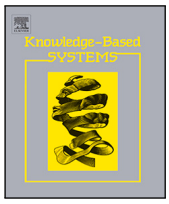
Knowledge-Based Systems 318 (2025) 113484

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



# Development of a runtime-condition model for proactive intelligent products using knowledge graphs and embedding

Fan Mo [a,b,c,*], Hamood Ur Rehman [a,f], Miriam Ugarte [d], Angela Carrera-Rivera [d], Nathaly Rea Minango [e], Fabio Marco Monetti [e], Antonio Maffei [e], Jack C. Chaplin [a]

[a] Institute for Advanced Manufacturing, University of Nottingham, Nottingham, NG8 1BB, United Kingdom
[b] Department of Computer Science, University of Oxford, Oxford, OX1 3QG, United Kingdom
[c] Centre for Human-inspired Artificial Intelligence, University of Cambridge, Cambridge, CB2 1SB, United Kingdom
[d] Mondragon Unibertsitatea, Arrasate-Mondragon, 20500, Spain
[e] Department of Production Engineering, KTH Royal Institute of Technology, Stockholm, 114 28, Sweden
[f] TQC Automation Ltd., Nottingham, NG3 2NJ, United Kingdom

ARTICLE INFO

*Keywords:*
Runtime condition
Data model
Intelligent system
Knowledge graph

ABSTRACT

Modern manufacturing processes' increasing complexity and variability demand advanced systems capable of real-time monitoring, adaptability, and data-driven decision-making. This paper introduces a novel runtime condition model to enhance interoperability, data integration, and decision support within intelligent manufacturing environments. The model encapsulates key manufacturing elements, including asset management, relationships, key performance indicators (KPIs), capabilities, data structures, constraints, and configurations. A key innovation is the integration of a knowledge graph enriched with embedding techniques, enabling the inference of missing relationships, dynamic reasoning, and predictive analytics.

The proposed model was validated through a case study conducted in collaboration with TQC Automation Ltd., using their MicroApplication Leak Test System (MALT). A dataset of over 9,000 unique test configurations demonstrated the model's capabilities in representing runtime conditions, managing operational parameters, and optimising test configurations. The enriched knowledge graph facilitated advanced analyses, providing actionable insights into test outcomes and enabling proactive decision-making.

Empirical results showcase the model's ability to harmonise diverse data sources, infer missing connections, and improve runtime adaptability. This study highlights the potential of combining runtime modelling with knowledge graphs to address the challenges of modern manufacturing. Future research will explore the model's application to additional domains, integration with larger datasets, and the use of machine learning for enhanced predictive capabilities.

## 1. Introduction

The increasing complexity and variability of modern manufacturing processes have created a critical demand for advanced systems capable of real-time monitoring, adaptability, and data-driven decision-making [1]. Industries such as aerospace, automotive, and healthcare now require sophisticated manufacturing solutions that integrate diverse operational data, optimise key performance indicators (KPIs), and address decision-making constraints within a unified framework. Meeting these challenges necessitates intelligent manufacturing environments capable of harmonising heterogeneous data sources, inferring missing relationships, and supporting predictive analytics.

Traditional manufacturing systems often fall short of meeting these requirements. They struggle to integrate fragmented data sources, rely on static models that fail to adapt to real-time changes, and lack robust mechanisms for handling imperfect or noisy data [2,3]. These limitations are particularly pronounced in dynamic and complex environments, where swift and accurate decisions are critical to minimising downtime, ensuring operational efficiency, and reducing waste. As a result, there is a growing need for innovative solutions that address these shortcomings by enabling seamless data integration, dynamic relationship management, and predictive decision support.

One of the key novelties in this work is the creation of a dynamic runtime condition model that embeds operational data into a knowledge graph, thereby enabling real-time inference and completion of missing relationships. Unlike static approaches, our method continuously adapts to changing manufacturing conditions, mitigating

* Corresponding author at: Centre for Human-inspired Artificial Intelligence, University of Cambridge, Cambridge, CB2 1SB, United Kingdom.
*E-mail address:* fan.mo@kellogg.ox.ac.uk (F. Mo).

https://doi.org/10.1016/j.knosys.2025.113484
Received 9 June 2024; Received in revised form 17 March 2025; Accepted 2 April 2025
Available online 15 April 2025

risks proactively and delivering robust decision support—even in the presence of imperfect or fragmented information.

To address these challenges, this paper introduces a novel runtime condition model that integrates knowledge graph embeddings to enhance data harmonisation, infer missing relationships, and enable dynamic predictive analytics. Unlike traditional static runtime models, the proposed approach dynamically represents and completes system relationships using embedding techniques, offering scalability and adaptability to real-world manufacturing environments. This framework effectively addresses key challenges, including data heterogeneity, imperfect information, and the increasing complexity of decision-making.

The key contributions of this paper are as follows:

(1) We propose a runtime condition model that captures the real-time operational state of manufacturing systems, integrating key components such as assets, constraints, configurations, capabilities, KPIs, and relationships. This dynamic framework is designed to adapt to real-time changes, ensuring continuous monitoring and decision-making support.
(2) The paper introduces a knowledge graph-based representation of runtime conditions, enabling the structuring and analysis of complex interdependencies in manufacturing systems. The use of graph embedding techniques enhances the graph's ability to infer missing relationships, enabling predictive analytics and dynamic reasoning.
(3) The proposed model is validated using the MicroApplication Leak Test System (MALT) in collaboration with TQC Automation Ltd. The case study demonstrates the model's ability to manage complex runtime data, optimise test configurations, and support advanced predictive decision-making.

By focusing on these critical areas, the paper bridges the gap between theoretical advancements and practical applications. A distinctive feature of the proposed model is its use of knowledge graph embeddings, which transform operational data into a dynamic framework capable of real-time updates, predictive reasoning, and fault prediction. This capability ensures that the model not only represents the current state of the manufacturing system but also anticipates and mitigates potential risks before they escalate into costly downtimes.

While existing runtime condition models primarily rely on predefined thresholds or rule-based decision-making, our approach introduces a dynamic, adaptive framework that uses runtime data, knowledge graph embeddings, and machine learning-driven analytics. Unlike previous methods, which often struggle with scalability and real-time fault prediction, our model continuously learns from operational data, enabling proactive fault detection and enhanced decision-making. This study systematically validates the effectiveness of this approach through an industrial use case, demonstrating its superiority over static runtime monitoring methods.

To validate the efficacy of the model, a case study was conducted with TQC Automation Ltd. The case study involves the MALT system, incorporating over 9,000 unique test configurations, which highlights the model's ability to manage diverse data, address imperfect information, and provide actionable insights for operational optimisation.

The remainder of this paper is structured as follows: Section 2 reviews related research on runtime condition modelling and manufacturing data standards. Section 3 outlines the design and implementation of the proposed runtime condition model and its integration with knowledge graphs. Section 4 presents the case study and empirical validation results, focusing on real-world application scenarios. Finally, Section 5 summarises the findings and discusses future research directions.

## 2. Related work

### 2.1. Runtime condition in manufacturing

The growing demand for efficient and responsive manufacturing systems in sectors such as aerospace, defence, automotive, and healthcare underscores the importance of advanced monitoring systems for manufacturing assets. Runtime condition monitoring enables real-time responses in manufacturing environments and is increasingly critical due to high maintenance and capital costs. Effective condition monitoring not only minimises unplanned downtime but also enhances machine health and performance, reduces the production of defective parts, and facilitates the implementation of data-driven maintenance programs. In addition, it supports automation by providing real-time condition data, thereby contributing to overall operational efficiency and system reliability.

Runtime refers to the periods during which manufacturing equipment is operational, excluding downtime. Monitoring systems are essential for maintaining flexibility, sustainability, and minimal human intervention in modern manufacturing [4]. The primary role of runtime condition monitoring is to optimise performance by capturing and analysing diverse data, including energy consumption, tool wear, production states, and inventory levels [5].

Two prominent approaches to condition monitoring include model-based and data-driven methods. Model-based techniques rely on mathematical or semantic models and require expert knowledge, while data-driven methods leverage machine learning algorithms applied to vast datasets. However, data-driven methods often face challenges in interpretability [6–8].

Effective runtime condition monitoring systems must address several key challenges. They need to manage the heterogeneity and mobility of data by handling diverse data types while ensuring coherence in dynamic environments. At the same time, capturing and aligning the dependencies between these various data types is crucial for robust data management. The system must also be capable of promptly capturing and predicting operational states, which is essential for timely decision-making. Furthermore, it should efficiently manage noisy or incomplete datasets to maintain reliability. In addition, supporting logical reasoning by analysing monitored conditions helps derive actionable insights, while using modelling formalisms that translate complex real-world concepts into understandable and actionable models is vital. Finally, these systems must provision resources efficiently to handle large volumes of data and complex models without compromising overall performance.

Runtime condition monitoring is defined as the process of capturing the internal condition or status of operational assets, including products and manufacturing equipment, to ensure the proper execution of manufacturing processes. It enables enhanced operations by identifying and predicting changes, violations, or malfunctions.

