LEVERAGING SEMANTIC INTEROPERABILITY IN INDUSTRY 4.0: AN OPPORTUNITY FOR AUTOMATED TASK PLANNING IN DISCRETE MANUFACTURING PROCESSES

Holger Weber*, Roland Glück*, Florian Krebs*

* German Aerospace Center (DLR), Institute for Structures and Design, Am Technologiezentrum 4, 86159 Augsburg, Germany

Abstract

Proprietary data formats in manufacturing and supply chain ecosystems impose significant manual effort to harmonise cross-system communication, spanning from shop-floor sensor data over manufacturing execution systems (MES) to enterprise resource planning (ERP) systems. This paper introduces a semantic interoperability framework powered by the Asset Administration Shell (AAS), a standardised digital twin model designed to unify heterogeneous systems. By leveraging AAS, the framework enables resilient, real-time production planning at the shop-floor level, facilitating dynamic process planning, synchronising available resources, and automated deviation detection via seamless integration into a planning/execution service.

In parallel, initiatives like Aerospace-X extend this paradigm in order to address broader supply chain challenges such as demand-capacity bottlenecks, quality management, and circular economy goals. These efforts establish federated data ecosystems which utilise standardised submodels, such as digital nameplates and digital product passports, to enhance visibility across multi-tier supplier networks, mitigate disruptions from material shortages, improve collaborative decision-making, and instantiate a market place for central services. The idea is that transitioning from rigid, inflexible supply chains to multilateral, AAS-enabled supply networks reduces lead times, optimises inventory management, and strengthens resilience against global disruptions.

To show the potential of AAS-driven manufacturing, we explored the feasibility to use AASs with their inbuilt capability and operation submodels to perform a discrete assembly task. In the FoF-X (Factory of the Future - eXtended) project we used these models to demonstrate autonomous task planning as well as autonomous error recovery. By combining planning via classic symbolic artificial intelligence (e.g. by using Planning Domain Definition Language (PDDL) [1]) with the modelling power of AAS submodels and perspectively formal semantic ontologies (e.g. OWL, RDF) for capability descriptions defined as AAS submodels, the framework achieves an agnostic and autonomous approach to planning, execution, non-conformity detection, and re-planning. The current implementation focuses on a toy problem to evaluate the technology. An application in an aerospace context is planned in the continuation project ASPIRO (Aerospace production using intelligent robotic systems). The results regarding implementation, caveats, and future perspectives are shown in this paper.

The results of this part of FoF-X emphasise the transformative role of semantic standardization in bridging vertical integration (shop-floor to ERP) with horizontal collaboration (supplier to OEM or M2M communications), offering a scalable blueprint for Industry 4.0 digitization, which enables higher efficiencies in distributed supply chains as well as higher resource utilization on the shop floor. This work underscores the critical importance of adopting AAS as a universal language for enabling intelligent and adaptive manufacturing ecosystems.

Keywords

Industry 4.0; Asset Administration Shell; Digital Twin; Semantic Interoperability; Automated Planning; PDDL; Manufacturing Execution Systems

1. INTRODUCTION

The ongoing digitalisation, globalisation and the rapid emergence of new technologies are creating an increasingly complex industrial landscape. Whether it is communication between machines on the shop-floor, human-machine interaction, or data exchange between companies, the multitude of proprietary data formats generates huge bureaucratic effort and hampers interoperability [2], [3].

The rapid rise of digital twins, sensor-rich machines, and cloud-based analytics has pushed *Industry 4.0* from a buzzword to a practical reality. Yet, the very diversity that promises agility also introduces a semantic bottleneck: devices, software, and humans still talk to each other in proprietary formats, heterogeneous environments making it impossible for systems to interact with each other in a sort of plug and play manner. Different interfaces and languages make it nearly impossible to exchange assets with those of competitors with near to no effort and lack of semantic harmonization.

Within the *Industry 4.0* context, the *Asset Administration Shell (AAS)* [4] has emerged as the successor to proprietary data-exchange formats, a formal, ISO/IEC-standardised data model that encapsulates all information about an asset (a machine, tool, product, or process). The AAS provides a *standardised data model* that acts as a *manufacturer-independent "common language"* and enables interoperable interfaces and the basis of cyber physical production systems [5]. AAS is part of the *Industry 4.0 Reference Model (RAMI 4.0)* and offers a common vocabulary for *identification, properties, sub-models, and operations*.

Typical AAS use cases include, for example:

- Product *life-cycle management* (Life-Cycle-Assessment)

doi: 10.25967/650173

- Condition-based monitoring and maintenance of assets
- Integration of manufacturing resources into higher-level value networks
- Implementation of a Product Passport

In the *Factory of the Future - eXtended* (FoF-X) project, a KUKA LBR iiwa robot and a Universal Robots UR10e were each wrapped in an AAS. Each robot's AAS exposed submodels describing *capabilities* derived by their API (e.g., move, position information, torque information) and their skill compositions (e.g., `pick`, `place`) as *operations* for assembly tasks. This enabled a shared planning layer to consume both robots through the same abstract standardised interface, eliminating vendor-specific wrappers.

