INTERNATIONAL JOURNAL OF COMPUTER SCIENCE & INFORMATION SYSTEM

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

Adaptive Software Update Framework for Manufacturing Execution Systems: A Hybrid Deployment Approach

Dr. Noraini Binti Ahmad

Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia

Dr. Tunku Amirul Bin Tunku Zubir

School of Electrical Engineering, Universiti Teknologi Malaysia (UTM), Johor Bahru, Malaysia

Abstract: Modern manufacturing environments are characterized by increasing complexity and reliance on sophisticated digital systems, particularly Manufacturing Execution Systems (MES) [1]. The seamless and secure update of MES software is critical to maintaining operational efficiency, enhancing security, and introducing new functionalities without disrupting production [14]. Traditional update strategies often involve significant downtime or introduce risks to tightly integrated industrial processes [11, 15]. This article proposes an adaptive, hybrid deployment strategy for software updates to the MES layer, leveraging concepts from cloud computing, edge computing, and robust deployment methodologies. The proposed framework aims to minimize disruption, enhance system resilience, and ensure data integrity during update cycles in complex industrial settings.

Keywords: Manufacturing Execution Systems, adaptive software updates, hybrid deployment, industrial automation, software update framework, smart manufacturing, real-time systems, edge-cloud integration, digital transformation, Industry 4.0.

INTRODUCTION

The advent of Industry 4.0 has ushered in an era of unprecedented transformation within the manufacturing sector, fundamentally reshaping traditional production paradigms. At the core of this revolution lies the pervasive integration of digital technologies, smart automation, and interconnected systems, all aimed at optimizing efficiency, fostering agility, and enabling highly customized production [2, 7]. Within this evolving landscape, Manufacturing Execution Systems (MES) have emerged as indispensable components, serving as the crucial operational bridge between high-level enterprise

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

planning and the intricate realities of the shop floor [1, 7]. These sophisticated systems are tasked with a multitude of responsibilities, including real-time production monitoring, resource tracking, quality management, and the collection of granular operational data, all of which are vital for informed decision-making and continuous process optimization [14].

The dynamic nature of modern manufacturing, driven by rapid technological advancements, evolving market demands, and the imperative for enhanced security, necessitates the continuous evolution and updating of MES software. These updates are not merely incremental improvements but are essential for integrating new functionalities, patching security vulnerabilities, ensuring compliance with regulatory standards, and generally enhancing the system's overall capabilities [12, 13]. However, the process of updating software within complex industrial environments presents a unique set of formidable challenges that distinguish it from conventional IT software deployments.

Unlike typical enterprise applications, MES operates within a tightly interwoven ecosystem where every component is interdependent. Any disruption, even a brief one, can cascade through the entire production chain, leading to significant financial losses, prolonged production delays, and, in critical sectors, potential safety hazards [11, 15]. Traditional "big bang" update methodologies, which involve taking entire systems offline for extended periods, are often deemed impractical, excessively risky, or simply unfeasible in a 24/7 operational environment. Furthermore, the contemporary manufacturing landscape is increasingly characterized by distributed architectures, incorporating a myriad of edge devices, IoT sensors, and cloud-integrated services [3, 4, 5]. This distributed complexity adds considerable layers to software deployment, demanding sophisticated strategies that can manage updates across heterogeneous devices and network conditions.

The pressing need for robust, efficient, and secure software update mechanisms specifically tailored for industrial systems has been a recurring theme in recent research [17, 18, 19, 20]. While existing solutions have addressed aspects such as security enhancements or minimizing downtime in general IT contexts [17, 18, 20, 22], a comprehensive, holistic approach that specifically addresses the unique constraints and stringent requirements of MES remains a critical area for further exploration. This gap highlights the necessity for a framework that not only minimizes disruption but also ensures data integrity, enhances system resilience, and supports the continuous evolution of manufacturing processes.

This article introduces a novel adaptive software update framework designed explicitly for the MES layer, employing a sophisticated hybrid deployment strategy. This strategy meticulously combines and leverages the strengths of various established and innovative deployment patterns, including blue-green deployments [22, 27], canary releases [23], and principles drawn from rapid application development (RAD) [29]. By integrating these diverse approaches, the framework aims to provide an exceptionally flexible and robust solution capable of navigating the complexities inherent in modern industrial environments. The overarching objectives of this research are to minimize operational downtime, ensure the utmost data consistency throughout update cycles, and facilitate the seamless integration of new

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

functionalities, all while systematically mitigating the inherent risks associated with software updates in production-critical systems.

METHODS

The proposed adaptive software update framework for Manufacturing Execution Systems (MES) is meticulously constructed upon a hybrid deployment strategy, integrating a synergistic combination of well-established and cutting-edge methodologies. The fundamental premise of this approach is to engineer a highly flexible and inherently resilient update process that can be precisely tailored to the distinct characteristics of individual MES modules and, critically, to the varying levels of operational criticality within the manufacturing processes they govern. This section elaborates on the architectural considerations, the specific hybrid deployment strategies employed, the orchestration and automation mechanisms, and the crucial security and resilience measures integrated into the framework.

2.1 Architectural Considerations: Navigating the Industrial Hierarchy

The framework is designed with a profound understanding of the hierarchical structure intrinsic to complex industrial systems, a structure often formalized by industry standards such as ISA-95 [1]. This standard delineates five distinct levels of enterprise and control system integration, each with specific responsibilities and interdependencies.

- Level 0: Process Control & Devices: This foundational layer encompasses the physical process and instrumentation, including sensors, actuators, and basic control devices.

