applied sciences

MDPI

Article

An Intelligent Human–Machine Interface Architecture for Long-Term Remote Robot Handling in Fusion Reactor Environments

Tamara Benito *,† and Antonio Barrientos

Centro de Automática y Robótica, Universidad Politécnica de Madrid—Consejo Superior de Investigaciones Científicas, 28006 Madrid, Spain; antonio.barrientos@upm.es

* Correspondence: tamara.benito@alumnos.upm.es

† Current address: GTD Science, Infrastructures and Robotics, Av. Leonardo Da Vinci, 28906 Getafe, Spain.

Abstract: This paper addresses the intricate challenge posed by remote handling (RH) operations in facilities with operational lifespans surpassing 30 years. The extended RH task horizon necessitates a forward-looking strategy to accommodate the continuous evolution of RH equipment. Confronted with diverse and evolving hardware interfaces, a critical requirement emerges for a flexible and adaptive software architecture based on changing situations and past experiences. The paper explores the inherent challenges associated with sustaining and upgrading RH equipment within an extended operational context. In response to this challenge, a groundbreaking, flexible, and maintainable human–machine interface (HMI) architecture named MAMIC is designed, guaranteeing seamless integration with a diverse range of RH equipment developed over the years. Embracing a modular and extensible design, the MAMIC architecture facilitates the effortless incorporation of new equipment without compromising system integrity. Moreover, by adopting this approach, nuclear facilities can proactively steer the evolution of RH equipment, guaranteeing sustained performance and compliance throughout the extended operational lifecycle. The proposed adaptive architecture provides a scalable and future-proof solution, addressing the dynamic landscape of remote handling technology for decades.

Keywords: remote handling; robotic; intelligent human–machine interface; software architecture evolution; fusion energy; modular architecture; software lifecycle

Citation: Benito, T.; Barrientos, A. An Intelligent Human–Machine Interface Architecture for Long-Term Remote Robot Handling in Fusion Reactor Environments. *Appl. Sci.* **2024**, *14*, 4814. https://doi.org/10.3390/app14114814

Academic Editors: Benjamim Fonseca, Daniel Schneider, António Correia and Tommi Kärkkäinen

Received: 11 April 2024
Revised: 14 May 2024
Accepted: 28 May 2024
Published: 2 June 2024

1. Introduction

Nowadays, the field of remote handling (RH) integrates many technical fields (electronics, communications, control, computer vision, artificial intelligence, augmented reality, etc.), and all of them, in general, work synergistically. Currently, there are outstanding new fields that add intelligence to RH robots; this is something very interesting because it simplifies the traffic and criticality of the information exchanged from/towards the operator, letting the robot act on its own [1]. Moreover, if we were able to model how the best operator carries out a series of complex tasks, we could reduce task times and human error risks introduced due to fatigue, boredom, or lack of experience.

These technological advances and others that RH is currently working on have made it possible to achieve great innovations in the nuclear fusion sector. Since 1930, scientists and increasingly also engineers have been continuously searching to discover how to recreate and harness nuclear fusion to provide virtually limitless clean, safe, and affordable energy to meet the world's energy demand. Fusion can generate four times more energy per kilogram of fuel than fission, which is used in nuclear power plants, and nearly four million times more energy than burning oil or coal.

More than 50 countries have carried out multiple nuclear fusion and plasma physics research, and fusion reactions have been successfully produced in many experiments.

Different designs and magnet-based machines have been developed to initiate fusion, like stellarators and tokamaks (this design is under study in this thesis), as well as approaches that rely on lasers, linear devices, and advanced fuels.

One of these countries is France, where the world's largest international fusion facility, named ITER, is being designed and manufactured across the world [2]. ITER endeavors to demonstrate the scientific and technological viability of producing fusion energy and validate technology and concepts for future fusion power plants that will generate electricity, referred to as DEMO. ITER will initiate its initial experiments in the latter half of the 2020s, and full-power tests should begin in the latter half of the 2030s [3]. The timeline for DEMO varies among different countries, but experts agree that an electricity-generating fusion power plant could be constructed and operational by 2050.

All these facilities have the same problem: the RH control systems will need to be operated from a dedicated remote control room that consists of many modules and technologies that must be operational over the lifetime of the fusion project (estimated at more than 30 years, including initial assembly and final decommissioning in the case of the ITER project). Therefore, the operation and supervision of these RH systems in environments requiring long-term flexibility and safety have increasingly necessitated the adoption of efficient graphical interfaces, prompting the need for a new innovative architecture proposal: Multiple Agent Managerial Intelligent Control (MAMIC). The MAMIC model allows the interaction of these systems with the operator in a unique, fast, and simple manner in response to varying situations and past experiences. Moreover, the repeatability required for industrialization underscores the significant importance of having reference models rather than each implementation being custom-built.

Section 2 reviews the evolution of software architecture in recent decades due to the increasing complexity of computing systems and the need to adapt to rapid technological changes and concludes that combining different approaches can lead to more robust architectures, such as those that RH facilities require. Section 3 presents the preliminary stages of the study (requirements and software lifecycle) aimed at developing a proposal for an architecture model that integrates multiple methodologies. Section 4 explores a completely new architecture model that addresses the anticipated needs in the technical specification phase while considering its lifetime, cost reduction, and risk mitigation over time. Last, the model application in the ITER case is summarized for convenience in Sections 5 and 6, including the model validation into the requirement phase, while Section 7 provides a short conclusion and future work. Following a list of acronyms, the article concludes with one appendix. Appendix A provides a detailed description of the remote robot handling intelligent human–machine interface architecture requirements.

2. Software Architecture Models Evolution Study

In the past decades, the software architecture landscape has witnessed significant evolution driven by the escalating complexity of computing systems and the imperative to adapt to a rapidly changing technological environment. During this period, diverse architectural frameworks and methodologies have emerged and evolved to address the unique challenges faced by software developers and architects. Among these architectures and methodologies, the most influential in the aforementioned environments are the following:

- Anthropomorphic approach: Rooted in human-centered design principles, the anthropomorphic approach places a strong emphasis on understanding and accommodating human needs and behaviors in system design [4]. While it enhances the user experience and adaptability, it may introduce complexity in implementation and require extensive user research.
- Predictive modeling: Leveraging data analytics and machine learning techniques, predictive modeling enables systems to anticipate future events and trends based on historical data. This fosters informed decision-making and resource optimization, but may be limited by data quality and the dynamic nature of real-world scenarios [5,6].

- Cognitive architecture: Inspired by human cognition, cognitive architecture seeks to replicate cognitive processes within software systems, enabling them to perceive, reason, and adapt like humans. While it enhances system intelligence and user interaction, it may entail significant computational overhead and require complex modeling [7].
- Hexagonal architecture: Offering a modular and adaptable design approach, hexagonal architecture decouples core system logic from external dependencies, promoting scalability and maintainability. However, it may introduce additional architectural complexity and require careful planning to ensure seamless integration [8].
- Design thinking: Design thinking fosters innovation by empathizing with user needs, ideating creative solutions, and iterating through prototyping and feedback cycles [9]. Its user-centric approach enhances solution relevance and usability, yet may lead to ambiguity in requirements and feasibility.
- Lean startup: Rooted in rapid experimentation and customer feedback, lean startup accelerates product development and validation. While it fosters agility and market responsiveness, it may prioritize speed over quality and overlook long-term scalability [10].
- Domain-driven design (DDD): DDD focuses on modeling complex business domains to align software architecture with business requirements [11]. By promoting a shared understanding of domain concepts, it enhances system flexibility and maintainability. However, it may require extensive domain expertise and introduce overhead in domain modeling.
- Agile development: Agile methodologies emphasize iterative development, collaboration, and adaptability to changing requirements [12]. They enhance transparency and stakeholder engagement but may pose challenges in managing scope and ensuring comprehensive documentation.
- Service-oriented architecture (SOA): SOA organizes systems as interoperable services, facilitating flexibility and reusability [13]. It promotes modular design and interoperability, yet may introduce complexity in service orchestration and governance.

While these architectures and methodologies offer a range of advantages, they also present unique challenges and limitations that must be carefully addressed. When evaluating and selecting an architecture or methodology for a specific project, it is crucial to consider the business needs and objectives, as well as the project's context and technical requirements. In the realm of complex system design, as mentioned before, the adoption of a singular methodology or approach may not suffice to address all dimensions and challenges that arise. It is within this context that the notion of combining different approaches and methodologies to create more robust and adaptable architectures emerges.

In this regard, the integration of approaches such as the anthropomorphic approach, predictive modeling, and cognitive architecture, along with the hexagonal architecture, can provide a more comprehensive and holistic vision in the design and development of advanced systems.