2. RELATED WORK

2.1. Semantic Interoperability in Manufacturing

Semantic interoperability in manufacturing has evolved from early standardization efforts to sophisticated digital twin implementations. The Reference Architecture Model for Industry 4.0 (RAMI 4.0), defined in DIN SPEC 91345, provides a comprehensive framework for structuring Industry 4.0 systems across multiple dimensions including hierarchy levels, life cycle phases, and functional layers [6]. Within this framework, the Asset Administration Shell serves as the standardised information model for representing assets and their capabilities.

Recent research has focused on enhancing AAS capabilities through advanced technologies. Xia et al. [7] investigated the integration of large language model agents for generating AAS instances, demonstrating improved semantic consistency in digital twin creation. Similarly, Lu et al. [8] explored the application of model-based design principles to enhance AAS-based production line design, emphasising the importance of semantic consistency across system boundaries.

The Manufacturing-X and Aerospace-X initiatives represent large-scale efforts to implement semantic interoperability in industrial ecosystems [9]. These initiatives leverage the Semantic Aspect Meta Model (SAMM) and Catena-X standards to create federated data spaces that enable secure and semantically consistent information exchange across organizational boundaries [10].

2.1.1. Cyber Physical Systems and Skills

The concept of capabilities and skills in manufacturing has been extensively explored. Nakamura and Harada [11] introduced a skill-based framework that demonstrates hierarchical structuring of primitive manufacturing skills (e.g., *MoveCartesian*, *Suck/Release*). RAMI 4.0 mapping studies analyze the integration of AAS into existing MES systems to evaluate flexibility gains [12].

2.1.2. Semantic Integration with OPC UA

Analogously to *OPC UA (Open Platform Communications Unified Architecture)*, the AAS follows the *RAMI 4.0 reference-architecture model*. RAMI 4.0 defines a *domain-neutral, semantically-oriented layered architecture* that helps transform rigid value chains into resilient value networks. The model is supported internationally by Germany, France, Italy, Japan, China, and other partners, and successful mappings to other reference architectures already exist.

In the Catena-X project, the concept of the AAS is extended through their Semantic Aspect Meta Model (SAMM), describing the semantic interconnection between submodels. Since OPC UA, AAS, and SAMM are deeply interconnected in their ambition to establish standardised concepts in a typically proprietary ecosystem, there are already proposals for leveraging each technology's strengths and integrating them.

2.2. Automated Planning in Manufacturing

Modern manufacturing processes are characterised by rapid and continuous parametric variations, nonlinear behaviors, and inherent uncertainties that significantly affect product quality and process efficiency. These dynamic manufacturing systems face unprecedented challenges in maintaining consistent quality and optimal performance due to time-varying parameters, unpredictable process dynamics, and environmental uncertainties.

Automated planning has been extensively applied to manufacturing scenarios, with the Planning Domain Definition Language (PDDL) serving as the de facto standard for representing planning problems [13]. The Fast Downward planner, developed by Helmert [14], has demonstrated exceptional performance in manufacturing planning scenarios, providing optimal solutions for complex resource allocation and scheduling problems. Recent advances in planning-as-a-service platforms provide extensible APIs to deploy planners in local or cloud environments, utilising tools like planutils for comprehensive planner libraries [15].

2.3. Integration of Semantic Models and Planning

The integration of semantic models with automated planning represents a relatively underexplored area in manufacturing research. Existing approaches typically focus on either semantic representation or automated planning, with limited integration between the two domains. The present research addresses this gap by providing a comprehensive framework that seamlessly combines AAS-based semantic representation with PDDL-based automated planning.

Here, we focus on two key technologies:

1) *Asset Administration Shell (AAS)* – the basis for standardised, manufacturer-independent data exchange, storing both *metadata* (digital nameplate, manufacturing life-cycle, safety constraints) and *operational data* (status, logs, process data).
2) *PDDL (Planning Domain Definition Language)* – a declarative language for modelling planning problems and automatically generating action sequences.

We present a concept that *combines these technologies* to define flexible, reusable manufacturing *skills* and to control their execution in an AAS-based environment.

3. FRAMEWORK ARCHITECTURE

3.1. Overview

The proposed semantic interoperability framework is designed as a multi-layered architecture that spans from physical manufacturing assets to high-level planning and execution services. The framework consists of five primary layers: Asset Layer, Semantic Layer, Planning Layer, Execution Layer, and Integration Layer.

The framework operates on the principle of semantic consistency, ensuring that manufacturing capabilities are maintained throughout the planning and execution process. The

Asset Administration Shell serves as the central semantic hub, providing standardised representation for manufacturing assets, their capabilities, and operational informations.

3.2. General Infrastructure

The infrastructure consists of several key components that enable semantic interoperability and automated planning:

TAB 1. Core Infrastructure Components

Component	*Description*
Asset Administration Shell (AAS)	Standardised data model for assets (machines, tools, products) with OPC UA, REST, MQTT interfaces
PDDL Planner	Automatic planner that generates action sequences from domain and problem definitions via REST API
Capability Repository	Service discovery and health-check; Library for available resources (e.g., UR10e, Kuka LBR iiwa) and their corresponding capabilities (e.g., *pick*, *place*) exposed as AAS sub-models
Orchestration Layer	Translates plans to AAS operation calls; Coordinates skill execution, handles errors and retries via asynchronous REST controller

The processing pipeline operates as follows:
1) User configures a product via the UI, sending a job request/product description to the *Planning Service*.
2) The Planning Service pulls all relevant AAS *capability sub-models* to create the planning domain.
3) Fast-Downward generates an optimal plan.
4) Orchestrator maps each action to its concrete AAS *operation* (e.g., `pick`).
5) Periodical health checks and quality assurance ensure system reliability.