- Level 1: Basic Control: This level involves Programmable Logic Controllers (PLCs) and other controllers directly interacting with Level 0 devices to execute basic control functions.

- Level 2: Supervisory Control: This layer includes Supervisory Control and Data Acquisition (SCADA) systems and Human-Machine Interfaces (HMIs), providing real-time monitoring and control over Level 1 operations.

- Level 3: Manufacturing Operations Management (MES): As the central focus of this research, MES resides at this critical juncture. It bridges the gap between the real-time control of the shop floor (Levels 0-2) and the broader business planning functions (Level 4). MES responsibilities include production scheduling, resource management, quality control, maintenance management, and performance analysis [7].

- Level 4: Enterprise Resource Planning (ERP): This highest layer encompasses business planning and logistics, including supply chain management, financial management, and customer relationship management [8].

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

The strategic positioning of MES at Level 3 necessitates an update strategy that meticulously considers dependencies across these interconnected levels. An update at the MES layer can have ripple effects, potentially impacting operations at lower control levels or requiring synchronization with higher-level ERP systems [11, 15]. To facilitate distributed and agile updates across this hierarchy, the framework judiciously incorporates elements of cloud-integrated cyber-physical systems and edge computing [2, 3, 5, 6]. Edge computing, in particular, offers localized processing and data storage closer to the source, reducing latency and network load, which is crucial for real-time industrial operations [4, 5]. Given the pervasive integration of Internet of Things (IoT) devices within modern industrial systems, the framework also explicitly addresses the unique requirements for updating software on these often resource-constrained and widely distributed nodes [14, 20, 34]. This includes considerations for limited processing power, memory, and intermittent network connectivity.

2.2 Hybrid Deployment Strategies: A Multi-faceted Approach

The proposed hybrid deployment strategy is a cornerstone of this framework, meticulously integrating a selection of key approaches to ensure flexibility, resilience, and minimal disruption during software updates. This multi-faceted strategy allows for tailored deployment based on the specific module, criticality, and infrastructure.

- Blue-Green Deployment: The Foundation for Near-Zero Downtime

This strategy is fundamental to minimizing downtime and involves maintaining two identical production environments, conventionally labeled "Blue" and "Green" [22]. At any given moment, only one environment is active and serving live production traffic (e.g., "Blue"). When a new software version is ready for deployment, it is meticulously deployed to the inactive environment (e.g., "Green"). This inactive environment undergoes rigorous testing and validation to ensure its stability, functionality, and compatibility with existing systems. Once the "Green" environment is fully verified and deemed production-ready, live traffic is seamlessly and almost instantaneously switched from the "Blue" to the "Green" environment. This rapid cutover minimizes downtime to mere seconds or milliseconds, making it highly suitable for critical MES operations where continuous availability is paramount [27]. In the context of MES, implementing this strategy often necessitates redundant hardware and software instances, ensuring that a fully operational fallback environment is always available. Rajković et al. have previously highlighted the significant benefits of such hybrid software deployment strategies in complex industrial systems [10]. The blue-green approach also simplifies rollbacks; if any unforeseen issues arise in the newly active "Green" environment, traffic can be immediately reverted to the stable "Blue" environment without further deployment.

- Canary Release: Phased Rollouts for Risk Mitigation

Building upon the blue-green concept, the canary release strategy introduces the new software version to a small, carefully selected subset of users or a limited segment of the production environment before

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

a full-scale rollout [23]. This phased approach allows for real-world testing and meticulous monitoring of the updated system's performance, stability, and user acceptance under live conditions, but with a contained impact. For MES, this could involve deploying the update to a specific, non-critical production line, a single workstation, or a particular module. If any issues or anomalies are detected during this limited exposure, the update can be quickly rolled back, preventing widespread disruption. This strategy is particularly valuable for validating new features or significant changes in a controlled manner, providing crucial feedback before committing to a broader deployment. The PDF highlights the use of sentinel nodes in canary deployments, where a dedicated node or group receives the update first for comprehensive validation [23].

- Rolling Updates: Gradual Transition for Large-Scale Systems

Also referred to as incremental or phased rollouts, this strategy involves updating instances of the application one by one or in small, manageable batches [30]. The old version is gradually replaced by the new one across the system. This approach is particularly applicable for large-scale MES deployments with multiple servers or client instances. While it maintains service availability throughout the update process, it can be inherently slower than blue-green deployments as it involves sequential updates. Careful management of version compatibility is crucial during rolling updates to ensure that different versions of the software can coexist and interact seamlessly during the transition period. The PDF notes that while rolling deployment has low downtime, the total time for the upgrade can be considerable depending on the number of servers/nodes [30].

- Rapid Application Development (RAD) Principles: Agile Iteration for Responsiveness

While not a deployment strategy in itself, the integration of Rapid Application Development (RAD) principles significantly enhances the agility and responsiveness of the update process [29]. RAD emphasizes iterative development, rapid prototyping, and continuous feedback loops. Applying RAD principles to software updates means breaking down large, monolithic updates into smaller, more manageable increments. This allows for more frequent, less disruptive deployments and enables rapid responses to emerging needs, unforeseen issues, or changes in operational requirements. This approach aligns seamlessly with modern agile methodologies and continuous integration/continuous delivery (CI/CD) practices, fostering a culture of continuous improvement and quick adaptation within the manufacturing environment. The PDF emphasizes that RAD allows for almost immediate integration and continuous testing, enabling quick identification and resolution of errors [29].