Smart and autonomous manufacturing systems exhibit *self-x* behaviours, such as self-organisation, self-configuration, self-learning, and self-diagnosis. These capabilities require a robust data model characterised by modularity, interoperability, scalability, and robustness [9].

### 2.2. Existing runtime standards in manufacturing

Building on the foundations of runtime data models, several manufacturing standards have been developed to provide structured approaches for asset digitalisation and operational performance. This section reviews key standards, highlighting their strengths and limitations.

MTConnect is an open-source standard designed to facilitate data exchange in manufacturing operations, particularly for CNC machines. It promotes plug-and-play environments by enabling data acquisition across equipment, systems, and software [10]. Its semantic data models encompass a hierarchical representation of equipment components

(Devices Information Model), a categorisation of real-time equipment data such as EVENT, SAMPLE, and CONDITION (Stream Information Model), a representation of non-device entities such as cutting tools (Asset Information Model), and a framework for managing error data (Error Information Model).

While MTConnect offers robust capabilities in data acquisition and semantic modelling, it faces several limitations. It is primarily designed for CNC machines, which may not fully support the diverse range of equipment in modern manufacturing environments. As systems grow in complexity, scaling MTConnect often requires significant modifications, and full interoperability with diverse and legacy systems frequently necessitates custom adapters or middleware. Moreover, keeping pace with emerging technologies requires continual updates, which can be resource-intensive.

OPC/UA provides an open, extensible framework for real-time data exchange between devices and systems using Internet protocols. Its object-oriented structures support dynamic modelling, enabling digital twins and advanced analytics [11].

However, challenges associated with OPC/UA include its complex implementation, as its comprehensive functionality often results in time-consuming deployment processes that require specialised expertise. Additionally, OPC/UA servers and clients can be resource-intensive, potentially impacting performance in large-scale deployments. Integration challenges may arise, particularly with legacy systems that often require additional interfaces or conversion layers, and maintaining performance and reliability at scale remains challenging.

Developed under Industrie 4.0, AAS links assets through standardised communication interfaces. Its structure includes static files (Type 1) where data is stored in formats like XML or JSON, runtime instances (Type 2) that combine static and interactive components, and proactive systems (Type 3) that enable independent communication and negotiation.

While Type 3 implementations represent the most advanced capabilities of AAS, they are rare in practice. Limitations of the AAS framework include the complexity of establishing and maintaining AAS—especially Type 3 instances—which requires significant technical expertise. Additionally, adoption remains limited due to the scarcity of tools and resources, and integrating AAS with existing systems often demands custom development. Variability in implementations can also hinder seamless interoperability due to standardisation gaps.

Standards such as ISO 23247 and ISO 22400 emphasise digital twins and performance assessment by categorising manufacturing data into static and dynamic types, thereby aiding simulation and decision-making processes.

However, these standards present limitations as well. They are often tailored to specific aspects of manufacturing, which limits their versatility across varied processes. Their rigid structured frameworks may require extensive modifications to adapt to evolving needs, and significant costs are associated with compliance, training, and system upgrades. Furthermore, updating these standards to reflect technological advancements is typically a lengthy process that often lags behind industry needs. In summary, existing runtime standards such as MTConnect, OPC/UA, AAS, and ISO frameworks provide structured approaches to manufacturing digitalisation and operational performance. However, they face limitations in scope, scalability, interoperability, and adaptability. These challenges highlight the need for dynamic and flexible approaches, such as the knowledge graph-based model proposed in this work, which aims to enable real-time data integration, dynamic relationship management, and advanced predictive analytics in manufacturing environments.

### 2.3. Knowledge graphs in the manufacturing domain

The adoption of knowledge graphs has proven pivotal in the manufacturing sector, offering robust tools for structuring and analysing complex and diverse datasets. A knowledge graph organises information into triples, comprising a head entity, a relationship, and a tail entity, such as *(Robot, operatesOn, AssemblyLine)* [12]. This structure facilitates the effective management of relationships and enables advanced data analysis and inference [13].

One key advantage of knowledge graphs over traditional databases is their inherent flexibility. New entities and relationships can be seamlessly incorporated without modifying the existing schema [14]. This adaptability makes knowledge graphs especially useful for addressing complex queries that involve interconnected data points, such as "What processes are required to assemble a specific product?"

A knowledge graph typically consists of two main layers: the schema layer and the entity layer [15]. The schema layer defines general concepts, properties, and relationships, while the entity layer represents specific instances of these concepts. For example, in the entity layer, *(Robot, operatesOn, AssemblyLine)* reflects a specific relationship, whereas the schema layer abstracts this to *(Equipment, operatesOn, Process)*.

In manufacturing, knowledge graphs have been used to optimise resource allocation [16], enhance decision-making processes [17], and streamline production workflows [18]. For instance, they enable real-time adjustments to production plans by integrating data from various stages of the manufacturing process. This view improves operational efficiency and supports the lifecycle management of products.

Additionally, knowledge graphs enhance data interoperability, enabling the integration of diverse datasets across formats and stages [19]. By providing a unified framework, they facilitate advanced analytics, such as identifying bottlenecks in production or predicting maintenance needs.

Despite their advantages, several challenges limit the adoption of knowledge graphs in manufacturing. One major issue is data integration complexity, as combining data from disparate sources with varying quality and formats can be time-consuming [19]. Additionally, building and maintaining a knowledge graph demands specialised technical skills, which can act as a barrier for some organisations [20]. The lack of standardisation further complicates matters, with variability in knowledge graph technologies hindering smooth integration with existing systems and workflows [21]. Moreover, many existing knowledge graphs in manufacturing are static, focusing solely on structured data and lacking support for real-time updates or dynamic changes [19]. Unlike traditional static knowledge graphs, which are limited to representing pre-defined relationships, dynamic knowledge graphs incorporate continuous enrichment techniques. For example, embedding models such as TransE transform entities and relationships into continuous vector spaces, enabling advanced tasks like link prediction and entity classification [22]. These embeddings capture latent patterns and semantic meanings, supporting sophisticated data analysis.

Knowledge graph completion, another key advancement, predicts missing triples in the graph, thereby improving its completeness and utility [23]. These techniques significantly enhance the capability of knowledge graphs to provide actionable insights and support real-time decision-making.

Building on these advancements, our proposed knowledge graph dynamically enriches its structure using embedding techniques. This approach enables the graph to seamlessly integrate new data in real-time, infer missing relationships to maintain data completeness, and support advanced predictive analytics by capturing latent and complex relationships.

For instance, applying TransE embeddings enables the discovery of patterns in operational data that traditional static models cannot identify. This facilitates tasks such as predicting the impact of equipment failure on production output or identifying optimal configurations for resource allocation. Unlike prior works [19], which focus on static representations, our approach offers a dynamic and scalable framework that adapts to the evolving needs of modern manufacturing environments.

Knowledge graphs provide a powerful framework for organising and analysing complex data in the manufacturing sector, offering significant advantages in reasoning and decision-making. While challenges such as data integration, expertise requirements, and standardisation remain, advancements in embedding and completion techniques are expanding their potential applications. The dynamic nature of the proposed knowledge graph sets it apart from static models, making it a valuable tool for modern manufacturing operations.

## 3. Methodology

In contrast to studies that focus solely on implementation, our work advances a comprehensive data-driven framework that integrates semantic, predictive, and adaptive components into a unified architecture. By combining knowledge graph embeddings with a dynamically evolving runtime model, we develop a scalable approach that not only meets local industrial needs but also offers broader theoretical insights into real-time condition monitoring. Accordingly, this paper proposes a runtime-condition model specifically designed to address key challenges in managing proactive, intelligent products, as detailed in Sections Section 2.1, 2.2, and 2.3. Our methodology effectively tackles the issues of handling heterogeneous and dynamic data, managing imperfect information, and supporting complex decision-making processes. Moreover, we present a complete process for generating a knowledge graph based on embedding techniques, which serves as the cornerstone for enhancing system monitoring and decision support.

Initially, the runtime model is defined. Subsequently, the initial schema layer of the knowledge graph is established. Following this, the data acquisition process is conducted based on various dataset formats, including customer documents, datasets, and sensor data. This leads to the generation of the entity layer of the knowledge graph and the subsequent updating of the schema layer. Then, the initial knowledge graph is constructed. Furthermore, through the application of graph embedding techniques, the knowledge graph completion process is carried out. Once completed, the knowledge graph is stored. Finally, the knowledge graph can be applied in various scenarios, such as semantic search and decision support.

### 3.1. Definition and contribution of the runtime condition model

The proposed *runtime condition model* is a framework designed to dynamically capture the real-time operational state of a manufacturing system. It integrates essential components such as assets, capabilities, configurations, constraints, key performance indicators (KPIs), data, and relationships, providing a holistic view of the manufacturing environment. Unlike static models, this runtime condition model continuously adapts and responds to real-time changes, ensuring continuous monitoring and optimal decision-making.

