Towards a Modular Architecture for Industrial HMIs

Muddasir Shakil, Alois Zoitl, *Member, IEEE*
LIT CPS Lab, Johannes Kepler University Linz
Linz, Austria
{muddair.shakil, alois.zoitl}@jku.at

***Abstract*—This paper presents a work in progress on a modular architecture design of industrial HMIs. The proposed approach is based on a micro frontend architecture style for developing web applications. It allows developers and integration engineers to build an HMI from distributed and modular shop-floor components called micro frontend. Each micro frontend is end-to-end integrated from its data acquisition to the user interfaces. This approach introduces an agile and modular development style of an industrial HMI for a modular and adaptive production system.**

***Index Terms*—micro-services, HMI , modular production systems, Cyber-Physical System**

I. Introduction

The fourth industrial revolution (I4.0) tries to adapt to the "Universal Plug-and-Play" like architectures in production, and product development processes. Therefore, concepts like "Plug-and-Produce" are introduced in Cyber-Physical Production Systems (CPPS) [1]. The "Plug-and-Produce" could be facilitated by applying interoperable and loosely coupled modules. On the other hand, Interoperability is realized by creating open interfaces for modules, which allow them to interact with each other and maintain the autonomous state. Whereas, loosely coupled modules add the required flexibility to the CPPS

A simple modular machine is illustrated in Fig. 1, which is decomposed into pluggable automation modules. A production system can adapt to the changing requirements by reorganizing modules and reconfiguring the production steps. A modular machine requires human interactions for accepting commands, monitoring process parameters, and notifying about alarms and process states. A Human Machine Interface (HMI) provides such interactions between the operator and the production machine. In the context of "Plug-and-Produce", it is important from the machine operator's point of view that these changes must be reflected on to the HMI. The command interfaces, process parameters, alarms, and events of newly added automation modules must be provided to the operator via HMI with minimum integration efforts.

The integration of new HMI objects requires reprogramming and redeploying of the HMI software. In this work, we present a new concept for HMI development based on a modular architecture. The main idea of the research work is to develop a modular HMI architecture that supports the "Plug-and-Display" environment. To achieve this, we first analyzed the basic requirements for the modular HMI architecture, followed by the first prototype implementation of the modular HMI using the micro frontend architecture. In the micro frontend architecture, a frontend application is divided into small modules depending upon features, functionalities, and team orientations. These independent modules are deployed as microservices for frontends known as micro frontend [2].

The rest of this paper is structured as follows: Section II provides the background knowledge and related work. The general architecture of a modular HMI is discussed in Section III. Followed by the requirement analysis for a modular HMI in Section III-B. In Section IV, the first implementation of the prototype is discussed. Whereas, a first assessment of the micro frontend architecture to build a modular HMI is presented in Section V. This paper is concluded with open points in the research in Section VI.

II. Background and Related Work

A. Service-Oriented Architecture (SOA)

The SOA was originally developed to provide functionality abstraction and standardized interaction between various business processes. SOA paradigm can be described as the set of policies, practices, and frameworks that facilitate applications to offer consumable functionalities as sets of services. These services can be invoked, published, and abstracted from implementation details by using standardized interface descriptions [3] [4].

H.Bloch in [5] shows that SOA approaches are suitable to full fill the requirements of modular process automation and there are appropriate patterns and reference models available for the realization of SOA in process automation. In

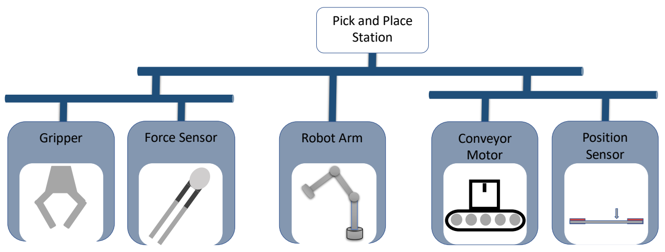

Fig. 1. A Simple Modular Machine

SIRENA [6] project, Web Services based framework was proposed for specifying and developing distributed applications built upon heterogeneous computing devices. It allowed seamless integration of device-level networks and enterprise-level networks.

B. Microservice Architecture (MSA)

O. Zimmerman in [7] presents an analysis that supports S. Newman's position of microservices in SOA. S. Newman in [8] defines the microservices architecture as a specific approach for SOA. Common characteristics like componentization, business orientation, decentralization, design for failures, independently deployable, and polyglot programming in multiple paradigms and languages are emphasized in the microservices implementation approach. Microservices are modelled against their business goals encapsulating implemented functionalities into independently deployable processes. They interact with each other using protocols like HTTP and message queues.