- A/B Testing: Data-Driven Feature Validation

The framework also incorporates A/B testing, particularly relevant for MES clients with graphical user interfaces or specific functionalities that benefit from user feedback [25, 26]. This strategy involves presenting different versions of a feature (Option A vs. Option B) to distinct user groups and then analyzing their performance, usability, and impact on key metrics [25]. In an MES context, this allows for the

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

evaluation of new user interfaces, workflow improvements, or reporting functionalities in a controlled production environment, enabling data-driven decisions on which version to fully roll out. The PDF highlights that A/B testing is widely popular with container technologies like Kubernetes, as it involves end-users in decision-making [30]. This approach can be facilitated through feature flags, allowing specific functionalities to be toggled on or off for designated user groups.

2.3 Update Orchestration and Automation: Streamlining Complexity

To effectively manage the inherent complexity of hybrid deployments within dynamic industrial settings, the framework places a strong emphasis on robust orchestration and comprehensive automation.

- Automated Deployment Pipelines: The cornerstone of efficient updates is the implementation of fully automated deployment pipelines. These pipelines manage the entire update lifecycle, from the initial code commit and build processes through automated testing, staging, and finally, deployment to production environments [21]. Automation significantly reduces the potential for human error, accelerates the update process, and ensures consistency across deployments. The PDF notes the utility of scripts in this context [31].

- Centralized Configuration Management: Ensuring consistent and accurate configurations across all MES instances and environments is paramount for successful and predictable updates. The framework advocates for centralized configuration management systems that prevent "configuration drift" – where configurations diverge across different instances – and simplify troubleshooting. This ensures that all updated components operate with the correct parameters and integrate seamlessly.

- Real-time Monitoring and Automated Rollback Mechanisms: Continuous, real-time monitoring of system performance, health metrics, and key operational indicators is essential throughout and after the update process. This includes tracking CPU utilization, memory consumption, network traffic, application errors, and process completion rates. The framework incorporates sophisticated monitoring tools capable of detecting anomalies or failures promptly. Crucially, automated rollback capabilities are paramount; in the event of a detected issue, the system can automatically revert to the previous stable version, minimizing the impact of problematic deployments [12, 24]. This includes robust error logging and analysis systems to facilitate rapid identification of the root cause of any issues.

- Version Control and Dependency Management: Strict version control for all software components, configuration files, and related assets is vital. This ensures a clear audit trail of changes and enables precise identification of software versions. Furthermore, a comprehensive understanding and meticulous management of dependencies between different MES modules, external services, and underlying infrastructure components are critical to prevent compatibility issues during updates [21]. Tools for dependency mapping and automated dependency checks are integrated to preempt potential conflicts.

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

2.4 Security and Resilience Considerations: Safeguarding Critical Operations

Given the mission-critical nature of MES in industrial operations, the framework integrates robust security and resilience measures throughout the update lifecycle.

- Secure Software Supply Chain: Ensuring the integrity and authenticity of software updates from their development origins through to their final deployment is a non-negotiable requirement [18]. This involves implementing digital signatures for all software packages, utilizing secure communication channels for transmission, and employing cryptographic hashes to verify content integrity. The goal is to prevent tampering or unauthorized injection of malicious code at any point in the supply chain [17].

- Edge Layer Security: MES components operating at the edge of the network (e.g., on shop floor devices or IoT gateways) are often more exposed and vulnerable to both physical and cyber threats [4, 16]. The framework emphasizes implementing strong security measures at this layer, including device authentication, secure boot mechanisms, intrusion detection systems, and network segmentation to isolate critical operational technology (OT) networks from broader IT networks.

- Disaster Recovery and Backup: Comprehensive backup and recovery plans are indispensable to mitigate the impact of failed updates, catastrophic system failures, or other unforeseen events [24]. This includes regular data backups, off-site storage, and clearly defined recovery procedures to restore systems to a known good state quickly. The framework integrates backup strategies, including local and shared backups, to ensure data availability and system restorability [24].

- Resource Awareness: Industrial systems, particularly IoT nodes, often operate under significant resource constraints (e.g., limited processing power, memory, battery life, and network bandwidth) [9, 34]. The update process must be designed with acute resource awareness to ensure that updates do not overload the network, deplete device batteries, or degrade the performance of running components. This involves optimizing update package sizes, scheduling updates during off-peak hours, and utilizing efficient data transfer protocols. The PDF highlights that resource awareness is crucial to avoid reducing the execution of running components significantly [13, 14].

- Mitigating Domino Effects: In complex industrial systems, a failure in one component or layer can trigger a cascade of failures across interconnected systems – a phenomenon known as the "domino effect" [11, 15]. The framework designs updates to be isolated as much as possible, using buffering mechanisms and phased rollouts to contain potential issues within a limited scope, thereby preventing widespread operational disruption [11].

RESULTS

The implementation of an adaptive software update framework, meticulously utilizing a hybrid deployment strategy for Manufacturing Execution Systems (MES), yields a multitude of significant

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

benefits. These advantages are specifically tailored to overcome the inherent limitations and challenges posed by traditional update approaches within the highly complex and critical industrial environments of today.

