Leveraging inconsistency management in the multi-view collaborative modelling of cyber-physical production systems

Birgit Vogel-Heuser1, Minjie Zou

1Institute of Automation and Information Systems, Technical University of Munich, Germany

Abstract: Cyber-physical production systems (CPPS) are interconnected systems producing customised products. Such a system appeals for flexibility and closer cross-disciplinary collaboration, which might introduce inconsistencies among models and lead to system failures. Detecting inconsistencies needs expert knowledge and is usually performed in an ad-hoc way. Knowledge base, a technology which stores knowledge formally for automatic problem-solving, can efficiently assist engineers in detecting inconsistencies across engineering models and between the models and their physical realisations. The approach is evaluated with a bench-scale CPPS.

1 Introduction

Cyber-physical production systems (CPPS) in the paradigm of Industry 4.0 are equipped with high flexibility, high customisation, high interconnectedness, and short product life cycles [1], which also introduces various challenges. First, customisation leads to the high variability of products with small batch sizes, which requires a quick (re)design of the manufacturing system and configuration of manufacturing resources [2]. With increasing complexity, CPPS expand beyond the boundaries of disciplines and call for cutting-edge cross-disciplinary interoperability, and synthesis. In CPPS, it is not only the physical system that is networked, but also the data, engineers, tools, and teams.

In CPPS development, disciplines (such as mechanical/electronic/software engineering) collaborate and shape different views (e.g. construction, dynamics optimisation, functional specification) on the same system by using models (e.g. graphs, math, text, simulation). A model is usually used by a particular team for a specific purpose, e.g. mechanical engineers apply 3D CAD models for mechanical design, while systems dynamics engineers use simulation models for validation and optimisation. Once these models are developed parallel on a system without proper synchronisation, inconsistencies emerge. An inconsistency by definition is when the artefacts in the models shaping different views on the same system are contradictory or cannot meet a specification [3]. Tiny unaware inconsistencies may lead to unintentional data errors or even system realisation failures, resulting in tremendous loss. Although various inconsistency identification approaches have been proposed [4], most of them are limited to tangible inconsistencies, e.g. syntactical or value inequivalence. Intangible inconsistencies related to domain knowledge are scarcely ever checked.

The major contribution of this paper is maintaining dependencies and detecting potential inconsistencies across multidisciplinary models during the CPPS design process by combining formal domain knowledge and modelling knowledge. This combination enables a semi-automated detection of inconsistencies, and therefore, reduces the requirements on engineers regarding the cross-disciplinary knowledge, the manual efforts in consistency checking, and the risks during the collaborative modelling.

2 Related work on inconsistencies management

Inconsistency management by definition is the process of handling dependencies in a way that the goals of stakeholders are concerned [3]. In this section, the related work of inconsistency management from software engineering, requirements engineering, mechatronic design, data engineering, and CPPS engineering are reviewed.

In software engineering, the consistency among software artefacts is essential. Nuseibeh et al. [5] and Finkelstein et al. [6] propose a general process to manage inconsistencies (including detection, diagnosis, and handling) during collaborative software development with respect to defined consistency rules. Based on this, Spanoudakis and Zisman [3] further investigate the methods and techniques (e.g. model checking, logic reasoning, human-based collaboration) to support inconsistency management. Towards a more efficient process, approaches have been proposed to either achieve a single underlying reference metamodel [7] or link separate metamodels [8]. In [8], Eggy et al. develop a cloud platform to assist engineers in exchanging and validating data. These studies do not cover the cross-model inconsistency problem, as they adopt the unified modelling language.

In the manufacturing systems design area, model-based systems engineering [9] is gradually adopted due to the increasing need for a common tool to integrate views and data from different domains. Systems modelling language (SysML) [10] and AutomationML [11] have gained attention in such a context. Consistency within AutomationML can be checked with logical rules or predefined query patterns [12]. Semantic inconsistencies can be prevented by using a shared reference library [13]. Feldmann et al. [14] identify typical inconsistency types during the cross-disciplinary engineering: type incompatibilities, violations of accepted theories and laws, different settings of the same object, and non-conformance to best practices. Based on this, a model-based approach is proposed to capture inter-model dependencies and check inconsistencies using triple graph patterns.

The knowledge base gains increasing application in handling inconsistencies, which stores data in a mutual and machine-processable language and processes them in a formal way. The knowledge base has been applied in various scenarios, e.g. data integration [15], manufacturing system configuration [16], and sensor data stream monitoring [2]. Most of these mainly focus on the data consistency without considering the multi-view modelling process and the modelling conventions.