A key feature of the model is its dynamic representation, allowing it to continuously update the state of manufacturing assets and processes, enabling real-time monitoring and adaptability. Additionally, it enhances data integration by harmonising heterogeneous data sources, ensuring seamless data flow and consistency across the system. By incorporating KPIs and constraints, the model improves decision support, facilitating complex decision-making processes to optimise performance and efficiency. Another critical capability is its robust handling of imperfect data, incorporating mechanisms to manage incomplete or noisy data, thereby ensuring reliability and accuracy in operational insights. Finally, the model is designed with scalability and extensibility in mind, making it modular and adaptable to accommodate the growth of manufacturing systems and evolving technological advancements. This comprehensive approach ensures that the runtime condition model remains a crucial tool for real-time system monitoring and intelligent decision-making in manufacturing environments.

The runtime condition model not only encodes domain-specific constraints and capabilities but also establishes a semantic foundation for embedding-based reasoning. Consequently, it serves as a conceptual bridge between classic manufacturing data standards and modern machine learning techniques, illustrating how semantic interoperability can be leveraged for predictive analytics in large-scale industrial settings.

The models and standards discussed in the previous sections address various individual components of production system needs. Yet, they do not comprehensively encompass all aspects, particularly in the context of runtime conditions. There is a clear necessity to expand further or adapt these standards and models to meet the specific demands of runtime conditions adequately. Consequently, a model must be developed that integrates aspects of all these standards and models, contextualised specifically for runtime conditions, and presents a systematic approach to represent the runtime condition for production systems at any given instance.

A production system model conducive to runtime conditions is conceptualised through several key components: configuration, capability, constraints, key performance indicators (KPIs), data, configuration and relationships.

These dependencies are crucial for runtime conditions in production systems as they capture all necessary information to accurately represent both the internal state of the production system and the external conditions of the production environment. The status of the manufacturing assets, such as equipment and products, is meticulously documented to facilitate monitoring and control in alignment with production execution. This representation of runtime conditions is essential to encapsulate the manufacturing operations comprehensively. Moreover, when combined with enabling technologies, it ensures that the manufacturing operations are conducted correctly, with proper identification, monitoring, and controlled behavioural changes.

The individual components that fulfil the requirements are detailed as follows:

- **Configuration:** Represents the internal state of the production system, assuming that the system comprises entities with multiple configurations and settings under specific constraints.
- **Relationship:** Captures the interconnections within the production system and its relationship with the environment. It includes concepts such as dependence, composition, matching, and compatibility, essential for maintaining system integrity.
- **Capability:** Defines the production system's ability to perform actions, distinguishing between cyber and physical capabilities.
- **Data:** Represents the data crucial for runtime conditions, encompassing aspects like data type, identification, value, and unit of measure, and also considers data aggregation from multiple sources.
- **Constraints:** Outlines the limitations on the entities within the production system, which can be internal or external.
- **KPIs:** Addresses the goals governing the production system, focusing on entity-level attributes such as cost, time, and quality, among others.

The runtime condition model demonstrates its dynamism by continuously capturing and adapting to real-time changes within the manufacturing environment. This is achieved through real-time monitoring, which continuously tracks the status of assets and processes to allow for immediate detection and response to anomalies; adaptive configurations that enable on-the-fly adjustments to asset settings and process parameters based on current operational data; predictive analytics that leverage both historical and current data to forecast future states, maintenance needs, and potential disruptions; and interoperability that seamlessly integrates with existing manufacturing systems and technologies to enhance overall responsiveness and efficiency. In turn, the model effectively supports manufacturing by enhancing operational efficiency through providing a comprehensive and up-to-date view of the production system that enables optimised resource allocation and

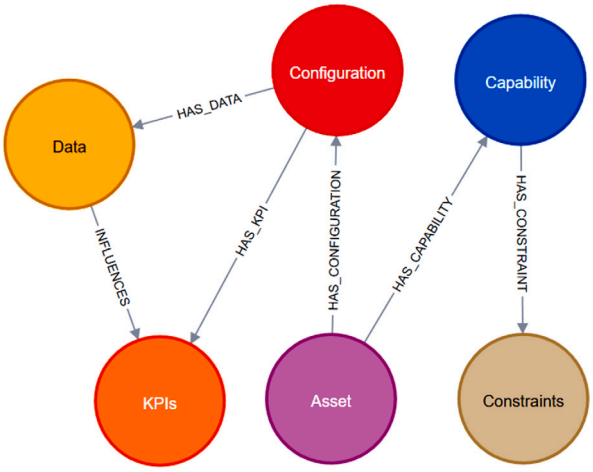



**Fig. 1.** A generalised representation of runtime conditions for a production system.

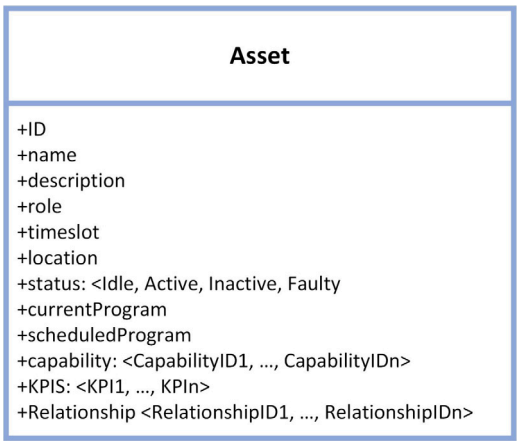



**Fig. 2.** UML representation for Asset modelling.

process improvements, improving quality control by ensuring that production processes adhere to quality standards and reducing defects and waste, facilitating scalability with its modular and extensible design that accommodates growth and technological advancements, and supporting decision-making by integrating KPIs and constraints to provide valuable insights and actionable data for strategic and operational decisions.

These features collectively ensure that the runtime condition model not only captures the current state of the manufacturing system but also actively contributes to its continuous improvement and adaptability, thereby providing substantial evidence and justification for its effectiveness in supporting modern manufacturing environments.

Traditional SCADA-based monitoring systems often require manual rule-setting, which limits adaptability in rapidly changing production environments. Our approach overcomes this limitation by introducing a framework that dynamically updates its knowledge representation. This significantly enhances its ability to optimise asset utilisation, and support real-time decision-making, which is not feasible with existing static models.

To our knowledge, this is the first study that systematically evaluates multiple knowledge graph embedding models for runtime condition modelling, providing empirical insights into their effectiveness in an industrial setting.

### 3.2. Details of the runtime condition model

This section offers a robust and nuanced understanding of the runtime condition data model by integrating visual representations with detailed component analysis and UML modelling. This approach not only meets the requirements set forth in the earlier sections but also facilitates a deeper insight into the system's operational framework, enhancing both its theoretical understanding and practical applicability. Furthermore, this runtime condition model will be utilised later for constructing the initial schema layer of the knowledge graph, thereby establishing a foundational structure for further knowledge integration and application (see Fig. 1).

#### 3.2.1. Asset

The *Asset* represents a crucial entity within the runtime data model of a manufacturing system, functioning as either a physical or logical agent. This entity is characterised by a set of defined attributes that articulate its roles and functionalities within the system.

The asset includes identification details such as "Name", "ID", and a "Description" that provides an overview of the asset. Its "Role" delineates its specific functions or responsibilities within the manufacturing process. The "Timeslot" specifies when the asset is active, and the "Location" indicates its position within the facility.

Operational details are captured in the "CurrentProgram" and "ScheduledProgram", which outline the tasks the asset is currently undertaking and scheduled to undertake, respectively. The "Status" of the asset is indicated as either Idle, Active, Inactive, or Faulty, which informs its operational readiness and health.

The asset is linked to pertinent datasets via the "Data" attribute, to its capabilities via the "Capability" attribute, and to studies enhancing those capabilities through the "CapabilityStudy". Performance metrics are tracked through "KPIs", and its interactions with other entities are defined under "Relationship".

The UML diagram in Fig. 2 visually represents these attributes and their interrelationships, providing a clear schematic for understanding the asset's integration within the broader system framework.

#### 3.2.2. Relationship

The *relationship* modelling represents the association information between two or more *Assets* that might function independently but can perform a manufacturing process or operation together. This association is vital for defining how assets interact and collaborate within the production environment.

Each relationship is uniquely identified by a "RelationshipId" and is further described by attributes such as "name" and "description", which provide a clear understanding of its nature and purpose. The attribute "type" is particularly important as it describes the hierarchy of the relationship with the related asset, which is identified by the attribute "associatedID". This hierarchical structuring is essential for mapping dependencies and operational sequences among assets.

Additionally, the field "isRequired" specifies whether the relationship is critical for the process or operation in which the assets are engaged. This information is crucial for prioritising relationships and ensuring that essential interactions are maintained to support uninterrupted manufacturing operations.

Fig. 3 illustrates the UML representation of these relationships, providing a standardised visual interpretation of how assets are interconnected within the system.

#### 3.2.3. KPIs

Key Performance Indicators (*KPIs*) are metrics utilised to quantify the success of a manufacturing process step or action. They serve as a quantifiable measurement that reflects the improvement or deterioration in the performance of an activity critical to the success of a process. KPIs can be set with specific targets, a range of acceptable values, or both to gauge how effectively a business is achieving its objectives.

