Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 55-10 (2022) 2725–2730

Detecting failure of a material handling system through a cognitive twin

R. D. D'Amico*, A. Sarkar**, H. Karray**, S. Addepalli*, J. A. Erkoyuncu*

** CDEM, Cranfield University, Cranfield, United Kingdom;(e-mail: r.damico@cranfield.ac.uk).*

***LGP-INP-ENIT, Université de Toulouse, Tarbes, France; (e-mail: mkarray@enit.fr)*

Abstract: This paper describes a methodology for developing a digital twin (DT) based on a rich semantic model and principles of system engineering. The aim is to provide a general model of digital twins (DT) that can improve decision making based on semantic reasoning on real-time system monitoring. The methodology has been tested on a laboratory pilot plant that acts as a material handling system. The key contribution of this research is to propose a generic information model for DT using foundational ontology and principles of systems engineering. The efficacy of the proposed methodology is demonstrated by the automatic detection of a component level failure using semantic reasoning.

Keywords: Digital twin, cognitive twin, ontology, BFO, IOF, CCO, knowledge graph, SPARQL, material handling systems, Festo MPS.

1. INTRODUCTION

Digital Twin (DT) is one of the technologies used to improve asset management, and the interest in this technology has gradually increased in the last years both in industry and academia (Jones et al., 2020). There are several definitions of DT and hence a lot of confusion around this terminology. The first time DT concept has been introduced by Michael Grieves (Grieves, 2015). In 2021, an ISO standard provided a standard definition of DT for manufacturing (ISO 23247-1:2021), which is a "*fit for purpose digital representation of an observable manufacturing element with synchronization between the element and its digital representation*.

Companies around the world (NASA, GE, Chevron, Siemens, Oracle, etc.) already use DTs to detect asset failures and improve efficiency (Tao & Qi, 2019). However, the rapid growth of DT is leading to a proliferation of stand-alone DT solutions. Without a standardized recipe for DT implementation, it often becomes challenging for the industries to deploy DT solutions quickly and flexibly. Moreover, a methodology to model unified DT is needed (Tao et al., 2019) so that the information model for various types of physical systems may be customized from generic templates by decreasing the development time. Antonova et al. pointed out the need for having common semantics for the data modelling and exchange for the DT to be able to integrate data from different sources (Petrova-Antonova & Ilieva, 2021). One of the solutions to address these needs is the use of a shared language approach (Erkoyuncu et al., 2020). Ontologies have been recently considered as one of the methodologies to ensure the shared language or the common semantics in the DT scenario (D'Amico et al., 2021)

The association between DT and ontology to break down the current siloed approach is gaining increasing attention. Among others, recent examples can be found in literature such as the concept of universal DT (Akroyd et al., 2021), and the concept of cognitive twin (CT) (Lu et al., 2020). Both of them use knowledge graphs and ontologies to ensure cross-domain interoperability (Lu et al., 2020).

Despite CT being a promising solution towards the achievement of a unified model (Tao & Qi, 2019) of DT, generality needs to be addressed not only by a set of common semantics but also by adopting common principles of systems engineering. While the former addresses data interoperability, the latter leverages the well-founded system theory to make the DT suitable for applying common system analysis (such as functional decomposition, function flow, material flow, energy flow, state-based activity analysis, and causal analysis). Furthermore, the need for physical demonstrators to show the efficacy and benefits of this unified method (Akroyd et al., 2021) is critical in ensuring increased uptake of the approach.