3.3. Asset Layer

The Asset Layer represents the physical and logical manufacturing assets within the production environment. This layer encompasses various types of manufacturing equipment, including robotic systems, the product, the planning service, and a quality inspection station. Each asset is encapsulated within an Asset Administration Shell that provides standardised semantic representation of the asset's capabilities, properties, and current operational state.
The AAS implementation follows the IEC 63278-1:2023 standard [4], ensuring compatibility with existing Industry 4.0 infrastructure. Each asset's AAS contains multiple submodels that describe different aspects of the asset, including technical specifications and operational capabilities. The AAS communication interface is implemented using RESTful APIs and optional OPC UA protocols, ensuring interoperability.

3.4. Semantic Layer

The Semantic Layer is responsible for aggregating, harmonising, and maintaining semantic consistency across all manufacturing assets. This layer implements core semantic interoperability mechanisms including modelling, mapping, and registering capabilities and data transformation services.
The layer utilises the Capability Description Submodel to ensure standardised semantic representation across different asset types and organizational boundaries, enabling consistent semantic interpretation across heterogeneous systems.

3.5. Planning Layer

The Planning Layer transforms available capability descriptions into executable planning domains and coordinates automated planning activities. This layer implements the novel AAS-to-PDDL transformation methodology, which systematically converts AAS skill models into PDDL action definitions, enabling automated reasoning over manufacturing capabilities.
The layer incorporates the Fast Downward planning system, which provides optimal solutions for complex manufacturing planning problems. The planner considers resource constraints, temporal dependencies, and quality requirements to generate executable task sequences that achieve specified manufacturing objectives.

3.6. Execution Layer

The Execution Layer coordinates the execution of planned task sequences and monitors ongoing manufacturing operations. This layer interfaces directly with Manufacturing Execution Systems (MES) and provides real-time feedback to the planning system regarding task progress, resource availability, and potential execution conflicts.
The layer implements sophisticated error detection and recovery mechanisms, enabling automatic adaptation to changing manufacturing conditions and unexpected events. When execution deviations are detected, the layer triggers replanning activities to maintain production continuity.

3.7. Integration Layer

The Integration Layer delivers standardised interfaces and communication protocols to facilitate seamless connectivity with existing manufacturing infrastructure. It supports various industrial communication standards, such as OPC UA and HTTP-based APIs, ensuring broad compatibility across diverse manufacturing systems. By default, the framework employs HTTP-based APIs; however, adopting OPC UA for communication is possible with minimal configuration effort by modifying the Asset Administration Shell (AAS) settings. For the integration of KUKA peripheral components, the proprietary EtherCAT protocol is utilised, but this complexity is encapsulated behind the asset layer, ensuring a unified interface to higher-level services.

4. CONCEPT OF A SKILL

A *skill* is a reusable, atomic manufacturing function described in a *skill sub-model* of the AAS.

4.1. Primitive Skills (Tasks)

Primitive skills represent basic manufacturing operations that can be composed into more complex workflows. The

following table describes fundamental skills used in the framework (TAB 2).

TAB 2. Primitive Manufacturing Skills

Skill	*Description*	*Parameters*
MoveCartesian	Positions the end-effector in Cartesian coordinates	target-X, Y, Z, speed
Suck/Release	Activates/deactivates the vacuum gripper	mode (suck/release)

A skill is modeled as a sub-model within the AAS with the following mandatory fields (TAB 3).

TAB 3. Skill Model Structure

Field	*Type*	*Example*
Sub-model ID	URI	https://www.dlr.de/bt/submodels#UR10e
Parameters	JSON	`{"X": float, "Y": float, "Z": float}`
Operation	URI	https://www.dlr.de/bt/submodels#UR10e#operations#MoveCartesian

Skills are invoked via standardised REST API calls to the AAS operation endpoints, enabling vendor-independent control of manufacturing assets.

4.2. Skill Composition

Complex manufacturing procedures are defined by *skill sequences* (workflows) that consist of a series of primitive skills. The sequence is modelled in PDDL as a *plan* and subsequently transferred to the AAS as a *composite skill*. The composition approach enables dynamic reconfiguration of manufacturing processes without requiring reprogramming of individual assets, significantly enhancing system flexibility and adaptability.

