Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

A Knowledge Based System for Managing Heterogeneous Sources of Engineering Information ⋆

Felix Ocker * **Birgit Vogel-Heuser** * **Matthias Seitz** *
Christiaan J. J. Paredis **

* *Institute of Automation and Information Systems, Technical University of Munich, Garching, 85748 Germany (e-mail: {felix.ocker; vogel-heuser; matthias.seitz}@tum.de)*
** *BMW Chair in Systems Integration, Clemson University, Greenville, SC 29607 USA (e-mail: paredis@clemson.edu)*

Abstract: As ever increasing amounts of engineering information become available, engineers require novel ways to manage this information. Especially in mechatronic engineering, e.g., the engineering of Cyber-Physical Production Systems, engineers from various disciplines are involved, and they represent their knowledge in heterogeneous ways. To be able to efficiently gain an overview of the available information and find the information required, engineers need support. Additionally, awareness regarding interdisciplinary dependencies should be increased to reduce late changes. Such dependency analyses allow engineers to better identify and thus manage potential inconsistencies, which can be expected to reduce development time. The Knowledge Based System presented supports engineers during the design process of production systems by providing an overview of available engineering knowledge, its representation, and associated stakeholders. Additionally, engineers can leverage the underlying holistic information model to create and manage digital twins.

Keywords: Knowledge Based Systems; Mechatronic Systems; Modeling; Information Model; Digital Twin

1. INTRODUCTION

Digitization leads to an increasing amount of information becoming easily available. This is valid specifically for production systems engineering, which is subject to a continuous increase in complexity and specialization, leading to various stakeholders with heterogeneous views being involved. For the knowledge intensive domains of business in an international context, such as engineering, digitization is a promising development. This is because the exchange of information is greatly facilitated even across the globe, and automated processing of the information can, in principle, be realized. Also, advanced technological ideas can be supported, such as the concept of a digital twin, i.e., a "dynamic virtual representation of a physical object or system across its lifecycle" (Bolton et al., 2018). However, such large amounts of heterogeneous information may be overwhelming if not managed properly. This makes purposeful information management crucial not only for interdisciplinary engineering, but also for distributed partial digital twins, which have to be integrated to mirror a larger system. An information model provides the means to cope with several resulting challenges. First, it allows engineers to find information when many stakeholders with heterogeneous backgrounds are involved *(C1)*. Second, it enables dependency analyses among different information concretizations, such as files *(C2)*. This is valuable because duplicate information and dependencies between information, i.e., information that is stored in several information concretizations, may result in inconsistencies which need to be managed. Third, the information model supports compatibility checks between the form in which information is available and the available tools *(C3)*. The form is characterized by the language the information is expressed in and the format it is stored in. This is particularly important whenever information is exchanged across the supply chain. In such cases, the different stakeholders might well use very different engineering tools, resulting in the information being expressed in diverse ways. Fourth, the information model can increase awareness of how changes propagate *(C4)*. These insights may be used to inform other stakeholders about changes that affect their design decisions. Finally, the information model serves as a basis on which digital twins may be built.

This paper is structured as follows. Section 2 gives an overview of related work. Subsequently, we introduce a Knowledge Based System (KBS) to support engineers that revolves around an information model. Section 4 gives details on the use case and the implementation, describes the feasibility study, and discusses the results. This paper concludes with a summary and an outlook in Section 5.

⋆ This work was supported in part by the BMBF via the project DAVID and in part by the German Research Foundation via the project CRC 768 T5.

Table 1. Overview of notions used in related work

	Ceusters and Smith (2015)	Giese et al. (2011)	Bézivin (2005)
information	information content entity	abstract syntax model	information
information concretization	information quality entity	concrete syntax model	model
information carrier	information artifact	-	-
format	-	concrete syntax	-
language	-	language	language
actor	agent	-	-
system	(entity)	system	system (/ object)

2. RELATED WORK

This section defines the term "information model" for the scope of this work and gives an overview of information created throughout the design process of production systems. Subsequently, it introduces a more philosophical perspective on information models and concludes with a discussion of concrete examples of information models from the engineering domain.