The advantages of this integration are manifold and significant. Whereas the anthropomorphic approach emphasizes the importance of understanding human needs and expectations in system design, leading to an enhanced user experience and greater acceptance of proposed solutions, predictive modeling offers the ability to forecast future events based on historical data, facilitating informed decision-making and resource optimization across various contexts.

Furthermore, the inclusion of cognitive architecture provides a framework for modeling human cognitive processes, enabling the design of systems that are more intuitive and adaptable to human interaction. Finally, the hexagonal architecture offers a flexible and modular structure for software development, promoting the separation of concerns and facilitating adaptation to technological and business changes.

By combining these approaches, an architecture can be achieved that is not only technically sound but also sensitive to human needs and behaviors, capable of anticipating

future events, and easily adaptable to changes in the environment. This integration offers a holistic approach to complex system design, allowing developers to effectively and efficiently tackle a wide range of challenges.

In the subsequent analysis (Table 1), how each of these approaches can complement and reinforce one another in the creation of advanced system architectures to transition from a flexible model to an intelligent open model is explored in detail.

Table 1. Analysis of the different architecture evolutions, focusing on the anthropomorphic approach, predictive modeling, cognitive architecture, and hexagonal architecture.

Architecture	Origin	Evolution	Current Application
Anthropomorphic Approach	This approach has deep roots in human–computer interaction and cognitive psychology. The aim is to create systems that better understand and adapt to human needs and preferences [4].	Over the decades, this approach has evolved with advances in understanding human psychology, ergonomics, and user-centered design [14].	This approach is used in the design of user interfaces, virtual assistants, artificial intelligence systems, and robotics [15].
Predictive Modeling: ARCH	It has existed in various forms throughout history but has undergone rapid development with the advent of computing and access to large amounts of data [5,6].	With the advancement of statistical and machine learning techniques, predictive modeling has become a powerful tool for forecasting future events in a variety of domains [16].	It is used in a wide range of applications, from weather prediction to product recommendations on e-commerce platforms. It is also fundamental in areas such as personalized medicine and financial risk management [17].
Cognitive Architecture: RCS	Developed in the fields of artificial intelligence and cognitive psychology to understand and replicate human cognitive processes [7,18–21]. An RCS is similar to other cognitive architectures by representing procedural knowledge through production rules and hldeclarative knowledge (abstract data structures: frames, classes, and semantic nets [22]). RCSs originated as a real-time intelligent control system designed to operate real machines interacting with actual objects in the tangible world.	Over the decades, researchers have developed a variety of cognitive architecture models to simulate human mental functions, such as perception, attention, memory, and reasoning. Throughout its developmental stages, all symbols within the RCS world model have been firmly linked to objects and states in the tangible world. The latest iteration, 4D/RCS, incorporates elements of Dickmanns' 4D approach to machine vision [23,24] within the RCS control architecture.	Applied in decision support system training and simulation systems. Initially, it served as a sensory-interactive, goal-directed controller for a laboratory robot [25]. Over time, RCSs have evolved into an intelligent controller applicable to industrial robots, machine tools, intelligent manufacturing systems, automated mail facilities, stamp distribution systems, mining equipment, unmanned underwater vehicles, and unmanned ground vehicles [26,27].
Hexagonal Architecture	Proposed by Alistair Cockburn (early 2000s) to improve the design of complex software systems. The key distinction from others is the business logic separation from technical implementation details, making business logic easier to test, evolve, and change independently of the underlying technology [8].	Since its initial proposal, hexagonal architecture has gained popularity as an effective approach for creating scalable, flexible, maintainable, and easily testable software systems [13].	Currently, hexagonal architecture is widely used in software development across a variety of domains, from web applications to embedded systems. It is especially popular in environments where high flexibility and adaptability to technological and business changes are required [13].

Combining all the mentioned approaches can be an interesting and potentially beneficial challenge, as each brings unique perspectives and can address different aspects of a complex system:

- Multidisciplinary analysis: Conduct a multidisciplinary analysis of the problem you are addressing, considering both the technical and cognitive/human aspects involved. This will help you better understand the context and the system's needs.
- User-centered design: Utilize the anthropomorphic approach to design systems that better adapt to users' needs, expectations, and behaviors. This may include designing intuitive interfaces and considering emotional and cultural factors in human–computer interaction.
- Cognitive modeling: Integrate cognitive models, inspired by cognitive architecture, to understand how users interact with and process information within the system. This can help identify behavioral patterns and optimize system usability.
- Prediction and optimization: Employ predictive modeling techniques to forecast future events and optimize system performance. For example, you can apply predictive analytics techniques to anticipate resource demand and optimize resource planning and allocation accordingly.
- Modular and decoupled architecture: Implement hexagonal architecture to organize the system design into clearly defined and decoupled layers. This will facilitate system modification and maintenance over time while providing flexibility to adapt to technological or requirement changes.
- Validation and feedback: Incorporate continuous feedback loops and validation with real users to iterate and constantly improve the system design. This can help ensure that the system meets users' expectations and needs, as well as identify improvement opportunities.

3. Software Lifecycle in the Remote Handling Environment

Defining a software lifecycle plays a fundamental role in designing an architecture that blends all the aforementioned approaches. Moreover, reliability, availability, maintainability, and safety (RAMS) are eminent requirements of RH systems. The primary software lifecycle phases extracted from ISO-12207 [28] are as follows:

- Software Requirement Analysis Process.
- Software Architectural Design Process.
- Software Detailed Design Process.
- Software Construction Process.
  - Coding.
  - Module testing.
- Software Integration Process.
  - Integration.
  - Integration testing.
- Software Qualification Testing Process.

The development and operation of a system of this complexity require rigorous adherence to safety engineering, quality assurance, validation and verification, requirement management, and requirement traceability processes to ensure reliability and fail-safe operation.

In short, an architecture that integrates multiple approaches must be designed, paying special attention to requirement gathering and the software lifecycle (Table 2). Both processes must be adapted to leverage the strengths of each approach and ensure that the resulting system meets the business needs and end-user's expectations.

Table 2. Requirement gathering and software lifecycle definition from the perspective of the various architecture approaches under study.

Architecture	Requirements	Software lifecycle
Anthropomorphic Approach	In this approach, the needs, expectations, and behaviors of end-users are crucial. Also, it emphasizes the importance of comprehending human characteristics to design systems that adapt to them. During requirement gathering, cognitive and emotional aspects influencing human–computer interaction must be identified.	In defining the software lifecycle, specific stages for user-centered design should be included, where usability testing is conducted and user feedback is gathered for iterative design.
Predictive Modeling: ARCH	This approach starts by analyzing historical data, trends, and behavioral patterns influencing system requirements over time, which are the main actions during requirement identification. Due to this phase, predictive modeling to anticipate future user needs is leveraged.	In this model, the software lifecycle plans and optimizes the development phase. This includes estimating necessary resources, predicting delivery times, and identifying potential obstacles or risks.
Cognitive Architecture: RCS	This approach searches for requirements associated with human cognitive processes: understanding how users interact with the system, which mental processes are involved, and how systems can be designed to better fit these processes.	The software lifecycle includes stages of modeling and validating human cognitive processes. This involves simulating user–system interactions to evaluate the effectiveness of the cognitive architecture in practice.
Hexagonal Architecture	This architecture prioritizes modularity and flexibility as its essential requirements. Identifying functional and non-functional requirements that determine the modular structure of the system is crucial for designing a scalable and adaptable architecture.	Defining the software lifecycle incorporates agile development practices that leverage the modularity and flexibility of the hexagonal architecture. This includes frequent iterations, unit testing, and continuous refactoring to maintain code quality and consistency.

4. MAMIC System: A New Model Generation

A reference model architecture delineates the functionalities, entities, events, relationships, and information flow occurring within and across functional modules. It furnishes a framework for delineating functional requirements, designing software to fulfill those requirements, and testing components and systems. By combining the anthropomorphic approach, predictive modeling, and cognitive architecture with the hexagonal architecture, more comprehensive and effective systems can be created that address both technical and cognitive/human aspects holistically.

It is important to note that the application of each approach should be tailored to the project's specific context and requirements, and the appropriate blend will depend on the domain characteristics and user needs.

Through the combination of these architecture approaches, a novel model emerges named the Multiple Agent Managerial Intelligent Control (MAMIC) system. The MAMIC system requires careful consideration of each component's functionality and interactions. In Table 3, some special aspects are detailed to understand how the anthropomorphic approach, predictive modeling, and cognitive architecture could be integrated into the hexagonal architecture.