5. AAS-TO-PDDL TRANSFORMATION METHODOLOGY

5.1. Task Decomposition

The transformation from AAS capability models to executable PDDL planning problems follows a systematic multi-stage process:

1) *Domain & Problem Construction in PDDL*
   - Formalise the goal state (e.g., *pyramid 3 by 3*) and the current state.
   - The scheduler creates *PDDL domain files* by querying the AAS registry for all available *capability* sub-models of the participating robots.
   - The *problem file* is derived from the current *system state* exposed by each robot's AAS, pyramid AAS and Quality Assurance AAS (e.g., current pose, held objects, pyramid surface layer state).
2) *Planner Execution*
   - Fast-Downward is invoked via a REST endpoint with parameters such as *timeframe*, as it iteratively converges to the best solution.
   - The planner returns a JSON representation of the plan: an ordered list of action calls.
3) *Mapping to AAS Skills*
   - Each planner action is linked to the corresponding skill sub-model.
   - The orchestrator resolves the *operation endpoint* and serialises the parameters into the expected JSON payload.
4) *Execution & Feedback Loop*
   - The orchestration layer invokes the skills via HTTP POST to the robot's AAS operation endpoint.
   - If the orchestration detects a status event (`design_deviation`, `robot_error`), the orchestrator updates the *global plan* state and triggers re-planning if necessary.

5.2. PDDL in Industry 4.0

PDDL (Planning Domain Definition Language) is a declarative language for describing planning domains (predicates, actions) and concrete planning problems (initial and goal states). It enables automatic generation of action plans by AI planners such as *Fast-Downward*.

TAB 4. PDDL Features in Industrial Context

PDDL Feature	*Industrial Application*
Domain Modelling	Define reusable robot skills (actions) and predicates (states)
Problem Definition	Encode current state and desired end-state for production scenarios
Automatic Planning	Generate optimal sequences using Fast-Downward, minimising production time while satisfying constraints

Why PDDL works well in Industry 4.0:

- Declarative nature permits *automatic optimization* across multiple robots.
- Planner can incorporate resource constraints (e.g., limited tool slots) into the decision process.
- Enables vendor-independent task sequencing and dynamic replanning.

5.3. Semantic Skill Modelling

The transformation methodology begins with the systematic representation of manufacturing capabilities as semantic skills within the Asset Administration Shell framework. Manufacturing capabilities are modeled as discrete, executable units that encapsulate specific manufacturing operations.

Each skill is represented as an AAS submodel that contains structured information about the skill's parameters, datatypes, and output information. The skill model utilises standardised property definitions to ensure semantic consistency and enable automatic processing by the transformation system.

The skill modelling approach distinguishes between three primary skill categories: manipulation skills (e.g., pick-and-place operations), safety skills (e.g., move to safe zones), and inspection skills (e.g., quality verification). Each category follows specific semantic patterns that facilitate consistent transformation to PDDL representations.

5.4. Problem Instance Generation

The framework provides automated problem instance generation capabilities that convert specific manufacturing

requirements into PDDL problem definitions. The problem generation process analyzes production orders, resource availability, and preconditions to create comprehensive problem instances that accurately represent manufacturing objectives.
The problem instance generation includes sophisticated constraint handling mechanisms that ensure generated problems are solvable while maintaining realistic manufacturing constraints. The system automatically handles resource constraints or concurrent operation limitations.

6. ERROR CLASSIFICATION AND RECOVERY

6.1. Error Classification

The framework implements a comprehensive error classification system that enables appropriate recovery strategies for different failure modes:

TAB 5. Error Classification and Recovery Strategies

Error Class	*Cause*	*Detection*	*Recovery*
Quality error	Faulty product (misaligned or missing parts)	Anomaly detection via vision system	Repick/resort, trigger re-plan
Actuator error	Malfunctioning robot, gripper, etc.	Status feedback from AAS	Safe-stop, swap robot, re-plan with alternative resources
Communication error	Network interruption	Timeout monitoring	Search for free capacities, re-connect, re-plan

The camera system checks after each layer the difference between design and actual state. Triggered by the orchestrator after each layer, it can detect how many layers were laid and identify deviations from the planned configuration.

6.2. Recovery Strategy

After an error flag is received from the robot's AAS event stream, the general recovery strategy of the orchestrator includes:
1) Query the affected resource's status and error codes.
2) Execute the corresponding safety or replanning step.
3) Ensure the current system state is safe to be continued.
4) Search for free production capacities via lookup in the AAS registry.
5) Validate needed capabilities.
6) Provide current state information (from camera) and desired product specification to the PDDL Planner.
7) Generate and execute new plan.

6.2.1. Quality Error Recovery

The camera detects when the product deviates from the planned solution. If the camera detects that the product is not as expected, it signals the orchestrator that the product does not match its specifications. As a recovery strategy, the orchestrator searches for free production capacities, provides the current state of the product and the desired product to the PDDL Planner, and triggers re-planning.
Limitations:
- The camera only evaluates the layer with the topmost ball.
- Visibility of the ArUco marker is necessary for accurate positioning.
- The camera must be placed within a certain distance range to differentiate layers (limited by camera resolution) and colours while not hindering robot movement.

6.2.2. Actuator Error Recovery

The actuator's health is checked via polling. If a given actuator is faulty, the orchestrator searches for free production capacities. The camera provides the current state of the product and the desired product specification to the PDDL Planner for re-planning.

6.2.3. Communication Error Recovery

Network interruptions are detected through timeout monitoring. The system attempts to re-establish connections and, if necessary, identifies alternative resources with equivalent capabilities through the AAS registry. Re-planning is triggered to accommodate the modified resource availability.