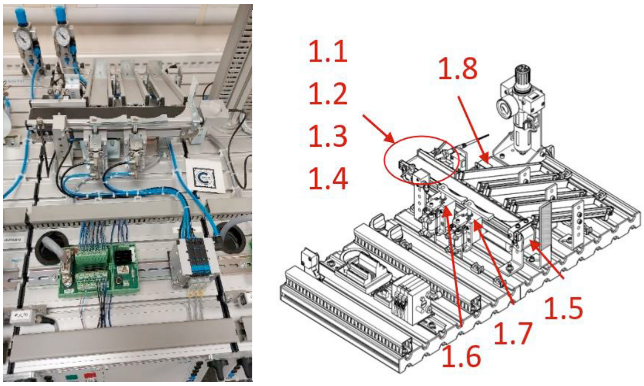
To address the abovementioned gaps, this paper adopts rich semantics (Poli et al., 2010), based on foundational terms borrowed from well-founded top-level ontologies with philosophical underpinning, for achieving utmost generality in the data modelling. At the same time, adopting the axiomatic design theory (Suh, 1998) for modelling the structure of the system based on functional decomposition and linking it to state-based activity modelling. This study grounds itself on principles of systems engineering.

To show the benefits and applicability of this unified approach, a physical demonstrator has been developed. The CT concept has been adopted in a use case to detect a component level failure in a laboratory pilot material handling system. The laboratory pilot system is the Festo sorting station module, which is part of a Festo modular production system (MPS) see Figure 1-top left. The purpose of this machine is to replicate in a lab the sorting feature of a material handling system. Moreover, the system has been modelled and validated using the principles of the aforementioned axiomatic design theory.

2. METHODOLOGY

The development of the CT starts with the characterisation of

10.1016/j.ifacol.2022.10.128



	Component	Action
1.1	Diffuse sensor	Part confirmation
1.2	Diffuse colour sensor	Part identification – colour
1.3	Inductive proximity	Part identification – material
1.4	Gate actuator	Part block and release
1.5	Conveyor belt motor	Part moving
1.6	Actuator 1	Sorting 1
1.7	Actuator 2	Sorting 2
1.8	Retro reflective sensor	End of cycle confirmation

Figure 1 - Festo sorting station, part of Festo MPS.

the system and the boundaries of the domain. Classification of the components, qualitative formalisation of parthood and connection relations between components (mereotopology), and relations between functions and capabilities of each component have been studied and reverse engineered. Each component has been classified as seen in Figure 1.

The Festo sorting station operates as follows:

1. When the workpiece enters the sensing unit, the system sends the data to the knowledge graph (KG). The KG associates a workpiece ID and the spatial region *s_in* to the timestamp received.

2. The sensing unit measures the colour of the workpiece and once measured, sends the order to the gate actuator to let the workpiece go.

3. When the gate actuator opens, the system sends the results of the sensing process to the KG, together with the related timestamp. The KG associates the workpiece ID, the spatial region *s_out*, and the colour to the timestamp received.

4. If the workpiece is *red*, the first actuator is activated sorting the part to the first collection point, whereas if it is *metal* the second actuator is activated sorting the part to the second collection point. No actuator is activated if the workpiece is *black*. A physical obstacle conveys the *black* workpiece through the conveyor system to the third collection point. In this case, data points for spatial regions *n1* and *n2* are associated with the relative workpiece ID.

5. When every workpiece is detected by the retro-reflective sensor, the system sends the data point to the KG. The KG associates the timestamp and spatial region w with the relative workpiece ID.

2.1 Ontology design of the CT

A set of Competency Questions (CQ) (Grüninger & Fox, 1995) have been created to formalise the pieces of information that the CT will be able to manage.

The CQs created for this use case are the following:

1. What are the components of the sorting station?

2. How long does the conveyor take to carry the workpiece X?

3. What is the time-point that the workpiece X entered the sensing unit?

4. What is the colour of workpiece X sensed after the sensing unit?

5. How many workpieces are at collection point X at time T?

6. Is workpiece X sensed by the sensing unit in the time interval T?

The predicates used to build the ontology model of the Festo sorting system extend Common Core Ontology (CCO) (https://www.cubrc.org) and the IOF-core (Karray et al., 2021). As those ontologies refer to a foundational ontology, or top-level ontology (TLO), called Basic Formal Ontology (BFO) (Smith et al., 2007), which is also an ISO standard (ISO 21838-2:2021), the application ontology becomes generic and interoperable.

