Methods

Fan Ji*, Maximilian Wünnenberg, Rafael Schypula, Juliane Fischer, Dominik Hujo, Michael Goedicke, Johannes Fottner and Birgit Vogel-Heuser

Inconsistency management in heterogeneous engineering data in intralogistics based on coupled metamodels

Inkonsistenz Management in heterogenen Engineering Daten in der Intralogistik auf Basis von gekoppelten Metamodellen

https://doi.org/10.1515/auto-2022-0128
Received October 4, 2022; accepted March 10, 2023

Abstract: During the development of intralogistics systems (ILS), heterogeneous models are created, which represent discipline-specific views, e.g., control software developed by automation engineers or discrete-event simulation models created by simulation engineers. These models represent discipline-specific views on the system but contain overlapping information. Thereby, keeping the information in different development models consistent is challenging and currently requires high manual effort, which highly depends on the developers' experience. To overcome this challenge, an approach to link heterogeneous model data and identify potential information inconsistencies within and between models automatically is proposed. The concept is evaluated with a use case containing three typical inconsistencies from five representative engineering models applied in ILS development.

Keywords: intralogistics systems; metamodel coupling; model inconsistency.

Zusammenfassung: Bei der Entwicklung von Intralogistiksystemen (ILS) werden heterogene Modelle erstellt, die disziplinspezifische Sichten auf das ILS darstellen, z.B., von Automatisierungsingenieuren entwickelte Steuerungssoftware oder von Simulationsingenieuren erstellte ereignisdiskrete Simulationsmodelle. Diese Modelle repräsentieren disziplinspezifische Sichten auf das System, enthalten aber überschneidende Informationen. Es ist herausfordernd, die Informationen in den verschiedenen Entwicklungsmodellen konsistent zu halten und erfordert derzeit einen hohen manuellen Aufwand, der stark von der Erfahrung der Entwickler abhängt. Um dies zu adressieren, wird ein Ansatz zur Verknüpfung heterogener Modelldaten und zur automatischen Ermittlung potenzieller Inkonsistenzen innerhalb und zwischen Modellen vorgestellt. Das Konzept wird mit einem Anwendungsfall evaluiert, der drei typische Inkonsistenzen aus fünf repräsentativen Modellen enthält, die in der ILS-Entwicklung eingesetzt werden.

Schlagwörter: Intralogistiksysteme; Metamodelkopplung; Modellinkonsistenz.

***Corresponding author: Fan Ji**, Technical University of Munich, Institute of Automation and Information Systems, Boltzmannstr. 15, 85748 Garching bei München, Germany, E-mail: fan.ji@tum.de
Maximilian Wünnenberg and Johannes Fottner, Chair of Materials Handling, Material Flow, Logistics, Technical University of Munich, Boltzmannstr. 15, 85748 Garching bei München, Germany,
E-mail: max.wuennenberg@tum.de (M. Wünnenberg), j.fottner@tum.de (J. Fottner)
Rafael Schypula and Michael Goedicke, Paluno – The Ruhr Institute for Software Technology, University of Duisburg-Essen, Gerlingstraße 16, 45127 Essen, Germany, E-mail: rafael.schypula@paluno.uni-due.de (R. Schypula), michael.goedicke@paluno.uni-due.de (M. Goedicke)
Juliane Fischer, **Dominik Hujo and Birgit Vogel-Heuser**, Technical University of Munich, Institute of Automation and Information Systems, Boltzmannstr. 15, 85748 Garching bei München, Germany,
E-mail: juliane.fischer@tum.de (J. Fischer), dominik.hujo@tum.de (D. Hujo), vogel-heuser@tum.de (B. Vogel-Heuser)

1 Challenges in ensuring model consistency during the development of intralogistics systems

Intralogistics systems (ILS) play a vital role in ensuring the successful satisfaction of customer demands within production companies. The term intralogistics system refers to

the internal material flow system of a company or factory, such as the transportation of goods during the production process [1]. Rapidly changing product portfolios, demand fluctuations, and alterations in supply chains require high flexibility of these systems, which means, e.g., re-assembling conveyor units to realize different material flow layouts according to changing requirements and capabilities for the adaption to different types of handling units. At the same time, they must satisfy throughput demands while being highly reliable [2]. Thus, developing successful ILS is a challenging task that incorporates experts from various engineering domains, e.g., mechanical engineering, electrical engineering, software development, or simulation engineering. Every expert works with domain-specific development tools, i.e., software tools that generate highly diverse model files. Contradictions between the information described in these models (subsequently referred to as inconsistencies) might be difficult to discover during the ILS development phase, but they can cause significant delays during the assembly, commissioning, and operation of the system. It is thus necessary to identify inconsistencies in the different models before the physical product is generated, which is a challenging task [3].

Recently, a study has been conducted to identify domain-specific views on ILS and the main causes of model inconsistencies in practice [4]. In the study, experts involved in different stages of the development process have assessed current practices and challenges in ILS development. The study results show that model inconsistencies occurring in the design phase are mostly caused by insufficient communications between stakeholders, especially when changing system requirements. Thus, improving the efficiency of data exchange between disciplines is highly demanded. In addition, the study reveals that most inconsistencies among model data are assessed manually, and no generic checking process exists in reality. Therefore, the quality and reliability of the results depend heavily on human experience and are oftentimes difficult to guarantee.

To overcome the challenges mentioned above, this paper addresses the following research question: How can discipline-specific models, which are created during the development of ILS, be automatically linked to detect potential information inconsistencies within and between these models? To answer this question, the approach is derived using insights from previously conducted expert interviews [4], takes formalized domain knowledge into account, and is evaluated using a prototypical implementation and a lab-sized use case. Contradictions between the different engineering views are automatically discovered by comparing the overlapping information contained in their models. To this end, the domain-specific formulation of model data is transformed so that a cross-discipline comparison can be executed.

As part of a production system, the ILS development poses similar challenges, requirements, and boundary conditions as the development of the general production systems. For example, both developments are interdisciplinary, requiring the participation and collaboration of different stakeholders who express different views of the system with discipline-specific models. In addition, both pursue small batch sizes to realize customized production in loosely coupled, discrete processes. Moreover, compared to other subsystems in factory automation, ILS exhibit a high degree of modularity in both hardware and software, which results in clearly defined interfaces and high reusability [5], and is reflected, e.g., in the mechanical layout as well as in the control software. Meanwhile, the domain knowledge that needs to be considered to identify inconsistencies is relatively limited. These characteristics make ILS a good starting point for investigating the inconsistency management approach in the production domain.

The remainder of the paper is structured as follows: Section 2 elaborates on five requirements for identifying model inconsistencies during ILS development, followed by the introduction of preliminaries and relevant research in this area in Section 3. In Section 4, the proposed concept is described in detail, which is prototypically implemented and evaluated in Section 5 with three concrete inconsistency cases to assess the fulfillment of the requirements. In the end, a summary and an outlook on future research directions are provided in Section 6.

2 Requirements for identifying model information inconsistencies in intralogistics systems

Due to the challenges in keeping the information in different development models consistent during the design of ILS, the following requirements need to be fulfilled by a systematic and comprehensive inconsistency management approach.

Firstly, as mentioned above, during the development of ILS, engineers with different professional backgrounds are involved, who model the same system from discipline-specific perspectives using different modeling tools (software) [6]. For example, simulation engineers use simulation tools to forecast the dynamic performance of a logistics

system, while automation engineers develop the Programmable Logic Controller (PLC) code to control different actuators and to implement the desired process control. Therefore, the resulting heterogeneity of the applied engineering models that express discipline-specific views on the system should be handled by the inconsistency management approach (*R1*: *heterogeneous models*).

Second, during the development phases, different types of inconsistencies may occur within and across engineering models. Due to the high time pressure, engineers may introduce inconsistencies within one model when changes are performed on short notice at a late stage of the development workflow or when copy-paste-modify is used to speed up the development process [7]. Additional inconsistencies result from overlapping model information, meaning that the same information about the system under development is contained in two different models [6]. These can be equivalent information, that is, when two values within two different models need to be the same, or the values may be dependent on each other [8]. Since developers from different disciplines work with their discipline-specific modeling tools, they are not always aware of these content-wise overlaps. Both types of inconsistencies are potential sources of errors during the integration of system functionalities and, thus, need to be detected by an inconsistency management approach as early as possible (*R2*: *various types of inconsistencies*).