Table 3. Component's functionalities of anthropomorphic approach, predictive modeling, and cognitive architecture used to combine them into the hexagonal architecture.

Architecture	Special Consideration
Anthropomorphic Approach	1. Design user interfaces and interactions that are intuitive and adaptive to human behavior. 2. Gather user feedback through surveys, interviews, or usability testing to understand user preferences and needs. 3. Implement features such as personalized recommendations or natural language processing to enhance user experience.
Predictive Modeling: ARCH	1. Collect and analyze historical data on user behavior, system performance, and external factors. 2. Develop predictive models to anticipate future user actions or system events. 3. Integrate predictive analytics into the system to dynamically adjust behavior or resources based on anticipated events.
Cognitive Architecture: RCS	1. Model cognitive processes such as perception, learning, and decision-making within the system. 2. Implement algorithms inspired by cognitive science to simulate human-like behavior or reasoning. 3. Design interfaces and interactions that align with human cognitive abilities and limitations.

Integrating these approaches within a hexagonal architecture involves structuring the system into layers with a clear separation of concerns:

- Domain layer: The domain is the innermost layer. It encompasses all core business logic and functionality of the system and remains agnostic of the other layers. It should only vary based on the needs of the business, uninfluenced by external factors. In this layer, the data model is located or "Read model is housed", and it is crucial to consider modeling it based on the requirements rather than the database model. Furthermore, within this layer, business validations are located in "Value Objects" and "Entities", as well as the ports of the MAMIC system (adapter interfaces and domain services).
- Application layer: The application layer contains the use cases needed for each "Bounded context" or unit of the MAMIC system. The role of application classes is to orchestrate the necessary classes to execute an action requested by the external systems. Additionally, it handles the translation of primitive data received into objects managed by the domain. The application layer should never be aware of infrastructure details.
- Infrastructure layer: The infrastructure layer is responsible for communication between the MAMIC system and external factors, and vice versa. Its classes are tightly coupled to specific technologies such as frameworks, database engines, specific queuing systems, external libraries, and other infrastructure components. Within this layer, some modules translate information received/sent by other components and entry points to the MAMIC system, among other functionalities.

The MAMIC model, Figure 1, facilitates the integration of these approaches by providing a modular and flexible structure. Each model can be encapsulated within its respective layer, allowing for independent development, testing, and evolution. Additionally, the architecture promotes the separation of concerns, making it easier to maintain and extend the system over time:

- Anthropomorphic approach: Implemented primarily in the application layer, where user interfaces and interaction components reside. User-centric design principles guide the development of interfaces and interactions.

- Predictive modeling: Integrated into both the domain and application layers. Predictive models are part of the business logic in the domain layer, while data processing and analysis components reside in the application layer.
- Cognitive architecture: Implemented within the domain layer, where cognitive processes are modeled and simulated. Algorithms inspired by cognitive science are embedded into the business logic to enable human-like behavior.

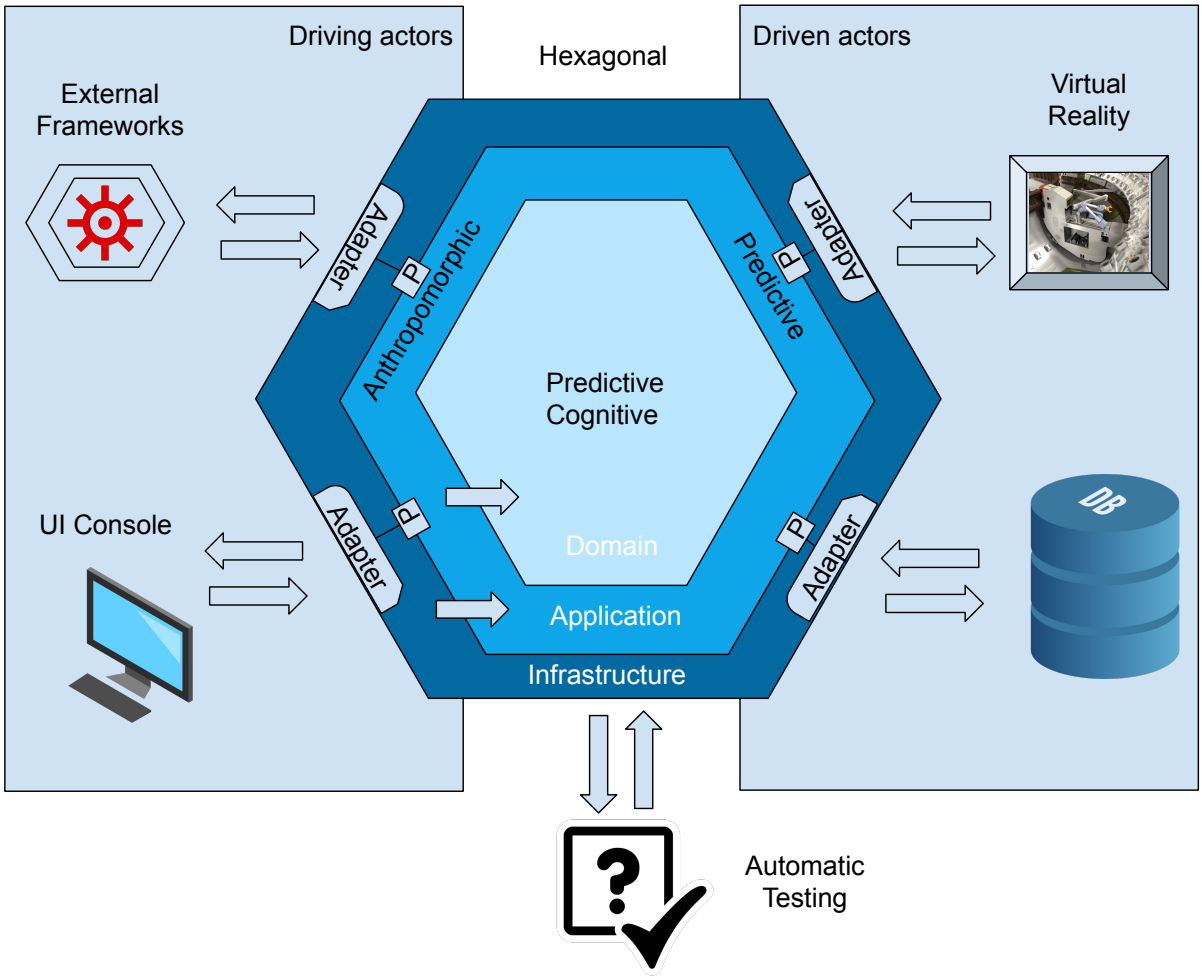

Figure 1. MAMIC model generation and integration.

In summary, while the MAMIC model proposal offers valuable insights and capabilities, it presents specific challenges related to complexity, resource utilization, technological constraints, and ethical implications.

- Initial complexity: Deploying this methodology could necessitate increased design and configuration efforts owing to its intricate structural nature.
- Resource overhead: The modular nature of this architecture may result in elevated layers and abstractions, which could pose maintenance challenges and overhead.
- Technological limitations: The MAMIC model could be limited by existing technological capabilities, impacting its ability to faithfully replicate human behavior or achieve precise predictive outcomes.
- Computational cost: Sophisticated predictive models may require substantial computational resources, impacting scalability and practical deployment capabilities.
- Ethical considerations: Applying MAMIC principles raises ethical considerations concerning human perception and technology-related expectations.

5. Application to a Specific Case Study: ITER Divertor Remote Handling System

5.1. ITER Environment

The ITER remote handling (RH) [29] system will be operated from a dedicated control room, approx. 1km away from the reactor and designed for full remote operation. The intention is to manage all the necessary equipment for specific RH tasks from a single operations bench known as the RH standard work cell. Given that numerous RH operations will be executed concurrently, the plan is to incorporate multiple work cells within the RH control room.

The ITER RH control system comprises various modules and technologies, necessitating sustained operational functionality throughout the ITER project's estimated 30-year lifespan, encompassing assembly and decommissioning.

To streamline initial integration and ongoing maintenance, a modular approach is advocated. ITER mandates a fixed modular architecture for all systems, emphasizing a straightforward principle to foster interoperability and maintainability: each control system element should meet its requirements without imposing specific solutions on other elements. For the implementation of the RH standard work cell, this modular approach was realized. Figure 2 illustrates the different modules that configure a work cell.