According to [24], KPIs must adhere to several key requirements to ensure their validity in engineering:

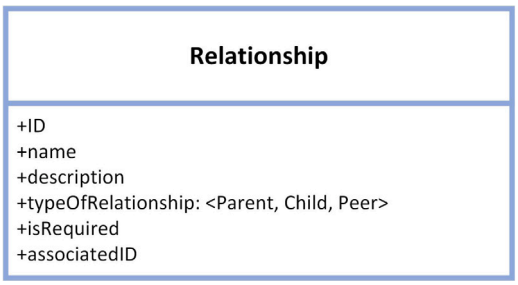



**Fig. 3.** UML representation for Relationship modelling.

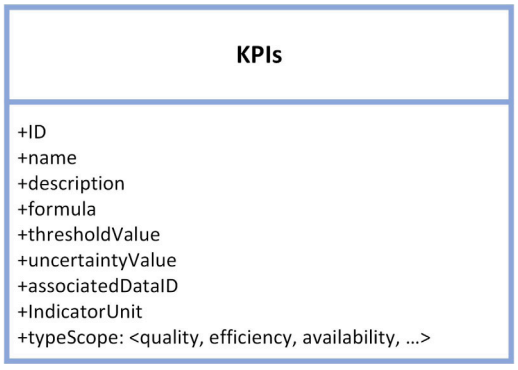



**Fig. 4.** Process modelling for Runtime condition.

(1) KPIs need to be quantifiable, meaning they must be measurable in quantitative terms.
(2) They should be sensitive to change, indicating that any variation in the manufacturing process should be directly observable in the KPI values.
(3) Reliability is crucial; the calculation of KPIs should be free from errors, ensuring their accuracy.
(4) Finally, KPIs should be efficient: their calculation must be cost-effective and quicker than the manufacturing processes they are measuring.

Fig. 4 illustrates the process modelling for the Runtime condition, which incorporates the structured evaluation and application of KPIs within the operational framework.

### 3.2.4. Capability

The concept of *capability* within a manufacturing system refers to the specific features, attributes, or skills of an asset. Each capability is uniquely identified and characterised by a *name* and a *description*, which precisely define the particular feature or skill of the asset. Capabilities can be inherent to *machines* or integral to *processes*, and they are often subject to a set of limitations or constraints, as outlined in Section 3.2.6.

Capabilities are crucial in evaluating and optimising the operational effectiveness of manufacturing systems. They can be assessed using capability indices, which provide a standardised measure of their performance. Details of this assessment methodology are elaborated in [25]. These indices help determine how effectively a capability meets the operational demands and challenges of the manufacturing environment.

To provide a holistic view, the data model not only includes detailed descriptions of each capability but also establishes links to the relevant constraints. This integration ensures that all potential limiting factors are considered when deploying or optimising these capabilities, enabling informed decision-making in real-world scenarios.

Fig. 5 illustrates the UML representation of capability modelling, providing a visual depiction of how these capabilities are structured and related to other components within the asset framework.

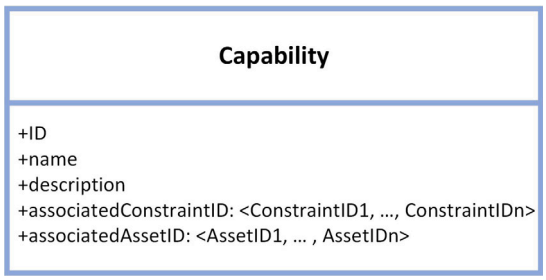



**Fig. 5.** UML representation for capability modelling.

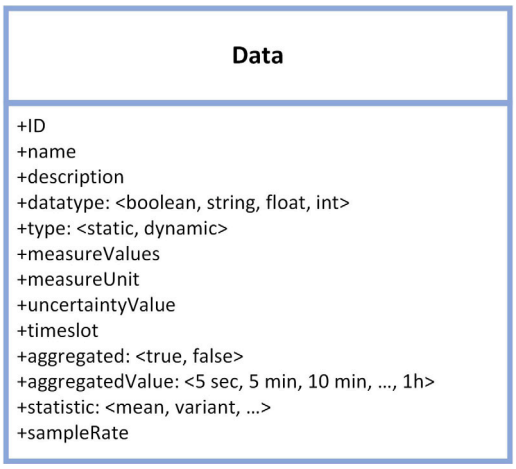



**Fig. 6.** UML representation for data modelling.

### 3.2.5. Data

The *data* object is essential for defining the information collected from various levels of the manufacturing floor via sensors in a clear and detailed manner. This object is characterised by several attributes that facilitate a understanding and utilisation of the data within the manufacturing system.

Key attributes include the *ID*, *name*, and *description*, which provide basic identification and details about the data. The *datatype* can vary among boolean, string, float, and int, reflecting the diverse nature of data that can be captured. The *type* of data, whether static or dynamic, determines how it is handled within the system.

Furthermore, each piece of data is associated with a *valueMeasure* and a *unitMeasure*, which describe the magnitude and the unit of measurement, respectively. The *uncertaintyValue* is also crucial as it helps assess the data's reliability and identify potential sources of error.

For enhanced data analysis, attributes such as *aggregatedData*, *aggregatedValue*, and *statistics* (like mean or variance) are vital. These allow for the consolidation and statistical analysis of data over specified intervals, aiding in the extraction of meaningful information. The *sampleRate* specifies how frequently data samples are collected, which is significant for both real-time monitoring and historical data analysis.

The decision to treat data as a separate node rather than merely a property of another node in the system provides several significant advantages. It enhances scalability, supports the mapping of complex relationships, improves query performance, maintains data integrity, facilitates dynamic data handling, and simplifies data governance. Each of these factors is crucial for a robust manufacturing data architecture, allowing the system to adapt efficiently to evolving technological and operational demands.

Fig. 6 displays the UML representation of the data object, illustrating the relationships and structure of these attributes within the data model.

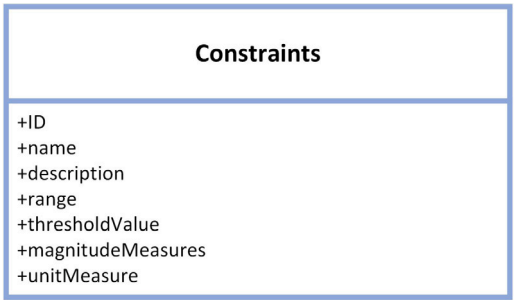



**Fig. 7.** Constraint modelling for Runtime Condition.

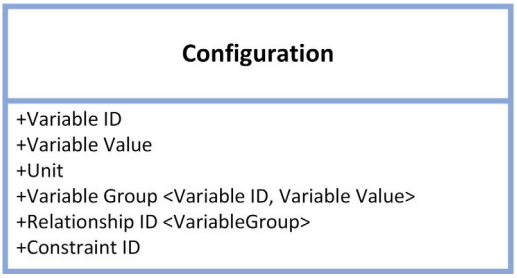



**Fig. 8.** Configuration modelling for Runtime Condition.

#### 3.2.6. Constraints

*Constraints* modelling for runtime conditions serves as an essential extension of the "Constraints ID" included in the capability model. It outlines the limits within which the production system's capabilities can effectively operate, defining the operational boundaries necessary to ensure the system functions within its design specifications. Each constraint is identified by a "Constraints ID" consistent with the one used in the capability representation and is detailed with a "Name" and a "Description", which aid in understanding the nature and implications of the constraint. Additionally, the "Range" and its corresponding "Unit" establish the measurable parameters, setting clear and quantifiable operational limits for system activities.

The *Threshold Value* represents the optimal level or condition that the constraint should maintain. Any variations from this threshold require the activation of control mechanisms designed to correct or compensate, ensuring the system continues to operate within optimal parameters. To provide further clarity in defining constraints, two key attributes are used: *magnitudeMeasures* and *unitMeasure*. The *magnitudeMeasures* attribute specifies the quantitative extent or boundary of the constraint, representing the measurable value that ensures the system remains within acceptable conditions. For instance, in the case of a temperature constraint, the magnitude might be *100*. The *unitMeasure* attribute defines the unit associated with the magnitude, ensuring standardisation and preventing misinterpretation. For the same temperature constraint, the unit might be $^{\circ}$ *C* (degrees Celsius). These attributes collectively provide a structured approach to defining constraints, ensuring precision and consistency in system operations.

This methodical approach to defining and integrating constraints is crucial for the stable and efficient functioning of production systems, providing a clear framework for monitoring and managing system capabilities. Fig. 7 illustrates the general syntax and modelling of constraints, offering a visual guide to their role and implementation in the runtime data model.

#### 3.2.7. Configuration

Configuration modelling is a fundamental aspect of the runtime data model, focusing on how "Variables", their "Relationships", and the "Constraints" that affect these variables are structured and managed. This model provides a systematic approach to capturing and maintaining the production system's operational parameters.

The configuration is primarily defined by "Variable ID" and its corresponding "Variable Value", which together establish the specific settings or conditions under which the production system operates. These variables are grouped into "Variable Groups", linking each ID to its respective value. This grouping facilitates the management and application of settings across the system while ensuring scalability and ease of organisation.