7. IMPLEMENTATION AND INTEGRATION

7.1. Technology Stack

The framework implementation utilises a modern, container-native technology stack that ensures scalability, reliability, and maintainability. Every component is encapsulated within its own Docker container to ensure isolation and security. The core implementation is based on Java and Python, providing robust enterprise-grade capabilities for handling complex manufacturing scenarios.
The Asset Administration Shell implementation utilises the FA³ST (Fraunhofer Advanced AAS Service Tools) framework, which provides comprehensive AAS management capabilities including data model validation, communication interface implementation, and semantic reasoning support [16]. This ensures full compliance with IEC 63278-1:2023 standards and compatibility with existing AAS infrastructure.
The planning system integrates the Fast Downward planner through a custom AAS wrapper that provides seamless integration with the semantic layer. The wrapper handles PDDL domain and problem generation, planner invocation, and solution parsing, ensuring efficient and reliable planning operations.

7.2. Modular Programming Architecture

The framework follows a modular architecture with clearly separated concerns:

7.2.1. Robot Asset Components

All Components are made for both robot assets, the UR10e and the Kuka iiwa robot. Whilst their capabilities are proprietary, skills are vendor-agnostic.
Capabilities: Serialization of commands and state information. State includes position, velocity, and operational status data.
Capability Skill Mapping: Matches provided capabilities to skills, enabling dynamic skill composition based on available resources (e.g. pick or place).

7.2.2. Planning and Execution (Orchestrator)

The planning service pulls PDDL-capabilities and goal state from AAS, building problem and domain representations. For solving, planning-as-a-service is used. The execution service maps plans to AAS operations and invokes them sequentially.

7.2.3. Planning-as-a-Service

Planning as a service (PaaS) provides an extendable API to deploy planners online in local or cloud servers. The service provides a queue manager to control a set of workers, which can easily be extended with one of several planners available in planutils. planutils is an open-source and adjustable library for planners provided via Docker image and reachable via REST endpoints.

7.2.4. Kuka LBR iiwa Server

A RESTful server for the Kuka LBR iiwa robot provides a comprehensive API for controlling the LBR, retrieving its current state, and performing various operations such as moving the arm to a specific position and executing predefined action sequences. It uses reflection to interpret methods and GSON [17] to serialise/deserialise data objects.
The server itself is asynchronous and non-blocking, allowing for efficient handling of multiple tasks. Most function calls are synchronous, allowing for a single task at a time as a safety aspect and making it easy to queue calls. Stop and safety-relevant features are non-blocking, as is the use of external signals (e.g., EtherCAT) to trigger the vacuum valve.

7.2.5. Computer Vision

The computer vision module includes preprocessing of depth images and matching of stereo images. It uses the OpenCV library for image processing and the ArUco marker library for marker detection. Post-processing steps normalise the depth image and stereo images via ArUco markers and include denoising.
The application is a Flask server wrapped by an AAS which provides its capabilities as operations. This operation provides:

- Information on detected ball positions and their colours in JSON format
- The depth image
- The RGB stereo image

7.2.6. Quality Assurance

The quality assurance application provides product quality checking. It takes a depth image, stereo image, and the design AAS to return an error array for the current state, providing information about wrong colours and missing balls.

7.2.7. User Interface

The user interface is a web application served by a Dash Python server. The UI provides a graphical interface for configuring the product, starting the planning process, viewing planning results (PDDL), initiating plan execution, and real-time visualization of system state and performance.

7.3. AAS Creator and Java Interface Wrapper

The AAS Creator uses Java reflection to read JAR libraries. By selecting a main file that reflects the skills of the robot (e.g., for KUKA it is the Sunrise LBR controller), the framework derives the skillset of the robot. Inheritance is taken into account. Functions are mapped as operations, and complex datatypes are broken down into their chained structure until reaching primitive datatypes, similar to primitive skills.

7.3.1. FA^{3}ST Custom Asset Connection

The custom Asset Connection requires the respectively mapped JAR, which is mounted automatically at startup. In the FA^{3}ST config file for the server, the operations of the AAS are mapped to the custom asset connection. If an operation is invoked, the custom asset connection interprets the input parameters and builds complex datatypes. The connection can handle overloaded functions, and complex datatypes are automatically translated to/from JSON via GSON.

7.4. FA^{3}ST Registry

Components automatically register their AAS at the registry, making them discoverable in the industrial context. Hosting capability submodels, the registry facilitates easy lookup of available resources and dynamically adapts to production needs, providing the PDDL planner with necessary information.

7.5. Manufacturing Execution System Integration

The framework provides comprehensive integration capabilities with existing Manufacturing Execution Systems through standardised interfaces and communication protocols. The integration layer supports various communication standards, including OPC UA and RESTful APIs.
The MES integration handles complex synchronization requirements, ensuring that planning decisions are properly coordinated with ongoing manufacturing operations and asynchronous task execution. The system provides real-time status updates, enabling dynamic replanning when manufacturing conditions change or unexpected events occur.

7.6. Configuration and Deployment

Docker containerization ensures consistent deployment across different infrastructure environments. The configuration management system supports dynamic reconfiguration of semantic models, planning parameters, and integration endpoints, enabling adaptation to changing manufacturing requirements without system downtime.

8. EVALUATION AND PRACTICAL APPLICATIONS

8.1. Experimental Setup

The framework evaluation was conducted using a discrete manufacturing scenario that represents typical Industry 4.0 production environments with a toy problem. The experimental setup included manufacturing assets such as robotic assembly stations and a quality inspection system.