For identifying different inconsistency types within, but especially across different development models, the content-wise overlap and dependencies between the different discipline-specific models need to be made explicit. However, due to the diversity and discipline-specific nature of the engineering models created, it is a time-consuming task to define relations among model information manually and, thus, not feasible to repeat for each specific ILS. Moreover, due to changing requirements during the development of a specific ILS, the respective models are updated frequently, which would require updating the relations. Therefore, automatic and dynamic coupling of project-specific models is essential to improve the efficiency of the inconsistency management approach (*R3*: *automatic model coupling*).

Based on the pre-study [4], most inconsistencies are assessed by experienced engineers in their respective disciplines. Thus, the accuracy of the checking results relies heavily on engineers' experience, including their awareness and understanding of standards and guidelines. This is called domain knowledge in the following. Therefore, to check the conformance of model information with domain knowledge, supplementary information from these sources should also be incorporated. Formalized domain knowledge also serves as a basis for defining reusable inconsistency queries, since some of them are caused by a lack of experience or awareness of domain standards (*R4*: *knowledge formalization*).

Ensuring the industrial applicability of the approach can be considered from three perspectives: reusability, extensibility, and tool support. First, to reduce the efforts of adapting the concept to industrial applications, most parts of the concept should be reusable. Besides, considering the increasing complexity of ILS and the diversity of modeling tools applied in engineering activities, the concept and resulting prototypical implementation should be extensible to further integrate models from other disciplines. In addition, engineers involved in the ILS development should be assisted with tool support to define and identify multi-types of model inconsistencies (*R5*: *industrial applicability*).

3 Preliminaries and state of the art in model inconsistency management

Subsection 3.1 provides a brief overview of Model-based Engineering (MBE) and Semantic Web Technologies (SWT). Next, apart from the underlying approaches (V-SUMM, SUMM), Subsection 3.2 focuses on the current state-of-the-art in model inconsistency management approaches in the domain of factory automation (discrete processes), limited to linked model approaches that follow the similar principle to V-SUMM.

3.1 Preliminaries: model-based engineering and semantic web technologies

Model-based engineering (MBE) is widely applied to address the growing complexity of ILS and to increase their flexibility to changing market demands. In MBE, engineers from different disciplines depict certain aspects of the system by using specific modeling tools, which are not limited to software development but also include other disciplines involved in the system development. For example, mechanical engineers use Computer Aided Design (CAD) tools such as AutoCAD to design 3D system layouts, while electrical engineers create the circuit diagram using Electrical CAD tools such as EPLAN or Zuken E3. Consequently, various engineering models are generated in the design processes. A model, regarded as an abstraction of real-world objects, describes a system from the specific viewpoint of its

developer and contains only discipline-related system information. To define the structure, semantics, and constraints for a certain kind of model [9], metamodels are utilized and often designed in advance. A metamodel expresses model syntax and semantics at a higher abstraction level, which enables a general description of a certain type of model. In this paper, the information contained in metamodels is considered as model knowledge.

Due to the heterogeneity of engineering models and discipline-specific expressions of views on a system such as an ILS, a common formalism for representing discipline-specific knowledge about the structure and semantics of these various models is demanded. According to Sabou et al. [10]. Semantic Web technologies (SWT) are advantageous in terms of flexible semantic modeling, intelligent knowledge integration, exploration of distributed datasets, knowledge quality assurance, and knowledge reuse. These characteristics make SWT an ideal concept to address the requirements mentioned in Section 2. As the basic building block in SWT, the Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web and is applied to integrate data from multiple sources. An RDF graph is a set of triples, each consisting of a subject, a predicate, and an object, which can be expressed as a node-arc-node link (e.g., *conveyor1-modeledBy – CAD-model*, *conveyor1-is_a-Beltconveyor*). Compared to RDF, Web Ontology Language (OWL) is designed to formulate not only relations between instances but also constraints and hierarchy among classes. Since both languages support rule-based reasoning, i.e., inferring new element relations based on the existing ones, the realization of *R3* (*automatic model coupling*) is made possible. In this paper, the reasoning ability of SWT is utilized to automatically generate links among model instances based on the previously coupled metamodels.

3.2 State of the art in model inconsistency management

Within the scope of MBE, many approaches have been raised to maintain or rebuild consistency (i.e., inconsistency management) among models. For example, Feldmann et al. propose to use links to define different types of information overlaps within and between engineering models in the domain of automated production systems (aPS) [6]. To check the consistency between models from different disciplines, they use a consistency model to trace links between the elements from different models in combination with a rule-based approach. Based on this work, Zou et al. [11] further classify the intra-model and inter-model inconsistencies links into five categories, which are *existence*, *equivalence*, *correspondence*, *refinement*, and *satisfaction*-links. In the proposed approach, the predefined links between metamodels, which represent implicit inter-model dependencies, are created based on this classification. However, in these two approaches, all model relations should be explicitly and manually defined in advance (*R3*: *automatic model coupling*), which greatly reduces their flexibility and extensibility. In addition, in the work of Feldmann et al. [6, 8], the lack of consideration of domain-specific knowledge (*R4*: *knowledge formalization*) and the lack of the ability to generate a combined view, including linked model information from all involved disciplines, lead to the limited utility in industrial applications. To represent heterogeneous model information in a uniform and formalized format, SWT have been widely used in modeling inconsistency management. For example, Herzig et al. [12] propose an SWT-based framework in which all models are represented and stored in a knowledge base in the form of RDF. They also design a semantic reasoner for querying and inferring inconsistencies based on deductive reasoning mechanisms. However, due to the lack of a hierarchy of modeled information in the knowledge base, all model relations need to be defined manually (*R3*: *automatic model coupling*), and domain-specific artifacts are not taken into consideration (*R4*: *knowledge formalization*).

To automatically couple different types of models, view-based development is widely applied in the software engineering domain in combination with MBE. A view, by definition, is regarded as a model which is generated to allow users to see the system from a specific viewpoint [13]. To maintain consistency between models, Atkinson et al. propose a Single Underlying Model (SUM) structure combined with its metamodel (SUMM), where all the system information is predefined [13]. They transform model instances into the SUM to prevent potential inconsistencies. However, the concept's application in practice is limited since it requires a 1:1 relationship between different metamodel elements, which cannot always be realized. To overcome this shortage, Kramer et al. propose VITRUVIUS using a V-SUMM (Virtual Single Underlying Meta Model) by connecting metamodels from different disciplines to construct a virtual metamodel [14], whereby "virtual" refers to a metamodel that is combined of various metamodels. In their work, the V-SUMM is built with a set of metamodels coupled with consistency preservation rules. Models for developing a concrete software system can thereby be derived and modified. However, V-SUMM is only applied in the software domain and targets solving inconsistencies among models formalized in major languages like the Unified Modeling Language (UML) profiles. The applicability of V-SUMM to the development of engineering systems such

as ILS is not investigated, which comparatively contain different disciplines, such as mechanical engineering, software development, and project management. Additionally, model inconsistencies depending on the domain knowledge are not captured in the approach.

Some studies have already been conducted to manage model inconsistencies in specific application domains. For example, in mechatronics engineering, Li et al. propose a concept to integrate multidisciplinary engineering models during the development phase using SysML4Mechatronic, which is a SysML profile aimed specifically at developing mechatronic systems [15]. In this approach, required information from heterogeneous engineering models is firstly imported into a SysML4Mechatronics plant model, which contains reusable components in each discipline and serves as the central model database for various application cases. The inconsistency checking process is performed on an Automation Markup Language (AML) file exported from this plant model. To store and formalize the consistency constraints and rules, a knowledge base is built additionally based on discipline-specific knowledge. However, after two steps of transformation, the completeness and correctness of model information may not be ensured. In addition, since the model data and domain knowledge are saved separately, their relationship is unknown to the system and should all be defined manually for each mechatronic plant, which cannot satisfy *R3* (*automatic model coupling*) and *R5* (*industrial applicability*). In addition, Ananieva et al. explore the application of VITRUVIUS framework in solving the inconsistencies of AML models in the automation domain [16]. They classify the consistency constraints into two types, namely *static* and *dynamic*, and prove the ability of the VITRUVIUS to detect both types of inconsistencies automatically. However, only a single modeling language, AML, is implemented, and the content-wise model relations (*R4*: *knowledge formalization*) are not taken into consideration in their work. In the intralogistics domain, Vogel-Heuser et al. propose a software architecture based on coupled metamodels to synchronize control programs and other models to support the software evolution of ILS [17]. To link heterogeneous model information in ILS, they introduce a three-layer architecture, which consists of metamodels, models, and real-world objects from top to bottom. This structure is also utilized in this paper to define the multi-level model information.