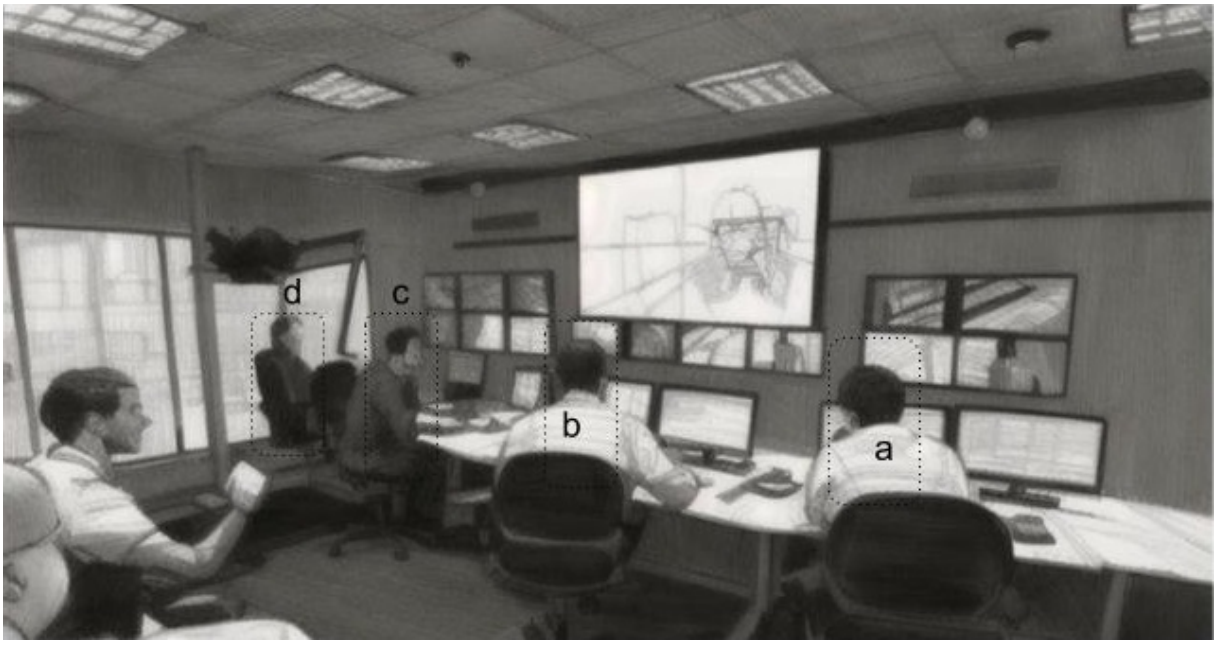

Figure 2. The four-person work cell tasks. (**a**) The responsible officer oversees operations. (**b**) The deputy manages cameras and support tools. (**c**) The "mover" operates casks, transporters, and cranes. (**d**) The "manipulator" controls the master arm and other manipulative devices. Reprinted with permission from Ref. [30]. 2024, D. Hamilton.

The ITER remote maintenance system (IRMS) [31] is required to provide full remote handling capability (including rescue) inside the in-vessel area. The IRMS remote handling equipment is the divertor remote handling system (RHS) (DTP2), which is illustrated in Figure 3:

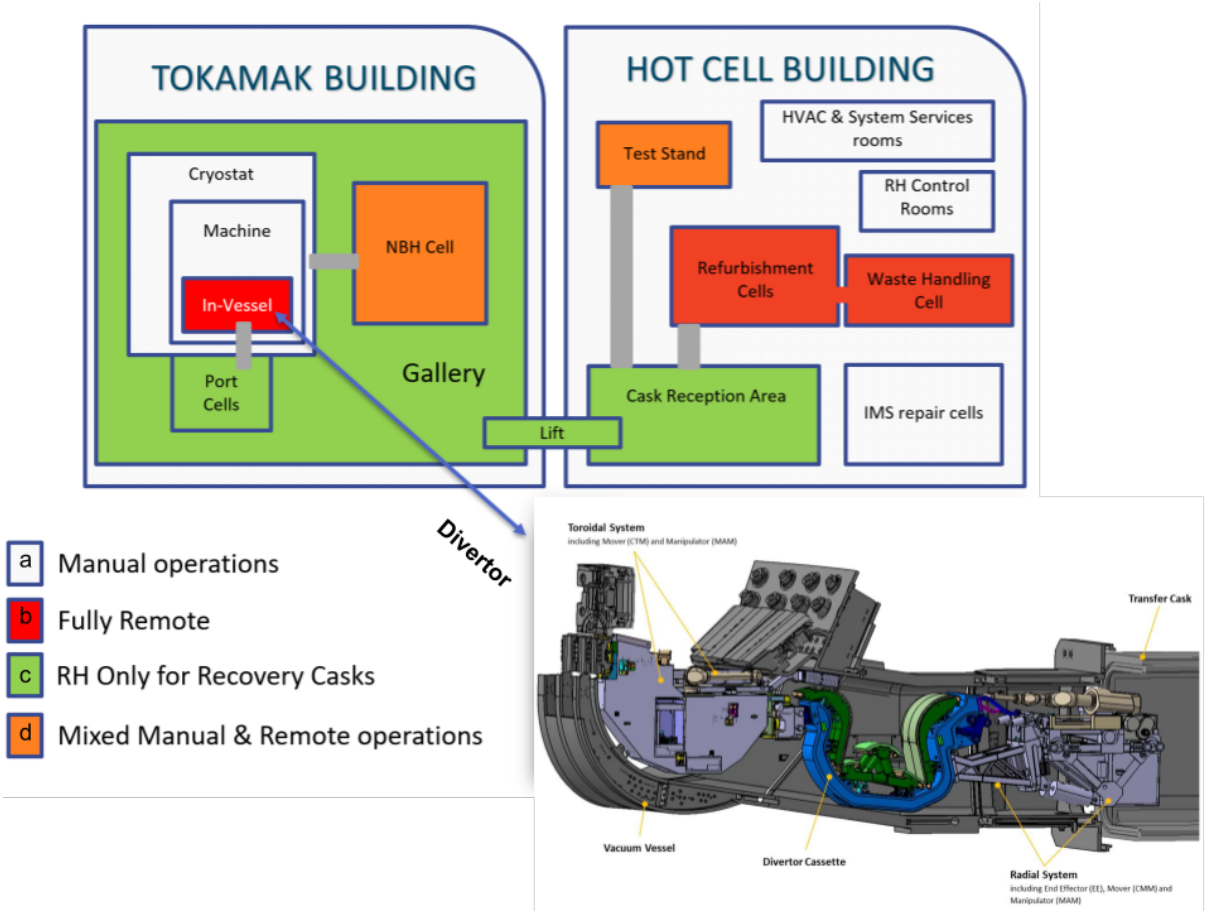

Figure 3. The ITER Maintenance System serves various areas within the ITER facility. (**a**) Full capability for its manual maintenance within the hot cell, NB cell, and test stand. (**b**) Full remote handling capability, including rescue operations, within the in-vessel, hot cell, and NB cell. (**c**) Complete remote handling capabilities for the recovery of casks within the lift, gallery, and hot cell. (**d**) Remote handling capabilities that can be seamlessly combined with local manual support within the NB cell, hot cell (part), port cell, and test stand area.

The divertor RHS (toroidal system) is required to perform the following key functions:

- Insertion/extraction of divertor cassettes and their transportation to/from a transfer cask docked at divertor-level RH ports.
- Insertion/extraction of divertor-level diagnostic assemblies and their transportation to/from a transfer cask docked at divertor-level RH ports.
- Removal/replacement of the divertor-level RH port primary closure plate (PCP).
- Dust removal within and around the divertor region during the cassette-removal process.

A work cell to control the divertor RHS will be established based on the latest ITER specifications for the control room and work cell design, incorporating the remote handling and control room software developed for F4E/ITER:

1. General-purpose controller (GENROBOT): GENROBOT is a software controller application built to command particular RH systems used across the ITER remote handling environment and is integrated to command RH systems via the RH-IHMI application.
2. Remote robot handling intelligent human–machine interface (RH-IHMI): The RH-IHMI software is a single application that provides the operator interface to control and command the different ITER remote handling systems (RHSs) via a customizable graphical user interface. The RH-IHMI software application also allows one to receive information about all the steps present in the presently active method and any step containing an SL instruction designed for RH-IHMI application.

