SN Computer Science (2023) 4:205
https://doi.org/10.1007/s42979-022-01572-5

ORIGINAL RESEARCH

Traceable Multi-view Model Integration: A Transformation Pipeline for Agile Production Systems Engineering

Felix Rinker[1,2] · Laura Waltersdorfer[2] · Kristof Meixner[1,2] · Dietmar Winkler[1,2] · Arndt Lüder[3,4] · Stefan Biffl[2,4]

Received: 1 November 2021 / Accepted: 16 December 2022 / Published online: 10 February 2023

Abstract

Purpose. Agile Production Systems Engineering (PSE) is characterised by parallel and iterative engineering of several disciplines. This multi-view engineering requires capabilities for tracing changes to support configuration management of PSE assets. Yet, traditional model transformation approaches in PSE do not preserve local views and hierarchies on concepts of PSE assets, such as plans and configurations. Thus, tracing multi-view changes to PSE assets is challenging.
Method. Following the Design Science approach, we (i) elicit requirements for tracing multi-view changes to PSE assets from a domain analysis in automotive manufacturing; (ii) introduce and evaluate the *Traceable Multi-view Model Transformation (TMvMT)* process; and (iii) propose the TMvMT pipeline architecture to provide traceable model integration capabilities for agile PSE.
Results. In a feasibility study on robot cell models, we evaluate the TMvMT process and architecture regarding the requirements for traceability compared to traditional approaches.
Conclusion. The proposed TMvMT approach provides traceability of changes in multi-view modelling as a basis through the separation of modelling transformation steps and provision of clear input and output artefacts to achieve traceable configuration management and validation of system designs for production system assets in agile PSE.

Keywords Production systems engineering · Domain-specific modelling · Model-driven engineering · Domain-specific languages · Model transformation · Multi-disciplinary engineering

Introduction

Innovative Cyber-Physical Production Systems (CPPSs) [1], envisioned in the Industry 4.0 (I4.0) Initiative[1] [2], aim at enabling sophisticated functionalities, such as predictive maintenance or digital twins. However, many of these techniques create large amounts of data, requiring new methods to facilitate the reproducible management and integration of engineering artefacts and data.

[1] Industry 4.0 Initiative: https://www.plattform-i40.de.

This article is part of the topical collection "Model-Driven Engineering and Software Development" guest edited by Slimane Hammoudi and Luis Ferreira Pires.

Laura Waltersdorfer, Kristof Meixner, Dietmar Winkler, Arndt Lüder, Stefan Biffl have contributed equally to this work.

✉ Felix Rinker
Felix.Rinker@tuwien.ac.at

Laura Waltersdorfer
Laura.Waltersdorfer@tuwien.ac.at

Kristof Meixner
Kristof.Meixner@tuwien.ac.at

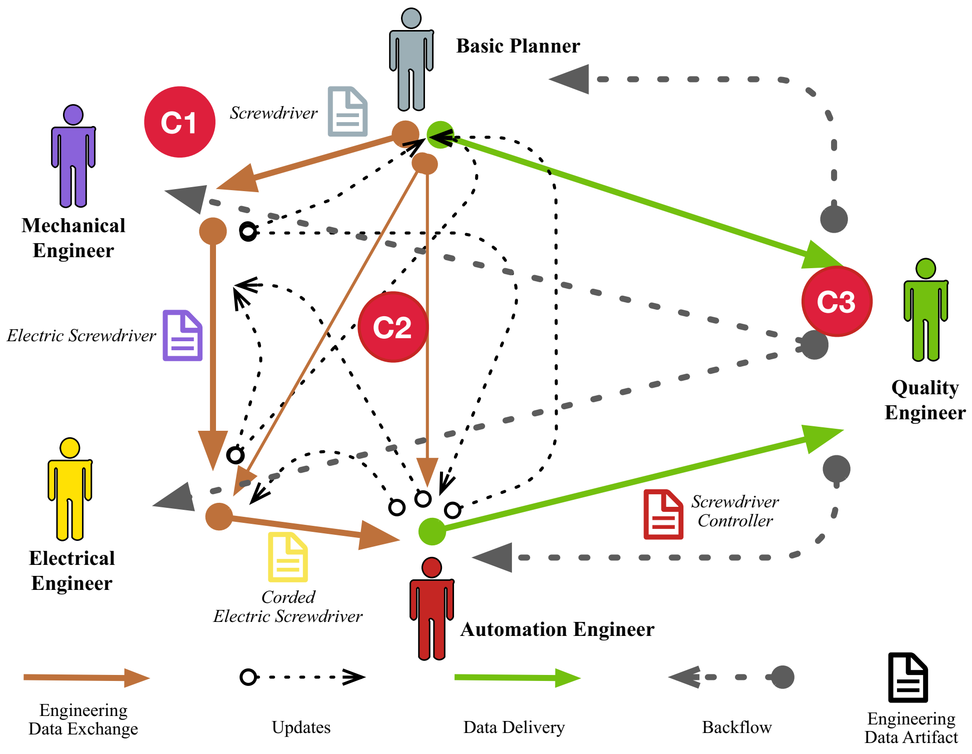
Dietmar Winkler
Dietmar.Winkler@tuwien.ac.at