Although many works have been carried out in managing model inconsistency, research gaps remain (see Table 1). In MBE, some approaches aim to integrate multidisciplinary engineering models based on the model-linking concept and also utilize multisource domain knowledge for consistency definition, but the automatic model coupling is not realized (*R1*), which brings great human efforts during the preparation phase and hinders their generic applicability. In the domain of view-based software engineering, V-SUMM architecture for automatic model coupling has been developed. However, the complete process to apply it in the development of real production systems such as ILS is still missing (*R5*). Further, content-wise (semantic) model inconsistencies depending on the application domain are not taken into consideration (*R4*). In addition, V-SUMM is only implemented on some major modeling languages such as UML and AML, while its ability to couple heterogeneous engineering models (*R1*) and to handle extensive model inconsistencies (*R2*) is not sufficiently researched Table 1.

4 Concept for automatic inconsistency identification based on coupled metamodels

The concept aims to automatically identify inconsistencies within and between development models applied in the intralogistics domain and to support engineers in resolving the identified issues. The developed concept is depicted in Figure 1 and consists of two main parts: a client module (CM) for information extraction and user interaction and an inconsistency checking module (ICM) for information modeling, storage, and inconsistency query.

Table 1: Fulfillment of requirements in state of the art. Fulfilled (+), partially fulfilled (o), not fulfilled (−), or not considered (n.a.). If not specified otherwise, the row entries refer to all sources in the heading.

Requirement	Atkinson (SUMM) et al. [13]	Kramer et al. (V-SUMM) [14]	Feldmann et al. [6, 8], Zou et al. [11]	Herzig et al. [12]	Li et al. [15]	Ananieva et al. [16]	Vogel-Heuser et al. [17]
R1: heterogeneous models	+	o	+	+	o	−	o
R2: various types of inconsistencies	n.a.	n.a.	+	−	o	+	n.a.
R3: automatic model coupling	+	+	−	−	−	+	−
R4: knowledge formalization	−	−	o [11]	o	o	o	−
R5: industrial applicability	−	o	+	+	+	o	o

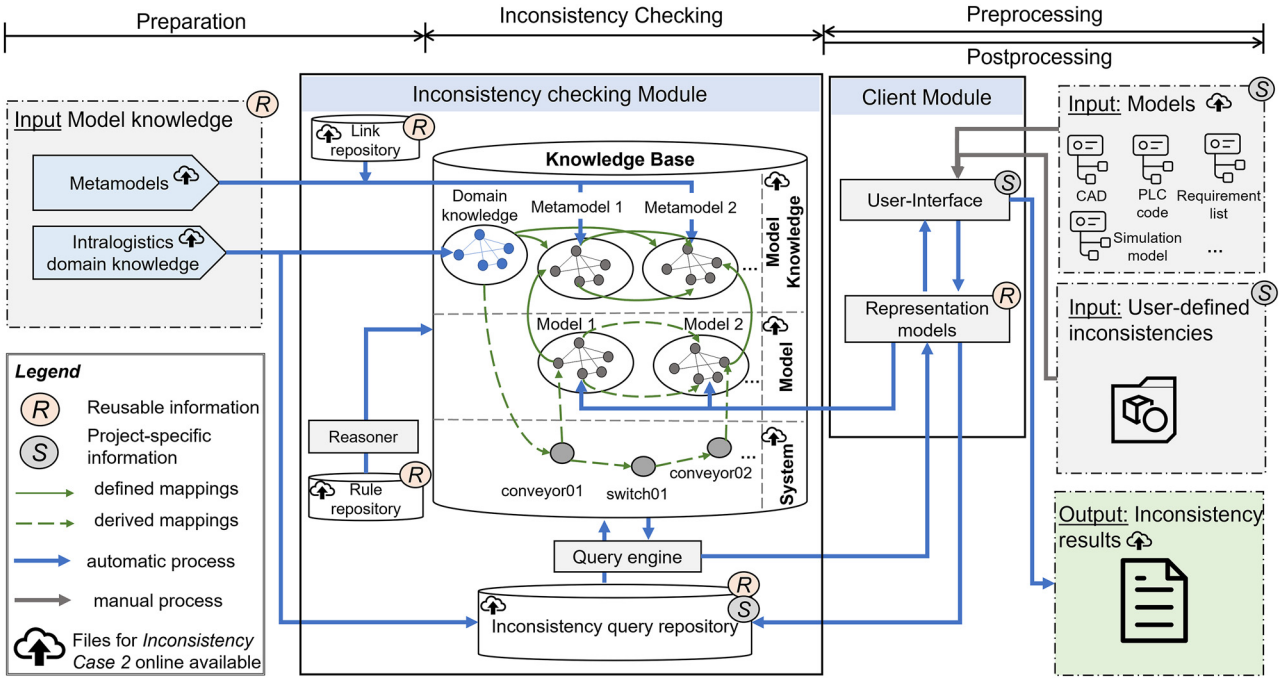

Figure 1: Overview of the concept for automatic inconsistency checking of model data in ILS.

The ICM contains a database to store the extracted model information, their metamodels, known dependencies between (meta-)model elements, and formalized knowledge from the intralogistics domain, which should be considered to identify inconsistencies. Since the database contains not only model-related data but also formalized knowledge embedded in metamodels and domain standards, it is called the knowledge base in the following. Based on the concept of model coupling proposed by Vogel-Heuser et al., this knowledge base is composed of three layers [17]. The elements at the lowest layer refer to real-world objects, which can be electrical or mechanical hardware or software in a system. The middle layer stores the information extracted from project-specific models (also called model instances). At the top layer, metamodels are interconnected to represent model-related knowledge. In addition, knowledge from selected intralogistics domain standards, together with domain experts' knowledge, is formalized as a complement to model knowledge (metamodels). The arrows in the knowledge base refer to the relationships between multisource data. Moreover, the IM also contains a reasoner to generate different types of data relations (see "derived links" in Figure 1) and a query engine to identify potential contradictions among model information. The CM mainly serves as an interface between engineers and ICM. It includes a user interface allowing engineers to upload models and self-defined consistency rules, trigger the checking process and get access to the inconsistency checking results.

Overall, the inconsistency identification process can be divided into four steps: First, domain knowledge is formalized, and the metamodels of the selected engineering models are imported into the ICM together with their dependencies (*Preparation*). After preparing the reusable part of the ICM, project-specific models should be uniformly formalized so that they can be integrated into the knowledge base (*Preprocessing*). Subsequently, the reasoner and query engine in the IM link these model instances automatically and check the inconsistencies with predefined and project-specific queries (*Inconsistency checking*). Finally, the query results are supplemented with additional information and presented to the engineers via the user interface in the CM (*Postprocessing*). Following these steps, details of the concept are provided in the following.