3. Virtual reality (VR): The VR software GUIs at ITER RH possess the capability to receive external commands via an interface port. Certain commands trigger script files, referred to as VR internal commands, which enable adjustments to the VR configuration and display. These script files are pre-programmed within the VR models and are also stored within them.
4. Input devices: Comprising a joystick and haptic arms. The area where the haptic arms are usually located will be used for exoskeleton/VR and new technologies.
5. Operation management systems (OMSs): The OMS software tools are a component of the RH software suite within the high-level control system (HLCS). It establishes communication with various other components of the HLCS, primarily situated within or near the remote handling control room (RHCR). The primary goal of the OMS involves the planning and execution of task procedures. These procedures involve the entire sequence of manual actions necessary to carry out tasks; in the end, the OMS tool essentially drives the remote handling process.
6. The RH database (RHDB): This constitutes an integral component of the OMS, along with the OMS task builder and the OMS task executor. It encompasses a comprehensive range of RH task information, covering the following:
    - Super-tasks, sub-tasks, methods, and steps.
    - Instances of RH tools and RH equipment (cross-referenced within super-tasks, sub-tasks, methods, and steps).
    - Associations with external multimedia files (referenced within super-tasks, sub-tasks, methods, and steps).
    - Records of task logs (inclusive of annotations and links generated during task execution).
    - Profiles of RH users and OMS login credentials, with varying permission tiers.
7. Mixed reality: Designs an interactive experience in which the sensor readings are combined with HMI-generated content. The content can span multiple sensory modalities, including visual, auditory, haptic, somatosensory, and olfactory. Moreover, other techniques, such as augmented reality, are designed as a system that incorporates three basic features: a combination of real and virtual worlds, real-time interaction, and accurate 3D registration of virtual and real objects.
8. Data analysis with artificial intelligence: Incorporating user perception and operator fatigue into the application model operation involves integrating computer vision techniques to effectively direct the operator's attention to the ongoing tasks. Additionally, to enhance configurability and usability for the operator, artificial intelligence (AI) is integrated into the general application. This requires further investigation into determining optimal operator inputs to train the AI model, including factors such as work hours, RH device activity, task development duration, and more. All these functionalities will be performed under the operator's supervision, allowing manual intervention at any time to restore the system to a safe state.

5.2. ITER Requirements

The essential requirements of ITER organization [32] for the RH-IHMI software in a critical environment are outlined in Appendix A.

In this table, it is remarkable that providing a user-friendly interface supporting real-time communications with RH systems is necessary. Additionally, the interface must be efficient in monitoring and visualizing large datasets and incorporate robust security measures to prevent unauthorized access. It should also be adaptable to various screen sizes and resolutions and comply with industry standards for reliability and safety. Furthermore, comprehensive logging and auditing features, along with multi-language localization support, are essential. Customizable dashboards and reporting tools, modular architecture for easy maintenance, built-in error handling, and support for integration with other systems complete the list of requirements for this critical software system.

Ultimately, the architecture design process will be guided by hazard and risk assessments, with a recommendation to adopt the ISO-10218 [33] standard for industrial robots to comply with the European Machinery Directive. Furthermore, the ITER nuclear safety control tier is utilized for I&C related to nuclear safety.

ISO-10218-1:2011 §5.4.2 [33]: Safety-related parts of control systems shall be designed so that they comply with PL=d with structure category 3, as described in ISO-13849-1:2006 [34], or so that they comply with SIL 2 with a hardware fault tolerance of 1 with a proof test interval of not less than 20 years, as described in IEC 62061:2005 [35].

5.3. ITER Software Lifecycle Definition

Due to the complexity of the proposed requirements (Annex 1), interaction with numerous external agents, modular integration across different environments, and maintainability over long periods, an iterative software development cycle was chosen. It offers numerous benefits, including increased flexibility, risk mitigation, and stakeholder satisfaction, making it a preferred approach for many software development projects. Integrating an iterative software development cycle within a MAMIC architecture involves structuring the development process to align with the principles of iterative development while leveraging the modular and flexible nature of the MAMIC architecture.

Several steps are necessary to be carried out for the culmination of an iterative software cycle, as has been followed in the case of ITER:

1. Incremental development: Breaks down the software development process into smaller increments or iterations, each focused on delivering specific functionality or features. This aligns with the iterative approach, allowing for continuous feedback and improvement.
2. Feedback loops: Establishes feedback loops at the end of each iteration to gather input from stakeholders, users, and team members. This feedback informs the next iteration, guiding adjustments and refinements to the system.
3. Modular design: Utilizes the modularity of the MAMIC architecture to encapsulate different components and functionalities within distinct modules. Each module represents a self-contained unit with well-defined interfaces, facilitating incremental development and testing.
4. Separation of concerns: Maintains a clear separation of concerns within the MAMIC architecture, with distinct layers for domain logic, application logic, and infrastructure. This separation ensures that changes and updates can be made to specific components without affecting the entire system.
5. Continuous integration and verification and deployment: Implements automated testing and continuous integration and verification practices to ensure that changes introduced during each iteration are seamlessly integrated into the codebase. This allows for rapid feedback and the early identification of potential issues.
6. Adaptability and flexibility: Embraces the flexibility of the MAMIC architecture to accommodate changes and updates throughout the development process. The modular design and separation of concerns enable developers to respond quickly to evolving requirements and priorities.
7. Iterative refinement: It continuously refines and improves the system with each iteration based on feedback and lessons learned. This iterative refinement process ensures that the software evolves incrementally, gradually meeting the desired objectives and quality standards.

Taking into account a classic software lifecycle and the considerations carried out in the MAMIC architecture, the following iterative development methodology (IDM) is presented in Figure 4.

Figure 4. Software development, verification, and validation lifecycle.

5.4. ITER MAMIC Model Definition

By integrating an iterative software development cycle within a MAMIC architecture, teams can effectively manage complexity, respond to change, and deliver high-quality software that meets the evolving needs of stakeholders and users. Focusing on the ITER environment, the MAMIC architecture presents an approach for building software systems with a centralized core that is separate from its interactions with external interfaces. This is realized through the utilization of ports and adapters, as depicted in Figure 5.

The driving agents (primary) are those that initiate the interaction: the external input controller transfers data to the application through a port, the RH task operations in structure language process it in an automatic or semi-automatic way, and third-party frameworks send the data through the adapter to carry out some actions. The driven agents (secondary) are those that respond to the application's request: a database adapter is called upon by the application to retrieve the data, an RH controller device is commanded by the application layer and also publishes status information through the port, and mixed reality is presented to visualize the in-vessel critical environment.

The domain layer is divided into several bounded contexts. These are a self-contained and delimited space where a specific set of business problems can be solved in isolation from the rest and without affecting anything beyond this context. Each context has its own set of concepts, terms, and specific domain models and may have its own technical implementation. The different bounded contexts depicted in the figure, which comprise the domain layer, are as follows:

- Configuration domain: The MAMIC architecture provides a flexible configuration mechanism that allows customization of the design and the internal logic behavior of

the RH-IHMI. This customization is achieved through configuration parameters that are stored in the configuration files and loaded during start-up time.

- Error handling domain: The error management process defines the software behavior when an error occurs during its execution. The basic steps in an error management process are detection, identification, handling, mitigation, and resolution or recovery. Moreover, this domain is in charge of gathering controller event messages from RH devices and VR, AR, RHDB, and OMS actions.
- Task prediction domain/application: This domain is based on predictive modeling, in which artificial intelligence can be included in the RH-IHMI to be more configurable, considering its usability by the operator. Therefore, it is necessary to incorporate the best operator inputs to train the artificial intelligence model, such as work hours, when the RH device is working, how much time has been spent on the development of the task, and so on. This domain allows the driving of processes in an automatic or semi-automatic way. This domain operates under continuous operator supervision, enabling manual intervention to return the system to a safe state at any time.
- World interaction domain: An interactive experience in which the sensor readings are combined with HMI-generated content. The content can span multiple sensory modalities, including visual, auditory, haptic, somatosensory, and olfactory. Moreover, other techniques, such as augmented reality, are designed as a system that incorporates three basic features: a combination of real and virtual worlds, real-time interaction, and accurate 3D registration of virtual and real objects.
- Assistant operation domain: In this domain, a cognitive architecture is applied, dividing the bounded context in different node behavior generation; world modeling, which interacts with the world interaction domain; sensory processing, adding user perception and operator fatigue state; and a value judgment process together with a knowledge database. In each node, there exists both a deliberative and a reactive component. At the lower level, each node completes a reactive control loop driven by feedback from sensors. At the higher level, each node formulates and executes plans intended to fulfill task objectives, priorities, and limitations communicated by commands from above. Within each node, deliberative plans are integrated with reactive behaviors.
- Communication domain: Different communication protocols need to be configured to establish efficient data handling with the RH devices, RH database, augmented reality, virtual reality, I/O input devices, third-party frameworks, CODAC workstations, and OMS applications.