A model, as understood by Pidd (2003), is a "representation of reality intended for some definite purpose". Models are often used in systems engineering, with a system being defined as a "combination of interacting elements organized to achieve one or more stated purposes" (Haskins, 2006). This kind of representation typically comes with some form of abstraction, i.e., reduction to what is considered important. In software engineering, models are usually classified into structural, behavioral, and information models (Bourque and Fairley, 2014, p. 9-5). While structural models describe the physical or logical composition of a system's elements, behavioral models define the functions of the same system. In contrast, an information model is "an abstract representation that identifies and defines a set of concepts, properties, relations, and constraints on data entities". Here, we think of information as know-what (Rowley, 2007). This is contrasted by the terms "data", which are symbols, and "knowledge", which refers to know-how, i.e., the application of information (Rowley, 2007). The additional layer of "wisdom", as proposed by Rowley (2007) is still controversial (Fricke, 2009).

In systems engineering in general, and in production systems engineering in particular, vast amounts of information are created in the design process. This trend has been intensified with the spread of effective Model Driven Engineering (MDE) approaches such as the one by Priego et al. (2015). Feldmann et al. (2019) highlight the heterogeneity of the stakeholders involved in the design process of Cyber-Physical Production Systems (CPPSs), and give an overview of the information created. This information includes, but is not limited to, requirements, Systems Modeling Language (SysML) models, Simulink models, P&IDs, data sheets, sensor and actuator lists, mechanical 3D models, and, e.g., state charts to describe the system's intended behavior. Koltun et al. (2019) identified 47 models expressed in diverse languages and stored in various ways, which had been developed over the course of twelve years within a single collaborative research center. To cope with this degree of heterogeneity, Feldmann et al. (2019) propose an approach to managing inconsistencies specifically between SysML models, Simulink models, and planning models.

From a more philosophical perspective, we want to highlight three papers we consider to be crucial for this work. Ceusters and Smith (2015) developed the Information Artifact Ontology (IAO) as a "domain-neutral resource for the representation of types of information content entities such as documents, data-bases, and digital images". The IAO core notion is the *information content entity*, which *is about*, i.e., describes, some entity. According to Ceusters and Smith (2015), *information artifacts* bear the concretizations of information content entities. With a background in Model Based Systems Engineering (MBSE), Giese et al. (2011) present the results of the Workshop on Multi-Paradigm Modeling. They distinguish *abstract syntax models*, which contain the essence of a model, from *concrete syntax models*, which correspond best to the concretizations of information content entities. A *metamodel* is understood as a model of a language which does not cover semantics (Giese et al., 2011). Bézivin (2005) elaborates on MDE and highlights the differences between the core notions in object technology and MDE. While object technology is based on inheritance and instantiation, MDE builds on the idea that a system is represented by a model, which in turn conforms to a metamodel. The relation *represented by* is inverse to the relation *is about* used by Ceusters and Smith (2015). It should be noted that literature with a background in MBSE focuses on models, but does not consider information carriers. To increase the understandability of the notions chosen for the information model developed here, we give an overview of how these notions fit in with the previous work in Table 1.

In contrast to these rather philosophical papers, some more specific frameworks have also been proposed to address the challenges *C1* - *C4* for several domains. The most generic one is possibly the lifecycle record of technical objects (DIN, 2018) which proposes a holistic approach to manage information created throughout the design of technical systems. The lifecycle record intends to capture information contained in document and data sets, and the physical objects, i.e., plants and their parts, described by this information. The lifecycle record also aims to support different viewpoints and suggests structuring principles to check the completeness of information, which distinguishes it from the approach pursued in this work. Focusing on the representation of physical systems, the OPC Foundation has developed the OPC UA information model (OPC Foundation, 2017). In contrast to this work, the OPC UA information model is primarily applied within runtime applications, though, e.g., using digital twins (Malakuti et al., 2019). Also in the production context, Petersen et al. (2017) propose an information model with a focus on the system description based on the Resource Description Framework (RDF). They focus on two use cases, tool