In the following sections, definitions of the concepts and relationships of the ontology model of the CT, including the related axioms using the foundational terms from BFO, CCO, and IOF-core, have been provided. For instance, *bears*, *realizes*, *participatesIn*, *OccurrentPartOf*, *occursOn*, *hasFirstInstance*, *hasLastInstance,* etc. have been taken from BFO. *hasOutput*, *ActOfLocationChange*, *ActOfMeasurement*, etc. have been taken from CCO. *System* from IOF-core.

2.2 Structure design

To capture the process flow in the Festo system, the structural parthood of the system needs to be constructed based on the corresponding functional decomposition. Axiomatic Design (AD) theory (Suh, 1998) is adopted to delineate the parts of the system in such a way that, for every level of decomposition, if the functions of each component are realised by some suitable processes, then the function of the parent component is also realised (Sarkar et al., 2020).

Figure 2 (left) presents structural mereology along with the corresponding functional decomposition for the Festo sorting station. The independence axiom of AD theory is reformulated by Sarkar et al. based on how the functions at some levels of decomposition are supported by the capabilities of the corresponding components. Two types of valid configurations: summation junction, and control junction (types of modulo-junction), may be derived from the relationships between functions and capabilities of components of a system,

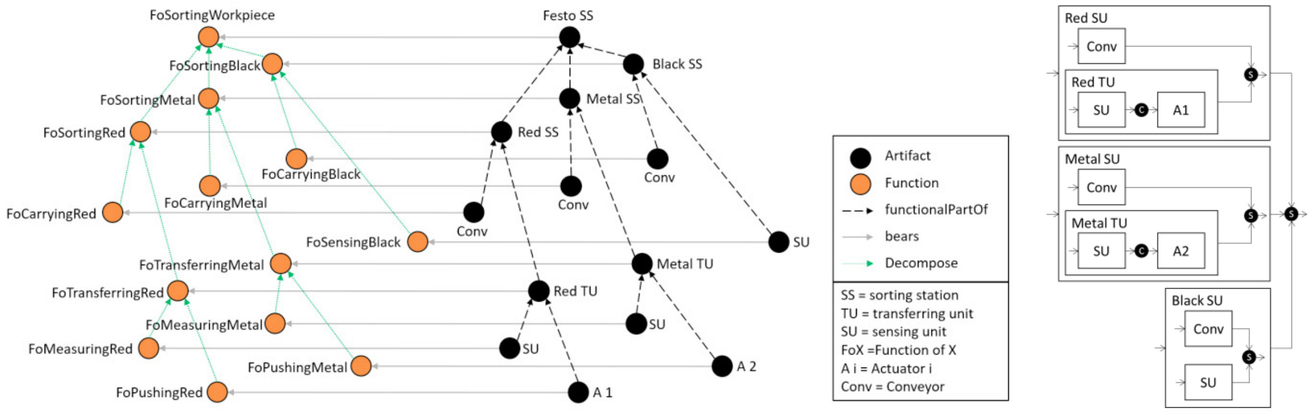


Figure 2 - Mereology of the system, and functions/capabilities of each component. (left) Modulo junction diagram (right).

following the diagonal and triangular design matrices (proposed in AD theory) respectively.

The predicate SummationJunction and ControlJunction hold between a component and two of its sub-components. For ControlJunction, the component in the second place controls the component in the third place. As presented in Figure 2 (right), the sorting station is a *SummationJunction* of the 'Black sorting station, 'Red sorting station, and 'Metal sorting station. Looking closer, the first two units are *SummationJunction* of a section of the 'Conveyor belt' and a 'Transferring unit', whereas the 'transferring unit' is a *ControlJunction,* in which 'Actuators' are controlled by the 'Sensing unit'. Note that there is no actuator for the last slope. The underlying relationships among the capabilities of the components and their functions are not included in this paper for brevity.