The ITER MAMIC model proposal aims to overcome the deficiencies or constraints of prior architectural models (anthropomorphic, predictive, cognitive, and hexagonal) by implementing the following strategies:

- Adopting a clear, well-defined separation of concerns (bounded context) so the initial complexity can be managed more effectively.
- Minimizing dependencies between modules and optimizing resource allocation, which can mitigate the risk of increased layers and abstractions.
- Designing elements to gracefully degrade or adapt based on available capabilities, allowing for progressive enhancement as technology evolves.
- Employing a scalable MAMIC model that can dynamically allocate resources based on workload can enhance scalability and reduce costs.
- Implementing mechanisms for facilitating user understanding and control over technology interactions.

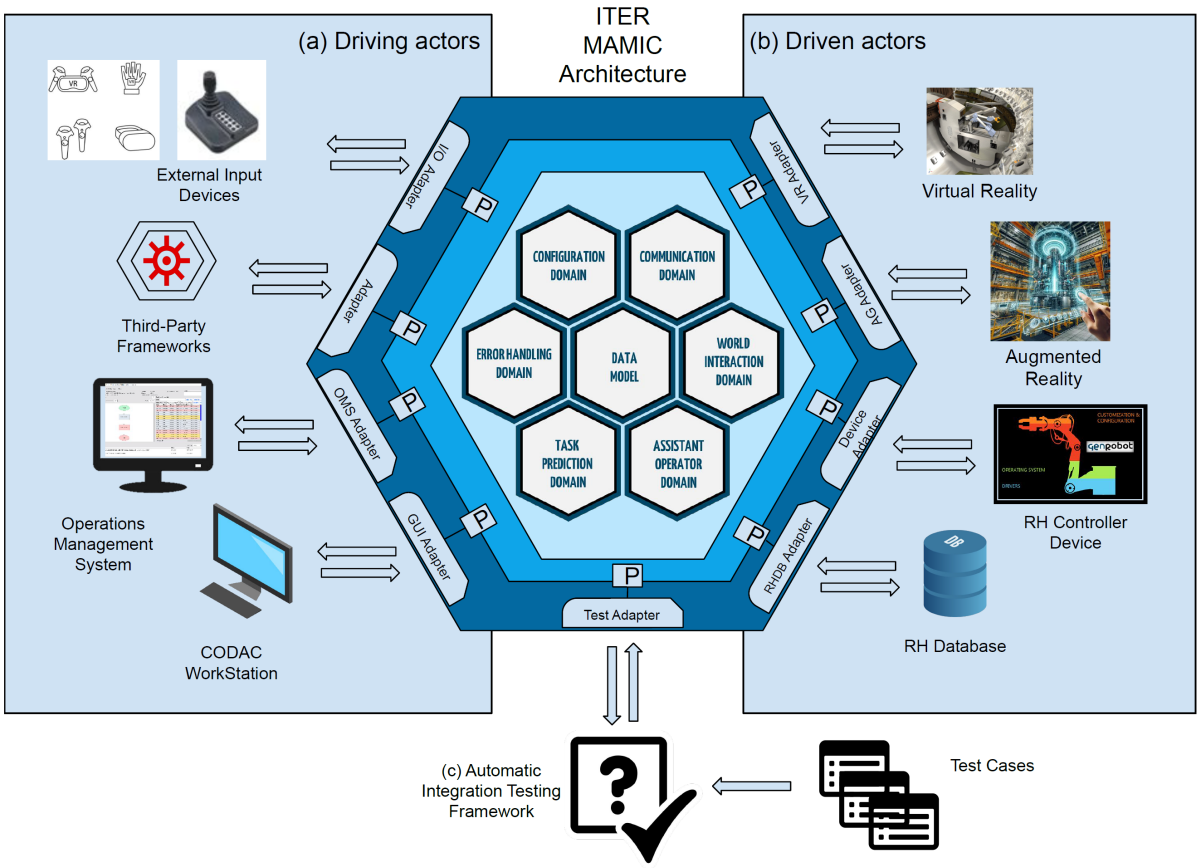

Figure 5. ITER MAMIC architecture generation integrating (**a**) driving actors, such as external input devices, third-party frameworks, OMS, and CODAC workstations, (**b**) driven actors, such as VR, AR, RHDB, and RH robots, and (**c**) an easy automatic integration testing framework into the different bounded context using the port–adapter pair.

6. Architecture Model Validation

In today's rapidly evolving technological landscape, ensuring the quality and reliability of any software system is crucial. One crucial aspect of this process is requirement validation, which represents the concluding phase of requirement engineering. The objective of this stage is to review the final draft of a requirement document to confirm that it accurately represents an acceptable description of the system to be implemented. Inputs to the validation process include the requirement specification, organizational standards, and implicit organizational knowledge. Outputs consist of a list of identified requirement issues and agreed-upon actions to resolve these issues. Applying this rigorous validation methodology, future developments can mitigate risks, enhance user satisfaction, and deliver iterative versions that meet or exceed expectations.

In this section, the validation process of the MAMIC architecture is conducted against a requirement matrix provided in Appendix A and consists of tracing the requirements established by ITER to address the scenarios defined in the RHS divertor against the software lifecycle and various layers that make up the reference model architecture, shown in Table 4. A detailed model validation analysis is included in the last column of this matrix.

This systematic approach verifies that each requirement is sufficiently addressed and implemented within the software solution, adhering to metrics outlined in ISO-9001 and CODAC SWIL-2 standards [36].

Table 4. Validation traceability used to link ITER requirements to the MAMIC architecture components.

Requirement Level	Validation Result
Applicable policies, processes, and requirements	1. The IDM ensures that the software planning activities produce the software plans and standards that direct MAMIC architecture development processes and others, such as configuration management and quality. 2. The IDM is developed following supplier QA processes to have good reliability and avoid delays in RH operations. 3. Risk management is performed systematically during the development to allow the reduction and control of the identified risks.
General requirements: application initialization and file management	1. The MAMIC architecture provides a flexible configuration mechanism that allows customization of the design and the internal logic behavior of the RH-IHMI. 2. The error management bounded context in the domain layer defines the software behavior when an error occurs during its execution. 3. The GUI adapter serves as an intermediary component responsible for facilitating communication between the core application logic and the CODAC workstation dependencies.
HMI design requirements	1. The application layer ensures that any action requested by the external systems is processed correctly and any primitive data received are translated into specific objects managed by the domain layer. 2. The error management context specifies the software's response and behavior in the event of an error occurring during its execution. 3. The GUI adapter functions as an intermediary module tasked with facilitating communication between the core application logic and the dependencies of the CODAC workstation.
Interfaces	1. Assistant operation domain ensures the execution of plans intended to fulfill task objectives, priorities, and limitations. 2. Communication domain establishes efficient data handling with the different external applications using specific protocols, such as HLC and RAPI. 3. Task prediction context allows the RH-IHMI to be more configurable, considering its usability by the operator.

In conclusion, the MAMIC model covers all the requirements established, with the advantage that it is flexible enough to introduce changes to its structure within short timeframes and with moderate effort.

7. Conclusions and Future Works

The MAMIC model proposal, in which the hexagonal architecture is integrated with an anthropomorphic approach, predictive modeling, and cognitive architecture, provides a robust strategy for developing flexible, maintainable, and testable software. Through the implementation of ports and adapters, a clear separation of concerns is achieved, enhancing the resilience of the RH-IHMI to technological shifts and enabling adaptation to evolving business needs. With the MAMIC architecture, the RH-IHMI is engineered to endure the passage of time, safeguarding the core business logic at its core and demonstrating resilience against future changes in technology and delivery mechanisms.

Moreover, using the MAMIC model, agents can operate autonomously until human supervision is required or a problem is detected by the supervisor. The MAMIC design strategy aims to create a versatile system applicable to different RH device controllers. We leverage the benefits of autonomous systems while allowing human control through a human–machine interface. The MAMIC architecture equips the supervisor with tools to interact with all processing levels of the RH device controllers. These interactions can rectify unexpected moves or address decisions that might lead an autonomous system into an unstable state. One of the MAMIC goals is to specify task assignments, teleoperate

agents, display sensory data, override process conclusions, and reconfigure the system in response to various sensory and agent failures.

Additionally, having a model architecture greatly aids industrialization by providing a common model and language for remote handling (RH) across the industry. This facilitates the integration of multiple systems and subsystems.

Due to the model generation having been validated with the ITER requirements, the MAMIC architecture is perfectly valid to cover the current and future needs of ITER facilities. Furthermore, considering the remote handling needs of future fusion facilities, such as IMIF-DONES [37–39] and DEMO [40–42], an update of the model could be made by including new bounded contexts in the domain layer and new ports and adapters in the application and infrastructure layers.

