International Journal of Precision Engineering and Manufacturing (2024) 25:1073–1086
https://doi.org/10.1007/s12541-024-00960-6

Online ISSN 2005-4602
Print ISSN 2234-7593

REGULAR PAPER

Platform Supporting Intelligent Human–Machine Interface (HMI) Applications for Smart Machine Tools

Il-Ha Park[1] · Joo Sung Yoon[2] · Jin Ho Sohn[3] · Dong Yoon Lee[4]

Received: 1 August 2023 / Revised: 3 January 2024 / Accepted: 5 January 2024 / Published online: 22 January 2024

Abstract
As the Internet of Things, artificial intelligence, and the fourth industrial revolution advance, smart factories and machines increasingly gain intelligent features that enable the integration of more sophisticated functionalities. Approaches to achieving this intelligence involve both internal systems, such as human–machine interface (HMI), and external systems, such as big data platforms and cloud services. Although current research leans toward studying external systems, accomplishing intelligent functions through such means poses more challenges in achieving real-time responses during machining processes than using internal systems. When intellectualizing machine tools through internal HMI systems, three critical issues must be addressed. First, HMI functions are structured to depend on the HMI itself, leading to a ripple effect where a problem occurring in one HMI function impacts the entire system. Second, owing to differences in development tools and programming languages, the interconnectivity between functions developed by multiple stakeholders to be loaded onto the HMI may suffer, leading to potential inefficiencies and increased maintenance costs. Third, although various types of computer numerical control (CNC) machines need to communicate with the HMI function, the diverse communication methods and development tools used by each CNC manufacturer result in identical intelligent functions being developed separately for each CNC type. To address these challenges, this study proposes an innovative HMI platform capable of executing and developing various intelligent functions. The HMI platform and its major components are designed and implemented through component-based development (CBD). Subsequently, the performance and effectiveness of the platform are validated using quality attribute scenarios.

Keywords Component-based development · Multi-vendors' computer numerical control · Intelligent HMI · HMI application · Smart machine tools

1 Introduction

As the Internet of Things (IoT) and artificial intelligence (AI) technologies continue to advance, the fourth industrial revolution is driving the transformation of smart factories and machine tools within them into intelligent entities. A machine tool is a machine for handling or machining metal, the machining process is to get the desired shape by removing the material from the larger piece of raw material through cutting [1]. Intelligent machine tools refer to autonomous machine tools that possess the capability to learn and make decisions on their own [2]. Several methods for intellectualizing machine tools are being explored, with active research focusing on advancing the functionality and software that can be integrated into these machines [3–15]. Table 1 provides a summary of the research about intelligent machine tools. Loading intelligent functions onto machine

✉ Joo Sung Yoon
jsyoon@kyungnam.ac.kr

1 Department of Systems Management Engineering, University of Sungkyunkwan, 2066, Seobu-Ro, Jangan-Gu, Suwon-Si, Gyeonggi-Do, Republic of Korea

2 School of Mechanical Engineering, Kyungnam University, Changwon 51767, South Korea

3 Department of Industrial & Management Systems Engineering, KyungHee University, Deogyeong-Daero, Giheung-Gu, Yongin-Si, 1732 Gyeonggi-Do, Republic of Korea

4 Digital Transformation R&D Department, Korea Institute of Industrial Technology, 143 Hanggaulro, Sangrok-GuGyeonggi-do, Ansan-Si 15588, Republic of Korea

Table 1 Intelligent machine tools researches

Method of intelligence	References
Machining algorithm, software	[3, 4, 6–12]
Application system	[13–17]
Software with external systems	[5, 18–25]
Software with internal systems	[26, 27]

tools can be achieved through two approaches: utilizing external systems, such as big data platforms and cloud services [16–25], or employing internal systems like the Human Machine Interface (HMI). While significant research has concentrated on external systems, this method of loading intelligent functions is susceptible to network failures, and real-time responsiveness during the machining process can be challenging when compared to using internal systems. Notably, DMG MORI and Okuma have released products like CELOS [26] and OSP Suite [27], respectively, which employ intelligent functions loaded on HMIs, presented in the form of apps for self-diagnosis and active control.

Intelligent machine tools utilizing internal systems, such as HMIs, must address three critical issues that may arise during HMI development. The first issue pertains to the design and implementation dependency of general HMI functions, as they function as subcomponents of the HMI application software. Consequently, a problem occurring in one function can affect the entire HMI, potentially leading to compromised system stability. The second issue arises from the diversity of stakeholders capable of developing functions to be loaded onto the HMI. Machine Tool Builders (MTBs) are developing intelligent functions independently, and on-site workers, as well as external vendors, can also directly contribute to developing the necessary intelligent functions to enhance work efficiency. With diverse developers working on intelligent functions, variations in the development tools and programming languages used can lead to inconsistencies when integrating these functions onto the same HMI. This, in turn, can hinder the data connection efficiency between intelligent functions and result in additional costs and time spent on maintenance tasks. The third issue highlights the significant diversity in the types of Computer Numerical Control (CNC) machines targeted by the HMI's intellectual functions. MTBs and external vendors are developing intelligent functions to cater to various CNCs from manufacturers such as Fanuc and Siemens, based on customer demand. As each CNC manufacturer offers distinct communication methods and development environments (development tools, software programming languages, etc.), identical intelligent functions may require separate development efforts depending on the CNC development tool being used. For example, Fanuc has communication functions for each data, whereas Siemens has one communication function that uses an address assigned to the data as input parameter. Furthermore, developers working on these intelligent functions must familiarize themselves with one or more development tools.

This study introduces an HMI platform capable of executing and developing diverse intelligent functions. The HMI platform offers a unified execution environment where independent intelligent functions can be executed in connection, as well as a common development environment that allows intelligent functions to be developed for various types of CNCs or by different developers. To design and implement the HMI platform, the component-based development (CBD) methodology is employed, which constructs software architecture in independent units of components [28]. As the HMI platform serves as an environment for various consumers and suppliers of intelligent functions to interact, it must be adaptable to accommodate the diverse requirements of all participants. Following the CBD methodology, the proposed HMI platform is designed and implemented with a combination of independent components, minimizing the impact on other components in response to changes driven by specific requirements. The reuse of independent components contributes to a reduction in development time and costs as the HMI platform continues to evolve.

The structure of this paper is as follows. In Sect. 2, a comprehensive review of the literature related to intelligent machine tools is presented. Section 3 outlines the collection of requirements for the design and implementation of the HMI platform. Subsequently, in Sect. 4, design and implementation strategies are established through the definition and analysis of use cases and quality attributes, culminating in the composition of the architecture governing the relationship between the components of the HMI platform. The actual implementation and validation of the HMI platform, based on quality attribute scenarios, are discussed in Sect. 5. Finally, in Sect. 6, the findings are summarized, and future work in this area is explored.

2 Related Research

2.1 Intelligent Algorithm and Application System

Various studies have focused on intellectualizing machine tools through the advancement of algorithms used during the machining process. These proposed algorithms serve different purposes and can be broadly categorized into applications before, during, and after machining processing. For real-time monitoring during the machining process, Bhinge et al. [8] introduced an intelligent tool abrasion monitoring software utilizing support vector machines (SVM). In the realm of control during machining, Li et al. [6] presented a fuzzy reasoning approach that optimizes the on/off state of the device, reducing resting time, while Saini et al. [7] proposed an intelligent wheel position searching algorithm to enhance processing precision. Furthermore, energy-efficient operation of machine tools has been addressed in research such as Maher et al. [10], who put forward an energy demand modeling method for processing, and Abdulshahed et al. [12], who utilized thermal imaging cameras and applied the fuzzy c-means clustering methodology for thermal error modeling of machine tools. In terms of prediction after processing, where analysis of processing data is crucial, Abdulshahed et al. [11] introduced an adaptive neural-fuzzy approach based on cutting force, enabling accurate prediction of surface roughness during end milling. Additionally, Jia et al. [9] developed an energy prediction model employing the Gaussian process regression method.