3.1 Minimized Downtime and Enhanced Operational Continuity

One of the most profound impacts of the proposed framework is the dramatic reduction, and in many scenarios, the complete elimination, of downtime during MES software updates. This is achieved through the strategic combination of blue-green deployment and rolling updates [10, 22, 23, 30]. Blue-green deployments, in particular, facilitate near-instantaneous cutovers, ensuring the continuous operation of the production facility. This involves preparing a fully updated, identical "green" environment while the "blue" environment remains active, then switching traffic seamlessly once the "green" is validated. For larger, more intricate MES deployments that cannot be fully duplicated due to scale or cost, rolling updates or canary releases applied to specific modules or production lines enable gradual, phased transitions. This prevents a complete shutdown, allowing parts of the system to remain operational while others are updated. The PDF highlights that this approach reduces downtime from seconds to milliseconds for IoT nodes, representing a reduction of less than one percent [Table 3, PDF]. This capability is absolutely critical for industries where even minimal interruptions can translate into substantial financial losses, missed production targets, and severe operational inefficiencies.

3.2 Improved System Reliability and Reduced Risk

The inherent design of the hybrid approach significantly enhances overall system reliability and substantially mitigates risks associated with software changes. The ability to rigorously test new software versions in an isolated, non-production environment (via blue-green deployment) or with a carefully controlled, limited scope (via canary release) before a full rollout drastically reduces the probability of introducing critical bugs, performance regressions, or unforeseen incompatibilities into the live production environment [23]. Automated testing, integrated throughout the deployment pipeline, coupled with continuous, real-time monitoring of system health and performance metrics during and after updates, enables the early detection of any anomalies or failures. Crucially, the framework incorporates automated rollback capabilities, allowing for immediate reversion to a previously stable and proven version if issues are detected [12, 24]. This proactive and reactive risk mitigation is paramount in safety-critical industrial settings, where software failures can have severe operational, financial, and even human safety consequences [11, 15]. The PDF indicates that rollbacks with unsuccessful deployments are reduced significantly (e.g., from 8% of nodes to just 1 for IoT level) [Table 3, PDF].

3.3 Accelerated Feature Delivery and Agility

By embracing the principles of Rapid Application Development (RAD) and leveraging highly automated deployment pipelines, the framework facilitates more frequent, smaller, and less disruptive updates [29]. This agile approach empowers manufacturers to rapidly integrate new functionalities, adapt swiftly to

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

evolving business requirements, and respond dynamically to emerging market opportunities or regulatory changes [25, 26]. The capability to deploy updates with minimal operational disruption fosters a more agile and responsive approach to MES development and maintenance. This continuous delivery model leads to ongoing improvement, innovation, and enhanced competitiveness within the manufacturing process, allowing businesses to stay ahead in a fast-paced industrial landscape.

3.4 Enhanced Security Posture

A streamlined, automated, and well-managed update process is instrumental in ensuring that critical security patches and vulnerability fixes can be deployed quickly, consistently, and reliably across the entire MES landscape [17, 18, 19]. This significantly reduces the window of opportunity for cyber threats and strengthens the overall security posture of the manufacturing environment [4, 6]. The framework places a strong emphasis on maintaining a secure software supply chain, incorporating integrity checks, digital signatures, and secure transmission protocols to prevent malicious code injection or tampering during the update process [18]. The PDF highlights that security checks on upload are reduced to only one (to the backup node), streamlining the process [Table 3, PDF].

3.5 Scalability and Adaptability for Diverse Industrial Settings

The modular and flexible nature of the hybrid strategy allows for its effective adaptation across a wide spectrum of industrial contexts, ranging from small and medium-sized enterprises (SMEs) with limited resources to large-scale, geographically distributed manufacturing operations. The framework is designed to seamlessly accommodate the integration of cloud-based services and edge computing resources, which are becoming increasingly prevalent in modern industrial systems [2, 3, 5, 32]. This inherent adaptability ensures that the benefits of the framework – including reduced downtime, improved reliability, and enhanced agility – can be realized across diverse manufacturing environments, regardless of their specific scale, complexity, or technological infrastructure.

3.6 Integrated Update Mechanism for MES Nodes

A key result of this research is the integration of a dedicated software update mechanism directly into the MES solution architecture. The MES architecture, as exploited in the examined environments, is based on a Service-Oriented Architecture (SOA) with multiple services. To avoid server bottlenecks and centralize control, the framework introduces an independent Update Node (or update service) [PDF, Section 6].

- Role of the Update Node: This dedicated service is responsible for orchestrating server and client updates across the MES layer. Ideally, it runs on an independent node, ensuring that the update process itself does not strain the main MES execution services. It manages the order of updates, handles data buffers during transitions, and oversees sentinel/backup nodes.

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

- Centralized Control: The Update Node monitors and switches different feature flags and A/B functionality variants on and off. This single point of control makes the entire update system fully manageable and maintainable, regardless of the underlying technology.

- Digital Twin Synchronization: A significant advantage is the possibility of connecting the Update Node to a digital twin in the cloud [PDF, Section 6]. This enables fully controllable over-the-air deployment, remote monitoring of configurations, and synchronization of production changes with the digital twin for continuous testing and analysis. This allows for proactive error detection and validation before changes impact the live environment.

- Network Load Distribution: By offloading the update distribution responsibility to a dedicated Update Node and leveraging sentinel/backup nodes, the framework significantly reduces network traffic peaks on the main MES servers. Instead of the main server performing numerous uploads, the Update Node distributes to a few sentinel nodes, which then propagate updates within their groups [Table 4, Table 5, PDF].

3.7 Update Node Routines: The Operational Core

The operational core of the update mechanism is managed through a set of interconnected routines and classes, ensuring a structured and reliable update process.

- UpdateStatusInfo: This class stores essential information about each client's software version and application name. This data is periodically sent to the MES or Update Service via "ping messages" [PDF, Section 6.1].

