Automated HMI Generation via Component-Based Virtual Engineering

Abdul Basit[1], Izhar Ul Haq[1], Muhammad Usman Qadir[1]
[1]Department of Mechatronics Engineering (University of Engineering & Technology, Peshawar).
*** Correspondence:** abdul.basitglt15@gmail.com and izhar@uetpeshawar.edu.pk

Citation | Basit. A, Haq. I. U, Qadir. M. U, "Automated Generation via Component-Based Virtual Engineering", IJIST, Vol. 7, Issue. 2 pp 817-829, May 2025
Received | April 17, 2025 **Revised |** May 13, 2025 **Accepted |** May 14, 2025 **Published |** May 15, 2025.

As system complexity rises and the demand for shorter time-to-market grows, there is a need to change our traditional methods of building automation systems. Developing code for Programmable Logic Controllers (PLCs) and HMIs is often a challenging and time-consuming part of designing automation systems. Typically, PLC and HMI codes are developed using vendor-specific tools and IEC-based languages. Secondly, code reuse usually involves a lot of manual copy-pasting, which is prone to errors. This method lacks proper version control and direct integration between PLC and HMI, making updates and maintenance not only challenging but also costly. This research provides a novel method to create an Auto HMI for component-based production machines by utilizing their associated virtual models. The production machine is initially modelled in Computer-Aided Design (CAD) tools and commissioned inside an emulated engineering environment to test and optimize the control behaviour. A methodology is presented to reuse the control data in the virtual models to build an Auto HMI. At last, the suggested solution is executed and verified on a conveyor-built system rig to demonstrate its feasibility.

Keywords: Human Machine Interface, Virtual Commissioning, Component-Based Automation, Programmable Logic Controller, Agile Manufacturing.

Introduction:

Manufacturing industries are increasingly confronted with challenges due to the relentless pressures of globalization, environmental concerns, economic instability, and rapid technological advancements. To remain viable within such a business climate, companies must develop an affordable approach for rapid and effective adaptation of their resources to suit the requirements of shifting marketplaces and client expectations without sacrificing production efficiency [1][2].

Manufacturing industries, specifically the automotive industry, are fully automated. However, the standard development process involves mechanical, electrical, and control engineering, each designed separately in a sequential manner. These components are integrated only during the final phase of commissioning. This normally leads to finding irregularities during the last phase, resulting in a protracted ramp-up period [3]. To meet the needs of mass customization, there is a considerable need for new types of production systems. Despite of sequential procedure, vendor-specific tools decrease ramp-up time and create complexity. As industries aim for better responsiveness, the drawbacks of using standalone tools in different departments are becoming increasingly obvious. These limitations lead to duplicated efforts, information loss, inconsistent data formats, and a lack of coordination across various engineering disciplines.

To minimize development duration, the manufacturing industry significantly depends on Virtual Commissioning (VC). VC requires developing and modelling a 3D design of the manufacturing machine. This technique allows control engineers to start coding the control unit before the machine development process. The control algorithms may then be evaluated using the three-dimensional virtual simulator to verify the control performance before the actual construction of the machine [4][5][6].

Generating auto code for PLC and HMI using a virtual engineering approach is an effective and error-free solution, eliminating the need for the conventional manual approach. The virtual model, created using an Integrated Engineering Tool, features predefined control behaviours, allowing for the reuse and transformation of the same data into the required control programming [7]. This method enhances the intuitiveness of control system programming by defining behaviours at a higher level, thereby reducing the challenges of low-level coding.

This research focuses on constructing a self-configurable HMI that reuses the data from virtual models. The HMI is constructed with many displays that serve as a graphical interface for operators to monitor and operate industrial operations. These displays are built utilizing a template-based technique, where the templates are filled with data specific to the machine retrieved from the virtual model.

Research Limitation and Scope:

The proposed concept has the potential to be implemented in various sectors, including Automotive Manufacturing, body-in-white manufacturing, airport baggage handling, and warehouse automation. Although this study primarily focused on generating code for Siemens, Schneider Electric, and PLCopen platforms, the methodology can also be modified for other platforms that use distributed systems and Service-Oriented Architecture (SOA).

Literature Review:

Recent studies have highlighted the importance of adopting HMI-based industrial automation systems. The structure of traditional PLC-based systems for data control and monitoring has been thoroughly examined [1][2][3][4][8][9][10]. In these systems, the HMI usually functions as a PC-based application that connects with the machine's PLC control system. The PLC is responsible for managing communication with the HMI, as well as handling input/output (I/O) signals for sensors and actuators through proprietary control networks. Data transfer between these components often depends on the Open Platform

Communications (OLE for Process Control) gateway, which enables the flow of data from the PLC to the HMI system [11][12]. HMI systems are typically programmed using specialized tools provided by the software vendor. These systems interact with the PLC control network to relay low-level PLC register data, which is then utilized to represent the machine's real-time status.

The HMI station features a lookup database that translates this raw machine data into human-readable descriptions, which are shown on interface screens. However, traditional development methods and maintenance can be quite complex and resource-heavy, necessitating highly skilled personnel to create and modify. When changes to the system are required, end users and Original Equipment Manufacturers (OEMs) often face difficulties, as engineers must first grasp the programmer's unique style before they can troubleshoot machine issues. The HMI and diagnostics frequently operate separately from the main control program and may not align with it's sequence. As a result, if the PLC program is altered, the HMI program might not be updated, leading to incorrect or outdated messages that reduce the system's effectiveness and usability.