Notably, beyond software-based approaches, some studies have also focused on application systems, which involve a combination of hardware and software execution. For monitoring and control during machining, Kang et al. [14] proposed a machine tool health monitoring and control system, while Wang et al. [15] designed a machine tool energy data analysis system for post-processing maintenance. Furthermore, Liu and Xu [16] introduced a fault diagnosis system.

2.2 Intelligent Machine Tools Architecture

A prominent research trend involves proposing architectures that facilitate the integration of intellectualization software, encompassing data collection, processing, and analysis. Chen et al. [17] introduced a cyber-physical system (CPS)-based structure that enables autonomous learning using data generated from intelligent machine tools. Similarly, Fei et al. [5] presented an information model and database, enabling machine tools to not only perform processing but also store data, employ data analysis algorithms, and utilize interfaces for these purposes. Furthermore, Liu et al. [18] implemented the concept of a digital twin to acquire and integrate manufacturing data, resulting in a machine tool architecture equipped with intelligent algorithms and a comprehensive database. Moreover, Liu et al. [19] proposed a machine tool architecture with three layers (NC, server, client) to acquire, store, and monitor real-time data, demonstrating its potential in predicting relevant relationships and processing outcomes.

In addition to architectural frameworks, certain studies have conceptualized intellectual functions based on specific architectures. For instance, Zuo et al. [20] developed and integrated modules capable of diagnosing defects, performing repair and self-maintenance of malfunctioning operations, and analyzing potential weaknesses in major components affecting machine reliability. This implementation was carried out within an architecture comprising application and cloud layers. Similarly, Gao et al. [21] devised and incorporated software for processing planning and tool path generation within an architecture composed of factory and cloud layers. Additionally, Tao et al. [22] designed and implemented software to monitor real-time device states and collect location data, utilizing an architecture consisting of five layers (resource, recognition, network, service, application). Other research efforts have focused on specialized services within CPS-based architectures. Lee et al. [23] proposed a diagnosis and maintenance management service applicable to CPS-based architectures. Cao et al. [24] addressed spindle maintenance, spindle state monitoring, and tool state monitoring and control, utilizing a multi-layer architecture encompassing sensing, decision-making, control, and physical CNC layers. Lastly, Mourtzis et al. [25] proposed a cloud-based monitoring function within an architecture incorporating physical factory, sensor network, machine tool operation, cloud infrastructure, and maintenance layers.

3 Requirements Analysis

This section conducts an analysis of the existing HMI to derive the requirements for the HMI platform's function (3.1), execution environment (3.2), and development environment (3.3). Additionally, the requirements are categorized as functional and non-functional, specifying the necessary conditions and capacities for the HMI platform. Functional requirements encompass criteria that the HMI platform must perform or enable the user to perform within the platform. Non-functional requirements comprise criteria essential for achieving optimal performance, such as ensuring the smooth execution of the HMI platform.

3.1 HMI Function

The HMI includes four fundamental and pivotal functions. First, the monitoring function identifies the currently executing NC file and the activated axis coordinate system, tool, and spindle data. Second, the file management function enables writing and execution of NC files, in addition to various file operations like copy and paste, to manage NC files effectively. Third, the tool management function is responsible for registering and deleting tools, along with managing the geometry and offset data of the registered tools. Lastly, the maintenance function identifies alarms occurring during processing and configures the necessary system parameters to compose the HMI function. Table 2 provides a summary of the NC data offered by each function, along with the corresponding data operation and communication methods.

- [R001] (Functional) The HMI platform supports reading/writing NC data (axis, spindle, tool, etc.).

Table 2 Function analysis

Function	Data	Operation	Communication
Monitoring	Axis Position, File path, File name, Tool number, Tool length, Tool radius, Feed speed, Feed override, Spindle speed, Spindle override, G modal, Part counter, Part run time	Read	Periodic
File management	Block string, Block counter, Sequence number, Program name, Program path, Program edited time	Create, Delete, Copy, Rename, Move	Non-periodic
Tool management	Tool number, Tool name, Tool offset, Tool length, Tool radius	Read, Write	Non-periodic
Maintenance	Alarm number, Alarm text, Axis limit plus/minus	Read, Write	Non-periodic

- [R002] (Functional) The HMI platform supports various NC file operations (create, delete, copy, paste, rename, move, etc.).
- [R003] (Functional) The HMI platform supports communication with NC in both periodic and non-periodic manners.
- [R004] (Functional) The HMI platform supports data communication between HMI functions.

To accommodate intelligent functions beyond the basic ones, the environment must allow for HMI function expansion. Moreover, to utilize results based on data analysis as an intelligent function, the data generated during processing must be stored permanently without volatilization. For instance, the condition analyzer app of CELOS saves and analyzes machine tool sensor data, enabling visual monitoring of the current device's availability and efficiency. Similarly, the OSP-AI app of OSP Suite saves and analyzes sensor data for self-diagnosing the state of the feed axes.

- [R005] (Functional) The HMI platform supports the expansion of HMI functions.
- [R006] (Functional) The HMI platform supports the storage of data generated during machining processing.

3.2 Execution Environment

To ensure efficient control and limited access to each HMI function, basic operations such as open and hide must be feasible. The system parameter change availability for the HMI system in Fanuc, for instance, requires altering the parameter value to true, while Siemens permits revision based on the granted authority for individual user accounts.

- [R007] (Functional) The HMI platform supports control of HMI functions (e.g., show, hide).
- [R008] (Functional) The HMI platform supports management of access rights that can be granted per HMI function.

To address concerns about system safety being compromised due to function dependency within the HMI, an independent execution environment must be provided. For instance, CELOS and OSP Suite load HMI intellectual functions in the form of apps [25, 27], ensuring independent execution. However, since an app is a self-contained software unit, the data exchange between independent functions becomes challenging when each function has separate execution environments (operating system, compile library, etc.). Therefore, a common execution environment for HMI functions is essential to facilitate fast and easy data exchange.

- [R009] (Non-functional) The HMI platform supports HMI functions to execute and operate independently.
- [R010] (Functional) The HMI platform supports a common execution environment that enables data exchange between various HMI functions.

3.3 Development Environment

The diverse range of HMI function developers, including on-site workers, CNC enterprises, and machine tool manufacturers, often involves multiple developers collaborating on a single function. To optimize data connection efficiency and minimize costs and time required for HMI function maintenance, a consistent development tool and function that all developers can use must be provided. Additionally, supporting an environment for seamless developer cooperation is essential.

- [R011] (Non-functional) The HMI platform supports a consistent development methodology (terminology, methods, etc.).
- [R012] (Non-functional) The HMI platform supports smooth cooperation between developers.

The various types of CNC that HMI functions aim to communicate with present a challenge, as each CNC enterprise offers distinct communication libraries. This can result in the need to develop separate communication protocols using different libraries for identical functions targeting different CNC types. To address this issue, the HMI platform should support a communication method that remains consistent regardless of the CNC type while also facilitating communication with various CNC types.

- [R013] (Non-functional) The HMI platform supports communication for various types of CNC.
- [R014] (Non-functional) The HMI platform supports an identical communication method regardless of CNC type.

4 HMI Platform Design

This section employs the CBD methodology to define the use cases of both functional and non-functional requirements and establish the analysis classes (4.1). Subsequently, the quality attribute is determined based on the non-functional requirements (4.2), and a design strategy, including design patterns, is derived (4.3). Finally, candidate components are identified, taking into account the quality attribute and use cases, and the relationships between the components are defined and structured (4.4).