2.3 Processes design

In every occurrence of a type of process, specific types of entities are engaged with different roles. We use two types of thematic roles (LIRICS - Linguistic InfRastructure for Interoperable ResourCes and Systems), i.e., instrument (that is the immediate cause) and patient (that is affected) to distinguish between the system and the material that the system handles. The latter material is the workpiece that the Festo sorting station sorts according to its colour. Three primary types of processes are carried out by different components of the Festo sorting station. For all of them, the workpiece acts as the patient but differs in how it is affected by the processes. For example, the first one of the following axioms dictates that for the sensors to realise the function of sensing colour by some process of *ActOfMeasuringColor*, colour information (the colour of the workpiece) becomes available at the end of every occurrence. Both *ActOfCarrying* and *ActOfActuating* aim at changing the location of the workpiece. The former changes the spatial location of the workpiece while the workpiece is seated at the same location (a *Site*) on the conveyor belt as it moves. The latter changes the location of the workpiece from one site of the conveyor to another site of the collection point (acting as final storage for the workpiece).

$$\forall p,f\ ActOfMeasuringColor(p) \wedge realises(p,f) \leftrightarrow ActOfMeasurement(p) \wedge \exists s\ \big(Sensor(s) \wedge isInstrumentOf(s,p)\big) \wedge \exists w\big(Workpiece(w) \wedge isPatientOf(w,p)\big) \wedge \forall t(occursOn(p,t) \rightarrow \exists c,ci\ (Color(c) \wedge ColorName(ci) \wedge hasOutput(p,ci) \wedge isMeasurementOf(ci,c) \wedge \exists t^e(hasLastInstant(t,t^e) \wedge inheresIn(c,w,t^e)) \quad (1)$$

$$\forall p,f\ ActOfCarrying(p) \wedge realises(p,f) \leftrightarrow ActOfLocationChange(p) \wedge \exists c\ \big(Conveyor(c) \wedge isInstrumentOf(c,p)\big) \wedge \exists w\big(Workpiece(w) \wedge isPatientOf(w,p)\big) \wedge \forall t\ (occursOn(p,t) \rightarrow \exists l,s^1,s^2,t^b,t^e\ (locatedIn(w,l,t) \wedge partOf(l,c) \wedge occupiesSpatialRegion(l,s^1,t^b) \wedge occupiesSpatialRegion(l,s^2,t^e) \wedge hasFirstInstant(t,t^b) \wedge hasLastInstant(t,t^e) \wedge (s^1 \neq s^2)) \quad (2)$$

$$\forall p,f\ ActOfActuating(p) \wedge realises(p,f) \leftrightarrow ActOfLocationChange(p) \wedge \exists a\ \big(Actuator(a) \wedge isInstrumentOf(a,p)\big) \wedge \exists w\big(Workpiece(w) \wedge isPatientOf(w,p)\big) \wedge \forall t\ (occursOn(p,t) \rightarrow \exists l,l',s^1,s^2,t^b,t^e,n,c\ (Slope(n) \wedge Conveyor(c) \wedge partOf(l',n) \wedge partOf(l,c) \wedge occupiesSpatialRegion(l,s^1,t^b) \wedge occupiesSpatialRegion(l',s^2,t^e) \wedge hasFirstInstant(t,t^b) \wedge hasLastInstant(t,t^e))) \quad (3)$$

Following the functional structure of the Festo Sorting Station given in Figure 2 (left), there are three sub-types of *ActOfMeasuringColor*, i.e., *ActOfMeasuringBlack*, *ActOfMeasuringRed*, and *ActOfMeasuringMetal*, based on the sub-types of colours they detect and the types of functions they realise, e.g., *realizes(ActOfMeasuringRed, FunctionOfMeasuringRed)*. Similarly, there are three types of *ActOfCarrying* depending on the end locations (e.g., the red workpiece is carried up to the '*Actuator* 1') of the workpiece. For the black workpiece being carried to the 'collection point 3' by the conveyor belt directly, no actuation process is required for the black workpiece and only two sub-types of *ActOfActuating* are declared, i.e., *ActOfActuatingRed*, and *ActOfActuatingMetal*.

For reasoning, the malfunctioned component, and the compositions of these processes are also needed. These processes are performed by the upper-level components and must have some occurrences of these processes as part. For instance, *RedTransferringUnit* is the instrument of some *ActOfTransferringRed*, for which the some *ActOfMeasuringRed* and *ActOfActuatingRed* are occurrent parts.