In [11][13], Edward et al. presented a component-based control strategy aimed at replacing conventional PLC/PC systems in the automotive engine manufacturing sector. The approach for creating HMI systems involves configuring HMI templates within the Process Definition Engineering toolset. To ensure a consistent and unified design, the end-user establishes a template that details both the PLC program structure and the HMI screen design, which must be followed by all machine builders. Although this approach ensures no inconsistencies between machine control logic and HMI system, but it does not offer adequate version control between PLCs and HMIs.

Chalmers University has conducted research focused on creating a system that reuses mechanical design data from a production unit to automatically generate PLC and HMI code. The framework is designed to identify sections of control code that can be created using data obtained from robot simulations and the design of the production cell. These components are arranged as function blocks, with each block corresponding to a specific device within the system. Although the primary focus of this research is to automate the control process, it does not address other aspects of control code, i.e., manual mode and integration of HMI [12][14].

Rockwell Automation has concentrated on creating a distributed platform that is adaptable and reconfigurable, supporting agent-based automation systems that are plug-and-play. PLC's flagship product, Logix™ control, is used to execute the agent-based method using real-time control agents and facilitate information sharing among the agents. Tags may be used to store and distribute data in the agent controls interface. Through an OPC communication bridge and their unique Java-based interface, external resources can access these tags. Using their proprietary implementation, the current industrial visualization systems based on HMI panels also work with these PLC tag values [15]. The limitations of tag-based representation necessitate re-loading and re-compiling the HMI program while undergoing reconfiguration, as noted in [15][16].

Bergert [17] created a method for auto HMI code generation using digital process data from production cells modelled in DELMIA Process Engineer. The process plan is sifted to keep the PLC-related information only and remove all extra information. Next, the filtered data is modified into a Sequential Function Chart (SFC) specifically suitable for use in Unity Pro. The resource library function blocks are then connected to this SFC via resource-specific modules using Unity Pro, including I/O signals, which define the performance of the different manufacturing resources. However, despite its potential, Bergert's research remains mostly theoretical with limited real-world application. The generated program still needs further manual adjustment before it can be deployed, because it still relies on the corresponding function blocks to map the SFC. Furthermore, the study is focused on the automation of

production cycles only; as diagnostics, mode control, and HMI integration are other important parts of comprehensive control programming, they are not considered.

In [18][19], Volvo Car Corporation used the automation designer tool from Siemens, used for virtual commissioning, to generate auto PLC and HMI code. The main objective was to produce code and screens early in the commissioning process and to reuse the data from process simulations stored in the database. However, some limitations were identified. The code was generated according to Volvo's standards but lacked overall sequencing, which complicated the validation process [5]. While the HMI screens followed the intended layout, the tag and screen naming conventions did not comply with the manufacturer's specified standards. While it was possible to change the names manually in the HMI software, the corresponding data blocks still needed to be maintained.

Novelty Statement:

Conventional methods for machine HMI systems often fail to address the needs of agile manufacturing. Key challenges include lengthy development timelines, no direct integration between PLC and HMI, and the requirement for highly skilled professionals, leading to greater chances of error-prone, time-consuming, and increasing complexity and obstacles among engineering collaborators in the machine development process. To overcome these challenges, this research provides a novel method to create an Auto HMI for component-based production machines by utilizing their associated virtual models to reuse the data, test, and optimize the control behaviour. For any changes and additions to the system, the changes are required in a virtual model only. Because the HMI software is designed generically, no direct changes are required in the HMI application. When a connection to the PLC is established, the HMI screens are automatically updated to reflect the latest changes in the Control Data Model (CDM), ensuring real-time synchronization with the system configuration.

Research Objectives:

1. To review existing approaches to HMI development in various sectors, including body-in-white manufacturing, airport baggage systems, and warehouse automation, to identify key limitations and time constraints.
2. To create and implement an approach for the auto-generation and integration of HMI Screens.
3. To evaluate the approach and analyze the time-saving impact of auto code generation compared to manual coding, using an automated test rig.

Methodology:

PLC Code Generation:

Figure 1 below illustrates the process of generating PLC code using the Integrated Engineering Tool (IET). After a production unit is successfully validated through virtual simulation, its data is transferred to the analyzer or Mapper in an open XML-based format. It offers an intuitive interface for configuring control logic, simplifying the process of defining I/O address mappings and assigning standardized function blocks to the actuators and sensors within the manufacturing cell. After configuring the sensors and actuators, control data is generated via Mapper by processing the given information. After that, control data is linked automatically with the PLC code specified by the end-user to produce an executable PLC program. In this case **SIMATIC S7-400** PLC was chosen, therefore, the generated code was transferred in multiple plain text formats. The detail is shown in Table 1 below.