management and energy consumption, and three data sources, namely sensor data, bills of materials, and data from the Manufacturing Execution System (MES). In the domain of space missions, Jenkins and Rouquette (2012) and Rouquette et al. (2005) use an implicit information model to support multi-view engineering and automate inconsistency management. In order to support engineers in the design of mechatronic products, Hehenberger et al. (2010) propose an approach that relies on hierarchical design models. However, this design process is not combined with an information model that would link the various models developed. Rooted in software development, the Open Services for Lifecycle Collaboration (Johnson and Speicher, 2013) also contributes to the integration of heterogeneous engineering tools *(C2, C3)*. In production systems engineering, model-based frameworks have been developed, e.g., Priego et al. (2015), which support engineers along the design process of such systems. Hildebrandt et al. (2018) present an approach to systematic modeling for creating a common understanding of information exchanged between technical systems. To do so, Hildebrandt et al. (2018) distinguish and link behavioral, functional, and structural information, but do not consider how and where this information is originally captured. To enable interdisciplinary data exchange, Lüder et al. (2019) suggest using AutomationML (AML). AML is a valuable contribution to the community but requires that all data is represented in supported formats. In contrast to these approaches, Kattner et al. (2019) address the challenge of dependencies among heterogenous models by manually mapping them.

Various frameworks have been proposed in the context of challenges *C1* through *C4*, some of which even introduce information models. However, none of them can be considered to fully address all four challenges, especially in the context of digital twins. Also, the domain specific information models, which focus on the system, have not yet been intertwined with the more philosophical perspective on information. With this paper, we intend to contribute to closing this gap.

3. KNOWLEDGE BASED SYSTEM FOR MANAGING HETEROGENEOUS ENGINEERING INFORMATION

The KBS developed here relies on an information model that links the heterogeneous information sources, cp. Figure 1. The information model revolves around the notions defined in the following. To achieve a common understanding of the terminology, we compared these with the notions used in existing work, cp. Table 1. The second part of this section describes the patterns we use to address challenges *C1* through *C4*.

3.1 Knowledge Representation

The core notion of the information model is, unsurprisingly, *information*. This quite abstract term is understood as defined by Merriam-Webster (2019):

INFORMATION =*def.* "the communication or reception of knowledge" (Merriam-Webster, 2019).

An alternative but not exclusive definition is that of information being "know what", i.e., information provides

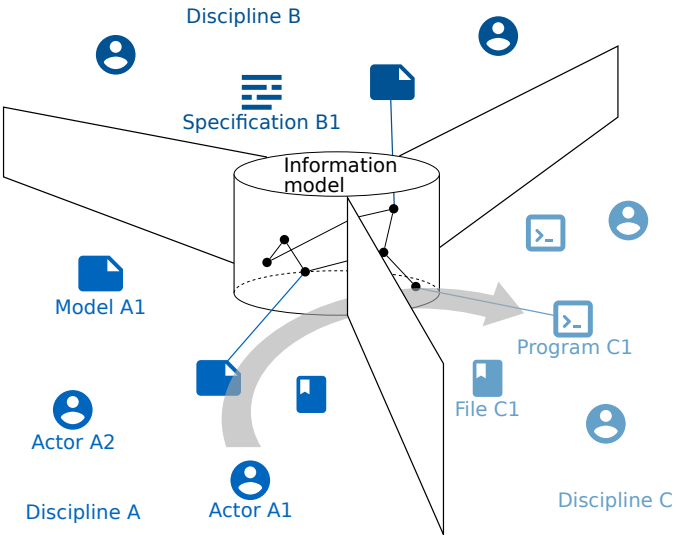

Fig. 1. KBS for representing information from heterogeneous sources

answers to who, what, where and when questions (Rowley, 2007). This notion corresponds to that of an information content entity defined by Ceusters and Smith (2015) and the idea of an abstract syntax model (Giese et al., 2011). We further distinguish structural and behavioral information, e.g., the structure of a production system and the processes it is able to execute, respectively. Information itself is abstract and can be expressed in multiple ways, i.e., using different *languages*. For instance, the behavioral information of a robot's movements may be expressed using state charts or the natural language English.

LANGUAGE =*def.* set of valid sentences that can be used to concretize INFORMATION.

In order to represent how information is expressed, we introduce the notion *information concretization*. An information concretization is *expressed in* a language and may be *stored* using some specific format. Examples of information concretizations are digital files, but also mental models existing only in human minds.

INFORMATION CONCRETIZATION =*def.* concretization of some INFORMATION, i.e., some INFORMATION being expressed in some LANGUAGE and possibly serialized in some FORMAT.