Arndt Lüder
Arndt.Lueder@ovgu.de

Stefan Biffl
Stefan.Biffl@tuwien.ac.at

[1] Christian-Doppler-Laboratory SQI, TU Wien, Vienna, Austria
[2] Institute of Information Systems Engineering, TU Wien, Vienna, Austria
[3] Institute of Ergonomics, Otto von Guericke University, Magdeburg, Germany
[4] Center for Digital Production, Vienna, Austria

Fig. 1 Challenges in point-to-point exchange of engineering artefacts in agile PSE [10]



The artefacts describe views on PSE assets with dependencies, e.g., joining products requires a correct level of force depending on consistent contributions from the mechanical, electrical, and software designs [3]. Consistent PSE asset integration is needed to validate the functionality and quality of modern CPPSs [1] and to reduce the risk of project and production delay due to late design changes.

To manage multi-disciplinary PSE, the Association of German Engineers published the VDI 3695 [4], a guideline regularly used in the PSE context to describe conceptual measures for engineering organizations. Particularly, VDI 3695 Part 2 (Processes) refers to an integral process called *Configuration Management* (CM). This process constitutes managing engineering work and its changes throughout the plant lifecycle.

Similar to CM in Software Engineering, it should enable a consistent and accurate documentation of various characteristics, such as requirements, design, or testing concerning the actual physical design [5]. However, production systems have reached a high level of complexity that makes conventional approaches to engineering knowledge integration inadequate [3, 6]. Therefore, this paper focuses on how to enable improved configuration management and, thus, traceability on the level of data.

We aim to track changes of property values of PSE assets and integrate discipline-specific engineering artefacts to improve capabilities for agile PSE. A challenge in PSE is that engineers typically conduct *point-to-point* artefact exchange (cf. Fig. 1), with only limited capabilities for multi-view model versioning or change trace management. Improving these capabilities requires a holistic view, combining several views of engineering stakeholders and project management, e.g., for quality management to test different configurations of planned designs. Currently, there is neither a holistic overview on nor a complete collection of concepts used in a typical PSE project, e.g., for virtual engineering [7], hampering traceability.

In practice, traceability is the capability to audit a change's origin to a common multi-view model element by following the data flow back to an engineering artefact in a particular stakeholder view. For instance, the value of a property *torque* of a screwing process should be linked to the relevant M-CAD tool data sheet (cf. Fig. 3). In the following, we will discuss several challenges to achieving adequate traceability.

Challenges to Traceability in agile PSE. From the industrial use case *Position-and-Screw Robot Cell* from automotive manufacturing (cf. Sect. 4.1 and literature [6, 8, 9]), we have identified several challenges. Figure 1 illustrates how engineers conduct data and model exchange in PSE across workgroup borders, in artefact-based transactions [9]. Furthermore, the figure shows the challenges marked with labels in red.

C1 Data artefact exchange instead of model exchange. Custom file formats encode information on data elements and hierarchies, in various forms, such as PDF, spreadsheets, and technical drawings. Information is often extracted manually and not integrated, after each update [8].

C2 Manual local model mapping and integration efforts. Changes to multi-view models in PSE artefacts are hard to trace using manual backtracking. For example, it is hard to decide whether an input file is new data or an update of a previous version, if tool data artefacts do not provide version identifiers, e.g., in the file name. Model transformation and integration require high effort due to custom data formats and implicit model and mapping knowledge, which is not formalised. These processes are not automated and associated with high effort for the data consumers [6], which can lead to error or loss of data.

C3. Missing common concept view. PSE tool suites are tailored to selected engineering domains' requirements and hence fulfil use-case specific purposes for individual disciplines [11]. However, such tool suites (e.g.: Creo[2], AutoCAD[3] or EPLAN[4]) do not provide a holistic view and data management beyond their limited scopes. For example, there can be up to 15 disciplines and 50 tools in a PSE project [12]. However, a partial overview is often only achieved in later phases of the project with increased efforts and when changes become more expensive and traceability more complicated.

To overcome these challenges, we investigate how agile software engineering approaches, such as model-based software engineering [13] and continuous development [14], have identified solution approaches to automate and maintain software development processes. However, these approaches assume only software artefacts and are not sufficiently tailored to traceable multi-view model changes in PSE [15].

Therefore, the main goal of this paper is to design and evaluate an approach that facilitates **traceable multi-view model transformations** in agile PSE. To address this goal, we apply the Design Science approach, building on our recent paper [7] with the following aims.

Aim 1. Traceability requirements. We define challenges and requirements for tracing multi-view changes to PSE concepts. Therefore, we build on the PSE guideline VDI 3695-2 for configuration management maturity [4] and on the results of a domain analysis [16, 17] on work cells in automotive manufacturing. This contribution targets guiding the design of an approach for traceable multi-view data modelling capabilities.