Table 1. Generated code as a plain text file

File Name	languages	File Format
I/O variables.asc	NA	ASC
UDT&DB.awl	Statement List	AWL
FBs.scl	Structured Control Language	SCL

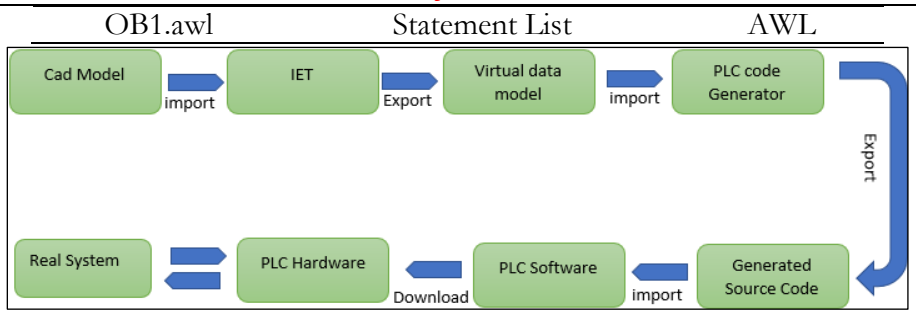

Figure 1. Auto PLC Code Generation Framework

HMI Generation:

The proposed framework for automated HMI generation, depicted in Figure 2, is built upon a virtual engineering approach and comprises three core components. While this section focuses exclusively on the HMI portion, it is worth noting that the PLC segment has been discussed earlier and is beyond the current scope.

Data Tags:

Data tags represent variables that may be either internal or external to the system. These tags establish a communication link between the HMI and external devices, enabling real-time data exchange essential for effective HMI operation. During runtime, external tags are dynamically retrieved from the Control Data Model (CDM). This ensures seamless synchronization and interaction between the PLC and HMI, enabling automated HMI generation and real-time system monitoring.

Screen Templates:

These are pre-defined user interface layouts that serve as standardized blueprints for machine monitoring and control. Templates are created once and dynamically updated using real-time data sourced from the CDM. This approach not only enhances consistency across HMI displays but also significantly reduces development time and manual configuration efforts.

Screen Generator:

The Screen Generator module is responsible for rendering operational screens used for machine control and diagnostics. When a screen is requested by an HMI device, the generator fetches the corresponding template and populates it with context-specific data from the CDM. This mechanism enables the real-time deployment of accurate and up-to-date graphical interfaces, facilitating efficient human-machine interaction.

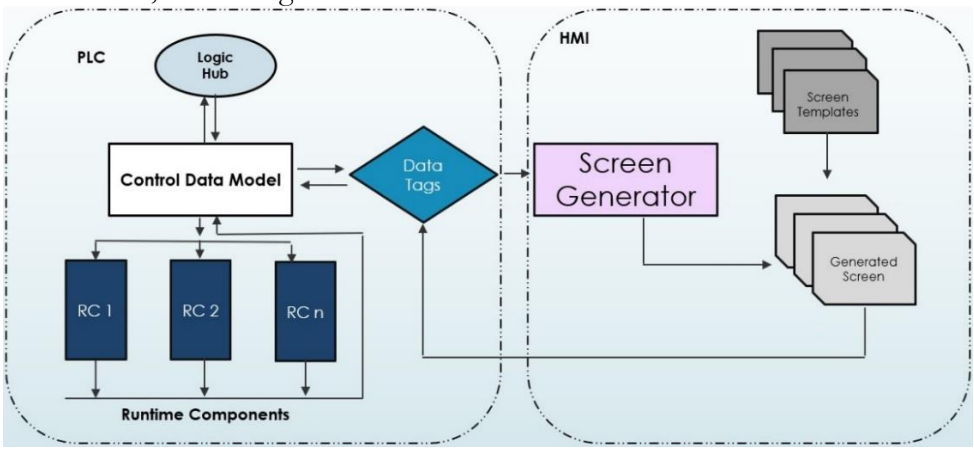

Figure 2. HMI System Development Tool Architecture

This paper primarily focuses on the automatic generation of HMI (Human-Machine Interface) screens, offering a detailed explanation of the underlying code generation process. In addition to the HMI screen creation framework, it introduces a complete methodology for

implementing the operator interface of a production unit through a virtual modeling approach, as illustrated in Figure 3.