The inclusion of "Variable Groups" does not inherently lead to redundancy, as it serves to structure variables with similar attributes or interdependencies. By encapsulating related variables, this grouping reduces the complexity of managing large-scale configurations and ensures clear mapping of relationships. However, careful implementation is required to avoid overlap in relationships or redundant data entries.

"Relationship ID" further extends the utility of the "Variable Group" by explicitly delineating interactions and dependencies among variables within the production process. This prevents redundancy by centralising the definition of interdependencies and ensuring consistency across the system.

The "Constraint ID" establishes a direct link between the configuration and the broader constraints model. This relationship ensures that all variable settings comply with predefined operational boundaries. Each configuration is inherently tied to one or more constraints, guaranteeing that system parameters remain within safe and optimal ranges. The direct relationship enhances the integrity and reliability of the runtime data model by integrating constraints into the configuration structure.

This structured configuration model clarifies how each system component is configured and ensures consistency and compliance with the designed operational limits. By integrating variables, relationships, and constraints, the model enhances system reliability, scalability, and performance. Fig. 8 illustrates the general syntax and structure of the configuration model, providing a visual representation of how variables are organised and related within the system.

### 3.3. Generation of knowledge graph

The methodology for constructing the knowledge graph using the runtime condition model involves several key steps. This subsection will describe the process and how knowledge graph embeddings are utilised to complete the knowledge graph, enhancing its utility and effectiveness for further applications.

#### 3.3.1. Constructing the initial schema layer

The initial schema layer construction is based on the proposed runtime condition model. Tools such as Protégé or other ontology development software are utilised to create this schema layer. The schema layer defines the classes, properties, and relationships that will structure the knowledge graph, ensuring consistency and alignment with the runtime condition model.

#### 3.3.2. Entity layer construction

To construct the entity layer from the runtime condition model, we first identify and extract all entities—including assets, capabilities, data points, constraints, configurations, KPIs, and relationships—from the model. Next, we convert the relationships between these entities into triples of the form $(h, r, t)$, where $h$ (head) and $t$ (tail) are entities and $r$ represents the relationship between them. The data used to build this layer comes from various sources, such as customer documents—processed through named entity recognition (NER), relation extraction, and attribute extraction—datasets that are analysed by using time/frequency analysis and pattern recognition techniques, and sensor data obtained through data acquisition and signal processing. First, we create the initial entity layer by aggregating all the identified entities from these diverse data sources.

### 3.3.3. Initial knowledge graph construction

Next, we build the initial knowledge graph by integrating the schema layer derived from the runtime condition model with the generated entity layer. This integration is achieved by first constructing the graph, in which all entities are added as nodes and all relationships as edges—each node representing an entity and each edge representing the connection between two entities. Subsequently, we ensure that the knowledge graph comprehensively and accurately reflects the relationships and constraints defined in the runtime condition model by seamlessly integrating the schema layer with the entity layer.

This process ensures that the knowledge graph accurately and comprehensively represents the operational framework defined by the runtime condition model, facilitating deeper insights and practical applicability.

### 3.3.4. Knowledge graph completion using embeddings

To enhance the knowledge graph, various embedding techniques can be used, such as knowledge graph embeddings and graph neural networks. In this paper, we employ the TransE model as an example to demonstrate the process.

(1) **Embedding Entities and Relations**: TransE represents each entity $e$ and each relation $r$ as a vector in a continuous vector space. The primary idea behind TransE is to interpret relationships as translations in this space. Specifically, if a relationship $r$ holds between entities $h$ and $t$, then the embedding of $t$ should be close to the embedding of $h$ plus the embedding of $r$:

$$\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t} \tag{1}$$

(2) **Training the Embedding Model**: The embeddings are learned by minimising a margin-based ranking loss. This loss function ensures that the correct triples $(h, r, t)$ are closer together in the embedding space than the incorrect ones $(h, r, t')$. The loss function is defined as:

$$\text{loss} = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} \left[\gamma + d(\boldsymbol{h}+\boldsymbol{r},\boldsymbol{t}) - d(\boldsymbol{h}'+\boldsymbol{r},\boldsymbol{t}')\right]_+$$

where:

- $S$ is the set of correct triples.
- $S'$ is the set of incorrect triples.
- $\gamma$ is the margin that separates the correct and incorrect triples.
- $d$ is the distance function (usually L2 norm).
- $[\cdot]_+$ denotes the hinge loss, which is defined as $\max(0, \cdot)$.

(3) **Knowledge Graph Completion**: Once the embeddings are trained, we can use them to infer missing triples and complete the knowledge graph. For a potential triple $(h, r, ?)$, we can predict the entity $t$ that minimises the distance:

$$\|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|.$$

By integrating these embeddings into our knowledge graph, we enhance its completeness and ability to represent and infer complex relationships within the production system.

Fig. 9 illustrates the embedding space where entities and relationships are represented, showing how the TransE model operates.

Consider a knowledge graph with incomplete information about the capabilities of certain assets. By applying the TransE model, we can predict the missing relationships and enrich the knowledge graph. For instance, if we know that an asset $A$ has a certain capability $C$, but lack information about another asset $B$, the embeddings can help infer that $B$ also possesses capability $C$ based on the learned vector representations and their similarities.

This methodology ensures that the knowledge graph not only represents the current state of the production system accurately but also becomes a powerful tool for predicting and reasoning about potential states and configurations, thereby supporting proactive and intelligent decision-making in manufacturing environments.

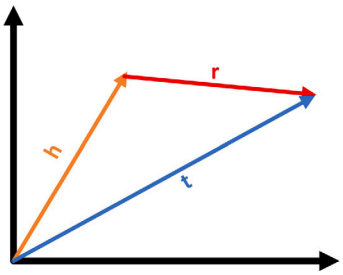



**Fig. 9.** Embedding space representation in TransE model.

### 3.3.5. Storage and application of the knowledge graph

To effectively utilise the constructed and completed knowledge graph, it is essential to consider its storage and practical applications in decision-making processes. The following steps outline this process:

Once the knowledge graph is completed, it can be stored in graph databases such as Neo4j, which efficiently manage and query complex graph structures through powerful query languages like Cypher. Integrating the stored knowledge graph with the runtime condition model enables real-time data updates and seamless interaction between the two systems, thereby facilitating continuous learning and adaptation based on the latest production data. This integration supports decision-making processes by providing a holistic view of the production system—allowing decision-makers to identify optimal asset configurations, predict potential issues, and evaluate the impact of different scenarios (for instance, by matching specific customer requirements to the most suitable assets and configurations). Furthermore, when used in combination with machine learning algorithms, the knowledge graph enhances predictive maintenance strategies by analysing relationships and historical data to forecast potential failures and recommend proactive maintenance actions, ultimately minimising downtime and optimising asset utilisation.

## 3.4. Data update mechanisms in the runtime condition model

Efficiently updating the runtime condition model when faced with massive, real-time data influx is crucial for maintaining system adaptability and accuracy. To address this challenge, the knowledge graph supports incremental updates, enabling new data points—such as sensor readings or test results—to be added without reconstructing the entire graph by updating only the affected nodes, relationships, and attributes. A streaming data pipeline, implemented using tools such as Apache Kafka or MQTT, ingests data continuously in micro-batches, ensuring low-latency updates while preserving computational efficiency. Data that is not immediately critical to decision-making (e.g., historical logs or archival data) is processed in larger batches during off-peak hours, reducing the computational burden on real-time operations. Furthermore, embedding techniques like TransE and RotatE are periodically retrained on the updated graph using incremental training methods, ensuring that the embeddings reflect new data and relationships without requiring a complete reprocessing of the dataset. The runtime condition model also divides data into smaller, context-specific subgraphs based on manufacturing units or operational zones, thereby reducing the computational complexity of global updates through hierarchical data partitioning. In addition, robust validation mechanisms check for conflicts and anomalies—triggering a data reconciliation process when, for example, two sensor readings provide inconsistent values—to ensure data reliability during updates. Finally, the knowledge graph is deployed on scalable graph databases, such as Neo4j, with distributed storage configurations that handle large-scale data efficiently and enable horizontal scaling as data volumes grow.

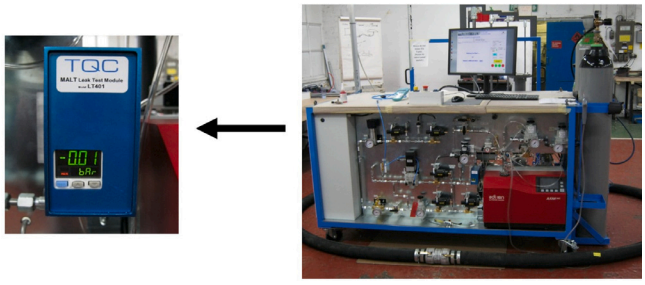



**Fig. 10.** MicroApplication Leak Test System.

## 4. Validation

The proposed runtime condition model was validated through a collaborative use case involving the University of Nottingham, the University of Cambridge, and TQC Automation Ltd. This use case integrates the model into an industrial dry-air leak testing system, the MicroApplication Leak Test System (MALT), developed by TQC Automation Ltd.