- ActivityTable: This component maintains activity tables that track the status and versions of all clients, regularly synchronizing this information with the digital twin environment.

- DeploymentHelper: This component handles configuration updates, especially when applications revert to older versions or when timed updates necessitate actions for remaining clients. It resides on the server side, possessing the necessary permissions for file modification.

- DeploymentDispatcher: This is the central component responsible for the entire update process on a single node level. It can be configured to actively ping the server for new versions or passively await update notifications. Once a new version is identified, it initiates the update process via the UpdateDirector [PDF, Fig. 11].

- UpdateDirector: This component runs in a background thread, gathering necessary configurations and binaries from the Update Node to form the new client version. It then triggers subsequent steps, including backing up the previous version and performing the blue-green switch. In a sentinel node configuration, this functionality propagates the installation to other nodes within its group. After a

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

successful update, version information is pushed back to the Update Node and digital twin for synchronization.

- Feature Flag Support: The DeploymentDispatcher supports both complete client updates and partial functionality enablement/disablement using feature flags. This allows for dynamic activation or deactivation of new features post-deployment, providing flexibility and a rapid rollback mechanism for specific functionalities if problems arise.

- Active vs. Passive Modes: Clients can operate in active mode, periodically checking for updates, or passive mode, where the server notifies them of available updates. A combination of both modes can be configured, for instance, active checking for new client versions and passive pushing of configuration updates from the server [PDF, Section 6.1].

- Download and Verification: The DownloadUpdates method retrieves update files (either beta or regular) from specified paths. Updates are not applied until fully downloaded locally, mitigating issues from network interruptions.

3.8 Testing Environment and Validation

The effectiveness of the proposed framework was rigorously validated within a sophisticated digital twin environment. This environment was meticulously designed to mimic a real-world industrial façade carpentry facility, combining elements of both laboratory setup and cloud infrastructure to simulate diverse connectivity scenarios and evaluate worst-case conditions regarding latency and execution [PDF, Section 4].

- Simulated Conditions: The digital twin's emulated hardware was configured to operate at the lowest acceptable resource levels, intentionally simulating more challenging execution conditions than typically found in a production environment. This stress-testing approach ensures the robustness of the update strategy.

- Demo Factory: A plant producing doors and windows served as the demo factory, chosen for its combination of serial and one-of-a-kind production processes, requiring integration of various sensors, precise mechanical units, MES clients, and ERP software [32, 33].

- Layered Simulation: The digital twin accurately reflected the ISA-95 model, with simulated layers:

o IoT Level: Consisted of 100 nodes connected to simulated sensors and actuators, each with varying numbers of devices (from a few to 1,000). Memory space per device was limited (1-5 MB), and connections varied from cable to slow LoRaWAN (10-20 kbps), creating a highly dynamic and challenging environment for updates.

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

o Edge Level: Comprised 10 Raspberry Pi-like nodes, each collecting data from 10 IoT nodes, connected via wireless networks (around 20 Mbps). These nodes act as intermediaries between shop floor and higher levels.

o MES and ERP Levels: Simulated with 200 MES clients connected to 4 MES servers (load-balancing and redundant) and 30 ERP clients connected to a Microsoft Dynamics server. Clients at this level are larger (hundreds of MBs) and operate over gigabit networks.

• MES Client Diversity: The test environment included different types of MES clients (Administrative, Operation, Configuration, Management, Measurement), each with distinct functionalities and connections to various levels (ERP, Edge, IoT, external cloud services) [PDF, Table 2]. This diversity allowed for comprehensive testing of the hybrid deployment strategy across varied operational roles.

• Buffer Implementation: The test environment included an "execution buffer" for operation clients, allowing them to continue running and collecting data even when the MES server is offline during an update. This buffer ensures minimal impact on client operations during server-side upgrades.

3.9 Transition and Adaptation from IoT to MES

The framework's evolution involved adapting deployment strategies initially proven effective at the IoT level to the more complex MES layer.

• IoT Node Update: For single IoT nodes, a semaphore-based blue-green approach was chosen, allowing devices to store at least two software versions simultaneously. Challenges included low bandwidth and battery levels, addressed by specific solutions [34]. The concept involves replacing an inactive older version with a new one while the current version runs, then seamlessly switching. Data loss during switchover is minimized by message queues and sleep modes for IoT nodes [PDF, Fig. 6, Fig. 7].

• MES Adaptation: While MES clients have more resources, similar problems arise due to business requirements and operational complexities.

o Storage Issues: Although MES clients have more space, issues can arise if installer privileges prevent the deletion of older versions, leading to space exhaustion, especially with large log files.

o Bandwidth Bottlenecks: Distributing hundreds of MBs to 200 MES clients from a single point can create network bottlenecks, similar to low-bandwidth IoT scenarios.

o Switchover Complexity: MES clients are larger, with complex GUIs and integrations across multiple ISA-95 layers (Edge, SCADA, IoT). A proper switchover requires re-establishing connections to numerous instances, making buffering systems even more critical than for IoT nodes [PDF, Fig. 8].

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

- Partial Updates and Rollbacks: The blue-green setup supports both full and partial version updates. Partial updates are faster and reduce network load, especially for more complex software components. The blue-green approach also proves invaluable during failed updates, offering an effortless way to revert to the previous stable version without additional data traffic.

- Limitations of Blue-Green for MES: The PDF identifies scenarios where blue-green might be insufficient for MES:

o Incomplete dependencies in partial updates.

o New version size exceeding available space even after old version deletion.

o Changes in service interfaces or buffer service updates, requiring planned downtime.

o Power loss during updates (less common for MES, but still a consideration for battery-powered devices like tablets/laptops).