Author Contributions: Conceptualization, T.B.; methodology, T.B.; validation, T.B.; formal analysis, T.B.; investigation, T.B.; resources, T.B.; data curation, T.B.; writing—original draft preparation, T.B.; writing—review and editing, A.B.; visualization, T.B.; supervision, A.B.; project administration, A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: We extend our sincere gratitude to the reviewers Juan Ramón Acarreta Rodriguez and José Acebrón Antón for their invaluable time, expertise, and constructive feedback, which significantly contributed to the enhancement and refinement of this paper. Their meticulous review and insightful suggestions undoubtedly elevated the quality and clarity of our work.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial intelligence
API	Application programming interface
AR	Augmented reality
CDN	Control and diagnostic network
CIP	Controller interface protocol
CODAC	Control, data acquisition, and communication
GENROBOT	General-purpose controller
HLC	High-level CIP
HLCS	High-level control system
HMI	Human–machine interface
IØ	Input/output devices
I&C	Instrumentation and control
IDM	Iterative development methodology
ITER	International Thermonuclear Experimental Reactor
IRMS	ITER remote maintenance system
ISO	International Organization for Standardization
MAMIC	Multiple Agent Managerial Intelligent Control
OMS	Operation management system
PCP	Port primary closure plate
RAPI	Remote API
RCS	Real-time control system
RH	Remote handling
RHDB	Remote handling database
RH-IHMI	Remote handling intelligent human–machine interface

RHS	Remote handling systems
SW	Software
SWIL	Software integrity level (CODAC)
UI	User interface
VR	Virtual reality

Appendix A. Remote Robot Handling Intelligent Human–Machine Interface Architecture Requirements

This appendix shows a summary of the essential requirements of the ITER organization necessary to support the ITER remote maintenance system in the divertor RHS.

This summary of requirements has been compiled after extensive collaborative efforts with the Fusion for Energy organization, encompassing all the specified needs outlined in references [32,43,44].

Table A1. Applicable policies, processes, and requirements.

Identifier	Description	Model Validation
IHMI-REQ-001	The IHMI system shall adhere to the requirements outlined in the RH Software Quality Policy [45] for Software Category 1 [ISO-9001 and QA based on CODAC SWIL-2].	Covered by the IDM.
IHMI-REQ-002	The model shall be responsible for conducting software module evaluations for the RH-IHMI system, utilizing the specified form outlined in the RH Software Quality Evaluation Procedure [43].	Covered by the IDM.

Table A2. General requirements: application initialization file management.

Identifier	Description	Model Validation
IHMI-REQ-003	The RH-IHMI system shall utilize a user rights file stored within a specific server.	Configuration domain, GUI adapter
IHMI-REQ-004	The RH-IHMI system shall establish operator roles by referencing the user log-in name specified in the user rights file.	Configuration domain, error handling domain, GUI adapter
IHMI-REQ-005	The RH-IHMI system shall be tasked with gathering event messages from external agents and storing them within a session log file.	Error handling domain, GUI adapter

Table A3. HMI design requirements.

Identifier	Description	Model Validation
IHMI-REQ-006	The HMI Style Guide HMI Style Guide and Toolkit [45] shall be adhered to as a set of guidelines for developing the RH-IHMI, ensuring a consistent IO style that includes the use of colors, text, symbols, and widgets.	Application layer
IHMI-REQ-007	The RH-IHMI shall offer different areas [46] to command and visualize data from the I/O devices, robot controller, VR module, IA module, RHDB module, and OMS module	Communication domain, error handling domain, data model, task prediction domain, assistant operator domain, application layer, device adapter, GUI adapter, I/O adapter, RHDB adapter, VR adapter
IHMI-REQ-008	The RH-IHMI system shall utilize the controller's EDD configuration file to ascertain the applicability of commands to the current operation mode.	Configuration domain, data model, application layer, GUI adapter
IHMI-REQ-009	The RH-IHMI shall offer functionality for the automatic execution of RHSL instructions [47].	Communication domain, error handling domain, data model, task prediction domain, application layer, device adapter, GUI adapter, OMS adapter

Table A4. Interfaces.

Identifier	Description	Model Validation
IHMI-REQ-010	The RH-IHMI system shall establish a connection to the gigabit Ethernet file network using a dedicated network interface card (NIC).	Communication domain, error handling domain, OMS adapter
IHMI-REQ-011	The RH-IHMI system shall utilize the file network [48] for various purposes, including general communications and data transfers, such as file transfers and loading applications and data from servers, accessing the OMSDB, communications with the OMS using the low-level communication (LLC) protocol, communications with the VR using the RAPI protocol, communications with the IA module.	Communication domain, error handling domain, OMS adapter
IHMI-REQ-012	The RH-IHMI system shall utilize the high-level Common Industrial Protocol (CIP) library, which incorporates the GENROBOT Remote Interface library. This communication library is specifically designed for interacting with GENROBOT-type controllers over the RH control network.	Communication domain, error handling domain, device adapter
IHMI-REQ-013	The RH-IHMI system shall subscribe to published events from controllers in the list of selected controllers.	Communication domain, error handling domain, device adapter
IHMI-REQ-014	The RH-IHMI application shall interface with the PostgreSQL database (OMSDB) through the RH file network to retrieve RHSL parameters. This procedure will utilize an encrypted password to ensure that it is executed by a user with appropriate privileges.	Communication domain, error handling domain, RHDB adapter, OMS adapter.
IHMI-REQ-015	The RH-IHMI system shall subscribe to published events from controllers in the list of selected controllers.	Communication domain, error handling domain, device adapter
IHMI-REQ-016	The RH-IHMI application shall interface with the virtual reality (VR) system to transmit external commands that make use of a script file name, thereby triggering actions such as altering the displayed model configuration.	Communication domain, error handling domain, world interaction domain, VR adapter.
IHMI-REQ-017	The RH-IHMI application shall interface with the data acquisition and IA processing module to process the sensor readings, which are combined with HMI-generated content.	Communication domain, data model, task prediction domain, assistant operator domain, OMS adapter, I/O adapter, RHDB adapter.
IHMI-REQ-018	The RH-IHMI application shall interface with the data acquisition and IA processing module to incorporate user perception and operator fatigue into the application model operation, which involves integrating computer vision techniques.	Communication domain, data model, assistant operator domain, OMS adapter, I/O adapter, RHDB adapter.
IHMI-REQ-019	The RH-IHMI application shall interface with the data acquisition and IA processing module to enhance configurability and usability for the operator.	Communication domain data model, task prediction domain, OMS adapter, I/O adapter, RHDB adapter.

References

1. Jaquero, V.; Montero, F.; Molina, J.P.; González, P. Intelligent User Interfaces: Past, Present and Future. In *Engineering the User Interface*; Redondo, M., Bravo, C., Ortega, M., Eds.; Springer: London, UK, 2009; pp. 1–12.
2. ITER Organization. *ITER Research Plan within the Staged Approach. ITR-18-003 Level III—Provisional Version*; ITER Organization: Saint-Paul-lez-Durance, France , 2018 .
3. Stroth, U.; Aguiam, D.; Alessi, E.; Angioni, C.; Arden, N.; Arredondo Parra, R.; Artigues, V.; Asunta, O.; Balden, M.; Bandaru, V. Progress from ASDEX Upgrade experiments in preparing the physics basis of ITER operation and DEMO scenario development. *Nucl. Fusion* **2022**, *62*, 042006. [CrossRef]