4.1 Use Case

Use cases represent actions performed by the system. Based on the functional and non-functional requirements defined in Sect. 3, the use cases of the HMI platform are summarized in Table 3 and elaborated below.

- [U001] NC Communication: HMI functions loaded onto the platform can request data read/write with periodic and non-periodic communication for NC.
- [U002] NC File Management: HMI functions can manage NC files by creating and deleting NC files.
- [U003] App Communication: HMI functions can request data read/write with periodic, non-periodic, and callback communication for other HMI functions.
- [U004] DB Communication: Each HMI function requires a database that can be generated, deleted, and accessed to read or revise data.
- [U005] App Control: HMI platform users can activate or hide HMI functions as well as execute and complete them.
- [U006] App Management: New HMI functions can be registered and deleted in the HMI platform.
- [U007] Authorization Management: The authority for reading, writing, and executing HMI functions can be registered, deleted, and checked.

4.2 Quality Attribute

Quality attributes represent the non-functional requirements that the software must fulfill. ISO/IEC 25010 [29] is a standard used to objectively assess software quality, classifying and defining representative attributes that the software should possess. In this study, a new quality attribute is specified based on the standard quality attribute of ISO/IEC 25010.

4.2.1 Commonality

"Commonality" refers to the degree of uniform communication with different NC types using an identical approach. As per the non-functional requirements, the HMI platform must facilitate communication with NCs, irrespective of their type, using the same approach ([R013], [R014]). Thus, "commonality" is defined as a quality attribute of the HMI platform, characterized by providing identical approaches for various CNC types based on adaptability (ISO/IEC 25010). Adaptability represents the effectiveness and efficiency of supporting the system across various usage environments, making it one of the standard quality attributes.

4.2.2 Flexibility

"Flexibility" indicates the level of support for independent execution of HMI intelligent functions. As stated in the earlier non-functional requirements, the HMI platform must enable independent execution of HMI intelligent functions

Table 3 Use cases

ID	Use case	R001	R002	R003	R004	R005	R006	R007	R008	R009	R010	R011	R012	R013	R014
[U001]	NC Communication	●		●								●		●	●
[U002]	NC File Management		●	●								●		●	●
[U003]	App Communication				●					●	●	●	●		
[U004]	DB Communication						●					●			
[U005]	App Control							●		●					
[U006]	App Management					●				●					
[U007]	Authorization Management								●						

([R009]). Consequently, "flexibility" is defined as a quality attribute of the HMI platform, denoting the characteristic that allows intelligent functions to execute independently and flexibly. This attribute is based on modularity (ISO/IEC 25010), representing the degree of individual composition with minimal impact among standard quality attributes, and modifiability (ISO/IEC 25010), representing the ability to efficiently revise the system or product without defects.

4.2.3 Consistency

"Consistency" pertains to the level of support for a uniform and straightforward use of the HMI platform. As specified in the preceding non-functional requirements, the HMI platform must provide a simple and consistent system to ensure smooth user experience ([R011], [R012]). Therefore, "consistency" is defined as a quality attribute of the HMI platform, characterized by supporting a communication system that is both simple and consistent. It relies on learnability (ISO/IEC 25010), representing the degree to which the stated target can be achieved by learning the method of use, and operability (ISO/IEC 25010), representing the ease with which the system or product can be operated and controlled among standard quality attributes.

4.3 Design Strategy

4.3.1 [Commonality] Common Data Model

To ensure communication with NC data across different NC types, a common data model is designed. This expandable and standardized data schema includes entities, attributes, and their relationships [30]. In the HMI platform, the common data model is structured to store various types of NC data in a uniform data structure.

4.3.2 [Commonality] Template-Method Pattern

To facilitate communication with NC data regardless of the NC type, the NC communication component is designed using the template-method pattern. This design pattern defines identical functions in a superclass and only requires subclasses to override the parts that need to be changed [31]. In the HMI platform, a representative method for communication with NC is created in a superclass, and the subclass inherits this method, allowing the execution of methods provided by NC libraries.

4.3.3 [Flexibility] Broker Pattern

To eliminate dependencies between HMI functions, the broker component is designed using the broker pattern. This design pattern is employed when independent components have different execution environments and require communication between them. Broker components are equipped with client and server information, enabling them to coordinate communication between components [32]. In the HMI platform, the broker component treats each HMI function as an independent component and mediates communication between them.

4.3.4 [Consistency] Facade Pattern

To provide a simple and consistent user experience with the HMI platform, the facade component is designed using the facade pattern. This design pattern offers a straightforward and integrated interface, minimizing dependencies and communication within a system comprising complex subsystems [31]. The facade component implements the functions provided by the subsystem classes and effectively processes the allocated tasks. By incorporating the facade component into the HMI platform, a unified and user-friendly interface is provided.

4.3.5 [Consistency] Standardized Uniform Resource Identifier

To ensure a smooth user experience, the data communication standard of the HMI platform is designed as a standardized uniform resource identifier (URI). The URI is a unique character string used to identify specific resources, composed of schemas, hosts, queries, and so on [33]. The HMI platform supports data communication not only for NC but also for various targets, such as apps and databases. By implementing and utilizing a URI communication system, consistent communication is achieved for diverse targets within the HMI platform.

4.4 Platform Architecture

The candidate components that could form the architecture are analysis classes responsible for the described behaviors in the defined use cases. These analysis classes fall into three types: the boundary class, acting as the interface between users and other systems; the control class, providing system control behavior; and the entity class, storing and managing system data [28]. The component candidates are refined to derive the components of the HMI platform architecture, as illustrated in Fig. 1. The components include the unified NC I/F, serving as the interface between external NC systems; Command I/F, providing the interface for users and HMI apps; and various managers, such as the File Manager, Security Manager, App Manager, Database Manager, and Communication Manager, which handle system control. The Security Manager contains the permission list, the App Manager holds the list of apps, the Database Manager

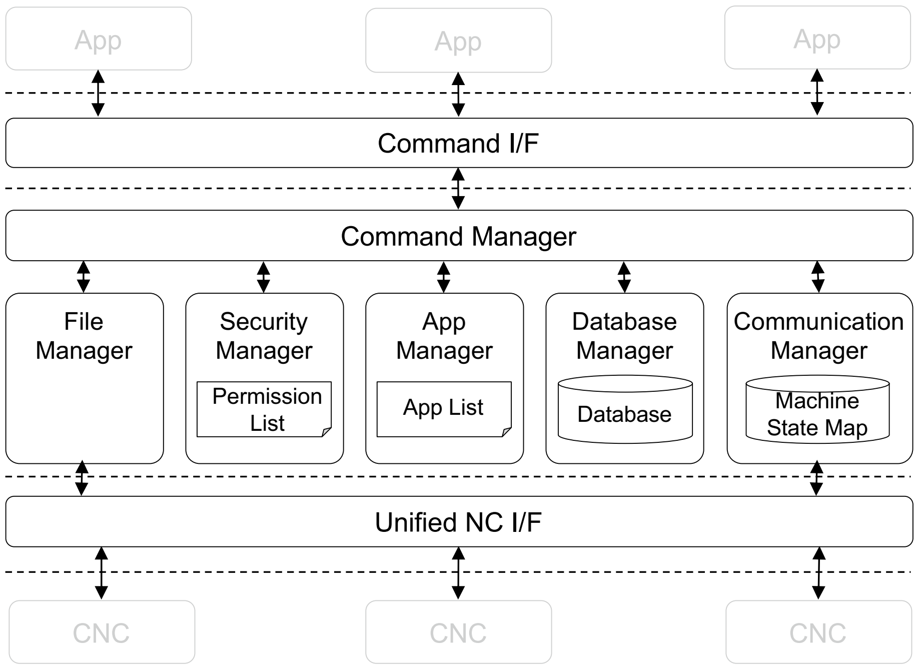


Fig. 1 HMI platform architecture

manages the databases, and the Communication Manager contains the data access point for the Machine State Map. The Command Manager acts as an intermediary for communication between the managers and HMI apps, as well as among different managers.