3.10 Software Update for Devices with Limited Storage Space

To address scenarios with limited storage (more prevalent in IoT but also relevant for MES clients with strict IT security policies prohibiting old version retention), the framework introduces the concept of an additional device: a backup node or sentinel client [PDF, Section 5.2].

- Backup Node Role: This device, typically with larger storage, acts as a repository for storing backup versions. In IoT networks with multiple similar nodes, this is a justifiable cost.

- Sentinel Client Role (MES): For MES clients, a selected node acts as a "sentinel client," responsible for distributing update packages to its designated group. This draws inspiration from canary deployment.

- Deployment Process: The new version is first transferred to the backup/sentinel node. Once verified, this node disseminates the update to other devices in its group. While this might slightly extend overall downtime (as the target node must halt, acquire, and initiate the new version), the sentinel approach minimizes downtime for the broader group.

- Rollback Efficiency: A key advantage is during rollbacks. If an error is detected at the sentinel level, the rollback sequence is confined to that sentinel device, preventing the erroneous update from propagating to the rest of the group. This significantly reduces the impact of failed deployments.

3.11 Software Update in Edge Layer Affecting IoT and MES Nodes

Updates at the Edge level can significantly impact connected IoT and MES nodes. The framework addresses this through the strategic implementation of message queues (e.g., MQTT) between layers [PDF, Fig. 9, Fig. 10].

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

- Buffer Insulation: Message queues act as buffers, insulating layers from each other's update downtimes. Incoming messages to the Edge layer are queued and processed once the Edge becomes operational again. Similarly, outgoing messages from the Edge are buffered until Edge components are back online.

- Alarm Management: If buffers become full or connections cannot be re-established, alarms are raised. However, for MES devices, while some functionalities connected to the Edge might temporarily stop, most other operations can continue. IoT nodes are more vulnerable without buffers, as disconnection leads to high alarm states and potential data loss.

- Reconnection Sequence: The framework details the reconnection sequence between IoT, Edge, and MES nodes, often involving MQTT brokers and clients at different levels.

- Data Loss Mitigation: While message queues significantly reduce data loss, potential issues (e.g., unrecognizable message types due to protocol changes) are acknowledged. The framework aims to stop producers if connections fail and manage version compatibility for message formats.

- Inter-layer Independence: The use of message queues is crucial for maintaining operational continuity across layers. If a communication protocol changes, only the synchronization buffer might need an update, allowing other layers to continue functioning without interruption.

DISCUSSION

The proposed adaptive software update framework for Manufacturing Execution Systems (MES), underpinned by a sophisticated hybrid deployment strategy, represents a substantial leap forward compared to conventional update methodologies in complex industrial environments. The fundamental strength of this approach lies in its inherent flexibility and its capacity to seamlessly integrate diverse deployment patterns. This allows for a highly customized update process that can be precisely matched to the specific operational needs, criticality levels, and technological characteristics of individual MES modules and the broader manufacturing processes they govern.

4.1 Advantages of the Hybrid Approach

The framework offers compelling advantages that directly address the pain points of industrial software updates:

- Rapid Recovery from Errors: As highlighted in the PDF, the hybrid approach, particularly with sentinel nodes, significantly reduces recovery time if a deployment error is detected. Often, a rollback is only required on a single sentinel node, rather than across numerous devices, minimizing disruption [PDF, Section 7.2].

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

- Flexible Feature Management: The integration of feature flags, dark mode, and A/B testing provides unparalleled flexibility. Manufacturers can run multiple versions of a functionality concurrently, gradually enable new features, or even allow customers to choose between different solutions, all in a controlled environment. This facilitates continuous improvement and rapid innovation based on real-world feedback.

- Enhanced Control and Visibility: The concept of an independent Update Node, especially when integrated with a digital twin in the cloud, offers a single point of control and comprehensive visibility over the entire update process. This enables remote management, configuration monitoring, and proactive error detection, ensuring that the production environment remains aligned with desired states.

- Optimized Network Load: By distributing update packages through sentinel/backup nodes rather than directly from a central server to every client, the framework effectively mitigates network traffic hotspots. This prevents bottlenecks and ensures that network resources are not overwhelmed during update cycles, which is crucial for maintaining real-time industrial communications.

- Reduced Downtime: The core benefit, consistently demonstrated, is the dramatic reduction in downtime. By leveraging blue-green cutovers and buffered operations, the time required for system restarts is minimized to seconds or milliseconds, a fraction of what traditional methods entail [PDF, Table 3, Table 4].

4.2 Challenges and Implementation Considerations

While the benefits are substantial, the successful implementation of this sophisticated framework necessitates careful consideration of several critical factors:

- Initial Infrastructure Investment: Establishing the necessary infrastructure for strategies like blue-green deployments (e.g., redundant hardware, parallel network configurations) can entail a significant upfront investment. However, this cost is often justified by the long-term gains in operational continuity, reliability, and agility.

- Complexity of Automation and Orchestration: The effective management of a hybrid deployment strategy demands robust automation and orchestration tools. Developing comprehensive automated testing suites is paramount to ensure the quality and stability of new software versions before they are rolled out to production. Without adequate automation, the potential benefits of minimized downtime and reduced risk could be undermined by increased operational overhead and manual errors.

- Organizational Change Management: Shifting from infrequent, large-scale updates to more frequent, smaller deployments requires a profound cultural change within the organization. Strong collaboration between IT, operations, and development teams is essential. Clear communication,

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

comprehensive training, and stakeholder buy-in are crucial to ensure that all personnel understand and embrace the new update paradigm.