Preparation: In the preparation phase, metamodels of the selected engineering models, user-defined consistency rules, relevant industry standards, and guidelines applied in the intralogistics domain are utilized as the inputs. With these inputs, a reusable upper structure (model knowledge layer) is constructed in the knowledge base, which is the core part of the ICM. To improve the extensibility and general applicability of the approach, the V-SUMM concept is applied, which utilizes coupled metamodels to achieve automatic model linking. Thus, dependencies among metamodels in this layer are manually defined (see "defined mappings" in Figure 1) and imported into a link repository in the ICM. The links "L" can be expressed as 3-tuple:

$$L = \langle \mathrm{MM}_i, \mathrm{MM}_j, \mathrm{LT}_k \rangle. \quad (1)$$

In Eq. (1), MM represents a set of metamodels that are imported into the ICM. MM_i, MM_j are the i-nd and j-nd metamodels in the set where $i, j = 1, \ldots, \mathrm{m}$ and $i \neq j$. LT refers to a set of link types, which is defined based on the classification of model dependencies [11], and LT_k is the k-nd type of link, $k = 1, \ldots, n$. In addition, a rule repository containing rules for automatically deriving links between model instances is prepared and integrated into the CM. The rules are constructed based on the first-order logic (FOL), which can be expressed as:

$$L_1(e_x, v_1) \wedge L_2(v_1, v_2) \wedge \ldots \wedge L_t(v_{t-1}, e_y) \Rightarrow L_{\mathrm{new}}(e_x, e_y). \quad (2)$$

$L_1, L_2, \ldots, L_t$ are the relations that were pre-saved in the link repository, and L_{new} is the inferred relation between elements in the model instances (entities). e_x, e_y refer to the entities to be linked, which are extracted from the model data. v_i are the variables that represent entities in the model data.

Preprocessing: Before checking the model inconsistency, heterogeneous model data should be preprocessed to gain a uniform format so that they can be merged into the knowledge base. The information extraction process is performed by the CM. In the CM, a representation model is developed for each selected engineering discipline (view). The extracted model data needed for consistency checking is written into respective representation models and each of which entails the information for one engineering view. For example, the representation model for the mechanical view contains information about the position, geometric dimensions, name, and assignment to an assembly of each part. Once the checking process is started, data from uploaded engineering models is parsed based on the representation models and transformed into RDF. In addition, self-defined consistency rules are also transformed by the CM in this step. All the processed data is sent together to the ICM.

Inconsistency checking: This process is performed by ICM and can be divided into two sub-steps: model merging and inconsistency querying. In the first step, uniformly formatted model data from the CM is merged into the knowledge base. Relying on the unique identification number assigned to each model during preprocessing, model instances are mapped to the corresponding metamodels. Until now, model data is still independent in the knowledge base. To generate relations between inter-model information, metamodel links, and the FOL-rules are utilized. Based on these two repositories (link- and rule-repository), an SWT-reasoner looks through all the model data and couples the cross-domain model elements automatically (see "derived links" in Figure 1) according to the FOL-rules.

After inserting the links between model instances, the knowledge base is prepared for inconsistency querying. In the query repository, some queries are pre-saved based on domain knowledge, such as searching for the reference velocity of conveyors based on their classification. In addition, user-defined inconsistencies, which are created based on project specifications or engineers' expertise, are also imported. In each query statement, starting from the source model information, the target information is searched along the generated model links, and the mathematical relations between the source and target model data are also explicitly defined (see Eq. (3) in Section 5.3).

Postprocessing: When inconsistencies are detected by the query engine, the results are then completed with additional information to assist engineers in resolving conflicts among models. The processed checking results include not only information about conflicting model information but also warning messages for engineers that are previously supplemented to each query.

After the four steps described above, heterogeneous model data are automatically extracted and connected. Different types of model inconsistencies are checked based on the formalized model and domain knowledge, and the results are presented to the engineers.

5 Evaluation using a prototypical implementation

In this section, the proposed approach is prototypically implemented and evaluated with three representative inconsistency cases coming from an industrial use case.

5.1 Prototypical implementation

To evaluate the concept described in Section 4, a prototypical tool shown in Figure 2 is developed. As mentioned in Section 4, the tool also consists of two modules.

The CM is written in the Java programming language and includes a graphical user interface (GUI). The representation models in the CM contain Java classes that need to be filled with extracted model data. Subsequently, they are transformed into RDF files using Apache Jena,[1] which is a Java framework for SWT. In addition, users can define their own consistency rules with the modeling tool Eclipse Papyrus.[2] To ensure that these rules correctly address model

1 https://jena.apache.org/.

2 https://www.eclipse.org/papyrus/.

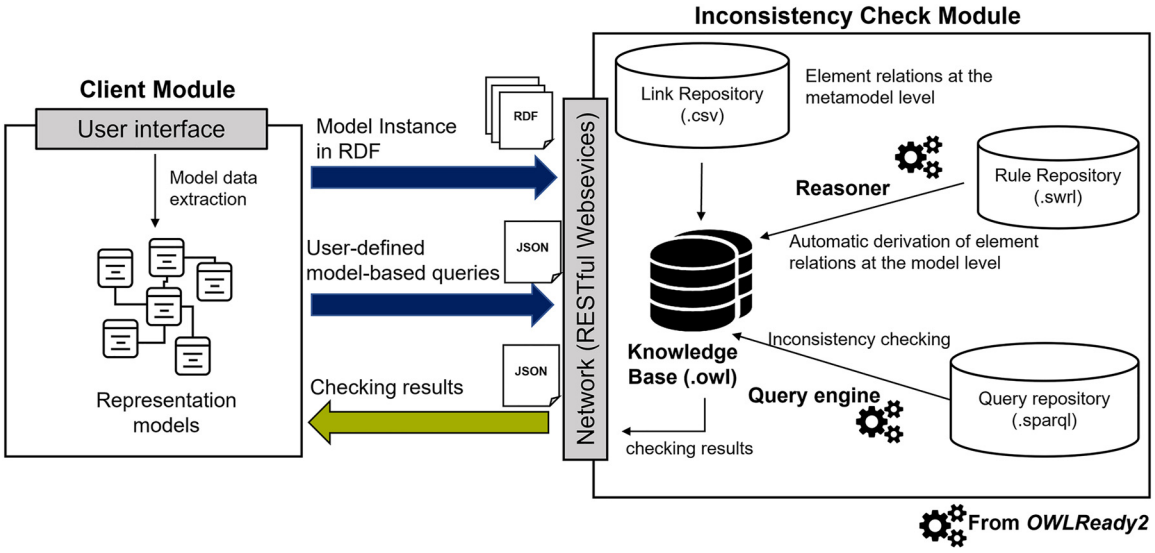

Figure 2: Prototypical implementation of the concept.

properties defined in the representation models, Papyrus model files with UML profiles are provided. Through the provided Papyrus model, user-defined rules are generated in UML format, which are further transformed into the form of JavaScript Object Notation (JSON) and sent to ICM together with extracted model information.

On the other hand, the ICM is composed of three repositories and a knowledge base. To provide a uniform format for representing heterogeneous engineering data and to infer potential information relations, SWT are applied. The knowledge base is modeled in OWL, which is an SWT language that is designed to process and integrate heterogenous information and supports relation reasoning. Thus, the knowledge is understandable for humans and also interpretable for machines. The link repository stores the links between elements at the metamodel level in a CSV file. Based on the predefined links and the model data received from the CM, a knowledge base is generated in OWL. To automatically couple the model instances, inference rules based on the FOL (see Eq. (2)) are defined and saved in a rule repository using the Semantic Web Rule Language (SWRL). In addition, in the query repository, the explicit relations between model elements are formalized in SPARQL, which is a semantic web query language. Once the checking process is triggered via the user interface, the query engine contained in the open-source Python packet *Owlready2* [18] queries through the knowledge base to detect potential inconsistencies. The synchronization of data and processes between the two modules is achieved by using RESTful web services.