2.4 Detecting malfunction analysis

A component malfunctions if the process, in which the component plays the role of the instrument, fails to realise its corresponding function. The effect of a component malfunctioning can be manifested not only at the component level but also at the system level. For instance, Actuator 1 not triggering may cause the red workpiece to be transferred to collection point 3 or the malfunction in the sensing unit may cause a Metal workpiece to be pushed into collection point 1. An error at the system level can be narrowed down to a component by applying several rules following the modulo-junction structure of the system given in Figure 2 (right). We define a predicate *malfunctionedFor* holding between a component, an occurrence of a process, in which the failure occurs, and a workpiece, which the component failed to handle. In other words, a process, in which that workpiece was a patient and the component an instrument, did not realise the function of the component. Below we provide a set of four rules for detecting an error in sorting red workpieces as an example. Other types of errors may have their own set of rules.

$$malfunctionedFor(c,p,w) \wedge RedSortingUnit(c) \wedge ActOfSortingRed(p) \wedge Workpiece(w) \wedge Conveyor(c') \wedge ActOfCarryingRed(p') \wedge isPatientOf(w,p') \wedge isInstrumentOf(c',p') \wedge \neg realizes(p','FunctionOfCarryingRed') \rightarrow malfunctionedFor(c',p',w) \quad (4)$$

$$malfunctionedFor(c,p,w) \wedge RedSortingUnit(c) \wedge ActOfSortingRed(p) \wedge Workpiece(w) \wedge RedTransferringUnit(c') \wedge ActOfTransferringRed(p') \wedge isPatientOf(w,p') \wedge isInstrumentOf(c',p') \wedge Conveyor(c'') \wedge ActOfCarryingRed(p'') \wedge isPatientOf(w,p'') \wedge isInstrumentOf(c'',p'') \wedge realizes(p','FunctionOfCarryingRed') \rightarrow malfunctionedFor(c',p',w) \quad (5)$$

$$malfunctionedFor(c,p,w) \wedge RedTransferringUnit(c) \wedge ActOfTransferringRed(p) \wedge Workpiece(w) \wedge SensingUnit(c') \wedge ActOfMeasuringRed(p') \wedge isPatientOf(w,p') \wedge isInstrumentOf(c',p') \wedge \neg realizes(p','FunctionOfMeasuringRed') \rightarrow malfunctionedFor(c',p',w) \quad (6)$$

$$malfunctionedFor(c,p,w) \wedge RedTransferringUnit(c) \wedge ActOfTransferringRed(p) \wedge Workpiece(w) \wedge Actuator(c') \wedge ActOfActuatingRed(p') \wedge isPatientOf(w,p') \wedge isInstrumentOf(c',p') \wedge \neg realizes(p','FunctionOfPushingRed') \wedge SensingUnit(c'') \wedge ActOfMeasuringRed(p'') \wedge isPatientOf(w,p'') \wedge isInstrumentOf(c'',p'') \wedge realizes(p'','FunctionOfMeasuringRed') \rightarrow malfunctionedFor(c',p',w) \quad (7)$$

In rule 1 (4), the cause of failure for *RedSortingUnit* (initial fact representing an error in sorting red workpieces) is transferred to the *Conveyor* if the related occurrence of *ActOfCarryingRed* fails (note that the negation of realizes implies that the state description stated in the right part of (2) is false, i.e., the required output state is not achieved). If rule 1 (4) doesn't trigger, that the *Conveyor* is not malfunctioning, rule 2 (5) will transfer the cause to the *RedTransferringUnit*. If *RedTransferringUnit* malfunctions, then rule 3 (6) and rule 4 (7) check if the malfunctioning is in *SensingUnit* or *RedTransferringUnit* respectively. As the *SensingUnit* controls the *RedTransferringUnit*, it is not required to check whether the *Actuator* also fails in rule 3 (6). On the contrary, the root cause of failure can only be attributed to the *Actuator*, if only the *ActOfActuatingRed* fails (doesn't realize the *Actuator's* function) but the preceding *ActOfMeasuringRed* succeeds (does realise the *SensingUnit*'s function).