Various studies like [1], [9], and [10] showed that microservices are suitable to implement more flexible, scalable, and modular CPPS. Afanasev [10] applied microservices to create a modular Computer Numerical Control (CNC) System. The suggested pattern showed that microservices make modules autonomous and simplifies scaling. Furthermore, Thramboulidis in [9] examined the potential of exploiting microservice architecture in the CPPS. Their performance measurements showed that the application of microservices introduces high latency. On the other hand, they offer great flexibility at the process plant level

C. HMI Development

Urbas in [11] presented a combination of two Model-driven HMI engineering approaches. The first approach is based on autoHMI which extracts information from Computer-Aided Engineering Data to derive a concrete UI. Whereas, the second approach is XVCML based to generate final UIs. The Cameleon Reference Framework was followed to develop the toolchain for context-adaptive HMI generation.

A distributed HMI platform based on web technologies and software architectures is presented in [12]. It distributed the integration of field device interfaces, control interfaces, and user management interfaces among three distinct components. These components offer communication with field devices, user command management, content personalization, presentation, and client interaction.

Mena in [13] proposed microservices and micro frontend based architecture to build a Progressive Web Application for acquisition of Geospatial data and IoT related information. In the solution, each component service is represented by its counterpart in the frontend as a micro frontend. Component Services are responsible for managing information related to the user's context.

Authors in [14] proposed web-based technologies to build modern and ergonomic HMIs for industrial systems. The focus of their work was on improving user experience and operator's performance. They did not consider the reactiveness of HMI display to reflect the flexibility and adaptability of modern manufacturing systems.

III. Modular HMI

The concepts of intelligent and flexible manufacturing systems require the automation systems to have decentralized autonomy and support "Plug-and-Produce" like features. On the other hand, the machine HMIs also need to dynamically update and adapt display screens and operator screen elements such as IO fields and buttons.

A. Proposed Concept

One way to achieve the adaptability of HMI screens is by decomposition of HMIs into decoupled and independently deployable HMI modules. An HMI module provides a logical grouping of automation components for visualization applications. As illustrated in Fig. 2, the logical grouping for an HMI module may consist of a singular unit (e.g. Sensors, actuators) or hierarchically structured automation components such as a Robotic arm and conveyor. The core idea of an HMI modularization is to build it from decentralized, intelligent and decoupled HMI modules served by independent automation components

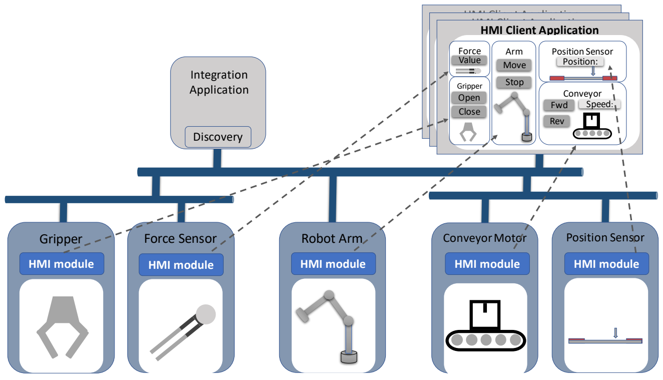

Fig. 2. A Generic Architecture for a Modular HMI

An integration application in the architecture performs the tasks of discovery, registration, and integration of distributed HMI modules at the production shop floor. It then serves the operator screen elements from distributed HMI modules to the platform-independent HMI client applications. The purpose of HMI client applications is to serve modular HMI screens to the Human interacting with machines.

The advantage of using modular HMI screens is that they can be served with different configurations. It allows companies to improve and re-adjust their production capabilities effectively. To facilitate such requirements, our research is focused on investigating and developing a "Plug-and-Display" architecture for modular HMIs.

B. Requirements for Modular HMIs

In this section, a basic set of requirements is presented for a modular HMI. It is a result of our initial functional requirement assessment on developing a modular HMI architecture. It is expected to grow as we investigate it further and analyze more use cases.

- **Auto Discovery**: In regard to a modular HMI this requirement is important to support the "Plug-and-Play" like environment. It allows integration applications to automatically discover and register newly added production modules on the production network.
- **Flexible Communication Network**: This requirement is also coming from "Plug-and-Produce" architecture. It basically means that the communication network used to exchange the information between HMI client applications and modules must be flexible to accommodate the restructuring of modular HMI and machines without the need of reconfiguring the network devices manually [15].
- **Independent Deployment**: Independent deployment is a key feature of a micro-service architecture, it reduces the scope of deployment of an HMI module to a specific automation component. Each module should have its independent deployment pipeline, which builds, tests, verifies, and deploys all the way to the production shop-floor [16].
- **Element Registration**: The screens in the HMI client applications are created by integrating and organizing operating screen elements. The element registration is similar to the "skill as a service" concept presented in [15], where a device offers skills as services to be used by other devices. In our case, the skills are operating screen elements that are offered by an HMI module as services to be used by other HMI client applications.
- **Auto (Re)Configuration**: The auto (re)configuration of operating screens, allows HMIs to adapt according to the current production state and machine structure without the need for any pre-configuration of operating screens.