5.2 Description of the use case

During the operation phase of ILS, individual logistics components may break down, e.g., due to aging. In the worst case, the broken component must be replaced by a new, comparable one if the original component type is out of production or has delivery delays [7]. Figure 3 shows a minimal example of ILS consisting of four belt conveyors built in a line. Although this use case is a minimal, lab-sized example, it is sufficient to represent the identified challenges, such as domain-specific models with different exchange formats modeled by different stakeholders and containing overlapping information about the system under development. In the considered scenario, belt conveyor 3 is broken and must be replaced by new hardware with similar functionality. In the new conveyor, a different drive is installed, requiring an update of the ILS's control software, and the mechanical conveyor parts have minor geometrical differences compared to the old components. Accordingly, during the preparation of the exchange, the model data needs to be updated. For example, the mechanical engineer imports the new conveyor's 3D model provided by its manufacturer into the 3D model of the entire system, while the automation engineer updates the configuration of the function block parameters. Meanwhile, to generate new process-related key performance indicators, e.g., the system's availability factor, a process simulation engineer re-simulates the transportation process with new simulation parameters of conveyor 3, such as the transportation time. If these model-related changes are not performed in time, or communicational problems

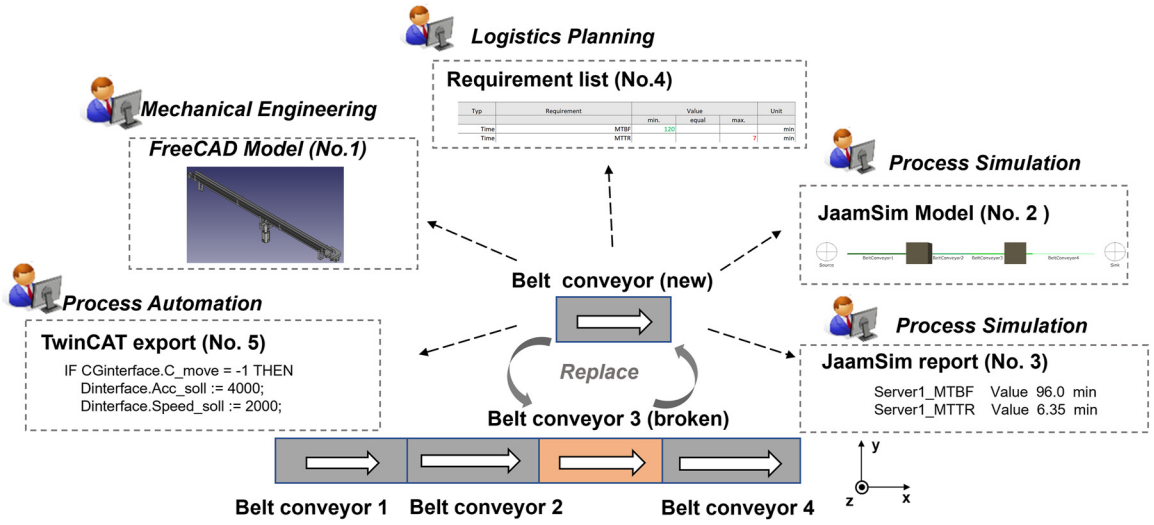

Figure 3: Use case replacement of a broken conveyor and the relevant engineering models.

between teams occur, conflicts among cross-domain engineering data may be caused.

The goal of this evaluation is to cover relevant engineering disciplines involved in the development of ILS without focusing too much on specific modeling tools from individual vendors. According to the pre-study [4], mechanical design, software development, material flow design, and project data analysis are the most relevant disciplines involved in ILS development. Thus, the following representative modeling tools are selected during the prototypical implementation: in the field of mechanical engineering, *FreeCAD* is selected (Model No. 1), to simulate the dynamic material flows, *JaamSim* is selected (Model No. 2), including generated system parameters (e.g., breakdown time, maintenance time) that are documented in a *JaamSim report* (Model No. 3) to check the fulfillment of project-specific requirements. The requirements list (Model No. 4) is stored in table format using *Excel* and a predefined template. For control software development, a *TwinCAT* export (Model No. 5) is generated, which contains all the control parameters and code to automate the transportation process. Each selected model targets a sub-aspect of the considered ILS, while there are information overlaps and dependencies between the models that may lead to errors during the design process, as highlighted in [4].

5.3 Potential inconsistencies

For evaluation purposes, three inconsistency cases are derived and simplified from the previously conducted expert interviews [4], which address different parts of the proposed concept. In this sense, they can be considered representative and serve as a first proof-of-concept evaluation.

5.3.1 Inconsistency case 1 (IC1): intra-model inconsistency (different heights of two adjacent conveyors)

When belt conveyor 3 is exchanged, its different interfaces need to be considered regarding consistency. Some geometrical conflicts of its mechanical interface can be detected directly by the *FreeCAD* tool, e.g., the spatial overlap of two objects. However, some inconsistencies still need to be checked based on domain knowledge. A simple example is the positions of the two adjacent conveyors on the *z*-axis (see Figure 3), which need to be identical to ensure a fluent and stable transportation process. In the use case, after updating the 3D model of conveyor 3 in *FreeCAD*, its height value is compared with the height values of its neighboring conveyors (conveyors 2 and 4) to avoid this inconsistency.

Since no metamodel is officially provided by the *FreeCAD* tool, a metamodel is developed during the implementation, which includes 24 classes representing the basic structure and information of a *FreeCAD* model. In the "*geometry*" class of the metamodel, three *positions* and three *directions* are defined as its attributes. In this case, the *equivalence* relations are utilized, which define whether two linked elements should be the same [11]. Thus, the link at the metamodel level is defined as "*z_position – equivalentTo – z_position*". In the rule repository, the inference rule to identify which conveyors are adjacent is derived from the material flow information contained in the simulation

model. Based on this rule, the reasoner automatically generates the "*has_neighbor*" relations among different conveyors. Based on this predefined link and the inferred relations, the query engine compares the z-position of all adjacent conveyors and detects an inconsistency if they are not equal. Additionally, tolerance of the height difference is also given according to domain knowledge and stored in the query repository.

5.3.2 Inconsistency case 2 (IC2): inter-model inconsistency (conveyor velocity)

To assess the fulfillment of *R1* (heterogeneous models), this inconsistency case is utilized. When changing the configuration of conveyor 3 regarding its geometrical data and control software, engineers in the process simulation domain also need to recalculate the transportation time of a workpiece on each conveyor according to the system throughput time. In the *JaamSim* model, the transport time of a workpiece on conveyor 3 ($t_{\text{conveyor3}}$, unit: *s*) is simulated. Moreover, the function block of conveyor 3 is replaced in the control program, including an update of the target rotational speed of the drive (v_{rotation}) on conveyor 3. The parameter v_{rotation} is converted to the linear target velocity of the conveyor ($v_{\text{conveyor3}}$, unit: m/s) in the control software. Overall, the parameters $t_{\text{conveyor3}}$ from *JaamSim and* v_{rotation} from *TwinCAT* are connected via the conveyor length ($l_{\text{conveyor3}}$, unit: mm) defined in the 3D *FreeCAD* model as follows:

$$v_{\text{conveyor3}} = \frac{l_{\text{conveyor3}}}{t_{\text{conveyor3}} * 1000} \tag{3}$$

Since this inconsistency case includes relations among three different engineering models, more details are provided here. As shown in Figure 4, three classes from three different metamodels are involved in this inconsistency case: "component" in *FreeCAD*, "simulationItem" in *JaamSim*, and "FunctionBlock" in *TwinCAT*. The metamodel link is defined between three attributes from these classes, which are "length", "transportationTime" and "velocity" respectively. Two types of links are applied in this case, which are *correspondence* and *satisfaction* respectively. The *correspondence* links ("*correspondTo*") focus more on convention-based model relations by which the mathematical relations among values of model attributes are built, whereas the *satisfaction* links ("*satisfiedBy*") depict the conformance of model elements with domain standards, guidelines, or project-specific requirements [11]. As denoted in Figure 4, links between corresponding parameter values extracted from concrete models are derived automatically, and their explicit relations (i.e., Eq. (3)) are saved in the query.