8.1.1. Assets Setup

The setup includes a KUKA LBR iiwa and a Universal Robots UR10e, each equipped with vacuum suction device and vacuum valves. Different technologies are used for external control: the UR10e has built-in 24V digital outputs,

TAB 6. Manufacturing Assets

Asset	Model	Key Features
KUKA LBR iiwa 14	7-DoF arm	14 kg payload, collaborative robot
UR10e	6-DoF arm	12.5 kg payload, industrial robot
Vision Camera	RealSense D435	3D depth via stereo vision, Resolution 1280×720

while the KUKA requires additional hardware for EtherCAT communication. A RealSense D435 stereo camera with infrared capability enables stereo image matching and depth image creation.
The workspace includes a board with four containers housing two colours for the two robots. Four ArUco markers on the board make the camera less volatile to disturbances, correct distortions, and normalise the depth view. With ArUco markers, the setup is camera-position agnostic (limited by marker visibility and camera resolution).
Each manufacturing asset was equipped with an Asset Administration Shell that provided comprehensive semantic representation of the asset's capabilities, current operational state, and parameter requirements. The AAS implementations followed IEC 63278-1:2023 standards and included multiple submodels representing different aspects of asset functionality.
The evaluation scenario involved the production of pyramid structures requiring assembly and quality inspection operations. Production orders specified various quality requirements, product constraints, and resource availability, creating complex planning challenges that effectively demonstrate the framework's capabilities.

8.2. Transformation Accuracy

The AAS-to-PDDL transformation methodology demonstrated high accuracy in converting semantic skill models into executable planning problems. Manual verification of generated PDDL domains confirmed that all essential manufacturing constraints, resource requirements, and operational dependencies were correctly represented in the planning formulation.
The transformation process successfully handled complex scenarios including concurrent resource usage, temporal constraints, and quality-dependent processing paths. Semantic consistency was maintained throughout the transformation process, ensuring that planning solutions accurately reflected manufacturing capabilities and constraints.

8.3. Planning Performance

The integrated planning system, utilising the Fast Downward planner, demonstrated excellent performance in generating optimal solutions for complex manufacturing scenarios. Planning times remained within acceptable limits for real-time production planning applications, typically completing within seconds for scenarios involving dozens of manufacturing operations and multiple resource conflicts.
The planner successfully identified optimal task sequences that minimised production time while satisfying all quality and resource constraints. The generated plans were validated and demonstrated high practical feasibility when executed in the experimental manufacturing environment.

8.4. Integration Effectiveness

The framework's integration capabilities were thoroughly evaluated through connection with Manufacturing Execution Systems and enterprise software infrastructure. The standardised communication interfaces enabled seamless data exchange with diverse manufacturing systems, including legacy equipment and modern Industry 4.0 infrastructure.
Real-time execution monitoring demonstrated effective coordination between planning activities and manufacturing operations. The system successfully detected execution deviations and triggered appropriate replanning activities, maintaining production continuity despite unexpected events and resource conflicts.

8.5. Scalability Analysis

The containerised architecture provides horizontal scalability through container orchestration, enabling dynamic resource allocation based on computational demand. This ensures that the framework can adapt to varying manufacturing workloads and support large-scale industrial deployments. The system needs further testing with increasingly complex manufacturing scenarios to validate performance at scale.

9. DISCUSSION

9.1. Potential Impacts

The integration of AAS and PDDL creates significant opportunities for transforming manufacturing systems:
Standardisation: Combining AAS and PDDL creates a manufacturer-independent, semantically rich data exchange framework. All partners within a value network can discover each other's capabilities automatically via the AAS registry. No custom connectors are required, significantly reducing integration effort and enabling rapid system reconfiguration.
Flexibility: Manufacturing skills can be composed dynamically and adapted to changing production conditions. The declarative planning approach enables automatic optimization across multiple resources, allowing the system to respond intelligently to resource availability changes, quality requirements, and production priorities.
Competitive Advantage: Companies can react faster to market changes because new products can be introduced by merely defining new skills and planning goals rather than extensive reprogramming. This significantly reduces time-to-market for new products and enables mass customization at scale.

9.2. Current Limitations

While the proposed framework demonstrates significant potential for enhancing semantic interoperability and automated planning in manufacturing environments, several limitations must be acknowledged. The current implementation focuses primarily on discrete manufacturing scenarios and may require adaptation for continuous processing applications.
The AAS-to-PDDL transformation methodology, while comprehensive, requires accurate semantic modelling that demands significant expertise in both manufacturing processes and semantic technologies. However, automated modelling concepts are emerging [7], and manufacturers

are increasingly adopting the AAS as an exchange format, which will reduce this barrier over time .
The framework's reliance on accurate semantic modelling requires significant expertise in both manufacturing processes and semantic technologies. As industry adoption of AAS increases and tooling matures, this limitation is expected to diminish.
While dockerization of decentralised components facilitates scalability across factory levels, centralised components such as PDDL planning require validation against more complex manufacturing scenarios to identify potential bottlenecks.