4.4.1 Communication Manager

The Communication Manager is identified as the NC communication control class. It determines the type of communication (periodic/non-periodic) and read/write operations, executes the requested communication, and saves the data in the Machine State Map. The Machine State Map is an entity class for NC data, designed using the common data model strategy for "commonality." It is internally supported by the Communication Manager.

4.4.2 File Manager

The File Manager is identified as the NC file management control class. It handles requests related to file creation, deletion, and renaming for the NC, as requested by the HMI app, and performs the specified file operations.

4.4.3 Unified NC I/F

The unified NC I/F is identified as the boundary class that interfaces with the NC. It provides a consistent interface for NC data, delivering and requesting NC data communication and NC file operations as requested by the Communication Manager and File Manager, respectively, to the NC.

4.4.4 Database Manager

The Database Manager is the component identified as the DB communication control class. It manages requested operations for generating, deleting, and reading/writing data in the database, as requested by the HMI app. The Database Manager performs the requested computations to store the data in the database, which is an entity class supported internally by the Database Manager.

4.4.5 App Manager

The App Manager combines three control classes, namely communication between functions, function control, and function management, into one component. It handles the management of HMI functions by registering or deleting them in the app list and executing requested operations for the HMI apps listed. The App Manager provides computations for the activation, hiding, execution, and completion of HMI apps, as well as read/write operations for HMI app data. The app list, which stores the identified and registered HMI app data as an entity class, internally supports the App Manager.

4.4.6 Security Manager

The Security Manager is the component responsible for access authority management. It handles the registration, deletion, and verification of authorities, such as read/write and execution of HMI functions, using the permission list. The permission list, which stores the authority data of the apps as an entity class, supports the Security Manager internally.

4.4.7 Command Manager

The Command Manager functions as a broker component, following the broker pattern as a design strategy for the "flexibility" quality attribute. It also serves as a facade component, following the facade pattern as a design strategy for the "consistency" quality attribute. The Command Manager acts as an intermediary for communication between HMI functions (broker component) and between HMI functions and the HMI platform (facade component). It mediates the execution of delivered commands between HMI functions.

4.4.8 Command I/F

The Command Interface is the boundary class that interfaces with HMI functions. It acts as a single interface for the Command Manager, serving as the interface for both the HMI manager and the HMI functions. The Command Interface receives commands from HMI users, functions, and managers, relaying them to the Command Manager, and returns the results of the command execution by the Command Manager.

5 HMI Platform Implementation and Validation

To validate the proposed HMI platform, the major components derived based on the quality attributes were implemented (5.1). Experimental scenarios were designed for each quality attribute, and the necessary experimental environment was set up accordingly (5.2). The validation of whether the HMI platform satisfied the quality attributes was performed through experimentation (5.3).

5.1 Component Implementation

5.1.1 Machine State Map

The Machine State Map is a component designed according to the common data model. To enable rapid data search, the map container from the standard template library (STL) provided by C++ was utilized for the implementation. In a previous study [34], a common machine tools state model that structuralized data components was proposed by analyzing the ISO 14649 process, machine tools, device data model, and commercialized CNC API to extract neutral data components. Based on these research findings, a data code capable of identifying NC data was assigned as the key of the map container, while the actual value of NC data was assigned as the corresponding value.

5.1.2 Unified NC I/F

The unified NC I/F was implemented using the template-method pattern design strategy to achieve the commonality quality attribute, as illustrated in Fig. 2. C++ in Visual Studio 2012 Professional, based on.NET Framework 4.5, was used for the implementation. Building on the results of previous research [34], the super class NC communication module, containing around 140 standardized functions, was implemented. Additionally, three subclass communication modules for Siemens, Fanuc, and CSCAM were developed, redefining the upper-class standard functions using Siemens Programming Package 4.7, Fanuc FOCAS2, and CSCAM HX20.

5.1.3 Command Manager and Command I/F

The Command Manager and Command I/F were implemented using the design strategies of flexibility and consistency, employing the broker pattern and facade pattern, respectively. C++ in Visual Studio 2012 Professional, based on.NET Framework 4.5, was used for the implementation. The Command Manager, acting as a broker component, mediates communication between HMI apps and incorporates functions enabling command and data requests for other HMI apps. As a facade component, it also implements functions provided by HMI managers that correspond to the subsystem, thereby facilitating minimized communication for several HMI managers. These functions are implemented in a library comprising 45 functions, which are distributed to HMI platform users and HMI app developers. Furthermore, communication mediation between HMI app-HMI app and HMI app-HMI manager is achieved using gRPC, a communication technology between processes that allows execution of functions or procedures from spaces with different

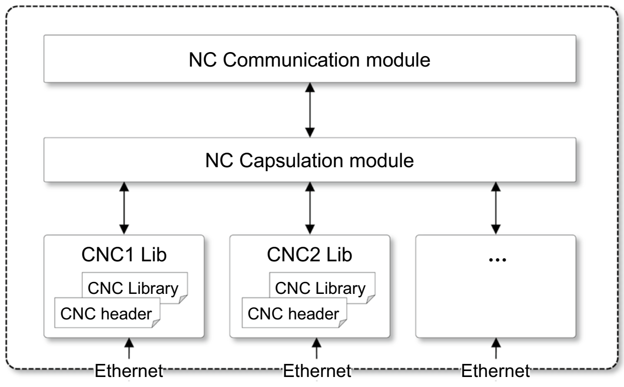


Fig. 2 Unified NC I/F [31]

addresses without separate coding. gRPC, an open-source advanced RPC framework developed by Google, enables communication between servers developed in different languages and frameworks. The communication rules are defined based on standardized URIs, ensuring consistency. URIs consist of four components: protocol, provider name, address, and filter, with the structure {Protocol}://{Provider_name}/{Address} ? {Filter}. The protocol represents the data address of the platform, provider_name is the name of the data provider (e.g., app or NC), address is the data address within the internal structure of the data provider, and filter is a specific condition that can identify data in detail, particularly in cases where the data provider is an NC.

- {Protocol}://{Provider_name}/{Address} ? {Filter}

5.2 Experimental Design

5.2.1 Definition of Experimental Scenario

The quality attribute scenario is determined based on six factors: source, stimulus, artifact, environment, response, and response measure [35]. For the three quality attributes, commonality, flexibility, and consistency, stimuli are generated for the artifact within a specific environment, and corresponding scenarios to measure the response are defined, as shown in Fig. 3. To validate "commonality," the HMI platform requests NC data communication in an environment where one CNC is connected to one app. The goal is to identify whether the data response values requested for the connected NC type are identical to the NC display for all three NC types. For "flexibility" validation, NC data communication is simultaneously requested for the HMI platform in an environment where one CNC is connected to two apps. The objective is to determine whether the remaining apps and HMI platform state operate normally when one app undergoes forced termination. To validate "consistency", the HMI platform requests NC data communication in an environment where one CNC is connected to two apps. Additionally, other app data communication is requested in another environment to identify whether a normal value has been returned, and the command that requests data is under an identical manner.

5.2.2 Establishment of Experimental Environment

C# in Visual Studio 2012 Professional, based on.NET Framework 4.5, was used to implement the apps. The apps, commandTester, commandTesterM, and commandTesterT, consist of three text boxes and one button. The first text box receives the address for data communication based on the address system rule as input, and the second text box filter receives the filter according to the address system rule for data communication as input. Clicking the read button executes the getData() function within the I/F library, using the address and filter inputs as parameters. The result of the getData() function is then displayed in the third text box.