3. COGNITIVE DIGITAL TWIN CONSTRUCTION

3.1 Ontology model development

The ontology is exported from the Protégé tool in the *.owl* file format. The ontology file created will then be imported into the data management tool together with all the other ontologies used (BFO, CCO, and IOF-Core). The ontology of the Festo sorting station has been created with the Protégé tool (https://protegewiki.stanford.edu/wiki/Main_Page). The data is collected and managed within the same data management tool, GraphDB (https://graphdb.ontotext.com/).

SPARQL (https://www.w3.org/TR/sparql11-overview) queries have been used to add information about the structure and functions of the system and its components, the related processes as well as the reasoning for the malfunctions. The following section describes how these queries transform the real-time monitoring data into state descriptions of the workpiece. Separate SPARQL queries have been used to infer the occurrences and their process types from the state descriptions as required. For example, if the analysis is conducted for a particular workpiece, all the occurrences in which the workpiece participated as a patient may be inferred from the related state descriptions using suitable SPARQL. The rules given from (4) to (7) are Horn rules and SPARQL Construct queries have been used to encode them. The ternary predicate *malfunctionedFor* is reified as a separate class (not modelled under any upper-level ontology) linking the instances of component, process, and the workpiece.

3.2 CT development

Figure 3 shows the architecture of the CT and how it works. Data points from the Festo sorting system have been sent to GraphDB, which is the tool selected for data management. GraphDB is a free tool that allows the creation of knowledge graphs (KG). The KG represents the network of all the entities that constitute the system (objects, events, data, etc.) and

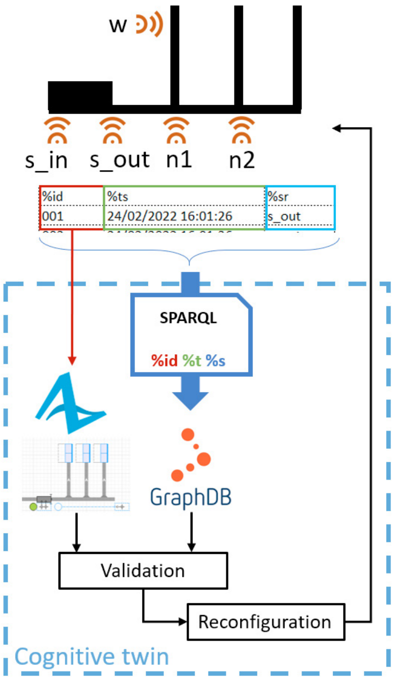


Figure 3 – Architecture of the CT

illustrates the relationships between them. It is the graph database that stores the data points received from the physical system and where it is possible to query the competency questions created before. It is a key element of the CT.

The system sends data points to the KG in 5 points: i) the diffuse sensor, called *s_in* (entering the sensing unit); the gate actuator, called *s_out* (leaving the sensing unit); the first actuator, called *n1* (pushed to the first collection section); the second actuator, called *n2* (pushed to the second collection section); and the retroreflective sensor, called *w* (workpiece stored in the slope). From each point, we get the timestamp and just at the end of the sensing process, we also get the colour from the combination of the sensors' output. Table 1 shows the combination of each colour based on the sensors' output. At each of the five points mentioned, the KG receives the data points. A simulated system has been created in the AnyLogic tool (https://www.anylogic.com/). The system sends the workpiece ID to the simulation model, which simulates the operations of the actual system for that workpiece providing the expected result. This simulated result is compared with the real-time status of the Festo sorting station for validation of its performance. In case of discrepancy, a suitable corrective measure may be applied to the physical system.

4. EXPERIMENTAL RESULTS

The CT described in the previous section, has been tested in a workstation with a CPU Intel Core i7-8750H and 16GB of

Table 1 - Sensors' output combination. (DS = Diffuse sensor, DCS = Diffuse colour sensor, IPS = Inductive proximity sensor).