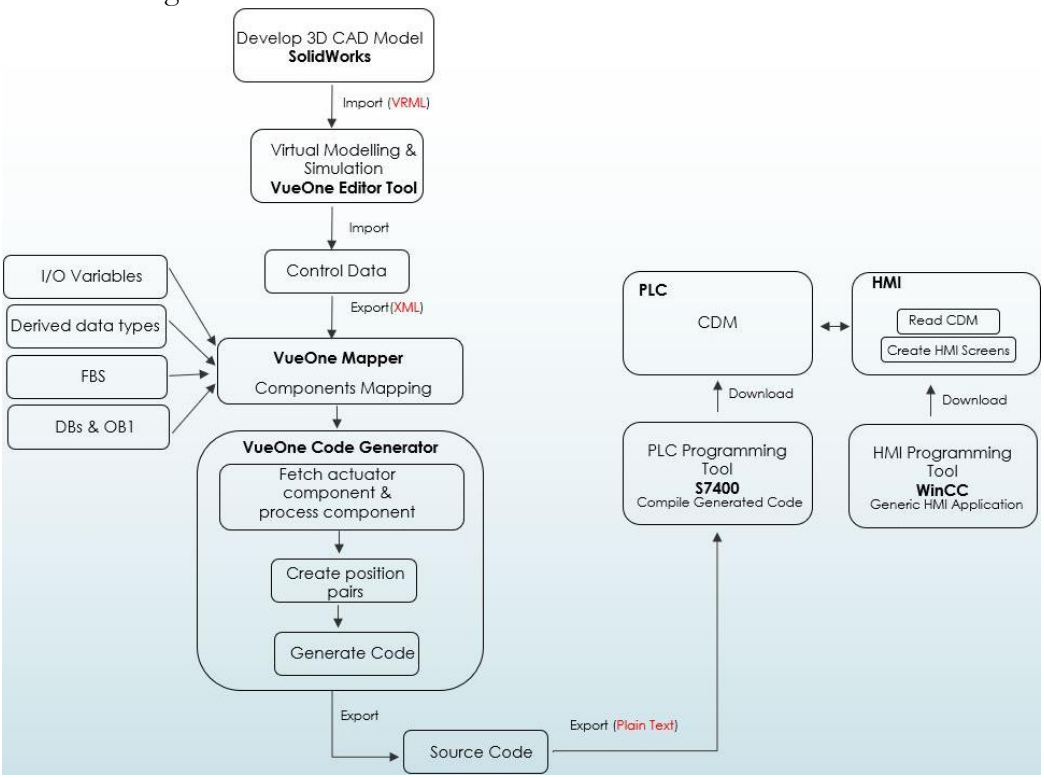

Figure 3. Flow Diagram of Methodology

HMI Generation Using a Virtual Model Approach:

A 3D CAD model of the test setup was initially created in SolidWorks and exported as VRML format to VueOne Editor. For the next design phase, the model is further commissioned (components behaviour and sequence of operation) and enhanced using the VueOne editor tool, allowing for a component-based representation suitable for virtual commissioning. Then the virtual model is exported as an XML format to VueOne Mapper. Further processing is discussed in the PLC code generation section. The data format is highlighted in red colour. The methodology adopted is based on a generic template approach, aimed at automating the creation of HMI screens.

To implement this approach, predefined screens such as the home screen, manual position screen, and logic tracking screen were created using the WinCC tool. These screens serve as template-based interfaces within the HMI generation framework.

The screen templates are classified into two main categories:

1. **Generic Screens** – These include standard layouts like the home screen, which are designed and integrated early during the template preparation stage. They are reused across different machines without modification.
2. **Machine-Specific Screens**. These are dynamically generated at runtime using the HMI Data Model (HDM) and an associated mapper function. These screens are tailored to the specific requirements of individual machine cells, ensuring adaptability and consistency within the overall HMI system.

The manual mode operator screens are uniquely configured for each manufacturing cell, giving each system its distinct identity. Typically, these screens contain pairs of pushbuttons corresponding to each actuator. Operators can interact with these buttons to move actuators from their home positions to their operational positions, enabling manual control over the system.

The code generator is designed to handle only the state behavior of actuator components, as these are the elements requiring manual operation. Once the screen templates are finalized, the mapper function is generated. This function links the HDM to the tags used in the screen templates, ensuring automatic tag mapping between template elements and the data model.

Following the deployment of the HMI software and its connection with the PLC, the system retrieves the necessary control data to populate and render context-specific HMI screens. These include interfaces for real-time monitoring, diagnostics, and manual operation. As changes occur in the data model, the HMI detects and reflects these in real-time, allowing the interface to reconfigure itself dynamically and maintain synchronization with the underlying control logic.

Case Study: Double Conveyor Test Setup:

The Double Conveyor test setup, as shown in Figure 4, serves as a representative case study to demonstrate the effectiveness of the proposed automatic HMI code generation approach. This setup was chosen for its suitability in illustrating the contrasts between auto-generated and traditional HMI coding methods. It simulates a typical transportation and manufacturing process, involving: Two conveyors, an ejector, A blocker, A gantry system for transporting processed items to storage.

These components collectively support material handling, inspection, and part processing within a simulated production environment.

For hardware implementation, the Siemens SIMATIC S7-400 PLC and WinCC HMI devices were selected due to their compatibility and industrial relevance.

To support virtual commissioning, the development process was divided into two key phases:

1. **Component Development Phase** – Focused on designing individual elements of the setup.
2. **Assembly Development Phase** – Emphasized the integration of components into a cohesive system.

This structured approach ensured both system flexibility and the seamless transition from virtual modeling to physical deployment, reinforcing the viability of automatic HMI generation in modern manufacturing environments.

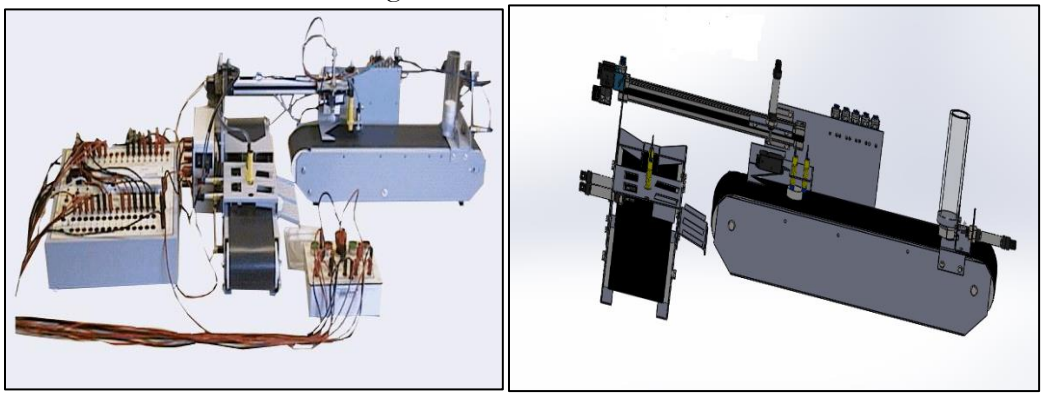

Figure 4. Real and virtual model of the Test setup

Component Architecture and Virtual Modeling:

The test apparatus comprises a total of 14 components, including 7 actuators and 4 sensors. The development process begins with the construction of these components and their associated function blocks, which define their behavior and control logic. When multiple components share identical or similar state behaviors, they can utilize a common function block, thereby simplifying code reuse and modularization. Figure 5 presents an example of this modular design, showcasing the function block implementation for a pusher actuator. Once the components have been defined, a virtual model of the test rig is assembled to

represent the complete system. This process entails assembling the individual components and integrating them through process logic, essentially a state-transition diagram that dictates the system's operational sequence and task flow.

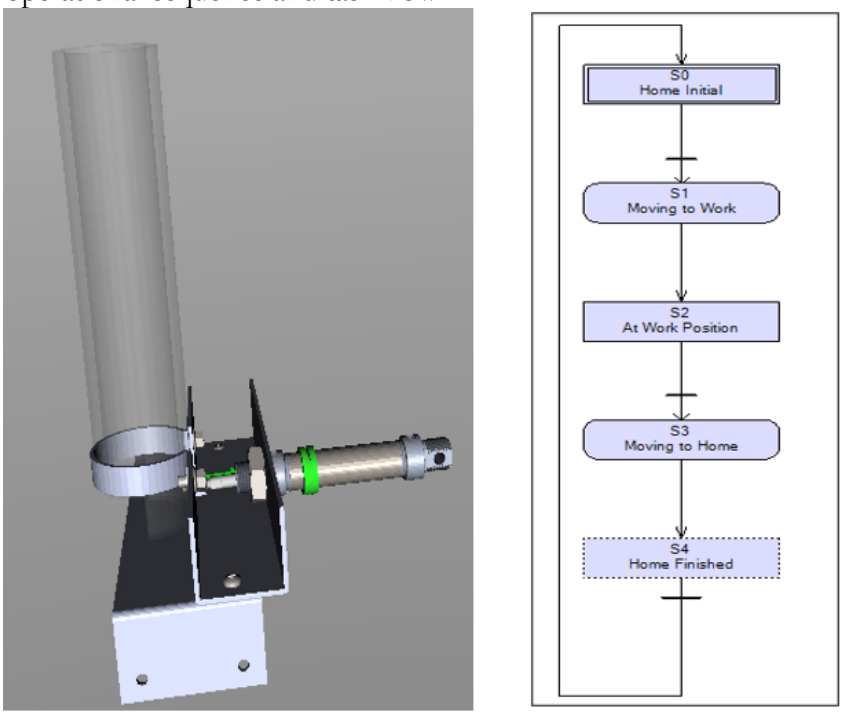

Figure 5. component – Pusher

Once the virtual modeling and validation process is complete, the corresponding PLC code is automatically generated and deployed to the Siemens SIMATIC PLC. In parallel, a standard SIMATIC WinCC project is created according to the architecture described in the methodology section. This project includes predefined HMI screen templates, which are then transferred to the HMI device.

Upon establishing a connection between the HMI and the PLC, the HMI screens are generated automatically based on the predefined templates and real-time data from the HMI Data Model (HDM). Figure 6 represents the created screens, i.e, Home Screen, Manual control Screen, and logic monitoring screen. Operation modes, controlling, and navigating to the other screens can be done via the Home Screen.

As a standard display page, it does not require automatic generation and remains consistent across all production machines. State transition diagrams, as outlined in the IET, dictate how many pushbuttons are needed for manual control operation. The HMI displays a specific number of rows, which is determined by the static locations of the actuator. The pusher is used as an example, as shown in Figure 7. Since the pusher has two positions, only a single row is generated.

Figure 8 illustrates how the PLC and HMI interact in real-time. We can illustrate this concept using a pusher. When the pushbutton on the HMI is activated to set the pusher in its work position, a command is sent from the HMI to the relevant manual control model, which then communicates with the Runtime Component (RC). After the RC receives the command, it processes the instruction and updates the relevant output variable, initiating the pusher's movement. Throughout this process, the RC continuously monitors the actuator's status and sends real-time updates.