4. Zdravković, M.; Luis-Ferreira, F.; Jardim-Goncalves, R.; Trajanović, M. On the formal definition of the systems' interoperability capability: An anthropomorphic approach. *Enterp. Inf. Syst.* **2017**, *11*, 389–413. [CrossRef]
5. Kuhn, M.; Johnson, K. *Applied Predictive Modeling*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 19–26.
6. Abbott, D. *Applied Predictive Analytics: Principles and Techniques for the Professional Data Analyst*, 1st ed.; Wiley: Hoboken, NJ, USA, 2014; pp. 1–82.
7. Archer, R.; Lebiere, C.; Warwick, W.; Schunk, D. Integration of Task Network and Cognitive Models to Support System Design. In Proceedings of the Collaborative Technology Alliances Conference 2003 Advanced Decision Architectures, College Park, MD, USA, 29 April–1 May 2003.
8. Cockburn, A.; Becker, A.P. *Crystal Clear: A Human-Powered Methodology for Small Teams: A Human-Powered Methodology for Small Teams*, 1st ed.; Addison-Wesley Professional: Boston, MA, USA , 2004.
9. Highsmith, J.; Descalzo Mascarós, A. *Application of Design Thinking Principles in UI/UX Design of Software Development*; TFGM, Universitat Politècnica de València: Valencia, Spain, 2022.
10. Ries, E. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*, 1st ed.; Crown Currency: New York, NY, USA, 2011.
11. Kapferer, S.; Zimmermann, O. Domain-Driven Service Design–Context Modeling, Model Refactoring, and Contract Generation. In Proceedings of the 4th Symposium and Summer School on Service-Oriented Computing, Crete, Greece, 13–19 September 2020.
12. Highsmith, J.; Cockburn, A. Agile software development: The business of innovation. *Computer* **2001**, *34*, 120–127. [CrossRef]
13. Khan, S.M . *Popular Software Architecture Used in Software Development*, 1st ed.; Kindle Publisher: Seattle, WA, USA, 2023.
14. Bettini, C.; Brdiczka, O.; Henricksen, K.; Indulska, J.; Nicklas, D.; Ranganathan, A.; Riboni, D. A survey of context modeling and reasoning techniques. *Permis. Mob. Comput.* **2010**, *6*, 161–180. [CrossRef]
15. Panetto, H.; Zdravkovic, M.; Jardim-Goncalves, R.; Romero, D.; Cecil, J.; Mezgár, I. New Perspectives for the Future Interoperable Enterprise Systems. *Comput. Ind.* **2016**, *79*, 47–63. [CrossRef]
16. Avouris, N.M. Intelligent interface design. *Hanbook Hum. Comput. Interact.* **2022**, *56*, 1–72.
17. Ding, Z.; Ji, Y.; Gan, Y.; Wang, Y.; Xia, Y. Current status and trends of technology, methods, and applications of Human–Computer Intelligent Interaction (HCII): A bibliometric research. *Multimed. Tools Appl.* **2024** . [CrossRef]
18. Langley, P.; Laird, J.E.; Rogers, S. Cognitive Architectures: Research Issues and Challenges, *Cogn. Syst. Res.* **2009**, *10*, 141–160. [CrossRef]
19. Newell, A.; Simon, H. *Human Problem Solving*, 1st ed.; Echo Point Books and Media: Brattleboro, VT, USA , 2019; Chapters 3 and 4.
20. Laird, J. *The Soar Cognitive Architecture*, 1st ed.; MIT Press: Cambridge, MA, USA, 2019; Chapters 1, 2 and 3.
21. Shavlik, J.W. *Extending Explanation-Based Learning by Generalizing the Structure of Explanations*, 1st ed.; Elsevier Science: Amsterdam, The Netherlands, 2014; Chapters 1 and 4.
22. Albus, J. *Brains, Behavior, and Robotics*; BYTE/McGraw Hill: Peterborough, NH, USA, 1983.
23. Albus, J. 4-D/RCS reference model architecture for unmanned ground vehicles. In Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Volume 4, pp. 3260–3265.
24. Kuvich, G. Integration of image/video understanding engine into 4D/RCS architecture for intelligent perception-based behavior of robots in real-world environments. In Proceedings of the SPIE—The International Society for Optical Engineering, Philadelphia, PA, USA, 25–28 October 2004 ; Volume 5608.
25. Barbera, A.J.; Albus, J.S.; Fitzgerald, M.L. Hierarchical control of robots using 10 microcomputers. In Proceedings of the 9th International Symposium on Industrial Robots, Washington, DC, USA, 13–15 March 1979.
26. Chadli, M.; Coppier, H. *Command-Control for Real-Time Systems*, 1st ed.; Wiley: Hoboken, NJ, USA, 2013; pp. 7–319.
27. Gazi, V.; Passino, K.M. *The RCS Handbook: Tools for Real-Time Control Systems Software Development*, 1st ed.; Wiley-Interscience: Hoboken, NJ, USA, 2001; Chapters 2.
28. *ISO/IEC 12207:2017*; Systems and Software Engineering—Software Life Cycle Processes. International Organization for Standardization: Geneva, Switzerland, 2017.
29. Haist, B. TO12 RH Control System Standard Work Cell—Final Report. *1110-PAR-P/002-TO012/B2208300/MAI/0001* ITER Organization: Saint-Paul-lez-Durance, France, 2012.
30. ITER. ITER NEWSLIN. 2004. Available online: https://www.iter.org/newsline/singleprint/-/1125 (accessed on 27 May 2024).
31. Tesini, A.; Rolfe, A.C. ITER Remote Maintenance Management System (IRMMS). *Fusion Eng. Des.* **2009**, *84*, 236–241. [CrossRef]
32. Hamilton, D. *Remote Handling Control System Design Handbook*; *2EGPEC v3.0*; ITER Organization: Saint-Paul-lez-Durance, France, 2017 .
33. *ISO 10218-1:2011*; Robots y Dispositivos robóTicos. Requisitos de Seguridad para Robots Industriales. Parte 1: Robots. ISO: Geneva, Switzerland, 2012.
34. *ISO 13849-1:2006*; Safety of Machinery—Safety-Related Parts of Control Systems—Part 1: General Principles for Design. International Organization for Standardization: Geneva, Switzerland, 2006.
35. *IEC 62061:2005*; Safety of Machinery—Functional Safety of Safety-related Electrical, Electronic and Programmable Electronic Control Systems. International Electrotechnical Commission: Geneva, Switzerland, 2005.
36. Hamilton, D. *RH Software Quality Policy*; *TF7PP2 v1.1*; ITER Organization: Saint-Paul-lez-Durance, France, 2017 .
37. Ibarra, A.; Arbeiter, F.; Bernardi, D.; Cappelli, M.; García, A.; Heidinger, R.; Krolas, W.; Fischer, U.; Martin-Fuertes, F.; Micciché, G. The IFMIF-DONES project: Preliminary engineering design. *IAEA Nucl. Fusion* **2019**, *58*, 105002. [CrossRef]

38. IFMIF International Team. *IFMIF Comprehensive Design Report, Broader Approach IFMIF/EVEDA*; IFMIF: Rokkasho, Japan, 2004.
39. Tien, K.; Arbeiter, F.; Ascott, M.; Crofts, O.; McIntyre, G.; Micciche, G.; Mitchell, G.; Qiu, Y.; Tóth, M.; Ibarra, A. Preliminary analysis on a maintainable test cell concept for IFMIF-DONES. *Fusion Eng. Des.* **2019**, *146*, 505–509.
40. Federici, G.; Bachmann, C.; Barucca, L.; Biel, W.; Boccaccini, L.; Brown, R.; Bustreo, C.; Ciattaglia, S.; Cismondi, F.; Coleman, M.; et al. DEMO design activity in Europe: Progress and updates. *Fusion Eng. Des.* **2018**, *136*, 729–741. [CrossRef]
41. Federici, G.; Kemp, R.; Ward, D.; Bachmann, C.; Franke, T.; Gonzalez, S.; Lowry, C.; Gadomska, M.; Harman, J.; Meszaros, B.; et al. Overview of EU DEMO design and R&D activities. *Fusion Eng. Des.* **2014**, *89*, 882–889.
42. Bachmann, C.; Ciattaglia, S.; Cismondi, F.; Eade, T.; Federici, G.; Fischer, U.; Franke, T.; Gliss, C.; Hernandez, F.; Keep, J.; et al. Overview over DEMO design integration challenges and their impact on component design concept. *Fusion Eng. Des.* **2018**, *136*, 87–95. [CrossRef]
43. Hamilton, D. *RH Software Quality Evaluation Procedure*; *SJNDD6 v1.0*; ITER Organization: Saint-Paul-lez-Durance, France, 2016 .
44. F4E . *PROVISION OF I&C INTEGRATION SERVICES: Development and Validation of the RH Command and Control Application*; *F4E-OFC-0811-01-16 v1.10*; ITER Organization: Saint-Paul-lez-Durance, France, 2021 .
45. Utzel, N. *HMI Style Guide and Toolkit 3XLESZ v3.8*; ITER Organization: Saint-Paul-lez-Durance, France, 2018 .
46. Tesini, A. *ITER Process for Human Machine Interface (HMI) Development*; *3T9UK2 v2.0*; ITER Organization: Saint-Paul-lez-Durance, France, 2016 .
47. Katragadda, Y.V.K.A. *RH Structured Language Implementation Using Xtext and Xtend*; *NNUNMC v1.0*; ITER Organization: Saint-Paul-lez-Durance, France, 2014 .
48. Hamilton, D. *IS-23.03-23.07-005 Interface between Cask and Plug RH System (PBS 23.03) and Remote Handling Control System (PBS 23.07): Communications*; *Q6GF4F v1.3*; ITER Organization: Saint-Paul-lez-Durance, France, 2017 .