Information concretizations are further described by their status, which may be complete or incomplete, and a timestamp, which indicates when the information concretization was created or changed last. In case an information concretization builds on a previous version, this may be expressed using the relation *predecessor*.
When thinking of information being stored digitally, we also need to distinguish different formats. For example, an ontology expressed in the Web Ontology Language (OWL) may be stored using RDF/XML or it may be serialized using Terse RDF Triple Language (Turtle). This is equivalent to the notion of a concrete syntax (Giese et al., 2011). Note that a format may be *based on* another format, e.g., RDF/XML is based on XML.

FORMAT =*def.* the concrete syntax (Giese et al., 2011) in which some INFORMATION CONCRETIZATION is stored.

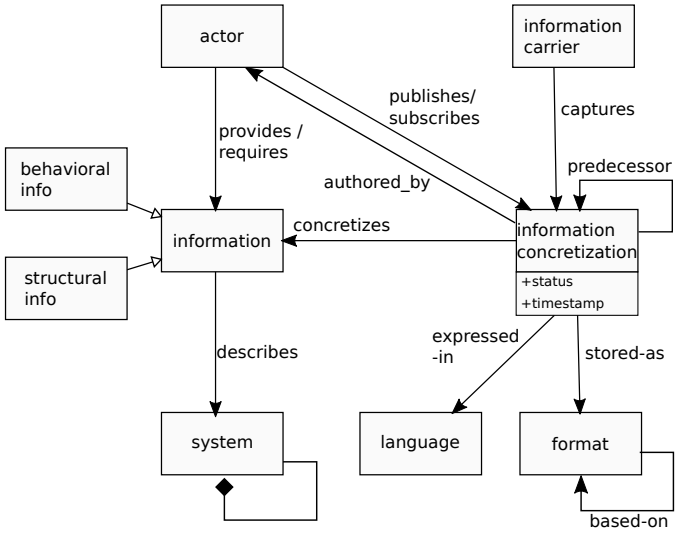

Fig. 2. Class diagram of the information model

In order to describe where an information concretization is captured, we use the notion *information carrier*.

INFORMATION CARRIER =*def.* physical artifact that *captures* some INFORMATION CONCRETIZATION.

Information carriers include devices such as hard drives, but a stone wall or even the mind of a person could be called examples of information carriers. Hence, the notion of an information carrier is used as a surrogate for these notions and many others.

Information is *provided* or *required* by actors. For instance, humans and software programs can be actors. In order to keep track of information, actors may *subscribe* to information concretizations. When providing information, actors do so by *publishing* information concretizations. Note that the relation *publishes* is inverse to the relation *authored by*.

ACTOR =*def.* anyone and anything that provides or requires information.

Within the scope of this work, we focus on information that is used to *describe* systems.

SYSTEM =*def.* "combination of interacting elements organized to achieve one or more stated purposes" (Haskins, 2006).

Figure 2 gives an overview of these notions.

3.2 Knowledge Extraction

Engineers may extract the knowledge necessary to cope with challenges *C1 - C4* when having a knowledge graph at hand that conforms to the information model developed.

The information model allows engineers to systematically identify relevant information. Knowledge extraction is especially useful to engineers in two scenarios. First, engineers can more easily identify existing information they require in the design process. Second, they can use the targeted search for information to configure bespoke partial digital twins. Such partial digital twins can be applied in environments, where the computational power is limited. In order to identify relevant information concretizations *?ic* created by others *(C1)*, engineers can search the information model based on the type of information *info-type-x* they need, i.e., structural or behavioral information, and the system *system-x* described. These restrictions are expressed in the following pattern, which also includes the information carriers in which the information concretizations are captured.

$\forall$ (?ic, ?carrier)
  $\exists$?info *is-a* INFO-TYPE-X
$\ni$?ic *concretizes* ?info $\sqcap$
  ?ic *is-a* INFO-CONCRETIZATION $\sqcap$
  ?info *describes* system-x $\sqcap$
  ?carrier *is-a* INFORMATION-CARRIER $\sqcap$
  ?carrier *captures* ?ic

This pattern may be extended by explicitly searching for information expressed in some specific language.
Similarly, engineers can identify information carriers *?ic* that capture the same information, even though this information may be expressed in different ways *(C2)*. Such duplicate information is problematic because it is likely to result in inconsistencies if not all concretizations are updated in case of changes. To reduce this risk, engineers can identify and handle unnecessary duplicates of information, or at least monitor them. The following expression also includes the information carriers *?carrier* by which these information concretizations are captured.