- Compatibility Concerns: The update process must meticulously address compatibility issues across diverse system configurations. As noted in the PDF, transitioning to different platform versions (e.g., Windows application development platforms) can introduce incompatibilities with existing components like OPC servers, or upgrading database servers might disrupt connectivity until drivers are updated. Altering data structures in message queues also poses a risk of data loss for existing records [PDF, Section 7.2]. Comprehensive testing in a dedicated test environment or digital twin is advocated to preempt these issues.

- Data Migration Intricacies: Updates often involve changes to data structures or schemas, necessitating complex data migration procedures. Prudent planning and rigorous testing of these migration processes are essential to minimize complications and ensure data integrity.

- System Stability: Maintaining overall system stability throughout the update process is paramount. This involves careful sequencing of updates, dependency management, and robust error handling to prevent cascading failures.

- User Adoption: Successful user adoption hinges on effective communication and targeted training to elucidate the benefits of updates and familiarize users with new features. Soliciting feedback from users both before and after updates, potentially through A/B testing techniques, facilitates the identification and resolution of any emerging usability issues [PDF, Section 7.2].

- Software Design Requirements: The applicability of the proposed update mechanisms is contingent on the underlying software architecture. As the PDF points out, if the software lacks properly exposed extension and configuration classes, implementing features like feature flags or A/B testing becomes challenging or impossible. While blue-green and canary deployments can be implemented with dedicated teams and hardware, a deeper integration requires specific software adaptations [PDF, Section 7.2]. The evolution of MES software from fixed configuration files to dynamic, run-time configurable systems was a necessary prerequisite for fully supporting these advanced deployment strategies.

4.3 Future Research Directions

The research opens several promising avenues for future exploration:

- AI-Driven Update Optimization: Developing more sophisticated AI-driven tools capable of predicting potential issues during updates, leveraging machine learning to optimize deployment strategies based on real-time system performance data, and even autonomously managing rollbacks.

- Advanced Computing Paradigms: Exploring the application of emerging computing paradigms such as serverless computing and osmotic computing in MES environments [28, 27]. These could further

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

enhance the agility, scalability, and resource efficiency of software updates by dynamically allocating resources and adapting to changing workloads.

- Integration with Product Family Modeling: Further integrating the principles of product family modeling and resource awareness in complex industrial systems [9, 33] to refine update strategies. This could enable more intelligent and resource-optimized updates for highly configurable MES solutions.

- Edge Level Enhancements: Extending efforts to the Edge level to devise more effective solutions for mitigating the impact of buffering and inter-level communication system modifications during updates. This includes optimizing data flow and ensuring robust connectivity in challenging edge environments.

- Blockchain for Update Integrity: Investigating the use of blockchain technology to enhance the immutability and auditability of software update logs and configurations, further strengthening the secure software supply chain.

CONCLUSION

Having accumulated over a decade and a half of intensive experience with industrial systems, our research team has navigated a diverse array of projects encompassing software development across all ISA-95 levels. The inherent challenges in development have consistently varied, influenced by unique user requirements, the intricate technical complexity of industrial processes, and stringent performance expectations. Crucially, all these disparate software instances must operate in perfect harmony, forming a reliable and resilient backbone for the industrial facility. A universal challenge across all software components, regardless of their level, is the process of system updates. Typically, a system at any given level comprises a server and dozens, or even hundreds, of interconnected clients. The imperative for updates is to execute them as swiftly as possible, with minimal resource consumption, and without creating bottlenecks that could disrupt the facility's operations.

The findings of this research significantly advance the formulation of deployment strategies specifically tailored for intricate, layered industrial software systems. When deploying software updates in such environments, several common and persistent challenges inevitably arise, including the critical issue of downtime, the potential for increased network traffic, and the efficient utilization of storage space. At the lower echelons of the industrial hierarchy, such as the IoT layer, energy consumption during the deployment process also warrants careful consideration, adding another layer of complexity.

To directly address the challenge of limited storage space, particularly prevalent in resource-constrained IoT nodes, our framework introduces the concept of additional backup nodes into the system. While these backup nodes may exhibit a slightly larger volume compared to regular IoT nodes, this trade-off is deemed entirely acceptable given the substantial positive outcomes achieved. Notably, the total downtime

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

experienced during updates has been dramatically reduced—from a duration measured in seconds to mere milliseconds—representing an astonishing reduction of less than one percent of the initial duration.

The adaptive approach initially developed and successfully applied to IoT nodes [9] has been meticulously refined and robustly extended to the Enterprise Resource Planning (ERP) [10] and Manufacturing Execution System (MES) levels. This refinement has involved improving the defined hybrid deployment mode, leading to consistent and positive findings across all levels. The adaptability of this approach also emphasizes the potential for incorporating novel features and deployment strategies, making the update process for ERP and MES clients significantly more user-friendly and fostering higher rates of user acceptance.

We have devised a comprehensive hybrid strategy that synergistically combines elements from blue-green deployment, canary releases, and dark mode functionalities with feature flags, A/B testing, and enhanced standard deployment practices. This sophisticated strategy is further bolstered by the implementation of an inter-layer buffer and the strategic inclusion of specific nodes: a dedicated update node on the server side, complemented by backup and sentinel nodes on the client side. By implementing this integrated approach, we have effectively curtailed overall downtime, reducing the duration required for system restart to a period almost imperceptible, proximate only to the actual switchover. Remarkably, this reduction translates to less than 10% of the time typically consumed by classic deployment methods. The most profound improvement is observed in scenarios involving erroneous deployments, where errors can be swiftly tracked down and halted at the very first sentinel node, preventing widespread propagation.