IV. Micro Frontend Implementation of a Modular HMI

In this section, an implementation of the modular HMI architecture is discussed. We adopted a micro frontend architecture style for the development and deployment of modular HMIs. At the bottom of Fig. 3, the independent shop-floor automation components are illustrated with block diagrams. The following components were used in this implementation: Supply Conveyor, Picking Robot, Sorting Robot, and Rack. Each component is connected to its respective micro frontend application via a backend server. A direct communication link between a micro frontend and an HMI module in the individual components can be established using a middleware communication protocol such as MQTT, AMQP, and OPC UA PUB/SUB.

The micro frontends are developed using the Vue.js framework [17]. In Fig. 3, micro frontends for each component are shown with different frameworks. It is to show that different technologies can be used to develop and deploy individual micro frontends. This allows module manufacturers

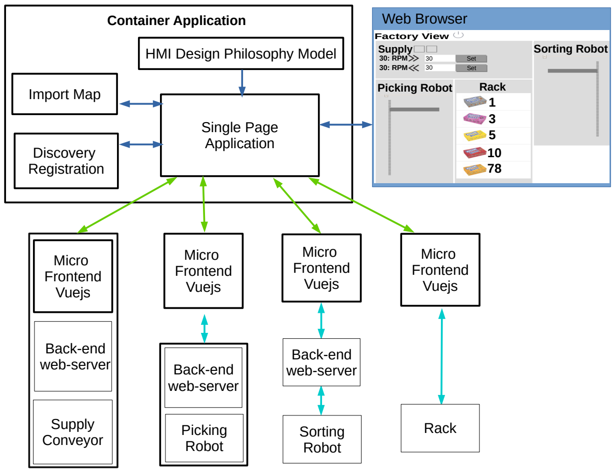

Fig. 3. Micro Frontend HMI Architecture

to maintain their autonomy over technologies that suit best for their devices.

A Container Application serves the purpose of an integration application presented in the architecture. It is developed using a Single Page Application. It enabled to build web-applications by interacting with the web-browsers by dynamically writing web-pages with the new data and events from backend servers. The Import Map specifies to the container application about which resources from micro frontends are dynamically loaded in Browser and added to the web-pages as necessary.

The HMI Design Philosophy Model provides the necessary information to support the auto-configuration requirement. It provides the layout information to generate operating screens, displaying size, colors, positioning, and define visual objects to display automation components like sensors, actuators, and conveyors. Moreover, it also provides information to the HMI page on how to display Alarms, trends, and dialog boxes. For the first prototype as a proof of concept, a simple CSS file is used to model the HMI Design Philosophy.

The dynamic discovery and registration of HMI modules are not supported in micro frontend architecture at the time of implementation. However, discovery mechanisms provided by OPC UA facilitated with a web-server that interacts with the Container Applications can enable auto-discovery of available HMI modules.

The Web-Browser serves as an HMI client application. It is a simple web-browser that can load Single Page Applications and dynamically update web-pages by reading new data changes and events from machine web-servers.

V. A First Assessment of the Proposed Micro Frontend Architecture

In this section, the result of our first assessment of a modular HMI implementation using micro frontend architecture is presented.

A. Micro Frontend Modules

In the architecture (see Figure 3), each module comes with its dedicated micro frontend. The micro frontend pattern requires the features to be developed as end-to-end integrated. It means that they contain their database, backend, and User Interface. Furthermore, the micro frontend architecture requires to have independent deployment pipelines to build, test, and deploy individual micro frontends.

B. Flexible Communication

One of the main objectives of the micro frontend architecture is to minimize cross-application communication as much as possible, as it can re-introduce the coupling. However, it is unavoidable in decentralized manufacturing systems due to distributed control and autonomy. Custom Events provide a basic mechanism to establish an indirect communication channel. But they are not easy to determine and apply strict contracts between micro frontends [16]. In this matter, the consumer-driven contract pattern must be investigated for flexible cross-application communication, which allows providers to get insight into the consumer's requirements and obligations.

C. HMI Design Philosophy Model

In the web-application development, CSS files describe how the HTML elements are displayed on screens. Furthermore, it can control the layout of web-pages, which is used to organize the operating screen elements. The dynamic loading of operating screen elements from HMI modules requires restructuring of the layouts. In the implementation, the layouts and screens are created using pre-configured CSS files. The dynamic generation of CSS files to support auto-configuration of operating screens by micro frontends is still work-in-progress.