$\forall$?duplicate-info
  $\exists$?ic *is-a* INFORMATION-CONCRETIZATION $\sqcap$
  $\exists$?carrier *is-a* INFORMATION-CARRIER
$\ni$?ic *concretizes* ?duplicate-info $\sqcap$
  count(?ic) > 1 $\sqcap$
  ?carrier *captures* ?ic $\sqcap$
  ?duplicate-info *is-a* INFORMATION

This second expression can be extended for chunks of information that depend on each other. Whenever two chunks of information describe the same system and are of the same type, they are likely to overlap. In such cases it becomes apparent that the KBS proposed is not a monolithic structure, but would benefit greatly from being combined with established approaches. For instance, when such dependencies are detected, established approaches for managing potential inconsistencies should be considered, e.g., the one introduced by Feldmann et al. (2019).

Also, compatibility between the formats of the information concretizations available in an organization and its available tools can be checked *(C3)*. We hereby distinguish two cases. There might be formats, for which there are no appropriate tools available, or the company in question does not have the necessary licenses. Either way, the organization cannot use the inaccessible information concretization *?inacc-ic*.

$\forall$?inacc-ic
$\nexists$?organization *has-license-for/supports* ?tool
$\ni$?inacc-ic *expressed-as* ?format $\sqcap$
  ?inacc-ic *is-a* INFO-CONCRETIZATION

This expression may be complemented by the following optional triple.

  ?tool *supports* ?format

That way, tools can be identified that support the format in question, but for which the organization does not yet have licenses.

These compatibility checks are valuable when multiple actors are involved in the design process who do not use the same tools. This is often the case for production systems, which are usually created by a complex supply chain. Ensuring that all information is available allows engineers to focus on engineering methods instead of having to deal with tooling problems. Also, when complex digital twins are created from partial ones, the information concretizations may be heterogeneous but must be integrated to work together seamlessly.

To cope with challenge *C4*, engineers can use the information model to track how changes propagate, even across information concretizations and disciplines. Using this information, other actors who are affected by this change can be identified and subsequently notified. For this to work, we rely on the publisher/ subscriber model. Engineers can manually subscribe to the information concretizations *?ic* they deem to be relevant for their own work. As soon as a new information concretization *info-concretization-x* is published, all the actors are identified who subscribed to the information concretizations that concretize the same chunk of information.

$\forall$ (?affected-actor, ?ic)
  $\exists$?info *rdf:type/rdfs:subClassOf** INFORMATION
$\ni$?affected-actor *rdf:type/rdfs:subClassOf** ACTOR $\sqcap$
  ?affected-actor *subscribes* ?info-concretization $\sqcap$
  ?info-concretization *concretizes* ?info $\sqcap$
  info-concretization-x *concretizes* ?info

The actors identified can then be notified efficiently. Additionally, we suggest automatically creating a subscription whenever an engineer provides or changes some information concretization.

4. IMPLEMENTATION AND FEASIBILITY STUDY

4.1 Use Case

Within the feasibility study, we use the lab-sized demonstrator Extended Pick and Place Unit (xPPU). The original Pick and Place Unit (PPU) comprises a stack, a crane, a stamp, and a conveyor with ramps and pushers. The demonstrator stamps and sorts white and black plastic as well as metallic workpieces. The stack stores the workpieces until they are picked up by the crane. The crane can pick up and move single workpieces between the stack, the stamp, and the conveyor. When the crane places a workpiece on the conveyor, the conveyor moves it to the ramp specified. When recognized by the sensors, the workpiece is pushed into the ramp by a cylinder. To distinguish workpieces of different colors and different materials, the PPU uses optical and inductive sensors. A detailed description of the PPU and its evolution scenarios can be found in the technical report (Vogel-Heuser et al., 2014). By now, the PPU has evolved into the xPPU, cp. Figure 3.

In contrast to the PPU, the xPPU features a more advanced transport system which consists of conveyors for re-feeding of workpieces and a linear handling module. To create a realistic demonstrator, the xPPU also includes a weighing module and safety cells. Various documentation for the xPPU, including code, is available publicly [1].