	DS	DCS	IPS
Black	1	0	0
Red	1	1	0
Metal	1	1	1

Figure 4 - Snapshot of the KG containing data related to workpiece ID001 in space s_in

RAM. Figure 4 shows a snapshot of the KG. As can be seen, BFO, CCO, and the IOF-core properties are imported together with the system ontology. Following the results from the CQs asked the KG:

1. What are the components of the sorting station?

	cl
1	"Sorting Unit Festo Sorting Station SN000"
2	"Translation Unit Festo Sorting Station SN000"
3	"Sensing Unit Festo Sorting Station SN000"
4	"Conveyor Festo Sorting Station SN000"
5	"Pushing Actuator Festo Sorting Station SN000"
6	"Diffuse Color Sensor Festo Sorting Station SN000"
7	"Diffuse Sensor Festo Sorting Station SN000"
8	"Inductive Proximity Sensor Festo Sorting Station SN000"
9	"Gate Actuator Festo Sorting Station SN000"

2. How long did the conveyor take to carry the workpiece X?

	w	p	time1	time2
1	:WFMPS001	_:node26516	"2022-02-24T16:01:26"	"2022-02-24T16:01:29"

3. What is the time-point that the workpiece X entered at the sensing unit?

	w	time	s
1	:WFMPS001	"2022-02-24T16:01:14"	http://cranfield.ac.uk/cdem/festo/Ss_in

4. What is the colour of workpiece X sensed after the sensing unit?

	w	s	color
1	:WFMPS001	http://cranfield.ac.uk/cdem/festo/Ss_out	"red"

5. How many workpieces are at the collection points X at time T?

	result
1	"12"^^xsd:integer

6. Is workpiece X sensed by the sensing unit in the time interval T?

Query took 0.1s, moments ago. YES

4.1 Detection of malfunction

A test scenario has been given below to demonstrate the reasoning to detect the malfunctioned component for a system error.

The machine monitoring status in GraphDB is compared with the simulation result. It can be seen that several workpieces are missing from collection point 1 as the number of workpieces in collection point 1 (CQ 5) is less than the number of workpieces simulated for the same collection point. Therefore, the system-level error is marked by the comparison script as shown in Figure 5, where the workpiece with ID 004 is one of the missing workpieces from collection point 1.

	RedSortingUnit	ActOfSortingRed	workpiece	malfunction
1	_:node26827	_:node26826	:WFMPS004	_:node26828

Figure 5 - malfunctioning of Red Sorting Unit

When the rules given from (4) to (7) are run on the KG, rule (5) is triggered because the *ActOfCarryingRed* realises the function of the *Conveyor* as given by the state description in Figure 6 (Workpiece 004 carried from *s_out* to *w*). Rule (5) will assert that the *RedTransferringUnit* has malfunctioned. Next, this assertion triggered only rule (6) as it is found that the output measurement is 'black' instead of 'red' and that violates the (3) as specialized for *ActOfActuatingRed*, resulting in the function of *SensingUnit* not being realized. Rule (6) will assert the *SensingUnit* as the malfunctioned unit as shown in Figure 7.

	Workpiece	Time	Location	Site
1	:WFMPS004	"2022-02-24T16:01:57"	http://cranfield.ac.uk/cdem/festo/Ss_in	http://cranfield.ac.uk/cdem/festo/site_A004
2	:WFMPS004	"2022-02-24T16:02:12"	http://cranfield.ac.uk/cdem/festo/Ss_out	http://cranfield.ac.uk/cdem/festo/site_A004
3	:WFMPS004	"2022-02-24T16:02:16"	http://cranfield.ac.uk/cdem/festo/Sw	http://cranfield.ac.uk/cdem/festo/site_C004

Figure 6 - State description of workpiece 004

	RedSensingUnit	ActOfMeasuringRed	Workpiece	Malfunction
1	_:node26846	_:node26847	:WFMPS004	_:node26848

Figure 7 - Malfunctioning of sensing unit

5. CONCLUSION

The work presented in this paper is part of an ongoing project and all the components of the CT are frequently updated. It aims to demonstrate the basic workflow of applying the CT to support the management and the health monitoring of an asset, specifically for a material handling system. Several pending issues need to be addressed before fully realising the expected target.