Unlike conventional runtime condition models that rely on static thresholds or predefined failure rules, our approach integrates a dynamic representation and knowledge graphs. The novelty lies in the automated inference of operational dependencies and the use of machine learning-based embeddings to predict potential faults before they occur. This methodology provides a scalable and adaptable framework that enhances industrial automation beyond traditional SCADA or rule-based systems.

### 4.1. Use case overview

The MALT system incorporates a Raspberry Pi gateway device and utilises a JAVA-based Asset Administration Shell (AAS) framework (BaSyx) alongside the AAS eXplorer (AASX) to represent and deploy runtime conditions, as illustrated in Fig. 10.

Although the MALT system is a specific industrial testbed, the architectural and semantic layers within our model are not restricted to a single company's workflows. The knowledge graph design and embedding strategies are equally applicable to other manufacturing domains—such as assembly lines, robotic workcells, and automated inspection systems—underscoring the framework's broader academic significance.

### 4.2. Dataset overview and runtime condition model application

The dataset generated from the leak testing comprises 9,076 unique test configurations, each defined by 12 parameters: Volume, Initialisation Delay, Test Pressure (mbar), Vent Off Delay, Fill Time, Stabilisation Time, Isolation Delay, Measuring Time, Venting Time, Offset Compensation, Alarm Leak Rate ($mm^3/s$ or mbar/s), and Alarm Differential Pressure (Pa or mbar). Each configuration is associated with one of four outcome classes, providing input data for the machine learning model. This dataset serves as an empirical foundation for validating the runtime condition model.

By systematically encoding each test configuration as an asset within the runtime condition model, we illustrate how even highly specialised domain knowledge—such as leak-testing parameters—can be mapped onto a general ontology for real-time monitoring and analytics. This ensures our validation transcends mere parameter-tuning and instead showcases an extensible, knowledge-driven architecture.

The model was systematically applied to the dataset to represent and analyse runtime information. The asset runtime representation is shown in Fig. 11.

(1) **Asset Representation:** Each test configuration is treated as an asset with the following attributes:
   - **ID and Status:** Unique identifiers and operational states (active, idle, or faulty).
   - **Implied Metadata:** Location and timeslot inferred from operational schedules.

(2) **Relationships:** Asset interactions and dependencies within the testing system are mapped to reflect operational interdependencies and sequence coordination.
(3) **Key Performance Indicators (KPIs):** Metrics such as leak rate, differential pressure, and test outcomes quantify asset and system performance.
(4) **Capabilities:** Each asset's capability is determined by its ability to perform tests under specific pressure and volume ranges, highlighting functional limits.
(5) **Data:** The dataset provides detailed operational parameters and outcomes, ensuring data alignment with system specifications for model validation.
(6) **Constraints:** Predefined operational limits, such as maximum allowable pressure and minimum leak rates, ensure system safety and reliability.
(7) **Configuration Management:** Test configurations, including pressure and volume settings, are optimised to meet testing requirements and demonstrate the model's adaptability.

The model learns from historical runtime data, allowing for real-time anomaly detection and predictive diagnostics, which is a novel aspect of this study.

### 4.3. Knowledge graph construction and enrichment

Compared to conventional manufacturing knowledge graphs that primarily serve as structured repositories, our model actively learns from operational data using embedding-based inference. This approach enables automated fault prediction and anomaly detection, moving beyond static knowledge representation. Furthermore, the scalability of our method ensures that as new test conditions and assets are introduced, the model dynamically updates its representation without requiring manual reconfiguration.

#### 4.3.1. Initial knowledge graph

As outlined in Section 3.3, a knowledge graph approach was employed to represent the runtime condition model. Knowledge graphs are valuable for structuring interconnected data, enabling advanced analytics, and facilitating reasoning through entity relationships. The initial knowledge graph comprised **1,034,667 nodes** and **1,104,741 relationships**, representing entities such as *Assets*, *KPIs*, *Configurations*, and *Relationships*. Fig. 12 is part of the generated knowledge graph.

The schema layer organises these entities and relationships, enabling seamless navigation, efficient querying, and scalable integration of additional data sources. This foundational graph structure allows the runtime condition model to represent complex interdependencies within the system, making it a pivotal tool for operational insights and predictive analytics.

#### 4.3.2. Model training and evaluation

Embedding techniques were employed to infer missing relationships and predict dependencies in the knowledge graph. A systematic evaluation framework was adopted, utilising standard link prediction metrics: *Hits@1*, *Hits@3*, *Hits@10*, and *Mean Reciprocal Rank (MRR)*. The detailed code is available on GitHub.[1]

[1] https://github.com/mf093087/KGEmbeddingLeakTesting.git

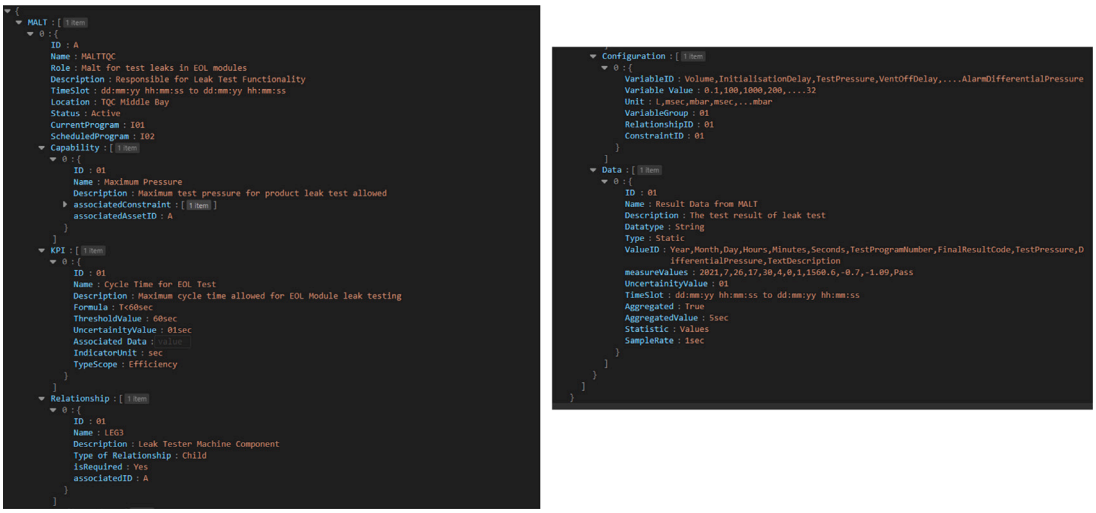



**Fig. 11.** MALT Asset Runtime Representation.

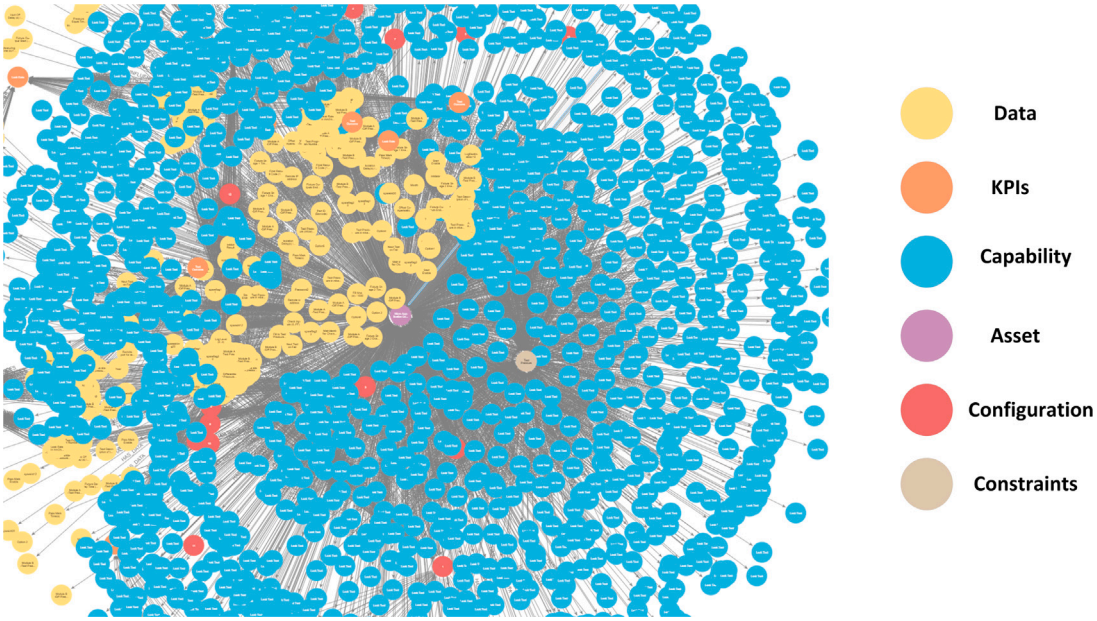



**Fig. 12.** Part of the Generated Knowledge Graph.

*Training and evaluation pipeline.* Embedding models were trained on a dataset derived from the original triples and node features, enhanced to ensure a balance between graph completeness and computational feasibility. The dataset augmentation process included incorporating additional node attributes and generating new relationships to capture nuanced interactions within the graph. The training was conducted on an NVIDIA 3070 GPU installed on a laptop, which provided sufficient computational power for medium-scale experiments while maintaining efficient resource usage. A negative sampling strategy was employed during training, where corrupted triples were generated by randomly replacing entities in valid triples. This approach enabled the models to effectively learn to distinguish meaningful relationships from noise.