Overall, despite the multitude of approaches to modelling manufacturing systems, semantic validation, and model verification are not supported inherently. By contrast, these are assured within the area of knowledge representation, whereas engineering entities in modelling are neglected. Thus, a combined approach is required to fulfil the following requirements: (R1) creating a cross-disciplinary understanding of the system to be...
forms from diverse models and to trace dependency on both the disciplines, (R3) automating the inconsistency detection process.

The final phase, inconsistencies can be identified through reasoning inconsistency identification across models define formal rules for consistency checking. Such rules can be contributed to generalising dependencies, repeating the procedures based on the identified rationale as consistency rules.

Inconsistency identification across models (R2) detecting inconsistencies across models and disciplines, (R3) automating the inconsistency detection process.

3 Concept of knowledge-based reasoning and inconsistency identification across models

In this section, a knowledge-based approach for automated reasoning and checking inconsistency is proposed for the collaborative and multi-view modelling of CPPS.

3.1 Approach to detecting inconsistencies across models

The approach is taken in three phases (cf. Fig. 1): dependency identification, transforming into the knowledge base, and inconsistency identification. These phases cross different knowledge abstraction levels: ontology, metamodel, and model instance. The approach starts with dependency detection on the model level, which can be either semantic dependent (elements representing the same phenomena) or functional dependent (elements modelling dependent phenomena). Consequently, a knowledge base is required to store information from dissimilar forms from diverse models and to trace dependency on both the metamodel level and ontology level. These abstract levels contribute to generalising dependencies, repeating the procedures and automating the process. The knowledge base also enables us to define formal rules for consistency checking. Such rules can be logically and generally expressed, such as modelling rationale. In the final phase, inconsistencies can be identified through reasoning based on the identified rationale as consistency rules.

3.2 Knowledge base

Since the engineering models in CPPS are in different forms. They need to be aggregated in the knowledge base in the same representation. The knowledge base stores knowledge distilled in three levels: domain ontology level, metamodel level, and individual instances (including objects in reality and modelling artefacts) in the lowest level. Domain ontology, which explains concepts, properties of concepts, and relations between concepts in a domain, can formally represent different forms of knowledge. These include the general concepts about the product, manufacturing process, and equipment (resources) to execute manufacturing tasks, which are otherwise known as the product-process-resource ontology. A domain ontology serves as a common understanding in semantics across disciplines.

In this paper, a light-weighted ontology in manufacturing automation from [15] is adopted (cf. Fig. 2). The product ontology takes account of the product property and requirement. The process ontology contains concepts of operations and actions. An operation is defined as a set of actions in a particular sequence. The resource ontology includes the station and the component concepts. The station is referred to machines conducting manufacturing operations (e.g. transporting, processing). The component includes elements performing individual actions.

In addition to the established domain ontology, related industrial standards can also be captured and stored in the domain ontology, such as eCl@ss dictionary (classification and description of component features) and IEC 62541 (the standard interface specification OPC Unified Architecture).

A Metamodel is a model that defines the language for expressing a model according to the meta-object facility standard, which specifies the essential components and relations in a model. Based on the metamodel, a model gives a simplified representation of a certain reality. Individuals include objects in reality and modelling artefacts, which are concretely substantiated and parameterised. By contrast, the ontology level and the metamodel level are in more general and abstract, which are concepts or classes representing a group of individuals with similar features.

For realising the knowledge base, the Web Ontology Language (OWL) is adopted, which is standardised by the World Wide Web Consortium for the knowledge representation. It introduces additional predefined vocabulary along with description logic semantics. OWL supports the specification of detailed constraints and relations between classes, instances, and properties, e.g. disjointness, enumerated classes, and characteristic of properties.

3.3 Inconsistency reasoning

Reasoning by definition is deriving facts that are not expressed in the knowledge base explicitly. The reasoning mechanism in this paper has an inference and a query part. The inference engine draws new conclusions based on logical rules. The knowledge in the knowledge base can be divided into the abstract and timeless concepts (terminological knowledge, TBOX) and the specific knowledge about individuals (assertional knowledge, ABOX). With reasoning, the existing relations in the knowledge base can be analysed. New relationships can be inferred from logical implications. After reasoning, inconsistencies can be detected by matching the existing elements and relations with the expected one (consistency patterns). In this paper, we mainly consider two types of inconsistencies across models (cf. Fig. 3): ⊗ inconsistency between modelling artefacts representing the same object, ⊖ mismatch of the association between the modelling artefacts and their target relation in reality.