To further check the conformance of the calculated conveyor velocity with domain-specific knowledge, the taxonomy of continuous conveyors from guideline VDI 4440 is integrated into the knowledge base, including the usual maximum conveying speed for belt conveyors indicated by domain experts. This domain knowledge is additional information beyond the model knowledge to identify additional inconsistencies. The system component "Belt conveyor 3" is automatically classified as a "Belt conveyor" in the taxonomy based on the analysis of string similarity. To further couple the "maximum velocity" of the belt conveyor

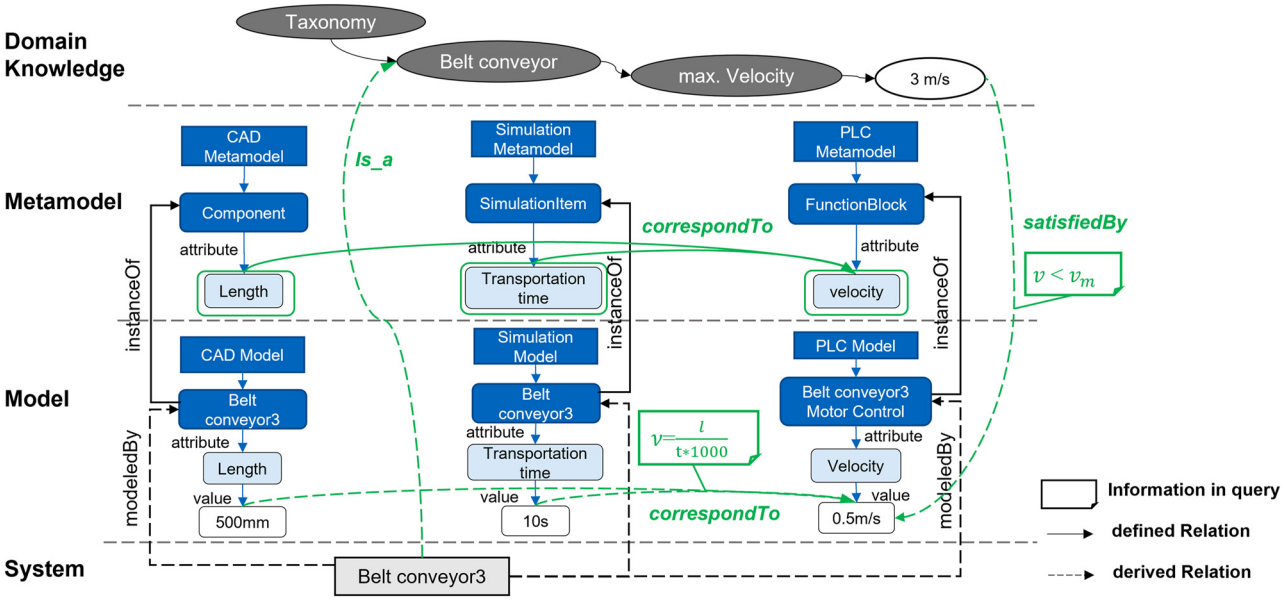

Figure 4: Relations between multi-level elements in inconsistency case 2.

and the velocity of conveyor 3 ($v_{\text{conveyor3}}$), the following FOL rule is formulated in Eq. (4) and saved in the rule repository:

$$\begin{aligned} &\text{Is\_a (conveyor, Beltconveyor)} \\ &\quad \wedge \text{ has\_parameter (conveyor, velocity\_conveyor)} \\ &\quad \wedge \text{ has\_max\_velocity (Beltconveyor, velocity\_max)} \\ &\quad \Rightarrow \text{satisfied\_by (velocity\_max, velocity\_conveyor)} \end{aligned} \tag{4}$$

By reasoning on this FOL rule, all the velocities of conveyors in a system can be linked to the corresponding velocity limits based on the conveyor type. Thus, the $v_{\text{conveyor3}}$ coming from *TwinCAT* is coupled with the velocity of belt conveyors in the taxonomy, and their consistency can be checked after performing a query. The materials for implementing inconsistency case 2, including an excerpt of the used metamodels, model instances in tool-specific formats, FOL rules, links on metamodel-level, SPARQL queries, as well as the checking results, are available online.[3]

5.3.3 Inconsistency case 3 (IC3): user-defined inter-model inconsistency (MTBF, MTTF)

According to the conducted pre-study [4], many inconsistencies in real ILS development are caused by requirement changes. In most cases, engineers need to manually look up specific parameters within their models that are affected by a changed requirement, which is time-intensive and error-prone. This IC3 addresses this challenge by defining consistency rules to realize automatic inconsistency checking during requirement changes. A typical requirement is the availability of the ILS, which represents the relationship between the total expected downtime and the total theoretical usable operating time. To ensure the economic efficiency of a plant and quantitatively forecast the probability of failure, two values are commonly applied, which are:

MTBF (*mean time between failures*) $= \frac{1}{\lambda} =$ expected value of the failure-free time between failures (λ – failure rate)
MTTR (*mean time to repair*) $= \frac{1}{\mu} =$ expected value of the failure time (μ – repair rate)

With these two values above, the availability of the system (A) can be calculated by the following equation [19]:

$$A = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}. \tag{5}$$

The availability of ILS is a project-specific value and is determined by custom requirements. As a result, when the logistics planner changes the required system availability, the model that provides these two values (MTBF, MTTR) should also be adjusted accordingly.

To forecast the availability of the four-conveyor-ILS described in Section 5.2, MTBF and MTTR are defined in the *JaamSim*, and their simulated values are documented in the *JaamSim* report. On the other hand, requirements targeting quantitative system parameters are specified in an *Excel* file with predefined value ranges. Since MTTR and MTBF are not pre-integrated parameters in the *JaamSim* but are simulation parameters self-defined at specific measuring points, their consistency rule cannot be prepared. Therefore, the framework is also developed to support user-defined inconsistency queries. In the prototypical implementation, the lowest boundaries of these parameters are specified in a requirement list. A consistency rule (i.e., "the simulated MTBF and MTTR values should not be lower than the required minimum values.") is defined manually with the graphical modeling tool provided by Papyrus. The generated rule file in the form of UML is uploaded together with the two models (i.e., *JaamSim* report and *requirement list*) via the user interface. Checking results are shown in Figure 5.

3 https://gitlab.lrz.de/materials-for-ic2implementation.

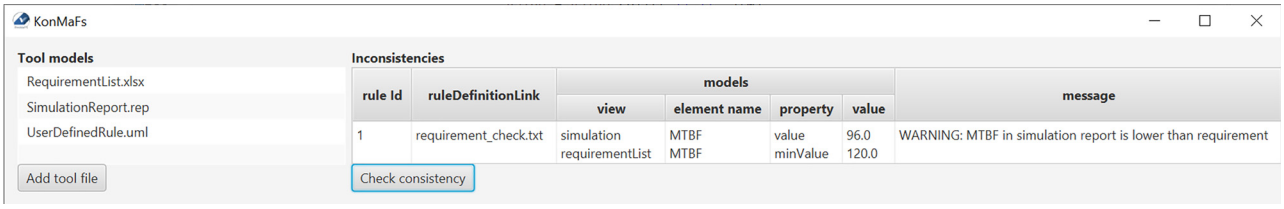

Figure 5: Checking results of inconsistency case 3.

5.3.4 Implementation results

To evaluate the concept, inconsistency cases described above are intentionally added to the respective models. After loading the corresponding models and user-defined consistency rule (IC3), all these three types of inconsistencies are successfully detected. Figure 5 shows a screenshot of the graphical user interface with the checking results for IC3. The left part of the interface lists all the model files uploaded by engineers, and the right part displays the checking results. In the result table, the first two columns show the internal number and rule file name, respectively, and the next four columns describe the affected engineering views, model types, and model elements. In the last column, a message that helps relevant engineers to resolve this inconsistency is provided.

5.4 Efforts for adapting to a different project

Although most parts of the concept are reusable, some changes are required when applying the approach during the development of different ILS. Three different adaptions are described below.

Since the metamodels described in this paper are not defined for specific modeling tools but for the selected disciplines (views), changing the tools for modeling will not require major changes to the metamodels in most cases. Most modeling tools within a discipline offer similar functionalities resulting in similar classes in the metamodels. For example, both *AutoCAD* and *FreeCAD* support modeling geometric characteristics of logistics components, and both own the class "*Geometry*" in their metamodel. This leads to a high potential for reusing previously defined classes and related links. However, the model parser in the CM needs to be adjusted when the new modeling tool has a different data exchange format.

On the other hand, when the consistency of a model from an additional discipline is required to be checked, its metamodel should be integrated into the knowledge base first. Some modeling tools provide their own metamodels, if not, they can also be newly developed or be selected from third-party metamodels [14]. Further, the relations of this metamodel with the existing ones saved in the knowledge base need to be defined. Although a basic classification of link types and a CSV-based definition of links are offered, human efforts are still required in this step.