The evaluation adhered to the filtered ranking protocol, which excludes existing valid triples from the ranking list during testing to eliminate biases. This ensures a fair and meaningful assessment of the models' link prediction performance.

The evaluation metrics used were Hits@K and Mean Reciprocal Rank (MRR). Hits@K is defined as the percentage of test triples for which the correct entity appears within the top $K$ predictions, reflecting the models' ability to rank the correct prediction near the top of the list. MRR, on the other hand, is the mean reciprocal rank of the correct predictions and provides an aggregated measure of overall ranking quality, with higher values indicating better performance.

*Performance analysis across models.* Table 1 summarises the performance of embedding models on the knowledge graph completion task, evaluated using standard link prediction metrics (*Hits@1*, *Hits@3*, *Hits@10*, and *MRR*). Each model's performance highlights its specific strengths and limitations.

TransE demonstrated competitive performance with Hits@1 of 0.5052, Hits@3 of 0.6618, and Hits@10 of 0.7513, along with an MRR of 0.5935; its simplicity yields high computational efficiency and scalability for large datasets, although it struggles with complex patterns such as symmetry or antisymmetry. RotatE, with Hits@1 of 0.4728, Hits@3 of 0.6026, Hits@10 of 0.6902, and an MRR of 0.5509, leverages rotational embeddings to effectively model symmetric and antisymmetric relationships, yet its moderate performance suggests challenges in handling noisy or diverse relational data. ComplEx emerged as the best-performing model overall, achieving the highest MRR (0.5960)

**Table 1**
Performance comparison of embedding models on knowledge graph completion.

| Model | Hits@1 | Hits@3 | Hits@10 | MRR |
|---|---|---|---|---|
| TransE [26] | 0.5052 | 0.6618 | 0.7513 | 0.5935 |
| RotatE [27] | 0.4728 | 0.6026 | 0.6902 | 0.5509 |
| ComplEx [28] | **0.5145** | 0.6504 | 0.7460 | **0.5960** |
| DistMult [29] | 0.5070 | 0.6437 | 0.7234 | 0.5870 |
| TransH [30] | 0.4903 | 0.6536 | 0.7555 | 0.5851 |
| TransD [31] | 0.4967 | **0.6727** | 0.7547 | 0.5941 |
| GraphSAGE [32] | 0.4972 | 0.6618 | 0.7579 | 0.5906 |
| R-GCN [33] | 0.4930 | 0.6522 | **0.7600** | 0.5869 |
| A2N [34] | 0.5012 | 0.6591 | 0.7598 | 0.5922 |
| CompGCN [35] | 0.4967 | 0.6512 | 0.7529 | 0.5885 |
| SE-GNN [36] | 0.4951 | 0.6461 | 0.7579 | 0.5874 |

and Hits@1 (0.5145), while DistMult, with an MRR of 0.5870 and Hits@10 of 0.7234, handles simple one-to-one relations effectively. TransH improves over TransE by projecting entities onto relation-specific hyperplanes, yielding Hits@10 of 0.7555 and Hits@3 of 0.6536 (MRR: 0.5851), and TransD shows strong results with the highest Hits@3 (0.6727) and competitive Hits@10 (0.7547) alongside an MRR of 0.5941. GraphSAGE, through neighbourhood aggregation, achieves high Hits@10 (0.7579) and an MRR of 0.5906, while R-GCN records the highest Hits@10 (0.7600) with Hits@3 of 0.6522 and an MRR of 0.5869, though its computational demands may limit scalability. Additionally, A2N (Hits@1: 0.5012, Hits@3: 0.6591, Hits@10: 0.7598, MRR: 0.5922), CompGCN (Hits@10: 0.7529, MRR: 0.5885), and SE-GNN (Hits@10: 0.7579, MRR: 0.5874) further illustrate that while each model offers distinct advantages for specific data types or relational structures, trade-offs exist between predictive accuracy and computational efficiency.

By systematically evaluating multiple embedding models for knowledge graph completion, we establish a benchmark for runtime condition modelling in industrial settings. This comparative analysis is the first of its kind in the domain of real-time asset monitoring, providing empirical insights into the trade-offs between computational efficiency and predictive accuracy. These findings inform the selection of optimal models for manufacturing applications requiring adaptive runtime intelligence.

*Explaining variations in performance.* Based on the evaluation metrics, ComplEx demonstrated the highest Hits@10 and MRR scores, proving effective for capturing complex relationships and asymmetric dependencies. In contrast, RotatE provided robust performance for diverse relational patterns, thereby complementing ComplEx in scenarios requiring relational diversity, while TransE was selected for its computational efficiency, particularly for large-scale datasets where scalability is critical.

Our knowledge graph framework, enriched by trained embedding models, refines representation and decision support for the MALT while addressing fault prediction in dynamic manufacturing. By structuring leak-testing parameters, assets, and constraints as interconnected entities, the graph provides a holistic view that reveals hidden interdependencies—such as specific volume and pressure combinations that lead to borderline outcomes—beyond a simple pass/fail table. The embedding-based fault prediction process collects historical leak-testing data and trains models like TransE, RotatE, or ComplEx to generate vector representations that capture co-occurrence patterns; new test configurations are then encoded and compared against these patterns, flagging those that match high-risk profiles to prompt preventive adjustments. Continuous retraining on live sensor data enables real-time detection of emerging bottlenecks and proactive remedial actions, while seamless integration of new sensor metrics and asset types ensures scalability and flexibility. Finally, detailed KPI monitoring combined with embedding-driven forecasting guides proactive maintenance, reducing downtime, shortening cycle times, and lowering production costs.

Our knowledge graph approach dynamically infers failure probabilities based on learned representations. This ensures adaptability to evolving production conditions, reducing dependency on manually engineered fault rules. The continuous update mechanism allows for incremental learning, improving fault prediction accuracy over time without requiring a full retraining cycle. This makes the approach particularly suitable for high-variability manufacturing environments where static models struggle to generalise.

Overall, these validation results demonstrate not only the feasibility of integrating runtime condition models with advanced embedding methods but also how such an approach can be scaled and adapted across different manufacturing contexts. By systematically evaluating multiple embedding algorithms, we illustrate the breadth and depth of potential analytic outcomes—from simple link prediction to sophisticated fault prevention—helping to bridge the gap between academic innovation and industrial necessity.

## 5. Conclusion

This paper presents the development and validation of a novel runtime condition model designed to enhance interoperability, data integration, and decision-making in intelligent manufacturing environments. The proposed model addresses the increasing need for systems capable of real-time monitoring, adaptability, and efficient decision-making by providing a framework for representing key manufacturing components, including asset management, relationships, key performance indicators (KPIs), capabilities, data handling, constraints, and configurations.

A major innovation of this work lies in its integration with a knowledge graph, which structures and organises operational data into a schema layer. By employing knowledge graph embedding techniques, such as the TransE model, the approach extends beyond static data representation to infer missing relationships, enrich the knowledge graph, and enable dynamic predictive analytics. This integration effectively addresses the challenges of heterogeneous data sources and supports proactive decision-making in real-world manufacturing scenarios.

The model was validated through a case study conducted in collaboration with TQC Automation Ltd., utilising their MicroApplication Leak Test System (MALT). Empirical results demonstrated the model's ability to integrate and manage complex runtime data, optimise operational parameters, and support predictive analysis. The rigorous testing of each component highlighted the model's capability to adapt to dynamic conditions and improve manufacturing efficiency.

The study revealed several significant contributions. The model successfully harmonises diverse data sources, providing a unified and comprehensive framework for operational analysis. The enriched knowledge graph enables the prediction of missing relationships and the optimisation of runtime decisions, addressing inefficiencies and minimising downtime. Furthermore, the model demonstrates flexibility in adapting to complex manufacturing setups, proving its relevance in real-world scenarios.

This study presents a novel approach to runtime condition modelling, integrating data, knowledge graphs, and machine learning-based to overcome the limitations of static runtime monitoring systems. By systematically evaluating different knowledge graph embedding models, we provide an empirical benchmark for future research in adaptive industrial control. The findings demonstrate that our model enables real-time diagnostics making it a scalable solution for evolving manufacturing environments.

Unlike traditional rule-based or SCADA-driven monitoring frameworks, our approach offers a self-adaptive mechanism. These insights contribute to the broader fields of digital twins, cyber–physical systems, and intelligent manufacturing. Future work will focus on expanding this framework to accommodate multi-source sensor fusion and optimising computational efficiency for large-scale industrial deployments.