D. Import Map

The Import Map is used as a registry of operating screen elements made available by HMI modules in the implementation. It has to be manually configured every time a new module is plugged-in/out, which does not completely satisfy the requirements for (auto) element registration.

E. Discovery and Registration

Dynamic discovery and registration are unexplored topics in micro frontend architecture. It is because micro frontend architecture is utilized for developing web applications, incremental features development, and code-base splitting. The modular and adaptable production is a unique use case for this architecture style. Therefore, topics like the dynamic discovery of micro frontends and run-time rendering of new HMI pages due to newly discovered modules are part of our ongoing research on modular HMI design.

VI. Conclusion

In this paper, we presented the ongoing work on the development of a modular HMI architecture for modular machines. The proposed solutions allows industrial HMIs to adapt to the current configuration of modular production systems. We presented a micro frontend architecture style as an implementation of such modular HMIs.

Some of the key benefits of the proposed solution over conventional HMI development approaches are: integration of automation modules without re-programming of HMIs screens, scalable organization of display screens, and role-based configuration of HMI screens can be presented.

There are still open points that must be addressed in future works. The list of open points in the research are:

- How to clearly define open interfaces for a micro frontend to interact with users and other micro frontend applications?
- How to develop a flexible cross-application communication without introducing any coupling between micro frontends?
- What modeling elements are required for HMI Design Philosophy Model and how to structure them?
- How to manage the import maps at runtime and provide dynamic discovery of micro frontends?

References

[1] A. Homay, A. Zoitl, M. De Sousa, and M. Wollschlaeger, "A survey: Microservices architecture in advanced manufacturing systems," *IEEE Int. Conf. Ind. Informatics*, vol. 2019-July, pp. 1165–1168, 2019.

[2] M. Geers, "Micro Frontends," *Micro Frontends*, Aug 2017. [Online]. Available: https://micro-frontends.org

[3] N. Komoda, "Service Oriented Architecture (SOA) in industrial systems," *2006 IEEE Int. Conf. Ind. Informatics, INDIN'06*, pp. 1–5, 2006.

[4] D. Sprott and L. Wilkes, "Understanding service-oriented architecture," *The Architecture Journal*, vol. 1, no. 1, pp. 10–17, 2004.

[5] H. Bloch, A. Fay, and M. Hoernicke, "Analysis of service-oriented architecture approaches suitable for modular process automation," *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, vol. 2016-November, pp. 1–8, 2016.

[6] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *IEEE Transactions on industrial informatics*, vol. 1, no. 1, pp. 62–70, 2005.

[7] O. Zimmermann, "Mircroservices tenets: Agile approach to service development and deployment," in *Proceedings of the Symposium/Summer School on Service-Oriented Computing*, 2016.

[8] S. Newman, *Building microservices: designing fine-grained systems.* " O'Reilly Media, Inc.", 2015.

[9] K. Thramboulidis, D. C. Vachtsevanou, and A. Solanos, "Cyber-physical microservices: An iot-based framework for manufacturing systems," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 232–239.

[10] M. Y. Afanasev, Y. V. Fedosov, A. A. Krylova, and S. A. Shorokhov, "An application of microservices architecture pattern to create a modular computer numerical control system," in *2017 20th Conference of Open Innovations Association (FRUCT)*. IEEE, 2017, pp. 10–19.

[11] L. Urbas, S. Hennig, H. Hager, F. Doherr, and A. Braune, "Towards context adaptive hmis in process industries," in *2011 9th IEEE International Conference on Industrial Informatics*. IEEE, 2011, pp. 244–249.

[12] A. Bozzon, M. Brambilla, P. Fraternali, P. Speroni, and G. Toffetti, "Applying web-based networking protocols and software architectures for providing adaptivity, personalization, and remotization features to industrial human machine interface applications," in *21st International Conference on Advanced Information Networking and Applications (AINA'07)*. IEEE, 2007, pp. 940–947.

[13] M. Mena, A. Corral, L. Iribarne, and J. Criado, “A progressive web application based on microservices combining geospatial data and the internet of things,” *IEEE Access*, vol. 7, pp. 104 577–104 590, 2019.

[14] T. Lojka, P. Šatala, J. Mocnej, and I. Zolotová, “Web technologies in industry hmi,” in *2015 IEEE 19th International Conference on Intelligent Engineering Systems (INES)*. IEEE, 2015, pp. 103–106.

[15] B. Madiwalar, B. Schneider, and S. Profanter, “Plug and produce for industry 4.0 using software-defined networking and opc ua,” in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 126–133.

[16] “Micro Frontends,” Apr 2020, [Online; accessed 20. Apr. 2020]. [Online]. Available: https://martinfowler.com/articles/micro-frontends.html

[17] “Vue.js,” Jun 2020, [Online; accessed 12. Jun. 2020]. [Online]. Available: https://vuejs.org