9.3. Data Space Integration

The framework is designed to operate within federated data space environments, supporting initiatives such as Manufacturing-X or in particular Aerospace-X that promote secure and semantic information sharing across organizational boundaries. The implementation includes comprehensive data sovereignty and security mechanisms that ensure controlled access to sensitive manufacturing information.
The data space integration utilises the Eclipse Dataspace Connector (EDC) technology to provide secure, policy-controlled data exchange capabilities [18]. This enables cross-organizational collaboration while maintaining data sovereignty and compliance with regulatory requirements, as well as providing Manufacturing-as-a-Service (MaaS) across the dataspace.
Organizations can offer their manufacturing capabilities through the data space, enabling flexible supply chain configurations and collaborative production scenarios. This supports emerging business models based on distributed manufacturing and MaaS.

10. FUTURE WORK

Several promising directions exist for extending and enhancing the framework:

10.1. Expanded Skill Catalogue

Future development should expand the skill catalogue with more complex actions, including collaborative robotics scenarios, adaptive quality inspection methods, and advanced material handling operations. This expansion will enable the framework to address a broader range of manufacturing applications and support more sophisticated production processes.

10.2. Machine Learning Integration

Advanced machine learning techniques present significant opportunities for enhancing the framework's capabilities. Future work may explore:
- Integration of reinforcement learning for dynamic planning optimization and continuous improvement of manufacturing strategies
- Use of Large Language Models (LLMs) for natural language processing and automated skill model generation from textual descriptions
- Computer vision enhancement for real-time quality assessment, process monitoring, and adaptive control
- Predictive maintenance integration based on AAS sensor data and operational history

Machine learning could potentially replace or augment PDDL-based planning, enabling learning-based optimization of skill parameters and manufacturing sequences.

10.3. Semantic Reasoning Enhancement

The framework's current implementation relies primarily on predefined skill models and static PDDL transformations. Future development should integrate advanced semantic reasoning capabilities that leverage formal ontologies such as OWL (Web Ontology Language) and RDF (Resource Description Framework) to enable intelligent inference over manufacturing knowledge.
Semantic reasoning would allow the system to:
- Automatically derive new capabilities from existing ones
- Identify equivalent operations across different manufacturing assets
- Optimise planning decisions based on inferred semantic relationships
- Support dynamic capability discovery and automated conflict resolution

By incorporating description logic reasoners and knowledge graphs, the framework could support context-aware planning and more sophisticated decision-making.

10.4. Extended Domain Support

Expansion to additional manufacturing domains represents important future research directions, including:
- Continuous manufacturing processes (e.g., chemical processing, food production)
- Hybrid discrete-continuous scenarios
- Assembly processes in aerospace contexts (ASPIRO project)

Each domain requires development of domain-specific semantic models and specialised transformation methodologies that address unique characteristics of these manufacturing environments.

10.5. Multi-Domain Optimization

Future work should move beyond single-cell optimization toward factory-wide or multi-site resource planning that simultaneously considers:
- Buffer management and inventory optimization
- Energy consumption and power constraints
- Multi-objective optimization (cost, time, quality, sustainability)

Federated orchestration approaches could enable collaborative planning across organizational boundaries while respecting data sovereignty requirements.

10.6. Real-Time Performance Enhancement

While current planning performance is adequate for many scenarios, highly dynamic production environments may require:
- Event-driven replanning with millisecond response times
- Distributed planning architectures for improved scalability
- Incremental planning algorithms that reuse previous solutions
- Predictive planning that anticipates future states and proactively generates contingency plans

10.6.1. Quantum Computing for Manufacturing Planning

The emergence of quantum computing presents unprecedented opportunities for solving computationally

challenging planning problems in manufacturing environments. While current PDDL-based planning with classical solvers like Fast Downward effectively addresses discrete manufacturing scenarios, quantum computing could fundamentally transform the scalability and optimization capabilities of planning systems.
Quantum optimization algorithms, could potentially provide exponential speedup for combinatorial optimization problems inherent in manufacturing planning. Complex scenarios involving multiple robots, competing resource constraints, and multi-objective optimization objectives could benefit significantly from quantum-enhanced solution methods.
Future work should explore:

- Integration of quantum annealing frameworks or quantum approximate optimization algorithms.
- Hybrid classical-quantum architectures that leverage quantum computers for specific optimization subproblems while maintaining classical planning for deterministic task sequencing
- Evaluation of quantum speedup on realistic manufacturing scenarios with abundent production resources and complex temporal dependencies

However, practical deployment remains distant due to current limitations in quantum hardware maturity and the lack of established quantum algorithms for general planning domains. Research collaboration with quantum computing providers and algorithm developers will be essential to evaluate whether quantum approaches offer genuine advantages over advanced classical.