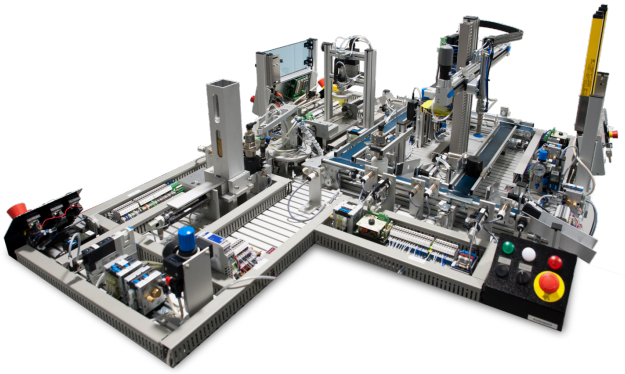

Fig. 3. Extended Pick and Place Unit (Vogel-Heuser et al., 2014)

4.2 Implementation

We formalized the information model in OWL using the python projects Owlready2 [2] and RDFlib [3]. These interfaces also allowed us to create configurable SPARQL And RDF Query Language (SPARQL) queries.

In order to increase reusability, we split the knowledge graph into a generic part that includes the notions depicted in Figure 2 and a more specific part which captures all the notions and instances needed for the feasibility study. All the queries developed are also available as individual python implementations to improve modularization. The entire implementation is available on github [4].

We acknowledge that domain experts might have trouble using Semantic Web Technologies (SWT) and the python implementation. To enable these engineers to use this KBS nonetheless, we implemented a prototypical interactive user interface using the python module tkinter.

4.3 Feasibility Study

Various engineering information is available for the xPPU. This makes it not only a realistic example, but also a suitable use case for the KBS presented. We exemplarily represented information necessary for the feasibility study as described in Section 4.2.

Usually, the purchasing department needs structural information of the plant to create a Bill of Materials (BOM). In case of the xPPU, such information is available from early design phases in the form of Unified Modeling Language (UML) class diagrams. These could easily be reused, but the purchasing department may be unaware of this information. Using the query developed for *C1*, employees from the purchasing department can purposefully search

[1] https://https://github.com/x-PPU last accessed: 2020-05-06
[2] https://bitbucket.org/jibalamy/owlready2/src/default/ last accessed: 2020-05-06
[3] https://github.com/RDFLib/rdflib last accessed: 2020-05-06
[4] https://github.com/felixocker/information-model-ifac2020 last accessed: 2020-05-06

for structural information of the xPPU and will find the UML class diagrams created in Eclipse Papyrus.

Just like in industrial practice, some of the information describing the xPPU is available in the form of several information concretizations. Specifically, the crane's behavior has been prescribed using UML activity diagrams, but was also implemented in IEC 61131-3 Structured Text using the CODESYS development system and TwinCAT. Additionally, PLCopen XML exports are available. To make things worse, some of these information concretizations may be captured in different information carriers. These redundancies may lead to potential inconsistencies and should be resolved whenever possible. If this is impossible, they should at least be monitored. Such monitoring can be achieved using the query developed for *C2*. For instance, the behavior of the xPPU's crane is concretized as a UML activity diagram, and in the form of code as Structured Text created with the CODESYS development system, created with TwinCAT, and in the exchange format PLCopen XML. Having identified these duplicates of information, engineers can automate transformations to avoid inconsistencies between the information concretizations.

An example for challenge *C3*, i.e., compatibility between formats of available information concretizations and available tools, results from proprietary tools and formats. For instance, suppliers might have created mechanical Computer Aided Design (CAD) files for the xPPU's crane in CATIA, and made them available as CATProduct files. If the institute does only have licenses for Inventor, they would not be able to use these CAD files, even though they are in principle available. By detecting such incompatibilities in format, suppliers can be encouraged to use other tools or provide the information concretizations in exchange formats, e.g., the Standard for the Exchange of Product Model Data (STEP). This challenge also highlights the importance of research efforts regarding such exchange formats for code, cp., e.g., Marcos et al. (2009).

The behavior of the xPPU's crane also shows how different engineers work with different information concretizations that depend on each other. However, the engineers may be unaware of the changes their colleagues make. This problem is reinforced when several actors along the supply chain work on dependent information concretizations. For instance, an engineer may change the UML diagrams that prescribe the crane's intended behavior. If that engineer is unaware of all the implementations created with the different Integrated Development Environments (IDEs), it can be assumed that these changes will not be incorporated by all implementations. As a consequence, the different implementations might behave in different ways leading to unexpected behavior of the plant. The KBS developed includes a query to identify all the actors, who are affected by changes to a specific information concretization. In case of the xPPU, a fictional employee named Eve might have subscribed to the crane's behavior expressed in Structured Text using CODESYS, while both Matthias and Felix subscribed to the crane's behavioral description expressed as a UML activity diagram. If other actors change the UML activity diagram, all three employees are identified as being affected by the change. This allows engineers who make such changes to efficiently notify their peers.