Depending on the targeted inconsistencies, domain standards, norms, or guidelines that help to define the model consistencies need to be updated or added. To include domain knowledge that defines the relations between different models, as in Eq. (3), only links or queries need to be updated. If the knowledge source provides additional information, such as the taxonomy of conveyors, it needs to be formalized in the knowledge base. Also, the relations between newly imported knowledge with the existing metamodels should be defined at the model knowledge layer (see Figure 1).

5.5 Assessment of the requirements' fulfillment

To evaluate the proposed concept, consistency checks are performed on five discipline-specific models (see Figure 3) involved in the use case described in Section 5.2 (*R1*: *heterogeneous models*). Accordingly, their metamodels are formalized in OWL and pre-coupled in the knowledge base, which enables the automatic coupling of concrete model information (*R3*: *automatic model linking*). In the evaluation, three types of representative inconsistencies are intensively introduced in the models and are successfully detected by the prototypical tool developed, which proves the ability of the concept to identify different types of information contradictions within and between models (*R2*: *various types of inconsistencies*). Apart from using model knowledge expressed by discipline-specific metamodels, in IC2, domain knowledge such as the conveyor taxonomy is also integrated and formalized in the knowledge base, which provides additional and supportive information for detecting model inconsistencies (*R4*: *knowledge formalization*). In terms of industrial applicability (*R5*), the CM in the proposed approach enables model developers to upload the models they are working with and check for consistency without experience in SWT. Meanwhile, the ICM can be configured and maintained by so-called knowledge management engineers, a newly arising role in the industry [20]. According to *Lupp*, it is possible to build and maintain reusable patterns for modeling links and rules, which can be provided via user interfaces to, e.g., intralogistics domain engineers, to ease their interaction with the SWT [20]. To improve the generalizability of the approach, the concept aims at using standardized data exchange formats such as PLCopenXML for control software. Unfortunately, standard formats are not supported for all used models, and thus, tool-specific files are also applied. However, *R5* is not completely satisfied because the small use case applied in the paper cannot cover all typical inconsistency cases that appear during the development of industrial-scale ILS. Additionally, most models applied in the implementation part are created with open-source modeling tools. Thus, the applicability of proprietary tools such as AutoCAD for 3D modeling needs to be investigated in future work to ensure industrial

Table 2: Assessment of the requirement fulfillment.

Requirement	Relevant inconsistency case	Assessment	Details regarding assessment
R1: heterogeneous models	IC2, IC3	Fulfilled	In IC2 and IC3, different engineering models are involved. Both cases illustrate how to identify inconsistencies between models from different disciplines.
R2: various types of inconsistencies	IC1, IC2, IC3	Fulfilled	IC1 describes how to check inconsistencies within a model (type: *equivalence*). In IC2 and IC3, different contradictions between models are detected (types: *correspondence* and *satisfy*)
R3: automatic model coupling	IC2, IC3	Fulfilled	In both cases, links among model instances are automatically generated based on the previously connected metamodels. (See "derived links" in Figure 4)
R4: knowledge formalization	IC2	Fulfilled	In IC2, additional domain knowledge is formalized and utilized to further check if the values of model attributes conform to the experience values.
R5: industrial applicability	IC3	Partially fulfilled	IC3 shows how the engineers can be supported to check model inconsistencies using project-specific consistency rules. In Section 5.5, efforts that should be made to adapt the approach to other projects are illustrated.

applicability. Details regarding the requirements' fulfillment are listed in Table 2.

6 Summary and outlook

This paper proposes a concept to automatically identify multiple types of model inconsistencies that emerge during the multidisciplinary development of ILS. Compared to the basic V-SUMM approach [14], consistency rules and links are extended to describe complex relations (including mathematical relations) between heterogeneous engineering models. Besides, domain knowledge is considered, and a third "system" layer is built to support checking various types of inconsistencies. Compared to the manual checking process, inconsistencies that are often easy to be overlooked can be defined as rules and queries, which ensures more reliable and user-independent checking results. In addition, engineers do not need to look up, compare and transform the parameters in different models, which significantly reduces human effort when dealing with more complex models in reality. Compared to other inconsistency management approaches developed for the production system development, the basic structure derived from the V-SUMM can realize automatic linking of model instances with pre-coupled metamodels, which cannot be addressed by, e.g., the work of Feldmann et al. [8]. Moreover, using V-SUMM, a so-called combined view (knowledge base), including the domain knowledge from all involved disciplines, is generated. For each modeled element, this knowledge base includes information about that element from all involved disciplines. Thus, to avoid inconsistencies caused by sequential, multiple modeling of an element in different views, information about the element already contained in the knowledge base is automatically displayed to engineers from different disciplines. Thereby, it is possible to generate discipline-specific models partially based on already entered information, which will be addressed in the future. In addition, different from the previous studies based on V-SUMM, which mainly focus on models in software development, the concept is able to detect inconsistencies among different models coming from multidisciplinary modeling tools. Furthermore, domain- and discipline-specific knowledge is also formalized in the knowledge base, which extends the scope of model inconsistencies that can be detected. The developed user interface enables engineers to check self-defined model inconsistencies according to the changing system requirements, which promotes the industrial applicability of the approach. By defining links between the metamodels and rules to be checked, a new potential source of errors is introduced, in the worst case leading to misidentified or overlooked inconsistencies. Nevertheless, automatic identification of parts of the included inconsistencies is a benefit compared to the so far completely manual inconsistency checking approach.

Some limitations still exist within the scope of the paper. First, validation of the proposed approach is limited to relatively simple inconsistency cases at the level of proof-of-concept. Additionally, the potential risks of misidentified or overlooked inconsistencies are still open. In the future evaluation, a real industrial use case will be selected,

including the application of real models and inconsistencies without simplification. The checking process and results will be evaluated by domain experts who have deep insights and experience in the intralogistics domain. Since no generic inconsistency-checking process exists in reality [4], the proposed approach will be assessed by these experts through a comparison with their current inconsistency-checking approaches.

In future work, the approach will be first applied to other ILS to further prove its generality. In addition, model consistency of other industrial production plants that include not only logistics systems will be tested with the proposed concept, accompanied by the incorporation of extensive engineering models and domain knowledge. Due to the similar characteristics of intralogistics systems and production systems in general, it is expected that the approach can be transferred to further parts of the production systems. For this purpose, additional model interfaces will be created in the CM to widely support modeling tools applied in the development phase, and the checking results will be intuitively presented to engineers in a graphical format. Meanwhile, knowledge not only from the logistics domain but also from other sub-parts of the production systems should be formally described and incorporated. Since defining links and rules still requires much manual work, future focus will be put on utilizing available information, such as unique equipment identifiers, which are used company-wide by some companies in practice, for linking models from different disciplines and, thus, reducing the manual activities involved. Furthermore, the scalability of the approach will be evaluated in the future to ensure, e.g., no cross effects will be caused by the coupling of additional model information. In addition, to investigate the impact of model changes, a future extension of the ICM is planned to allow versioning of the models uploaded to the knowledge base in order to identify inconsistencies resulting from model updates.

Author contribution: All the authors have accepted responsibility for the entire content of this submitted manuscript and approved submission.
Research funding: This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 451550676.
Conflict of interest statement: The authors declare no conflicts of interest regarding this article.