The HMI platform is installed on one PC (CPU: INTEL i7 6,700K, RAM: DDR4 16GB X4EA, VGA: NVDIA GTX1070 8G), which connects to three types of CNCs: Fanuc (31i-B), Siemens (840Dsl), and CSCAM (Hz) controller simulator through Ethernet communication. The communication setting (machineList.xml) file is revised to establish the connection between the HMI platform and NC. This file contains fields for ID, LIB, Name, IP Address, and Port. The ID of the NC that will be connected is inserted (default: 1), and the NC library and NC dll names for connection are input in LIB and Name, respectively. The IP Address and Port fields are used to input the IP and Port of the NC Ethernet communication that will be connected. To register the developed app into the HMI platform, an app info file is created, registering the app ID and app name for each app, and the corresponding file is added to a certain path.

5.3 Validation Experiment

5.3.1 Commonality

The HMI platform was connected to one of the three NC types: Fanuc, Siemens, or CSCAM. NC data communication

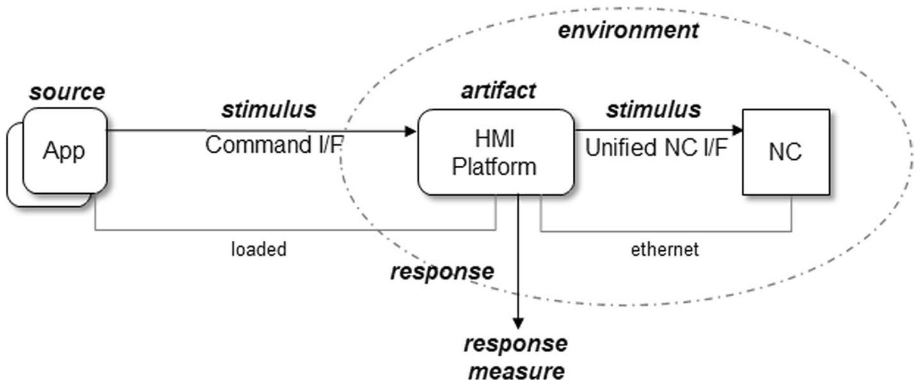


Fig. 3 Experimental concept

was requested for the HMI platform using commandTesterM. The NC data were virtualized based on the Machine State Map, which established a hierarchical relationship between data entities, and each NC data was assigned an address. The data requested in the experiment was the machine coordinate value (machine position), which is a lower attribute of the machine-channel-axis entity. The NC address for the machine coordinate value was as follows:

- NC address: Data://machine/channel/axis/machineposition?machine = 1&channel = 1&axis = 1

The return value from commandTesterM was compared to the output value displayed on the screen to determine whether they were identical. The experimental results depicted in Fig. 4 indicated that when the HMI platform was connected to the first CNC Siemens, the machine coordinate value 1,000 of the 3rd axis of Siemens was identical to the displayed value of 1,000 (a). Similarly, when the HMI platform was connected to the second CNC Fanuc, the machine coordinate value − 19.255 of the second axis of Fanuc 2 matched the displayed value of − 19.255 (b). Moreover, when connected to the third CNC CSCAM, the machine coordinate value 176 of the first axis of CSCAM was identical to the displayed value of 176 (c).

The results demonstrated that a consistent address system was used for reading communication requests across the three types of NCs, and the return values from the communication were identical to the displayed values. This is a important difference from other HMI where identical intelligent functions may require separate development efforts, and confirmed that communication for various types of CNCs was supported, satisfying the non-functional requirement for identical approaches regardless of the NC type [R014]. The HMI platform's implementation using inheritance for the unified NC I/F and standardized shared memory enabled consistent communication for other types of CNCs, fulfilling the quality attribute of "commonality" [R010].

5.3.2 Flexibility

The HMI platform was connected to the Fanuc NC, and two data requests were made: one for NC monitoring using commandTesterM to retrieve the machine position value of the first axis and another for tool monitoring using commandTesterT to retrieve the name of the first tool. Additionally, a forced termination of commandTesterM was performed during the experiment.

The experimental results displayed in Fig. 5 indicated that commandTesterM successfully read the machine coordinate value of 213.943, and commandTesterT concurrently retrieved the value of TOOL1. The data request logs for both commandTesterM and commandTesterT were simultaneously recorded. Furthermore, the results of the forced termination of commandTesterM revealed that the machine coordinate value request was abnormally halted, but the tool name request of commandTesterT and the overall platform continued to operate smoothly without any errors.

These findings demonstrate that despite the forced termination of commandTesterM due to an error, commandTesterT and the HMI platform operated independently and effectively. This is a significant difference from other HMI where a problem occurring in one function can affect the entire HMI, and confirms the fulfillment of the non-functional requirement for supporting independent execution of HMI functions [R009]. The HMI platform's design, which separates apps by mediating communication between them using a Command Manager as a broker component, ensures that apps remain independent, and any issues occurring within one app do not adversely affect other apps.

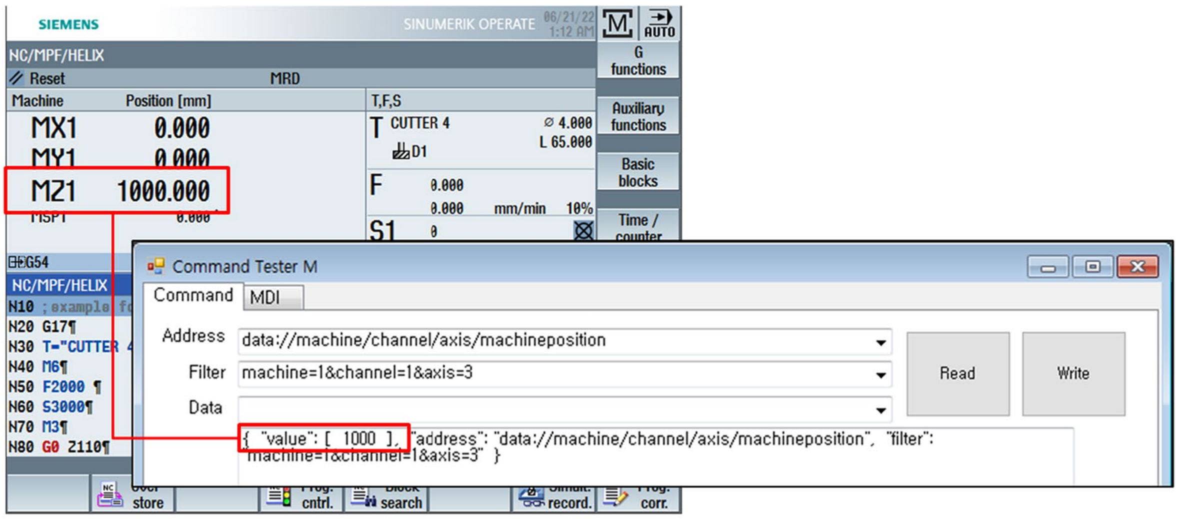
5.3.3 Consistency

The HMI platform was connected to Siemens, and two data requests were made: one for NC monitoring using commandTesterM to retrieve the machine coordinate value of the second axis, and another for app data using commandTester to retrieve the app name data of the "App Sample" using the following app address:

- NC address: Data://machine/channel/axis/machineposition?machine = 1&channel = 1&axis = 1
- App address: Data://app sample/data

The normal return of the app name value of the requested App Sample by commandTester is identified. The experimental results showed that commandTesterM returned the value of − 17.31, and the app name value of the requested "App Sample" by commandTester was returned as {"name": [app sample]}.

