I. Qasim, "Model-driven development of mobile applications: A systematic literature review," in *Proc. IEEE 9th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Nov. 2018.
- [24] H. Tufail, M. Waseem Anwar, I. Qasim, and F. Azam, "Towards the selection of optimum alarms system in leading industry automation software," in *Proc. 8th Int. Conf. Ind. Technol. Manage. (ICITM)*, Mar. 2019.
- [25] I. Majumdar, B. Banerjee, M. T. Preeth, and M. K. Hota, "Design of weather monitoring system and smart home automation," in *Proc. IEEE Int. Conf. Syst., Comput., Automat. Netw. (ICSCAN)*, Jul. 2018.
- [26] B. Kaur, P. K. Pateriya, and M. K. Rai, "An illustration of making a home automation system using raspberry Pi and PIR sensor," in *Proc. Int. Conf. Intell. Circuits Syst. (ICICS)*, Apr. 2018.
- [27] Z. M. Zain, M. A. C. Munaaim, and M. Mat-Noh, "Solar powered microcontroller module for real-time sewerage treatment plant monitoring system: Prototype development," in *Proc. 7th IEEE Int. Conf. System Eng. Technol. (ICSET)*, Oct. 2017.
- [28] P. Li, X. Yang, and Y. A. W. Shardt, "Simultaneous robust, decoupled output feedback control for multivariate industrial systems," *IEEE Access*, vol. 6, pp. 6777–6782, 2018.
- [29] H. T. H. Thabet, "Design and simulation of a monitoring electrical energy consumption system based on PLC techniques," in *Proc. 1st Int. Sci. Conf. Eng. Sci.-3rd Sci. Conf. Eng. Sci. (ISCES)*, Jan. 2018.
- [30] A. Toroman and E. Mujcic, "Application of industrial PLC for controlling intelligent traffic lights," in *Proc. 25th Telecommun. Forum (TELFOR)*, Nov. 2017.
- [31] J. P. S. Da Silva, M. Ecar, M. S. Pimenta, G. T. A. Guedes, L. P. Franz, and L. Marchezan, "A systematic literature review of UML-based domain-specific modeling languages for self-adaptive systems," in *Proc. 13th Int. Conf. Softw. Eng. Adapt. Self-Manag. Syst. (SEAMS)*, 2018.
- [32] H. Marouane, A. Makni, R. Bouaziz, C. Duvalliet, and B. Sadeg, "Defining a UML profile for the consistency of design patterns," in *Proc. IEEE/ACS 13th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Nov. 2016.
- [33] J. Thangaraj and S. Ulaganathan, "Model reusability and multidirectional transformation using unified metamodel," in *Proc. IEEE Distrib. Comput., VLSI, Electr. Circuits Robot. (DISCOVER)*, Aug. 2018.
- [34] H. Wang, D. Zhong, T. Zhao, and F. Ren, "Integrating model checking with SysML in complex system safety analysis," *IEEE Access*, vol. 7, pp. 16561–16571, 2019.
- [35] *Aggregate Widgets*. Accessed: Sep. 2019. [Online]. Available: <https://aggregate.tibbo.com/technology/visualization/widgets.html>
- [36] *Open Automation Software*. Accessed: Sep. 2019. [Online]. Available: <https://openautomationsoftware.com/>
- [37] I. Mahmood, T. Kausar, H. S. Sarjoughian, A. W. Malik, and N. Riaz, "An integrated modeling, simulation and analysis framework for engineering complex systems," *IEEE Access*, vol. 7, pp. 67497–67514, 2019.
- [38] M. Hentati, A. Trabelsi, L. Benammar, and A. Mahfoudhi, "Search-based software engineering for optimising usability of user interfaces within model transformations," *IET Softw.*, vol. 13, no. 5, pp. 368–378, Oct. 2019.
- [39] G. Dupont, S. Mustafiz, F. Khendek, and M. Toeroe, "Building domain-specific modelling environments with papyrus: An experience report," in *Proc. 10th Int. Workshop Modeling Softw. Eng. (MiSE)*, 2018.
- [40] O. M. Winzer, A. S. Conti-Kufner, and K. Bengler, "Intersection traffic light assistant—An evaluation of the suitability of two human machine interfaces," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018.
- [41] R. Zhang, S. He, X. Yang, X. Wang, K. Li, Q. Huang, Z. Yu, X. Zhang, D. Tang, and Y. Li, "An EOG-based human-machine interface to control a smart home environment for patients with severe spinal cord injuries," *IEEE Trans. Biomed. Eng.*, vol. 66, no. 1, pp. 89–100, Jan. 2019.
- [42] Z. Sheng, C. Ji, and S. Hua, "Application of siemens PLC and WinCC in the monitoring-control system of bulk grain silo," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Jun. 2018.
- [43] A. Amjad, F. Azam, M. W. Anwar, W. H. Butt, M. Rashid, and A. Naeem, "UMLPACE for modeling and verification of complex business requirements in event-driven process chain (EPC)," *IEEE Access*, vol. 6, pp. 76198–76216, 2018.
- [44] G. Guo, "Design and implementation of smart campus automatic settlement PLC control system for Internet of Things," *IEEE Access*, vol. 6, pp. 62601–62611, 2018.
- [45] *Model-Driven Mobile HMI Framework (MMHF)*. Accessed: Oct. 2019. [Online]. Available: <https://ceme.nust.edu.pk/ISEGROUP/MHCG/index.html>



IQRA QASIM received the B.S. degree in software engineering from Fatima Jinnah Women University, Rawalpindi, Pakistan, in 2014, and the M.S. degree in software engineering from the National University of Sciences and Technology (NUST), Islamabad, Pakistan, in 2019. Her research interests include model driven architecture (MDA), human-machine interface, and industrial control systems.



MUHAMMAD WASEEM ANWAR is currently pursuing the Ph.D. degree with the Department of Computer and Software Engineering, CEME, National University of Sciences and Technology, Pakistan. He is also a Senior Researcher and an Industry Practitioner in the field of model-based system engineering (MBSE) for embedded and control systems. His major research area is MBSE for complex and large systems.



FAROOQUE AZAM is currently an adjunct Faculty Member of the Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Pakistan. He has been teaching various software engineering courses, since 2007. His areas of interests are model driven software engineering, business modeling for Web applications, and business process reengineering.



HANNY TUFAIL received the M.Sc. degree in computer science from Quaid-i-Azam University Islamabad, Pakistan, in 2011, and the master's degree in software engineering from the National University of Sciences and Technology (NUST), Pakistan, in 2019. His research interests are model-driven software engineering, context aware systems, mobile applications, and industrial control systems.



WASAI HAIDER BUTT is currently an Assistant Professor with the Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Pakistan. His areas of interests are model driven software engineering, as well as Web development and requirement engineering.



MUHAMMAD NOUMAN ZAFAR received the B.S. degree in software engineering from the Government College University, Faisalabad, in 2014, and the M.S. degree in software engineering from the College of Electrical and Mechanical Engineering (CEME), National University of Science and Technology (NUST), Islamabad, in 2018. He is an Active Member of the Eclipse and Microsoft Community as well as the Model Driven Software Engineering Research Group at CEME, NUST.

His research interests are model driven architecture, distributed industrial control systems, the Internet of Things, big data, and robotics.

...