References

[1] M. ten Hompel, T. Schmidt, and J. Dregger, *Materialflusssysteme: Förder- und Lagertechnik*, 4th ed. Berlin, Heidelberg, Springer, 2018.
[2] C. Lieberoth-Leden, D. Regulin, and W. A. Günthner, "Efficient messaging through cluster coordinators in decentralized controlled material flow systems," *MATEC Web Conf.*, vol. 81, p. 6005, 2016.
[3] J. Grundy, J. Hosking, and W. B. Mugridge, "Inconsistency management for multiple-view software development environments," *IEEE Trans. Software Eng.*, vol. 24, no. 11, pp. 960—981, 1998.
[4] M. Wünnenberg, D. Hujo, R. Schypula, J. Fottner, M. Goedicke, and B. Vogel-Heuser, "Modellkonsistenz in der Entwicklung von Materialflusssystemen: Eine Studie über Entwicklungswerkzeuge und Einflüsse auf den Produktentstehungsprozess," *ZWF*, vol. 116, no. 11, pp. 820—825, 2021.
[5] M. Spindler, T. Aicher, B. Vogel-Heuser, and J. Fottner, "Engineering the control software of automated material handling systems via drag & drop," *Logist. J.*, vol. 2017, no. 10, pp. 1—8, 2017.
[6] S. Feldmann, M. Wimmer, K. Kernschmidt, and B. Vogel-Heuser, "A comprehensive approach for managing inter-model inconsistencies in automated production systems engineering," in *2016 IEEE International Conf. on Automation Science and Engineering (CASE)*, 2016, pp. 1120—1127.
[7] T. Aicher, J. Fottner, and B. Vogel-Heuser, "A model-driven engineering design process for the development of control software for Intralogistics Systems," *Automatisierungstechnik*, vol. 70, no. 2, pp. 164—180, 2022.
[8] S. Feldmann, K. Kernschmidt, M. Wimmer, and B. Vogel-Heuser, "Managing inter-model inconsistencies in model-based systems engineering: application in automated production systems engineering," *J. Syst. Software*, vol. 153, pp. 105—134, 2019.
[9] S. J. Mellor, *MDA Distilled: Principles of Model-Driven Architecture*, Boston, Addison-Wesley, 2004.
[10] M. Sabou, "An introduction to semantic web technologies," in *Semantic Web Technologies for Intelligent Engineering Applications*, S. Biffl and M. Sabou, Eds., Cham, Springer International Publishing, 2016, pp. 53—81.
[11] M. Zou, H. Li, and B. Vogel-Heuser, "A framework for inconsistency detection across heterogeneous models in industry 4.0," in *2019 IEEE International Conf. on Industrial Engineering and Engineering Management (IEEM)*, 2019, pp. 29—34.
[12] S. J. I. Herzig, A. Qamar, and C. J. J. Paredis, "An approach to identifying inconsistencies in model-based systems engineering," *Proc. Comput. Sci.*, vol. 28, pp. 354—362, 2014.
[13] C. Atkinson, D. Stoll, and C. Tunjic, *Orthographic Service Modeling*, 2011.
[14] M. E. Kramer, E. Burger, and M. Langhammer, "View-centric engineering with synchronized heterogeneous models," in *Proc. of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*, 2013.

[15] H. Li, M. Zou, G. Hogrefe, et al., “Application of a multi-disciplinary design approach in a mechatronic engineering toolchain,” *Automatisierungstechnik*, vol. 67, no. 3, pp. 246—269, 2019.
[16] S. Ananieva, E. Burger, and C. Stier, “Model-driven consistency preservation in automationml,” in *2018 IEEE 14th International Conf. on Automation Science and Engineering (CASE)*, 2018, pp. 1536—1541.
[17] B. Vogel-Heuser, M. Konersmann, T. Aicher, J. Fischer, F. Ocker, and M. Goedicke, “Supporting evolution of automated material flow systems as part of CPPS by using coupled meta models,” in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, 2018, pp. 316—323.
[18] J. Lamy, “Owlready: ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies,” *Artif. Intell. Med.*, vol. 80, pp. 11—28, 2017.
[19] D. Arnold and K. Furmans, “Planung von Materialflusssystemen,” in *Materialfluss in Logistiksystemen*, Berlin, Heidelberg, Springer, 2009, pp. 233—328.
[20] D. P. Lupp, “A higher-level view of ontological modeling: rule-based approaches for data transformation, modeling, and maintenance,” Ph.D. dissertation, University of Oslo, 2019.

Bionotes

Fan Ji
Technical University of Munich, Institute of Automation and Information Systems, Boltzmannstr. 15, 85748 Garching bei München, Germany
fan.ji@tum.de

Fan Ji received an M.Sc. in Mechanical Engineering and Management from the Technical University of Munich (TUM) in 2018. She is currently pursuing a Ph.D. at the Institute of Automation and Information Systems at TUM. Her main research interests include ontology-based knowledge formalization and inconsistency management in interdisciplinary engineering.

Maximilian Wünnenberg
Chair of Materials Handling, Material Flow, Logistics, Technical University of Munich, Boltzmannstr. 15, 85748 Garching bei München, Germany
max.wuennenberg@tum.de

Maximilian Wünnenberg is currently a research associate pursuing his Ph.D. at Technical University of Munich (TUM), Chair of Materials Handling, Material Flow, Logistics (fml), where he received his M.Sc. in Mechanical Engineering in 2020. He is responsible for the research project “Consistent Development of Material Flow Systems using a Model-based approach”. His main research interests are Model-based Systems Engineering, Material Flow Systems and Data Analytics.

Rafael Schypula
Paluno — The Ruhr Institute for Software Technology, University of Duisburg-Essen, Gerlingstraße 16, 45127 Essen, Germany
rafael.schypula@paluno.uni-due.de

Rafael Schypula, M. Sc., studied Applied Computer Science with a focus on industry and management at the Ruhr University Bochum. After five years as a software developer in the private sector, he joined the paluno in 2017. There he works on the design and development of software systems.

Juliane Fischer
Technical University of Munich, Institute of Automation and Information Systems, Boltzmannstr. 15, 85748 Garching bei München, Germany
juliane.fischer@tum.de

Dr.-Ing. Juliane Fischer received an M.Sc. in Mechanical Engineering from the Technical University of Munich (TUM) in 2017 and a Ph.D. degree in Mechanical Engineering from TUM in 2022. She is currently working at the Institute of Automation and Information Systems at TUM. Her main research interests are the design of modular, reusable control software and methods from the field of static code analysis to enhance the reuse of variant-rich legacy control software via identification of potentials for software improvement.

Dominik Hujo
Technical University of Munich, Institute of Automation and Information Systems, Boltzmannstr. 15, 85748 Garching bei München, Germany
dominik.hujo@tum.de

Dominik Hujo received the M.Sc. degree in Mechatronics and Information Systems in 2020 from the Technical University of Munich (TUM), where he is currently working toward the Ph.D. degree. He is currently a Research Assistant with the Institute of Automation and Information Systems. His research interests include heterogeneous distributed networked control systems and model-based assessments and engineering, where his primary focus is communication latencies at different automation levels.

Michael Goedicke
Paluno — The Ruhr Institute for Software Technology, University of Duisburg-Essen, Gerlingstraße 16, 45127 Essen, Germany
michael.goedicke@paluno.uni-due.de

Prof. Dr. Michael Goedicke, has been Professor of Practical Computer Science/Specification of Software Systems at the University of Duisburg-Essen since 1994. He studied computer science at the University of Dortmund and completed his doctorate there in 1985 on specification languages on p. 16 for embedded systems. He then conducted research in the areas of specification of software architectures and description of software components. In 1993 he completed his habilitation on the topic of specification of software components.

Johannes Fottner
Chair of Materials Handling, Material Flow, Logistics,
Technical University of Munich, Boltzmannstr. 15, 85748
Garching bei München, Germany
j.fottner@tum.de

Johannes Fottner is a professor for technical logistics at TUM, Chair of Materials Handling, Material Flow, Logistics (fml). His research areas are innovative identification technologies, digital planning of logistics systems and human factors in logistics. After obtaining his Ph.D. at TUM, chair fml in 2002, he worked in several management positions at Swisslog before becoming managing director of MIAS Group. Since 2015, he also has worked at the Association of German Engineers (Verein Deutscher Ingenieure, VDI) as chairman for Bavaria and vice-chairman for manufacturing and logistics.

Birgit Vogel-Heuser
Technical University of Munich, Institute of
Automation and Information Systems,
Boltzmannstr. 15, 85748 Garching bei
München, Germany
vogel-heuser@tum.de

Prof. Dr.-Ing. Birgit Vogel-Heuser received a Diploma degree in Electrical Engineering and a Ph. D. degree in Mechanical Engineering from RWTH Aachen. Since 2009, she is a full professor and director of the Insititute of Automation and Information Systems at the Technical University of Munich (TUM). Her current research focuses on systems and software engineering. She is member of the acatech (German National Academy of Science and Engineering), fellow of IEEE, editor of IEEE T-ASE, and member of the science board of MIRMI at TUM.