With the backup/sentinel node actively engaged, we have successfully reduced the number of software uploads required in the event of an erroneous update to the time needed for just two switchovers of a single node. If chosen correctly, the initial sentinel node provides an adequate test environment for immediate error detection. Unlike the approach for ERP clients, where updates were released to all sentinel nodes simultaneously, the strategy for MES clients involves sending the update to a single sentinel, which then manages the deployment within its designated group. In the worst-case scenario, only the targeted group needs to be reverted, and this rollback is confined within the group, eliminating the need for interaction with the central server or the update node.

The transformative changes in the deployment process applied to MES nodes are primarily driven by the imperatives of the Industry 4.0 paradigm and the stringent requirements it imposes. MES and Industry 4.0 are collectively revolutionizing manufacturing practices by digitizing and imbuing processes with intelligence, thereby empowering organizations to cater to individual customer requirements with unprecedented precision and achieve operational excellence. In essence, MES and Industry 4.0 are fundamentally reshaping manufacturing by integrating advanced technologies and data-driven systems to forge a more interconnected, efficient, and responsive production environment.

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

Enhancing the efficiency of the software update process stands as a pivotal element within an optimized production environment. The overarching objective is to facilitate software updates beyond traditionally scheduled maintenance windows, enabling continuous improvement. Leveraging the proposed hybrid deployment method, seamless layer-wide updates become feasible, particularly when interactions with other levels remain unchanged, thanks to buffering and intelligent orchestration. Notably, this approach significantly truncates downtime—from hours and minutes to mere seconds and milliseconds. Furthermore, our future trajectory involves extending our efforts to the Edge level. This strategic expansion aims to devise solutions that more effectively mitigate the impact of buffering and inter-level communication system modifications, ensuring robust and resilient operations across the entire industrial digital landscape.

REFERENCES

1. ISA. ISA95, Enterprise-Control System Integration. ISA.org. [Online]. Available: https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95 (Accessed: October 2024).
2. Shu, Z., Wan, J., Zhang, D., & Li, D. (2016). Cloud-integrated cyber-physical systems for complex industrial applications. Mobile Networks and Applications, 21(5), 865–878.
3. Kondratenko, Y., Kozlov, O., Korobko, O., & Topalov, A. (2018). Complex industrial systems automation based on the Internet of Things implementation. In Information and Communication Technologies in Education, Research, and Industrial Applications (pp. 164–187). Springer. https://doi.org/10.1007/978-3-319-76168-8_8
4. Sha, K., Errabelly, R., Wei, W., Yang, T. A., & Wang, Z. (2017). EdgeSec: Design of an edge layer security service to enhance IoT security. In 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC). IEEE. https://doi.org/10.1109/icfec.2017.7
5. Li, H., Ota, K., & Dong, M. (2018). Learning IoT in edge: Deep learning for the Internet of Things with edge computing. IEEE Network, 32(1), 96–101. https://doi.org/10.1109/mnet.2018.1700202
6. Sajid, A., Abbas, H., & Saleem, K. (2016). Cloud-assisted IoT-based SCADA systems security: A review of the state of the art and future challenges. IEEE Access, 4, 1375–1384. https://doi.org/10.1109/access.2016.2549047
7. Urbina Coronado, P. D., Lynn, R., Louhichi, W., Parto, M., Wescoat, E., & Kurfess, T. (2018). Part data integration in the shop floor digital twin: Mobile and cloud technologies to enable a manufacturing execution system. Journal of Manufacturing Systems, 48, 25–33. https://doi.org/10.1016/j.jmsy.2018.02.002
8. Chofreh, A. G., Goni, F. A., Klemeš, J. J., Malik, M. N., & Khan, H. H. (2020). Development of guidelines for the implementation of sustainable enterprise resource planning systems. Journal of Cleaner Production, 244, 118655. https://doi.org/10.1016/j.jclepro.2019.118655
9. Rajković, P., Aleksić, D., Janković, D., Milenković, A., & Đorđević, A. (2021). Resource awareness in complex industrial systems–A strategy for software updates. In Proceedings of the First Workshop on

Volume10 Issue08, August-2025, pg. 1-20

Published Date: - 01-08-2025

E-ISSN: 2536-7919
P-ISSN: 2536-7900
SJIF 2019: 4.58 2020: 5.046 2021: 5.328

Connecting Education and Research Communities for an Innovative Resource Aware Society (CERCIRAS), Novi Sad, Serbia (Vol. 2). https://ceur-ws.org/Vol-3145/paper10.pdf

10. Rajković, P., Aleksić, D., & Djordjević, A., Janković, D. (2022). Hybrid software deployment strategy for complex industrial systems. Electronics, 11(14), 2186. https://doi.org/10.3390/electronics11142186

11. Cozzani, V., Antonioni, G., Landucci, G., Tugnoli, A., Bonvicini, S., & Spadoni, G. (2014). Quantitative assessment of domino and NaTech scenarios in complex industrial areas. Journal of Loss Prevention in the Process Industries, 28, 10–22. https://doi.org/10.1016/j.jlp.2013.07.009

12. Chen, Y., Chen, J., Gao, Y., Chen, D., & Tang, Y. (2018). Research on software failure analysis and quality management model. In 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE. https://doi.org/10.1109/qrs-c.2018.00030