The paper shows how the CT approach helps to better define the knowledge about the system, using the foundational ontology and performing reasoning to support the decision-making process. The ontology model of the CT is built using systems engineering paradigms, such as functional decomposition, function flow, material flow, energy flow, state-based activity analysis, and causal analysis.

Future works include completing the development of the CT, testing it, and validating it for different kinds of malfunctions and operating conditions. Another future opportunity would be applying the same methodology to another kind of asset, in a different domain and validating both the CT performance as well as the semantic interoperability that the TLO approach claim.

ACKNOWLEDGMENT

This work was funded by the EPSRC as part of the DigiTOP project. https://digitop.ac.uk; EP/R032718/1.

REFERENCES

Akroyd, J., Mosbach, S., Bhave, A., & Kraft, M. (2021). Universal Digital Twin-A Dynamic Knowledge Graph. *Data-Centric Engineering*, *2*, 14.

D'Amico, R. D., Addepalli, S., & Erkoyuncu, J. A. (2021). Is a top level ontology based digital twin the solution to human-machine interoperability? *10th International Conference on Through-Life Engineering Service*, 16–17.

Erkoyuncu, J. A., del Amo, I. F., Ariansyah, D., Bulka, D., Vrabič, R., & Roy, R. (2020). A design framework for adaptive digital twins. *CIRP Annals*, *69*(1), 145–148.

Grieves, M. W. (2015). Digital Twin : Manufacturing Excellence through Virtual Factory Replication This paper introduces the concept of a A Whitepaper by Dr . Michael Grieves. *White Paper*, *March*.

Grüninger, M., & Fox, M. S. (1995). *The Role of Competency Questions in Enterprise Engineering*. 22–31.

Jones, D., Snider, C., Nassehi, A., Yon, J., & Hicks, B. (2020). Characterising the Digital Twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, *29*, 36–52.

Karray, M. H., Otte, N., Kiritsis, D., Ameri, F., Kulvatunyou, B., Will, C., Arista, R., Rai, R., & Smith, B. (2021). The Industrial Ontologies Foundry (IOF) perspectives. *Industrial Ontology Foundry (IOF) - Achieving Data Interoperability Workshop, International Conference on Interoperability for Enterprise Systems and Applications*.

Lu, J., Zheng, X., Gharaei, A., Kalaboukas, K., & Kiritsis, D. (2020). Cognitive Twins for Supporting Decision-Makings of Internet of Things Systems. *Lecture Notes in Mechanical Engineering*, 105–115.

Petrova-Antonova, D., & Ilieva, S. (2021). Digital twin modeling of smart cities. *Advances in Intelligent Systems and Computing*, *1253 AISC*, 384–390.

Poli, R., Healy, M., & Kameas, A. (2010). Theory and applications of ontology: Computer aplications. *Theory and Applications of Ontology: Computer Applications*, 1–576.

Sarkar, A., Šormaz, D., Koonce, D., & Farah, S. (2020). Developing a Resource-Based Manufacturing Process Capability Ontology. *Communications in Computer and Information Science*, *1407 CCIS*, 293–306.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S. A., Scheuermann, R. H., Shah, N., Whetzel, P. L., & Lewis, S. (2007). The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. In *Nature Biotechnology* (Vol. 25, Issue 11, pp. 1251–1255).

Suh, N. P. (1998). Axiomatic Design Theory for Systems. *Research in Engineering Design*, 189–209.

Tao, F., & Qi, Q. (2019). Make more digital twins. *Nature 2021 573:7775*, *573*(7775), 490–491.

Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics*, *15*(4), 2405–2415.

2022-10-26

Detecting failure of a material handling system through a cognitive twin

D'Amico, Davide R.

Elsevier

https://doi.org/10.1016/j.ifacol.2022.10.128