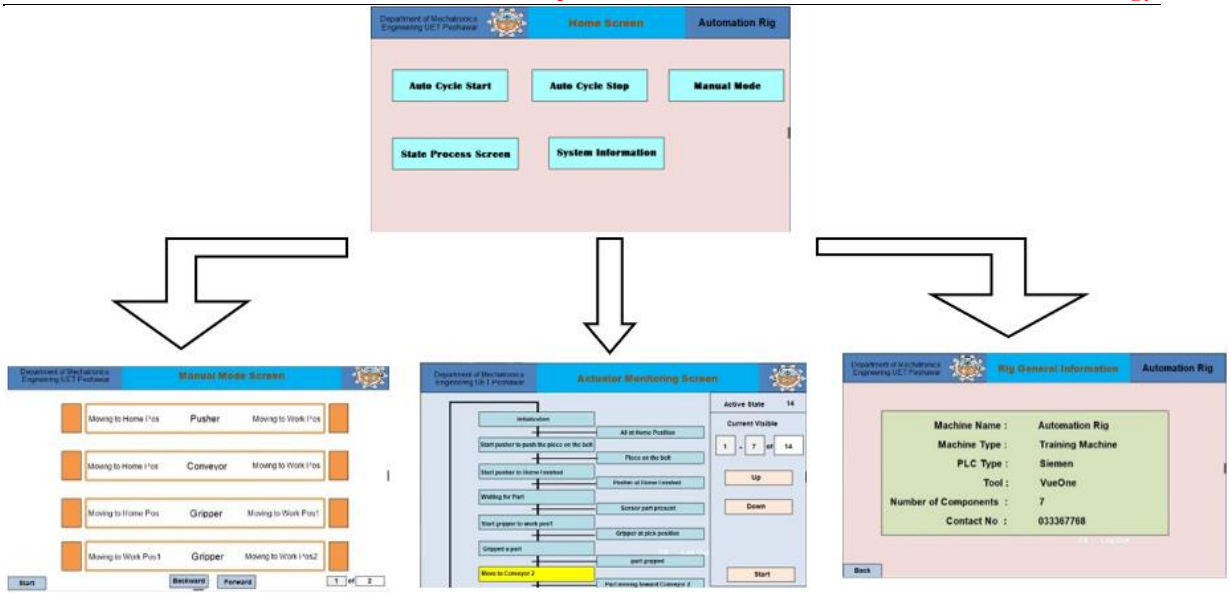

Figure 6. HMI Display Pages

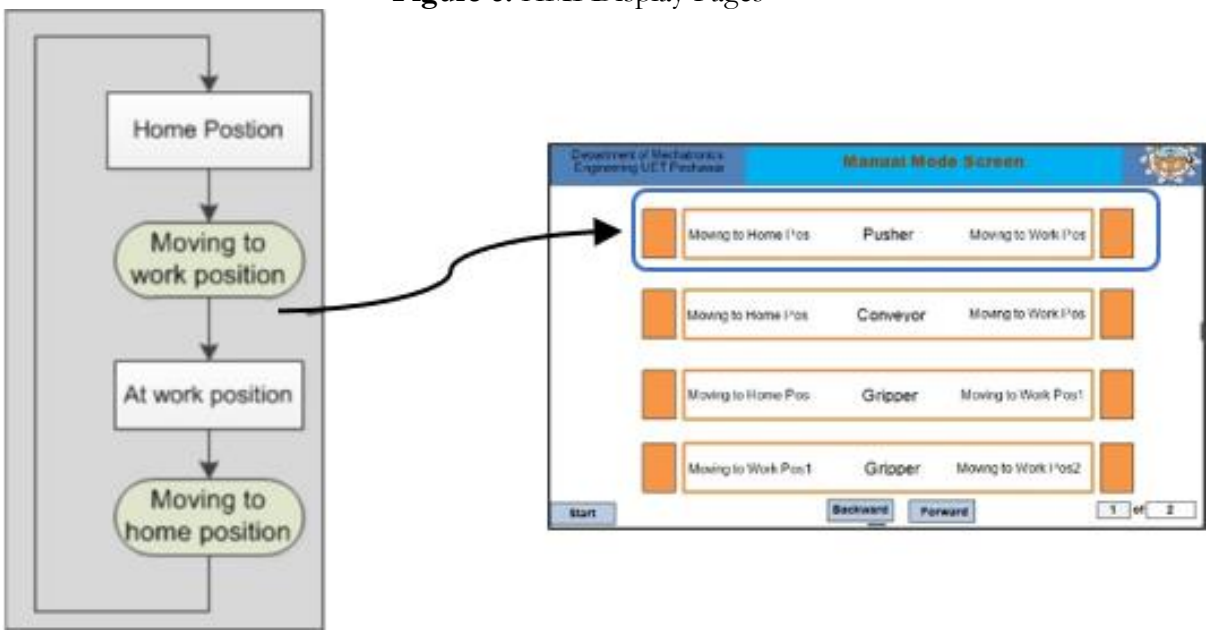

Figure 7. Auto Auto-generated manual mode display to reflect the state behavior of relevant components

To implement any necessary adjustments, the virtual model must be reconfigured first. After completing the reconfiguration in the IET unit, the new system information is reimported into the mapper. Once the essential input-output arrangement and Runtime Component analysis are completed, the CDM automatically updates to reflect the changes made to the system in the data model, ensuring accurate tracking and synchronization. Because the HMI software is designed generically, no direct changes are required in the HMI application. When a connection to the PLC is established, the HMI screens are automatically updated to reflect the latest changes in the CDM, ensuring real-time synchronization with the system configuration.

Results:

A comprehensive assessment of time consumption for both auto code generation and traditional manual coding was carried out to evaluate the efficiency and feasibility of implementing automated code generation in automation systems. The development and

Reconfiguration timelines for manual programming and auto-generated code were meticulously recorded and broken down into individual activities. This breakdown is essential to understanding the impact of automation on the engineering workflow, particularly in the context of component-based HMI systems.