These results displayed in Fig. 6 indicate that the NC data request address system and the app data request address system were consistent according to the standard communication criterion {Protocol}://{Provider_name}/{Address}?{Filter}. This meets the non-functional requirements [R12, R11] that the HMI platform supports smooth collaboration between developers and consistent development rules, unlike existing development environments where standardized development rules are not provided when multiple developers collaborate to develop HMI functions. The design and implementation of the HMI platform involved mediating communication between apps and managers, with the Command Manager serving as a facade component, and providing a simple and consistent communication system to one or more components through the command I/F. Consequently, the communication standards for NC



NC/MPF/HELIX

Machine	Position [mm]
MX1	0.000
MY1	0.000
MZ1	1000.000

T CUTTER 4 ⌀ 4.000 L 65.000 D1

F 0.000 0.000 mm/min 10%

S1 0

```
N10 ;example
N20 G17¶
N30 T="CUTTER 4
N40 M6¶
N50 F2000 ¶
N60 S3000¶
N70 M3¶
N80 G0 Z110¶
```

Command Tester M

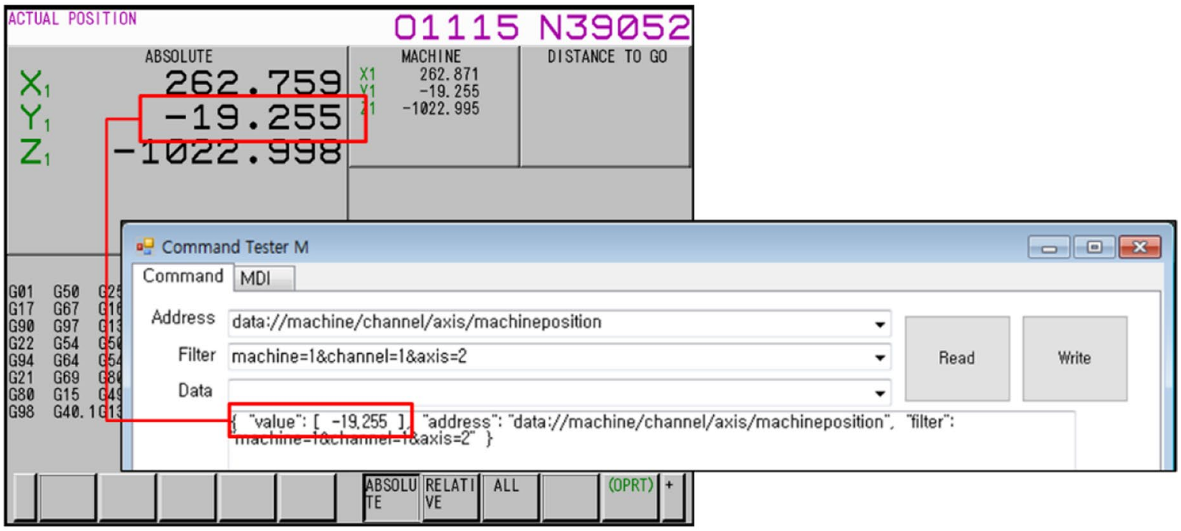
Address: data://machine/channel/axis/machineposition

Filter: machine=1&channel=1&axis=3

Data:

```
{ "value": [ 1000 ], "address": "data://machine/channel/axis/machineposition", "filter": "machine=1&channel=1&axis=3" }
```

(a) Siemens NC display and data communication



ACTUAL POSITION O1115 N39052

	ABSOLUTE	MACHINE	
X1	262.759	X1 262.871	
Y1	-19.255	Y1 -19.255	
Z1	-1022.998	Z1 -1022.995	

```
G01 G50
G17 G67
G90 G97
G22 G54
G94 G64
G21 G69
G80 G15
G98 G40.1
```

Command Tester M

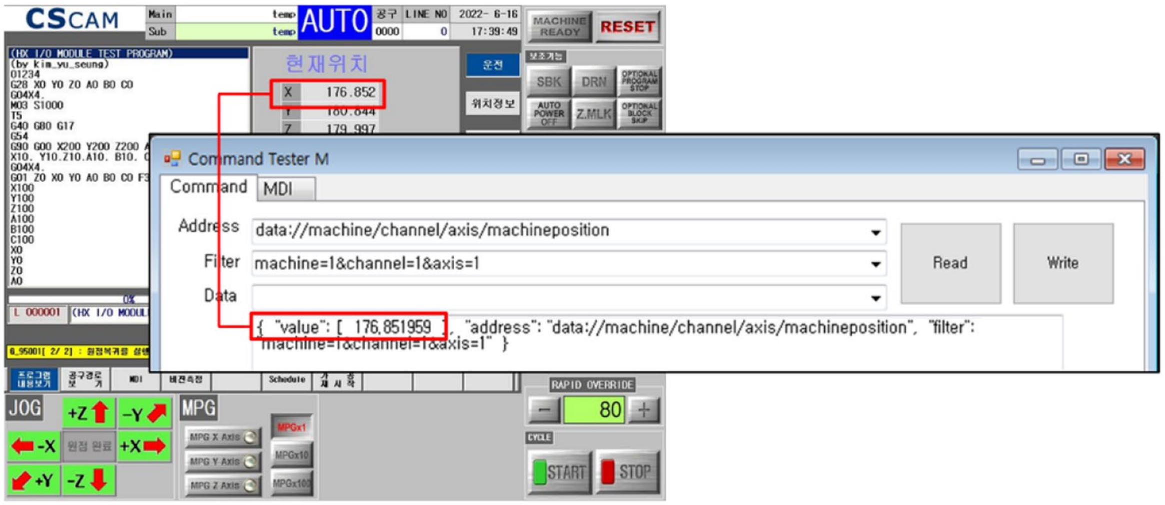
Address: data://machine/channel/axis/machineposition

Filter: machine=1&channel=1&axis=2

Data:

```
{ "value": [ -19.255 ], "address": "data://machine/channel/axis/machineposition", "filter": "machine=1&channel=1&axis=2" }
```

(b) Fanuc NC display and data communication



현재위치

X 176.852

Command Tester M

Address: data://machine/channel/axis/machineposition

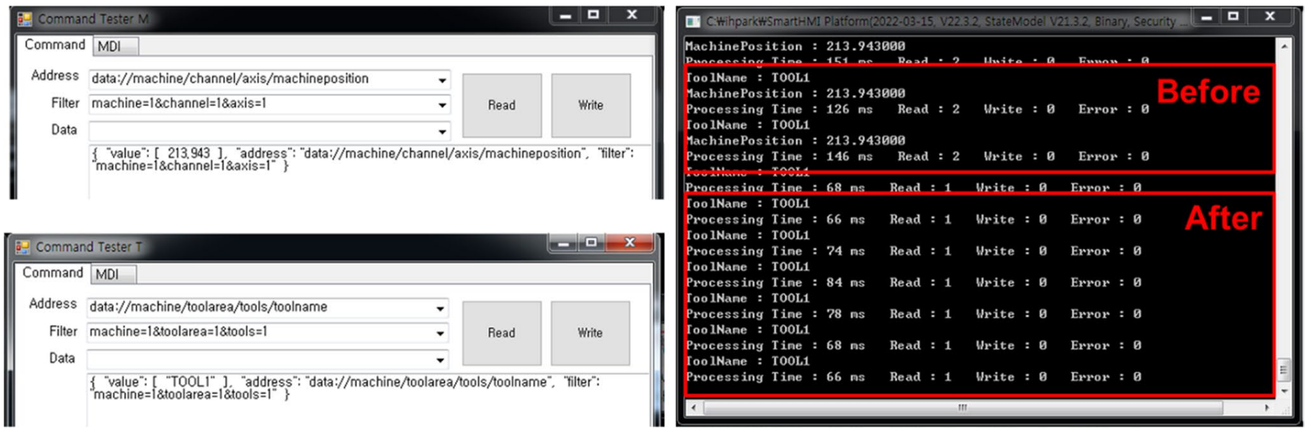
Filter: machine=1&channel=1&axis=1

Data:

```
{ "value": [ 176.851959 ], "address": "data://machine/channel/axis/machineposition", "filter": "machine=1&channel=1&axis=1" }
```