4.4 Discussion

As demonstrated by the feasibility study, the KBS can be expected to support engineers in coping with challenges *C1 - C4*. This way, the KBS contributes to both the interdisciplinary engineering and the creation of partial and distributed digital twins. By aligning the information model with abstract concepts from the modeling and the ontology community increases its potential for usability and reusability. Also, this degree of genericness greatly reduces limitations, e.g., to represent information concretized in the form of mental models in the minds of people.

Creating a single but rather generic KBS also allows engineers to combine it with existing approaches for the individual challenges. An efficient way to find relevant engineering information *(C1)* can be expected to be useful for various scenarios, including feasibility feedback for designers in early design phases (Ocker et al., 2019). Duplicate information identified *(C2)* can be used as a basis for more rigorous inconsistency management approaches, e.g., (Feldmann et al., 2019). Regarding change management, the KBS could be combined with existing approaches for change management, such as the one proposed by Rostami et al. (2017).

Even though this KBS can be considered very promising for the challenges specified, the creation of the ontology's assertional component remains as a core challenge. While we populated the information model manually, this would probably not be feasible for an industry-scale application. Hence, further research should be conducted regarding the automatic population of the information model. Here, different kinds of information pose different challenges. Lists of human actors can be reused from the human resources departments, and both authors and timestamps of information concretizations are meta data that can be accessed easily. Also, a file system crawler could analyze file extensions to automatically relate information concretizations with certain tools. However, linking information concretizations with abstract information is more challenging. Even though natural language processing seems promising for analyzing textual information concretizations, we might still require manual input of the engineer responsible.

5. SUMMARY AND OUTLOOK

This paper presents a KBS for supporting production systems engineering and the creation of partial, possibly distributed, digital twins using an information model. Specifically, the four challenges *C1 - C4* are addressed to support engineers in coping with the vast amounts of heterogeneous information available nowadays. The information model developed is aligned with more abstract terminology to enable reuse and close the gap between the philosophical perception of information and application oriented models from MBSE. We formalized the information model using OWL, and we populated it exemplarily for the use case of the xPPU. To extract relevant information, we implemented queries using SPARQL.

In future work, integration with established approaches for coping with the challenges *C1 - C4* should be pursued, as

described in Section 4.4. Also, further research is necessary to automate the population of the information model. Additionally, variant management and related similarity analyses in the scope of the information model should be researched in greater detail. From a more application oriented perspective, the information model should be extended to also support status checks of the information created along the design process of production systems and their digital twins. To do so, engineers would have to specify when information concretizations are created. Combined with milestones and status information, this would allow managers to monitor the project's progress. Relevant information can be extracted, e.g., from existing process descriptions in the form of Business Process Model and Notation (BPMN) diagrams. Finally, extracting and formalizing heterogeneous kinds of engineering knowledge and integrating it with online data from production systems is a promising and necessary research direction in the context of digital twins.

REFERENCES

Bézivin, J. (2005). On the unification power of models. *Software and Systems Modeling*, 4(2), 171–188.

Bolton, R.N., McColl-Kennedy, J.R., Cheung, L., Gallan, A., Orsingher, C., Witell, L., and Zaki, M. (2018). Customer experience challenges: bringing together digital, physical and social realms. *Journal of Service Management*, 29(5), 776–808.

Bourque, P. and Fairley, R.E. (eds.) (2014). *Guide to the Software Engineering Body of Knowledge SWEBOK. A Project of the IEEE Computer Society*. IEEE, 3 edition.

Ceusters, W. and Smith, B. (2015). Aboutness: Towards Foundations for the Information Artifact Ontology.

DIN (2018). Lifecycle record of technical objects - Part 1: Structural and content-related specifications.

Feldmann, S., Kernschmidt, K., Wimmer, M., and Vogel-Heuser, B. (2019). Managing inter-model inconsistencies in model-based systems engineering: Application in automated production systems engineering. *Journal of Systems and Software*, 153, 105–134.