Future research will focus on extending the model's applicability to broader manufacturing domains and incorporating larger datasets to enhance its predictive and analytical capabilities. The integration of advanced machine learning techniques with the runtime condition model holds significant potential for providing deeper insights and fostering innovation in intelligent manufacturing systems. Additionally, the incorporation of multimodal data sources and advanced graph neural network methods will further enhance inference accuracy and scalability, reinforcing the model's role as a transformative tool for next-generation manufacturing.

## CRediT authorship contribution statement

**Fan Mo:** Writing – original draft, Validation, Software, Methodology, Conceptualization. **Hamood Ur Rehman:** Writing – original draft, Validation, Software, Methodology, Conceptualization. **Miriam Ugarte:** Writing – original draft, Conceptualization. **Angela Carrera-Rivera:** Writing – original draft, Conceptualization. **Nathaly Rea Minango:** Writing – original draft, Conceptualization. **Fabio Marco Monetti:** Writing – original draft, Conceptualization. **Antonio Maffei:** Writing – review & editing, Supervision, Conceptualization. **Jack C. Chaplin:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This research is supported by the DiManD Innovative Training Network (ITN) project funded by the European Union through the Marie Skłodowska-Curie Innovative Training Networks under grant agreement number no. 814078. The research has also been funded by the EPSRC, United Kingdom grant EP/T024429/1 Elastic Manufacturing Systems, the support of which is gratefully acknowledged.

## Data availability

Data will be made available on request.

## References

[1] Chunquan Li, Yaqiong Chen, Yuling Shang, A review of industrial big data for decision making in intelligent manufacturing, Eng. Sci. Technol. Int. J. 29 (2022) 101021.
[2] Jay Lee, Jun Ni, Jaskaran Singh, Baoyang Jiang, Moslem Azamfar, Jianshe Feng, Intelligent maintenance systems and predictive manufacturing, J. Manuf. Sci. Eng. 142 (11) (2020) 110805.
[3] Abdessamad Ait El Cadi, Ali Gharbi, Karem Dhouib, Abdelhakim Artiba, Joint production and preventive maintenance controls for unreliable and imperfect manufacturing systems, J. Manuf. Syst. 58 (2021) 263–279.
[4] Panagiotis Stavropoulos, Dimitrios Chantzis, Charalampos Doukas, Athanasios Papacharalampopoulos, George Chryssolouris, Monitoring and control of manufacturing processes: A review, Procedia CIRP 8 (2013) 421–425.
[5] Zude Zhou, Bitao Yao, Wenjun Xu, Lihui Wang, Condition monitoring towards energy-efficient manufacturing: A review, Int. J. Adv. Manuf. Technol. 91 (9) (2017) 3395–3415.
[6] Jianhui Mao, Liqian Chen, Runtime monitoring for cyber-physical systems: A case study of cooperative adaptive cruise control, in: 2012 Second International Conference on Intelligent System Design and Engineering Application, IEEE, 2012, pp. 509–515.
[7] Hans Fleischmann, Simon Spreng, Johannes Kohl, Dominik Kißkalt, Jörg Franke, Distributed condition monitoring systems in electric drives manufacturing, in: 2016 6th International Electric Drives Production Conference, EDPC, IEEE, 2016, pp. 52–57.
[8] Andrew D. Kenyon, Victoria M. Catterson, Stephen D.J. McArthur, John Twiddle, An agent-based implementation of hidden Markov models for gas turbine condition monitoring, IEEE Trans. Syst. Man, Cybern.: Syst. 44 (2) (2013) 186–195.
[9] Fan Mo, Fabio Marco Monetti, Agajan Torayev, Hamood Ur Rehman, Jose A Mulet Alberola, Nathaly Rea Minango, Hien Ngoc Nguyen, Antonio Maffei, Jack C. Chaplin, A maturity model for the autonomy of manufacturing systems, Int. J. Adv. Manuf. Technol. 126 (1) (2023) 405–428.
[10] William Z. Bernstein, Thomas D. Hedberg Jr., Moneer Helu, Allison Barnard Feeney, Contextualising manufacturing data for lifecycle decision-making, Int. J. Prod. Lifecycle Manag. 10 (4) (2017) 326–347.
[11] Pranay Jhunjhunwala, Udayanto Dwi Atmojo, Valeriy Vyatkin, Applying skill-based engineering using OPC-UA in production system with a digital twin, in: 2021 IEEE 30th International Symposium on Industrial Electronics, ISIE, IEEE, 2021, pp. 1–6.
[12] Fan Mo, Jack C. Chaplin, David Sanderson, Giovanna Martínez-Arellano, Svetan Ratchev, Semantic models and knowledge graphs as manufacturing system reconfiguration enablers, Robot. Comput.-Integr. Manuf. 86 (2024) 102625.
[13] Xiaohan Zou, A survey on application of knowledge graph, J. Phys.: Conf. Ser. 1487 (1) (2020) 012016.
[14] Xiaolu Lu, Soumajit Pramanik, Rishiraj Saha Roy, Abdalghani Abujabal, Yafang Wang, Gerhard Weikum, Answering complex questions by joining multi-document evidence with quasi knowledge graphs, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 105–114.
[15] Bo Qiao, Kui Fang, Yiming Chen, Xinghui Zhu, Building thesaurus-based knowledge graph based on schema layer, Clust. Comput. 20 (2017) 81–91.
[16] Fan Mo, Jack C. Chaplin, David Sanderson, Svetan Ratchev, Advancing capability matching in manufacturing reconfiguration with large language models, in: Proceedings of the International Conference on Flexible Automation and Intelligent Manufacturing, Springer, 2024, pp. 215–222.
[17] Kaze Du, Bo Yang, Keqiang Xie, Nan Dong, Zhengping Zhang, Shilong Wang, Fan Mo, Llm-manuf: an integrated framework of fine-tuning large language models for intelligent decision-making in manufacturing, Adv. Eng. Inform. 65 (2025) 103263.
[18] Hamood Ur Rehman, Fan Mo, Jack C Chaplin, Leszek Zarzycki, Mark Jones, Svetan Ratchev, A modular artificial intelligence and asset administration shell approach to streamline testing processes in manufacturing services, J. Manuf. Syst. 72 (2024) 424–436.
[19] Georg Buchgeher, David Gabauer, Jorge Martinez-Gil, Lisa Ehrlinger, Knowledge graphs in manufacturing and production: A systematic literature review, IEEE Access 9 (2021) 55537–55554.
[20] Agniva Banerjee, Raka Dalal, Sudip Mittal, Karuna Pande Joshi, Generating digital twin models using knowledge graphs for industrial production lines, in: Proceedings of the 2017 ACM on Web Science Conference, 2017, pp. 425–430.
[21] Jingshu Yuan, Hongqi Li, Research on the standardization model of data semantics in the knowledge graph construction of oil & gas industry, Comput. Stand. Interfaces 84 (2023) 103705.
[22] Niannian Guan, Dandan Song, Lejian Liao, Knowledge graph embedding with concepts, Knowledge-Based Syst. 164 (2019) 38–44.
[23] Tong Shen, Fu Zhang, Jingwei Cheng, A comprehensive overview of knowledge graph completion, Knowledge-Based Syst. 255 (2022) 109597.
[24] Ella Roubtsova, Vaughan Michell, Modelling and validation of KPIs, Proc. BMSD (2013).
[25] Fan Mo, Hamood Ur Rehman, Fabio Marco Monetti, Jack C. Chaplin, David Sanderson, Atanas Popov, Antonio Maffei, Svetan Ratchev, A framework for manufacturing system reconfiguration and optimisation utilising digital twins and modular artificial intelligence, Robot. Comput.-Integr. Manuf. 82 (2023) 102524.
[26] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, Oksana Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in Neural Information Processing Systems, vol. 26, 2013.
[27] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, Jian Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, 2019, arXiv preprint arXiv:1902.10197.
[28] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, Guillaume Bouchard, Complex embeddings for simple link prediction, in: International Conference on Machine Learning, PMLR, 2016, pp. 2071–2080.
[29] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, Li Deng, Embedding entities and relations for learning and inference in knowledge bases, 2014, arXiv preprint arXiv:1412.6575.
[30] Zhen Wang, Jianwen Zhang, Jianlin Feng, Zheng Chen, Knowledge graph embedding by translating on hyperplanes, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2014.
[31] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, Jun Zhao, Knowledge graph embedding via dynamic mapping matrix, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2015, pp. 687–696.
[32] Will Hamilton, Zhitao Ying, Jure Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).

[33] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, Max Welling, Modeling relational data with graph convolutional networks, in: The Semantic Web: ESWC 2018 Proceedings, Springer, 2018, pp. 593–607.

[34] Trapit Bansal, Da-Cheng Juan, Sujith Ravi, Andrew McCallum, A2N: Attending to neighbors for knowledge graph inference, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 4387–4392.

[35] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Partha Talukdar, Composition-based multi-relational graph convolutional networks, 2019, arXiv preprint arXiv:1911.03082.

[36] Ren Li, Yanan Cao, Qiannan Zhu, Guanqun Bi, Fang Fang, Yi Liu, Qian Li, How does knowledge graph embedding extrapolate to unseen data: A semantic evidence view, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 5781–5791.