(c) CSCAM NC display and data communication

Fig. 4 Commonality experiment results



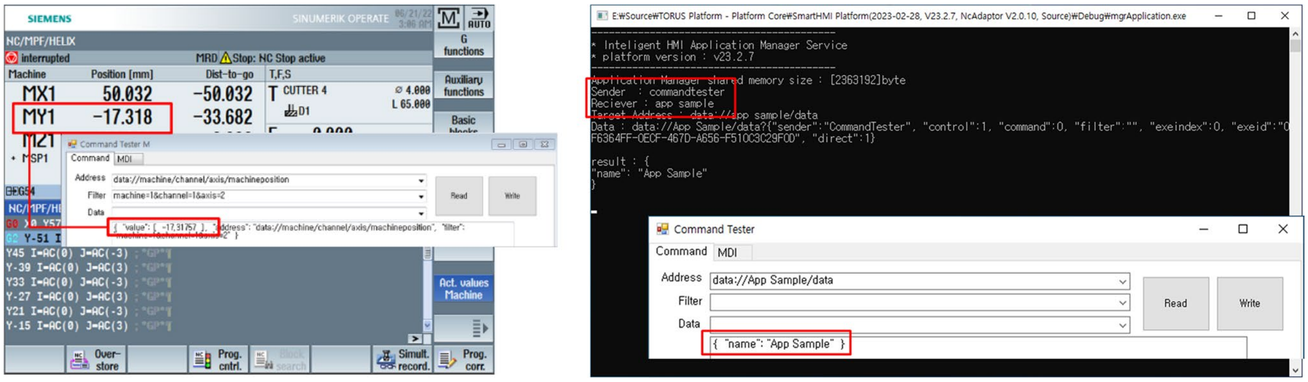
Address data://machine/channel/axis/machineposition
Filter machine=1&channel=1&axis=1
Data
{ "value": [213.943], "address": "data://machine/channel/axis/machineposition", "filter": "machine=1&channel=1&axis=1" }

Address data://machine/toolarea/tools/toolname
Filter machine=1&toolarea=1&tools=1
Data
{ "value": ["TOOL1"], "address": "data://machine/toolarea/tools/toolname", "filter": "machine=1&toolarea=1&tools=1" }

C:₩ihpark₩SmartHMI Platform(2022-03-15, V22.3.2, StateModel V21.3.2, Binary, Security ...

```
MachinePosition : 213.943000
Processing Time : 151 ms   Read : 2   Write : 0   Error : 0
ToolName : TOOL1
MachinePosition : 213.943000
Processing Time : 126 ms   Read : 2   Write : 0   Error : 0
ToolName : TOOL1
MachinePosition : 213.943000
Processing Time : 146 ms   Read : 2   Write : 0   Error : 0
ToolName : TOOL1
Processing Time : 68 ms   Read : 1   Write : 0   Error : 0
ToolName : TOOL1
Processing Time : 66 ms   Read : 1   Write : 0   Error : 0
ToolName : TOOL1
Processing Time : 74 ms   Read : 1   Write : 0   Error : 0
ToolName : TOOL1
Processing Time : 84 ms   Read : 1   Write : 0   Error : 0
ToolName : TOOL1
Processing Time : 78 ms   Read : 1   Write : 0   Error : 0
ToolName : TOOL1
Processing Time : 68 ms   Read : 1   Write : 0   Error : 0
ToolName : TOOL1
Processing Time : 66 ms   Read : 1   Write : 0   Error : 0
```

Fig. 5 Flexibility experiment results



NC/MPF/HELIX
interrupted MRD Stop: NC Stop active

Machine	Position [mm]	Dist-to-go
MX1	50.032	-50.032
MY1	-17.318	-33.682

T,F,S
T CUTTER 4 ⌀ 4.000 L 65.000 D1

```
Y45 I=AC(0) J=AC(-3)
Y-39 I=AC(0) J=AC(3)
Y33 I=AC(0) J=AC(-3)
Y-27 I=AC(0) J=AC(3)
Y21 I=AC(0) J=AC(-3)
Y-15 I=AC(0) J=AC(3)
```

Address data://machine/channel/axis/machineposition
Filter machine=1&channel=1&axis=2
Data
{ "value": [-17.31757], "address": "data://machine/channel/axis/machineposition", "filter": "machine=1&channel=1&axis=2" }

E:₩Source₩TORUS Platform - Platform Core₩SmartHMI Platform(2023-02-28, V23.2.7, NcAdaptor V2.0.10, Source)₩Debug₩mgrApplication.exe

```
-----------------------------------------
* Inteligent HMI Application Manager Service
* platform version : v23.2.7
-----------------------------------------
Application Manager shared memory size : [2363192]byte
Sender   : commandtester
Reciever : app sample
Target Address : data://app sample/data
Data : data://App Sample/data?{"sender":"CommandTester", "control":1, "command":0, "filter":"", "exeindex":0, "exeid":"0
F6364FF-0ECF-467D-A656-F510C3C29F0D", "direct":1}

result : {
"name": "App Sample"
}
```

Address data://App Sample/data
Filter
Data
{ "name": "App Sample" }

Fig. 6 Consistency experiment results

communication and communication between apps remain identical, ensuring a coherent and standardized approach within the HMI platform.

6 Conclusions

This study presented an HMI platform capable of expanding HMI intelligent functions while supporting a common execution and development environment for such functions. Using the CBD development methodology, the requirements of the HMI platform were gathered, defined, and further refined as use cases and quality attributes. Based on the analysis of these use cases and quality attributes, the HMI platform was designed and implemented using a combination of independent components. The quality attributes chosen for the HMI platform were commonality, flexibility, and consistency, and the platform's validation was conducted based on scenarios devised for each quality attribute.

The developed HMI platform allows for the expansion of HMI functions within a single machine tool. Additionally, it provides an environment where HMI functions can be executed commonly, thereby enabling efficient support for the connectivity between HMI functions. Moreover, the platform offers a development environment with a standardized approach, ensuring that various types of CNC are not dependent on specific NC types. This resolution addresses the issue of developers having to understand different library development methods for each NC and eliminates the inefficiency of developing identical software and intellectual functions twice. As the type and amount of data stored in the HMI platform's database increases, intelligent functions can be further advanced by applying AI technology to this data.

The HMI platform resulting from this research is currently available at the Intelligent HMI platform (http://www.torus.co.kr/), and it is freely accessible to individuals, non-profit organizations, and businesses. While the scope of this research focused on the intellectualization of a single machine tool, it is evident that future advancements would involve extending the platform to accommodate multi-machine tool intellectualization. For the realization of smart manufacturing and production sites and their digitalization in the future, the HMI platform will be expanded to enable interconnection with IoT devices capable of acquiring sensor signals and PLC-based non-processing devices. Additionally, it will facilitate the integration of multiple CNC devices into the HMI platform proposed in this research.

Acknowledgements This work is supported by the Ministry of Trade, Industry and Energy, Korea (20012807).