Fricke, M. (2009). The Knowledge Pyramid: a Critique of the DIKW Hierarchy. *Journal of Information Science*, 35(2), 131–142.

Giese, H., Levendovszky, T., and Vangheluwe, H. (2011). Summary of the workshop on multi-paradigm modelling: Concepts and tools. *Models in Software Engineering*.

Haskins, C. (2006). *Systems engineering handbook*. INCOSE, 3 edition.

Hehenberger, P., Poltschak, F., Zeman, K., and Amrhein, W. (2010). Hierarchical design models in the mechatronic product development process of synchronous machines. *Mechatronics*, 20(8), 864–875.

Hildebrandt, C., Scholz, A., Fay, A., Schröder, T., Hadlich, T., Diedrich, C., Dubovy, M., Eck, C., and Wiegand, R. (2018). Semantic modeling for collaboration and cooperation of systems in the production domain. In *International Conference on Emerging Technologies and Factory Automation*. IEEE.

Jenkins, J.S. and Rouquette, N.F. (2012). Semantically-Rigorous Systems Engineering Modeling Using SysML and OWL. In *International Workshop on Systems & Concurrent Engineering for Space Applications*.

Johnson, D. and Speicher, S. (2013). Open Services for Lifecycle Collaboration - Core Specification Version 2.0. Technical report, Open Services for Lifecycle Collaboration.

Kattner, N., Bauer, H., Basirati, M.R., Zou, M., Brandl, F., Vogel-Heuser, B., Böhm, M., Krcmar, H., Reinhart, G., and Lindemann, U. (2019). Inconsistency Management in Heterogeneous Models - An Approach for the Identification of Model Dependencies and Potential Inconsistencies. *International Conference on Engineering Design*, 3661–3670.

Koltun, G.D., Romero Viturro, C.A., Buchholz, J., Wissel, J., Zaggl, M., Ocker, F., and Vogel-Heuser, B. (2019). Effective Innovation Implementation of Mechatronic Product-Service Systems Considering Socio-Technical Aspects. *International Conference on Engineering Design*, 3051–3060.

Lüder, A., Pauly, J.L., Rinker, F., and Biffl, S. (2019). Data Exchange Logistics in Engineering Networks Exploiting Automated Data Integration. In *International Conference on Emerging Technologies and Factory Automation*. IEEE.

Malakuti, S., Schmitt, J., Platenius-Mohr, M., Grüner, S., Gitzel, R., and Bihani, P. (2019). A four-layer architecture pattern for constructing and managing digital twins. In *European Conference on Software Architecture*. Springer Verlag.

Marcos, M., Estevez, E., Perez, F., and Van Der Wal, E. (2009). XML exchange of control programs. *IEEE Industrial Electronics Magazine*, 3(4), 32–35.

Merriam-Webster (2019). Definition of Information.

Ocker, F., Vogel-Heuser, B., and Paredis, C.J.J. (2019). Applying Semantic Web Technologies to Provide Feasibility Feedback in Early Design Phases. *JCISE*, 19(4).

OPC Foundation (2017). OPC Unified Architecture - Part 5: Information Model. Technical report.

Petersen, N., Halilaj, L., Grangel-González, I., Lohmann, S., Lange, C., and Auer, S. (2017). Realizing an RDF-based information model for, a manufacturing company – A case study. In *International Semantic Web Conference*. Springer Verlag.

Pidd, M. (2003). *Tools for thinking: Modeling in Management Science*. John Wiley and Sons Ltd, New York, USA, 2nd edition.

Priego, R., Armentia, A., Estévez, E., and Marcos, M. (2015). On applying MDE for generating reconfigurable automation systems. In *International Conference on Industrial Informatics*. IEEE.

Rostami, K., Heinrich, R., Busch, A., and Reussner, R. (2017). Architecture-Based Change Impact Analysis in Information Systems and Business Processes. In *International Conference on Software Architecture*. IEEE.

Rouquette, N.F., Wasserman, G.M., and Carson, V.D. (2005). OWL for Space Mission Systems development at JPL with semantic architecture styles. In *OWL: Experiences and Directions*.

Rowley, J. (2007). The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, 33(2), 163–180.

Vogel-Heuser, B., Legat, C., Folmer, J., and Feldmann, S. (2014). Researching Evolution in Industrial Plant Automation: Scenarios and Documentation of the Pick and Place Unit.