11. CONCLUSION

The research presented here is a comprehensive semantic interoperability framework, that successfully bridges the gap between standardised data representation and automated decision-making in manufacturing systems. By combining the semantic capabilities of the Asset Administration Shell with automated planning techniques based on PDDL, the framework enables autonomous task sequencing, resource allocation, and production optimization while maintaining semantic consistency across organizational boundaries.
The key innovations include a novel AAS-to-PDDL transformation methodology that systematically converts semantic skill models into executable planning problems, an integrated architecture that supports real-time production planning and execution monitoring, and comprehensive integration capabilities that enable deployment within existing manufacturing infrastructure. The framework successfully demonstrates how semantic standardization can bridge vertical integration (shop-floor to MES) with horizontal collaboration (machine-to-machine communications).
Experimental evaluation in the Factory of the Future - eXtended (FoF-X) project demonstrates the framework's effectiveness in handling complex manufacturing scenarios while maintaining high performance and scalability. The use of heterogeneous robotic systems (KUKA LBR iiwa and Universal Robots UR10e) validates the vendor-independent nature of the approach, confirming that standardised AAS representations enable true plug-and-play integration of manufacturing assets.
The approach addresses critical Industry 4.0 challenges by providing a foundation for autonomous production planning that leverages standardised semantic representations.
By providing standardised mechanisms for semantic interoperability and automated planning, the framework supports the evolution toward truly autonomous manufacturing systems that can adapt dynamically to changing requirements and conditions. The results emphasise the transformative role of semantic standardization in enabling higher resource utilization on the shop floor.
The foundation established by this work provides a solid basis for continued advancement in semantic interoperability and automated planning for Industry 4.0 applications. Future extensions in machine learning integration, semantic reasoning, and multi-domain optimization will further enhance the framework's capabilities and industrial applicability. This work underscores the critical importance of adopting AAS as a universal language for enabling intelligent and adaptive manufacturing ecosystems.

Contact address:

Holger.Weber@dlr.de

References

[1] Malik Ghallab, Craig Knoblock, David Wilkins, Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, David Smith, Ying Sun, and Daniel Weld. Pddl - the planning domain definition language. 08 1998.

[2] H. Kagermann, W. Wahlster, and J. Helbig. Recommendations for implementing the strategic initiative industrie 4.0, 2013.

[3] Michael Schluse and Juergen Rossmann. From simulation to experimentable digital twins: Simulation-based development and operation of complex technical systems. *2016 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–6, 2016.

[4] International Electrotechnical Commission. Asset administration shell for industrial applications - part 1: Asset administration shell specification, 2023. Standard. https://webstore.iec.ch/publication/65628.

[5] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2):621–641, 2016. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2016.06.005.

[6] DIN Deutsches Institut für Normung. Reference architecture model industrie 4.0 (rami4.0), apr 2016. Specification. https://www.dinmedia.de/de/technische-regel/din-spec-91345/250940128.

[7] Yuchen Xia, Zhewen Xiao, Nasser Jazdi, and Michael Weyrich. Generation of asset administration shell with large language model agents: Towards semantic interoperability in digital twins in the context of industry 4.0, 03 2024. DOI: 10.48550/arXiv.2403.17209.

[8] Quanbo Lu, Xinqi Shen, Jiehan Zhou, and Mei Li. Mbd-enhanced asset administration shell for generic production line design. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(9):5593–5605, 2024. DOI: 10.1109/TSMC.2024.3408296.

[9] Boris Otto, Jakob Seidelmann, Julian Schmelting, Oliver Sauer, Hartmut Rauen, and Gunther Koschnick.

Manufacturing-x data space study: Architecture, basic services and organization, taking into account the specific features of the equipment industry, 2023. Study commissioned by VDMA and ZVEI. https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Pressebereich/2023_059_Manufacturing_X/Manufacturing-X_Data_Space_Study_ZVEI-VDMA-Fraunhofer.pdf.

[10] Eclipse Foundation. Semantic aspect meta model (samm) specification. https://eclipse-esmf.github.io/, sep 2025. Version Snapshot, accessed on 12.09.2025.

[11] Akira Nakamura and Kensuke Harada. Evaluation standard of error recovery planning focused on revival process from failures in robotic manufacturing plants. *Proceedings of International Conference on Artificial Life and Robotics*, 29:399–404, 02 2024. DOI: 10.5954/ICAROB.2024.OS14-1.

[12] Jyotsna Singh, Lakshay Mundeja, Armando Walter Colombo, and Bilal Ahmad. Digitalization of the components of a mini factory with the implementation of rami 4.0 asset administration shell. In *2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 1–8, 2024. DOI: 10.1109/ICPS59941.2024.10640019.

[13] Marco Aiello and Ilche Georgievski. Introduction to ai planning, 2024. https://arxiv.org/abs/2412.11642.

[14] M. Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, July 2006. ISSN: 1076-9757. DOI: 10.1613/jair.1705.

[15] Christian Muise, Florian Pommerening, Jendrik Seipp, and Michael Katz. Planutils: Bringing planning to the masses. In *ICAPS 2022 System Demonstrations*, 2022.

[16] Michael Jacoby, Friedrich Volz, Christian Weißenbacher, and Jens Müller. Fa³st service – an open source implementation of the reactive asset administration shell. 09 2022. DOI: 10.1109/ETFA52439.2022.9921584.

[17] Inc. Google. Gson – a java serialization/deserialization library to convert java objects into json and back, 2024. https://github.com/google/gson.

[18] Eclipse Foundation. Eclipse dataspace connector: Trusted data sharing with sovereignty. https://newsroom.eclipse.org/eclipse-newsletter/2021/october/eclipse-dataspace-connector-trusted-data-sharing-sovereignty, October 2021. Accessed on 12.09.2025.