References

1. Mostaghimi, H., Park, C. I., Kang, G., Park, S. S., & Lee, D. Y. (2021). Reconstruction of cutting forces through fusion of accelerometer and spindle current signals. *Journal of manufacturing processes, 68*, 990–1003.
2. Chen, J., Hu, P., Zhou, H., Yang, J., Xie, J., Jiang, Y., Gao, Z., & Zhang, C. (2019). Toward intelligent machine tool. *Engineering, 5*(4), 679–690.
3. Nam, J. S., & Kwon, W. T. (2022). A study on tool breakage detection during milling process using LSTM-autoencoder and gaussian mixture model. *International Journal of Precision Engineering and Manufacturing (IJPEM), 23*(6), 667–675.
4. Mostaghimi, H., Park, S. S., Lee, D. Y., Nam, S., & Nam, E. (2023). Prediction of tool tip dynamics through machine learning and inverse receptance coupling. *International Journal of Precision Engineering and Manufacturing (IJPEM), 24*(10), 1739–1752.
5. Fei, Z., Li, S., Chang, Q., Wang, J., & Huang, Y. (2018). Fuzzy petri net based intelligent machine operation of energy efficient manufacturing system. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 1593–1598. IEEE.
6. Li, G., Zhou, H., Jing, X., Tian, G., & Li, L. (2017). An intelligent wheel position searching algorithm for cutting tool grooves with diverse machining precision requirements. *International Journal of Machine Tools and Manufacture, 122*, 149–160.
7. Saini, A., Vanraj, D. G., Pabla, B. S., & Dhami, S. S. (2017). Intelligent tool wear monitoring in machining TI6AL4V alloy using support vector machines.
8. Bhinge, R., Biswas, N., Dornfeld, D., Park, J., Law, K. H., Helu, M., & Rachuri, S. (2014). An intelligent machine monitoring system for energy prediction using a Gaussian Process regression. In *2014 IEEE International Conference on Big Data (Big Data)*, pp. 978–986. IEEE.
9. Jia, S., Tang, R., & Lv, J. (2014). Therblig-based energy demand modeling methodology of machining process to support intelligent manufacturing. *Journal of Intelligent Manufacturing, 25*(5), 913–931.
10. Maher, I., Eltaib, M. E. H., Sarhan, A. A., & El-Zahry, R. M. (2015). Cutting force-based adaptive neuro-fuzzy approach for accurate surface roughness prediction in end milling operation for intelligent machining. *The International Journal of Advanced Manufacturing Technology, 76*(5), 1459–1467.
11. Abdulshahed, A. M., Longstaff, A. P., Fletcher, S., & Myers, A. (2015). Thermal error modelling of machine tools based on ANFIS with fuzzy c-means clustering using a thermal imaging camera. *Applied Mathematical Modelling, 39*(7), 1837–1852.
12. Abdulshahed, A. M., Longstaff, A. P., & Fletcher, S. (2015). The application of ANFIS prediction models for thermal error compensation on CNC machine tools. *Applied Soft Computing, 27*, 158–168.
13. Klancnik, S., Brezocnik, M., & Balic, J. (2016). Intelligent CAD/CAM system for programming of CNC machine tools. *International Journal of Simulation Modelling, 15*(1), 109–120.
14. Kang, H. S., Lee, J. Y., & Lee, D. Y. (2020). An integrated energy data analytics approach for machine tools. *IEEE Access, 8*, 56124–56140.
15. Wang, C., Lin, W.-Y., & Young, H.-T. (2014). An intelligent fault diagnosis system for machine tools. *International Journal of Automation and Smart Technology, 4*(3), 150–156.
16. Liu, C., & Xu, X. (2017). Cyber-physical machine tool–the era of machine tool 4.0. *Procedia Cirp, 63*, 70–75.
17. Chen, J., Yang, J., Zhou, H., Xiang, H., Zhu, Z., Li, Y., Lee, C., & Xu, G. (2015). CPS modeling of CNC machine tool work processes using an instruction-domain based approach. *Engineering, 1*(2), 247–260.
18. Liu, C., Vengayil, H., Zhong, R. Y., & Xu, X. (2018). A systematic development method for cyber-physical machine tools. *Journal of Manufacturing Systems, 48*, 13–24.
19. Liu, W., Kong, C., Niu, Q., Jiang, J., & Zhou, X. (2020). A method of NC machine tools intelligent monitoring system in smart factories. *Robotics and Computer-Integrated Manufacturing, 61*, 101842.
20. Zuo, Y., Wang, H., Wu, G., Gu, Y., & Qiao, W. (2019). Research on remote state monitoring and intelligent maintenance system of CNC machine tools. *The Journal of Engineering, 2019*(23), 8671–8675.
21. Gao, W., Zhang, C., Hu, T., & Ye, Y. (2019). An intelligent CNC controller using cloud knowledge base. *The International Journal of Advanced Manufacturing Technology, 102*(1), 213–223.
22. Tao, F., Zuo, Y., Da Xu, L., & Zhang, L. (2014). Iot-based intelligent perception and access of manufacturing resource toward cloud manufacturing. *IEEE Transactions on Industrial Informatics, 10*(2), 1547–1557.
23. Lee, J., Bagheri, B., & Kao, H.-A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters, 3*, 18–23.
24. Cao, H., Zhang, X., & Chen, X. (2017). The concept and progress of intelligent spindles: A review. *International Journal of Machine Tools and Manufacture, 112*, 21–52.
25. Mourtzis, D., Vlachou, E., Milas, N., & Xanthopoulos, N. (2016). A cloud-based approach for maintenance of machine tools and equipment based on shop-floor monitoring. *Procedia Cirp, 41*, 655–660.
26. DMG Mori (2023). Celos. Retrieved July 25, 2023. https://en.dmgmori.com/products/digitization/celos
27. Okuma (2023). OSP suite. Retrieved July 25, 2023. https://www.okuma.com/osp-suite
28. Brown, A. W. (2000). *Large-scale, component-based development* (Vol. 1). Prentice Hall PTR.
29. ISO/IEC 25010 (en) Systems and software engineering (2011). Retrieved July 25, 2023. https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en.

30. Microsoft. (2022). Common data model. Retrieved July 25, 2023. https://learn.microsoft.com/en-us/common-data-model/
31. Gamma, E. (2012). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.
32. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (2013). *Pattern-oriented software architecture*. Wiley.
33. Berners-Lee, T., Fielding, R., & Masinter, L. (2005). RFC 3986: Uniform resource identifier (uri): Generic syntax.
34. Yoon, J. S., Park, I. H., Sohn, J. H., & Kim, H. J. (2018). Development of unified interface for multi-vendors' CNC based on machine state model. *Journal of the Korean Society for Precision Engineering, 35*(2), 151–156.
35. Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice*. Addison-Wesley Professional.

Il-Ha Park received the M.S. degree in Industrial Engineering from Dongguk University. She worked at Korea Institute of Industrial Technology, until April 2021. She is currently researcher in the Advanced Institute of Convergence Technology and a Ph.D. student in Sungkyunkwan university. Her current research topics are anomaly detection and big data.

Joo-Sung Yoon received the Ph.D. degree in Industrial Engineering from POSTECH. He worked at Korea Institute of Industrial Technology, until June 2020. He is currently Assistant Professor in the School of Mechanical Engineering, Kyungnam University. His current research topics are smart machine tool system and intelligent HMI of CNC.

JinHo Sohn received a master's degree in Computer Engineering from TECH UNIVERSITY OF KOREA. He worked at LS MECAPION, until November 2012. He is currently a senior researcher with the SW Development team #2 in HYUNDAI WIA CORP. and he is a doctoral student in the Department of Industrial & Management Systems Engineering at Kyung Hee University. His current research topics are smart factory platforms based on machine tools, collaborative robots, and AMRs.

Dong Yoon Lee received the Ph.D. degree in Mechanical Engineering from KAIST. He worked at Samsung Corning Precision Glass, until June 2006. He is currently a Principal Researcher with the Digital Transformation R&D Department, Korea Institute of Industrial Technology. His current research topics are in-process monitoring/control via CNC communication and sensors, virtual machining, and process optimization.