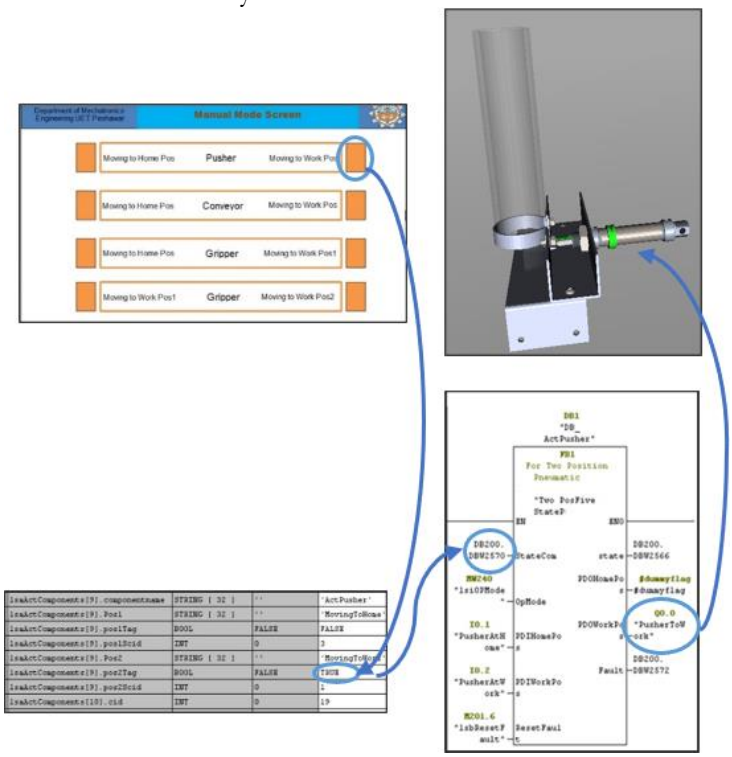

Figure 8. HMI system operation sequence overview

Development Time: Manual vs Auto Code Method:

The recorded time durations for both approaches, using the same set of control functionalities, are presented in Tables 2 and 3.

Table 2. Time Utilization in Manual Code Development

Activity	Time Duration (hr)
Design and implementation of templates	1.5
Creating Function Blocks and Data Blocks, and testing	9
I/O allocation or mapping	0.5
Creating a Sequential Function Chart	9
HMI Screen development	5
Commission and installation	9
Total time	34

Table 3. Time Utilization in Auto Code Generation

Activity	Time Duration (hr)
Design and implementation of templates	2
RCs development and testing	9
Specifying control logic and behaviour in VueOne	4
Virtual commissioning	1
I/O and RCs allocation	1
Commission and installation	1
Total time	18

Reconfiguration Time: Manual vs Auto Code Method:

Table 4 provides a side-by-side comparison of the time required for reconfiguration in both methods.

Table 4. Reconfiguration Time

Manual Code approach		Auto Code generation approach	
Activity	Time(hr)	Activity	Time(hr)
Modification of PLC programming	1	Reconfiguration of the virtual model	0.7
Modification of HMI	0.4	Virtual commissioning	0.4
Recommissioning and installation	0.6	Recommissioning and installation	0.1
Total time	2	18	1.2

Discussion:

The manual method involved several time-intensive steps, such as designing templates, creating function blocks, testing, HMI development, and commissioning. A total of 34 hours was spent on the entire cycle. On the other hand, auto code generation reduces development time. A total of 18 hours was recorded for this process.

These findings suggest that the auto code generation method resulted in a time saving of approximately 16 hours, a reduction of nearly 47% in the total development timeline. From Tables 2 and 3, it is clear that the Auto code generation method significantly reduced installation and commissioning times compared to the traditional manual programming approach. The main reason for this was the validation of control logic during the virtual commissioning phase, which utilized the VueOne editor tool and was carried out offline. This approach helped to enhance the project's critical path. Since the screens are generated automatically, therefore no need to develop when any modification is required. This improvement is particularly impactful in fast-paced manufacturing environments where time-to-market is a key success factor.

The auto-generated system required only 1.2 hours for reconfiguration, compared to 2 hours needed for the manual method. Although the difference may appear small in absolute terms, in cumulative large-scale applications or frequent reconfiguration scenarios, this 40% time saving translates into major efficiency gains.

The results demonstrate that auto code generation significantly reduces the overall engineering effort, with a time saving of nearly 47% in initial development and 40% during reconfiguration tasks.

Conclusion:

This paper examines a new approach to creating HMI screens in component-based automation systems. Conventional methods for machine HMI systems often fail to address the needs of agile manufacturing. Key challenges include lengthy development timelines and the requirement for highly skilled professionals, leading to an obstacle among engineering collaborators in the machine development process.

From the direct-integration perspective, any modifications made to the Control System Data Model are instantly updated in all three sections of the control software. Initially, the Double Conveyor test setup was developed and tested within a virtual three-dimensional platform. Generated and implemented screens automatically for HMI using a verified and validated virtual model, eliminating the need for manual changes.

The case study emphasizes the benefits of adopting a dynamic HMI development approach as opposed to traditional static methods. It allows for greater flexibility across multiple platforms and is simpler to implement. This strategy is expected to enhance virtual commissioning by decreasing the software development time in automation systems. Additionally, this strategy boosts responsiveness, versatility, and reusability, allowing it to better address user needs in today's business world.