4 Use case open research demonstrator

A bench-scale manufacturing system Pick & Place Unit (PPU, cf. Fig. 4) is used to illustrate the challenges and inconsistencies in the multi-view collaborative modelling of CPPS. The bottleneck part of PPU is a crane, which picks up various types of workpieces from a stack and delivers them either to the stamp for processing or
to a conveyor system for further transportation. In this paper, small excerpts out of various models of PPU [Various models of PPU are online available: https://github.com/x-PPU] are selected (cf. Fig. 4): a domain-spanning SysML model for specifying the system structure and parameters, and a Simulink model for the specification and simulation of the system behaviour. Additionally, a component plan represents the unique PPU to be realised in reality.

In the PPU evolvement and redesign process, systems engineers update the SysML model, while simulation engineers modify simulation models parallel, and the component plan provides a view on the current PPU. Implicit dependencies and constraints exist in these models and serve as consistency rules. For example, the motor is modelled in both models: SysML considers its attributes and association, while Simulink focuses on its dynamics (① in Fig. 4). In addition, in SysML model, the crane contains a turning table connected to a sensor (micro-switch) for position detection, which should be matched with the component plan, i.e. the PPU to be developed in reality (② in Fig. 4).

In order to manage the inconsistencies in PPU, following problems are addressed by applying the approach proposed in this paper: Can the models with different views map to the same realisation without contradictions (cf. requirement R1)? Can the dependencies between models be captured for engineers to detect potential inconsistencies (R2)? Is it possible to automate the detection process (R3)?

For a common understanding of the system, a knowledge base for PPU is set up to associate the metamodels and the domain ontology. The terminological part (cf. Fig. 5) consists of the metamodels of SysML and Simulink and the product-process-resource ontology. Metamodels depict the semantics of modelling languages. For example, the stereotypes within the SysML profile include blocks, interfaces, ports, and functionalities, which are associated with each other. While a Simulink model is composed of blocks with ports (inputs/outputs) and flows between ports.

Based on the knowledge base, additional relations between individuals and additional type definitions of individuals can also be derived automatically through reasoning. Regarding the two examples aforementioned, it is given that a motor (ID:m11x), a turning table (ID: m1), and a micro-switch (ID: S-3-BE) are specified in the component plan for resource planning. The motor is parallel modelled in SysML and Simulink. Following dependencies can be reasoned out: the motor blocks in both models should be associated (cf. ① in Fig. 6); the crane and micro-switch should be facilitated in the component plan as designed in SysML (cf. ② in Fig. 6). After reasoning, the consistency can be checked by matching the facts during the modelling process with the knowledge base with reasoning results as the consistency patterns.

The reasoning process is automated with the following procedure: (see Fig. 7 ) The checking results will be returned to the modelling engineers, respectively to assist them to have a cross-boundary view of other parallel modelling processes. Regarding the implementation, the knowledge base of PPU is stored in OWL. Rules are defined in the Jena Rule Language syntax. Inconsistency checks are conducted with SPARQL queries, which consist of a clause identifying the type of query, a pattern to be matched against the knowledge base, and when retrieving information, a list of information to be returned. OWL API and Apache Jena API are used to access the knowledge base and conduct reasoning.

5 Discussion, conclusion, and future work

A knowledge-based approach is proposed to handle the inconsistency problems in the CPPS design by combining formal domain ontology and engineering models. The ontology is utilised to store essential concepts and relations in manufacturing systems, while the metamodels are used to capture mappings between heterogeneous model syntaxes. Challenges and the application are illustrated with a bench-scale PPU.

Based on PPU, the application process is projected and shows how the approach can support engineers to systematically check inconsistencies. Besides, three examples are given representing
different consistency rules, e.g. dependency across models (R2), relations between models and their physical realisations, and the adaptation of manufacturing resources for product requirements (R1). In the application process, the domain ontology captures the fixed relations in manufacturing (R1), while linking metamodels maintains inter-model dependencies (R2). The combined knowledge base allows semantic interoperability and automated inconsistency checking (R3).

Although adopting a holistic modelling language instead of using heterogeneous models can help to avoid the problem addressed in this paper. Our approach is superior in that it allows engineers to stick to the tools and languages that they are used to.