Acknowledgments: We would like to express our sincere gratitude to the Department of Mechatronics, University of Engineering and Technology, Peshawar, Pakistan, for their

invaluable support throughout this research. Our deepest appreciation goes to Dr. Izhar ul Haq, whose expertise and guidance greatly contributed to this study. We also thank our colleagues and participants for their cooperation and insights, which were instrumental in shaping our findings.

Conflict of interest: The authors declare that there is no conflict of interest regarding the publication of this manuscript in the IJIST.

References:

[1] M. R. Anwar, O. Anwar, S. F. Shamim, and A. A. Zahid, "Human machine interface using OPC (OLE for process control)," Student Conf. Eng. Sci. Technol. SCONEST 2004, pp. 35–40, 2004, doi: 10.1109/SCONES.2004.1564766.

[2] S. Da'na, A. Sagahyroon, A. Elrayes, A. R. Al-Ali, and R. Al-Aydi, "Development of a monitoring and control platform for PLC-based applications," Comput. Stand. Interfaces, vol. 30, no. 3, pp. 157–166, Mar. 2008, doi: 10.1016/J.CSI.2007.08.008.

[3] C. Şahin and E. D. Bolat, "Development of remote control and monitoring of web-based distributed OPC system," Comput. Stand. Interfaces, vol. 31, no. 5, pp. 984–993, Sep. 2009, doi: 10.1016/J.CSI.2008.09.027.

[4] G. DiFrank, "Power of automation: An overview, technology, and implementation," IEEE Ind. Appl. Mag., vol. 14, no. 2, pp. 49–57, Mar. 2008, doi: 10.1109/MIA.2007.914275.

[5] E. H. Mikael Andersson, "Automatic generation of PLC programs using Automation Designer Based on simulation studies and function block libraries," Prod. Eng. Chalmers Univ. Technol. Göteborg, Sweden, 2010, [Online]. Available: https://publications.lib.chalmers.se/records/fulltext/133762.pdf

[6] Joakim Davidson and Tobias Sennö, "INTERACTIVE CONTROL OF A VIRTUAL MACHINE," Lund Univ. Dep. Ind. Electr. Eng. Autom. Sweden, 2005, [Online]. Available: https://www2.iea.lth.se/publications/MS-Theses/Short article/5208_Smf_Interactive_Control_of_a_Virtual_Machine.pdf

[7] B. Ahmad, X. Kong, R. Harrison, J. Watermann, and A. W. Colombo, "Automatic generation of Human Machine Interface screens from component-based reconfigurable virtual manufacturing cell," IECON Proc. (Industrial Electron. Conf., pp. 7428–7433, 2013, doi: 10.1109/IECON.2013.6700369.

[8] K. H. Lee, E. C. Tamayo, and B. Huang, "Industrial implementation of controller performance analysis technology," Control Eng. Pract., vol. 18, no. 2, pp. 147–158, Feb. 2010, doi: 10.1016/J.CONENGPRAC.2009.09.011.

[9] H. C. Lin, "A remote monitoring and control-based precise multilocation riveting system," Comput. Appl. Eng. Educ., vol. 13, no. 4, pp. 316–323, Jan. 2005, doi: 10.1002/CAE.20057.

[10] M.-J. Y. Vu Van Tan, Dae-Seung Yoo, "A Novel Framework for Building Distributed Data Acquisition and Monitoring Systems," J. Softw., vol. 2, no. 4, 2007, [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=638b831d922b25af6347a64c49f7d3153afd4b39

[11] E. W. Mellor, A. A. West, and R. Harrison, "A Component-Based Human Machine Interface System for Automotive Manufacturing Machines," Proc. 7th Bienn. Conf. Eng. Syst. Des. Anal. ESDA 2004, vol. 2, pp. 361–366, Nov. 2008, doi: 10.1115/ESDA2004-58368.

[12] J. Richardsson and M. Fabian, "Automatic generation of PLC programs for control of flexible manufacturing cells," IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA, vol. 2, no. January, pp. 337–344, 2003, doi: 10.1109/ETFA.2003.1248719.

[13] 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," 2014.

[14] S. Garbrecht, "The Benefits of Component Object-Based Supervisory System Application Development versus Traditional HMI Development in Water Systems Operations Management," Proc. Water Environ. Fed., vol. 2008, no. 8, pp. 7358–7370, Sep. 2012, doi: 10.2175/193864708788809365.

[15] P. Vrba et al., "Rockwell automation's holonic and multiagent control systems compendium," IEEE Trans. Syst. Man Cybern. Part C Appl. Rev., vol. 41, no. 1, pp. 14–30, 2011, doi: 10.1109/TSMCC.2010.2055852.

[16] Process-Industry-News, "Rockwell Automation launches signal conditioners as first process component offering," Assem. Autom., vol. 30, no. 3, pp. 59–63, Aug. 2010, doi: 10.1108/AA.2010.03330CAD.002/FULL/XML.

[17] M. Bergert, C. Diedrich, J. Kiefer, and T. Bär, "Automated PLC software generation based on standardized digital process information," IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA, pp. 352–359, 2007, doi: 10.1109/EFTA.2007.4416789.

[18] Volvo Car Corporation, "Programming Instructions for PLC Systems," Simatic, 2008.

[19] K. Güttel, P. Weber, and A. Fay, "Automatic generation of PLC code beyond the nominal sequence," IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA, pp. 1277–1284, 2008, doi: 10.1109/ETFA.2008.4638565.