However, there still remain limitations. First, the inconsistencies in this paper are limited to static semantics. This is sometimes problematic because the behaviours in physics are not always consistent with models due to the model fidelity and uncertainties in reality. Secondly, the approach still requires manual efforts in specifying dependencies. In this paper, we take the dependency capture as a pre-condition for our approach. In this way, we focus on the desired dependencies as critical positions for potential inconsistencies. In our future work, dependency detection will be conducted in a more automated way of detecting more potential inconsistencies. Dynamic models will be investigated to cover the time-dependent factors in CPPS during design and simulation. Furthermore, scalability and performance for large-scale CPPS modelling projects will be evaluated.

6 Acknowledgment

The authors acknowledge the German Research Foundation (DFG) for funding this work in the CRC 768/3-D1 at TUM.

7 References

[1] Vogel-Heuser, B., Hess, D.: ‘Guest editorial industry 4.0 - prerequisites and visions’, IEEE Trans. Autom. Sci. Eng., 2016, 13 (2), pp. 411–413
[2] Zinnikus, I., Antalaki, A., Kapahnee, P. et al.: ‘Integrated semantic fault analysis and worker support for cyber-physical production systems’. IEEE 19th Conf. on Business Informatics, July 2017, pp. 207–216
[3] Spanoudakis, G., Zisman, A.: ‘Inconsistency management in software engineering: survey and open research issues’, in ‘Handbook of software engineering and knowledge engineering’, Volume I: Fundamentals (World Scientific, 2001), pp. 329–380
[4] Zou, M., Vogel-Heuser, B.: ‘Feature-based systematic approach development for inconsistency resolution in automated production system design’. IEEE Int. Conf. on Automation Science and Engineering (CASE), August 2017, pp. 687–694
[5] Nuseibeh, B., Easterbrook, S., Russo, A.: ‘Leveraging inconsistency in software development’, Computer, 2000, 33 (4), pp. 24–29
[6] Finkelstein, A., Spanoudakis, G., Till, D.: ‘Managing interference’. Joint Proc. of the Second Int. Software Architecture Workshop and Int. Workshop on Multiple Perspectives in Software Development, San Francisco, California, USA, 1996, pp. 172–174
[7] Ananieva, S., Klare, H., Burger, E. et al.: ‘Variants and versions management for models with integrated consistency preservation’. Proceedings of the 12th Int. Workshop on Variability Modelling of Software-Intensive Systems, Madrid, Spain, 2018, pp. 3–10
[8] Eyed, A., Zeman, K., Hehenberger, P. et al.: ‘Maintaining consistency across engineering artifacts’. Computer, 2018, 51 (2), pp. 28–35
[9] Estefan, J.A.: ‘Survey of model-based systems engineering methodologies’, Incnse MBSE Focus Group, 2007, 25 (8), pp. 1–12
[10] Object Management Group: ‘Systems modeling language (SysML)’, version 1.4, 2015
[11] Lüder, A., Pauli, J., Rosendahl, R. et al.: ‘Support for engineering chain migration towards multi-disciplinary engineering chains’. IEEE 14th Int. Conf. on Automation Science and Engineering (CASE), Munich, 2018, pp. 671–674
[12] Wimmer, M., Mazak, A.: ‘From AutomationML to AutomationQL: a by-example query language for CPPS engineering models’. IEEE 14th Int. Conf. on Automation Science and Engineering (CASE), August 2018, pp. 1394–1399
[13] Hildebrandt, C., Schoz, A., Fay, A. et al.: ‘Semantic modeling for collaboration and cooperation of systems in the production domain’. 22nd IEEE Int. Conf. on Emerging Technologies and Factory Automation, Limassol, 2017, pp. 1–8
[14] Feldmann, S., Kernschmidt, K., Wimmer, M. et al.: ‘Managing inter-model inconsistencies in model-based systems engineering: application in automated production systems engineering’. J. Syst. Softw., 2019, 153, pp. 105–134
[15] Sabou, M., Ekaputra, F.J., Biffl, S.: ‘Semantic web technologies for data integration in multi-disciplinary engineering’, in ‘Multi-disciplinary engineering for cyber-physical production systems’ (Springer, Cham, 2017), pp. 301–329
[16] Ferrer, B.R., Ahmad, B., Lobov, A. et al.: ‘An approach for knowledge-driven product, process and resource mappings for assembly automation’. 11th IEEE Int. Conf. on Automation Science and Engineering (CASE), August 2015, pp. 1104–1109