Aim 2. Traceable process. We introduce the *Traceable Multi-view Model Transformation (TMvMT) process* with knowledge representation in intermediate models for PSE data integration. This process and knowledge representation provide the foundation for implementing flexible model transformation workflows for a defined scope of work. For evaluating the viability in the scope of a manufacturing work line, we conduct the process with selected changes to engineering artefacts (a) to integrate the domain knowledge scattered over several engineering views and (b) to verify the traceability for changes to property values in the integrated model.

Aim 3. Software architecture. We refine the Multi-view Model Transformation (MvMT) software architecture building on the flexible pipeline software design [7], to automate tasks in the TMvMT process and consider traceability as a main goal of the workflow.

This work builds on and extends our previous work on continuous model integration [7] and domain-specific modelling [18]. In a first step, a more detailed model transformation process is described, resulting in an integrated engineering graph. Next, the *Multi-view Modelling Framework (MvMF)* design towards traceable model transformation is refined to automate asset property value propagation. Furthermore, we extended the Product-Process-Resource (PPR) Domain-Specific Language (DSL) to define dependencies of requirements and integrated it into the TMvMT. Finally, the model transformation process is evaluated in comparison to alternative approaches.

In summary, our contributions in this paper to the *Computer Science (CS)* community are as follows:

(i) We provide insights to CS researchers on PSE domain concepts and traceability issues.
(ii) The *TMvMT process* to define traceable and flexible multi-view model transformation pipelines for building intermediate models and an integrated PSE model, based on agile workflows to support distributed modelling.
(iii) The paper explores the *TMvMT architectural system design* for such a modelling transformation pipeline.
(iv) A feasibility study for the TMvMT approach by providing a demonstrative use case and implementation based on the PSE standard AutomationML (AML) and automated with a Continuous Integration (CI) system.

The remainder of this paper is structured as follows: Section Related Work summarises related work. Section Research Questions and Approach motivates the research questions and approach. Section Engineering Use Case and Requirements presents an illustrative use case on robot-based *positioning* and *joining* of car parts and requirements for traceability. Section Traceable Multi-view Model Transformation describes (a) the *Traceable Multi-view Model Transformation (TMvMT) process* to combine stakeholder view models into an integrated engineering model with capabilities for tracing changes to PSE assets

[2] Creo: https://www.ptc.com/creo.

[3] AutoCAD: https://www.autodesk.com/autocad.

[4] EPLAN: https://www.eplan.de.

and (b) the TMvMT software architecture to automate the TMvMT process, an improved model transformation workflow based on continuous integration and model engineering. Section Evaluation with a Feasibility Study demonstrates the feasibility of the TMvMT approach with the illustrative use case *Position-and-Screw Robot Cell* and compares the capabilities of the TMvMT approach to alternative model transformation approaches in PSE. Section Discussion discusses the research results and limitations. Section Conclusion and Future Work summarises the research findings and proposes future research.

Related Work

This section summarizes related work on modelling interoperability, integrated production system modelling for Industry 4.0, and model-based IT Operations (DevOps).

System Modelling

Modelling interoperability is an essential topic in (domain-specific) system modelling, which has already gained research attention over the last decades: Tolk and Muguira [19] generalise conceptual interoperability and distinguish five classes of models, from complete black-box applications to white-box applications with harmonised data structures. The *Athena* framework [20], a reference framework for enterprise applications, considers three dimensions: (a) conceptual, (b) applicative, and (c) technical integration. All three layers are essential components for ensuring exchange and integration of system models.

Model operations, especially model transformations, are another essential measure to achieve the interoperability of domain models. The Model-Driven Engineering (MDE) community has established best-practice concepts and frameworks: the ATLAS Transformation Language (ATL)[5] framework conforms to the Meta Object Facility (MOF)[6] specifications for model transformations and provides the foundations for assuring modelling interoperability [21]. EMF Compare [22] is a framework for model comparison, conflict detection and merging, mainly for technically convergent application contexts.

In PSE, the technical divergence and multi-disciplinary nature of the domain require the specification of explicit dependencies and relations between different system perspectives, based on implicit knowledge. Vogel-Heuser et al. conceptually describe two approaches to overcome inconsistencies, either *a priori* or *a posteriori*, based on multiple use cases [3].

Both the modelling community and the PSE community require adequate multi-view modelling processes and frameworks [23, 24]. Atkinson et al. [25] suggest minimum requirements for multi-level modelling, including: "*some fundamental notion of abstracting a multitude of model elements to a common classifier*". Tunjic and Atkinson [26] conceptualise a Single Underlying Model (SUM) as a common unified model, which could be automatically populated based on the information from the single views. Therefore, the authors describe a modelling approach to preserve local views by defining mappings and representations between local and common views.

To summarise, there are many established approaches from model engineering ranging from the generation of software code to model comparison, which are beneficial for automation and quality assurance purposes. However, these approaches do not offer extended domain-specific capabilities for the PSE context. To fill this gap, evolving initiatives in industrial PSE increasingly have proposed designs, which we will discuss in the following subsection.

Integrated System Modelling for Industry 4.0

PSE aims to create a consistent set of engineering models required to physically set up the intended production system [4]. According to the guideline VDI 3695 [4], engineering organizations might expose different levels of quality related to their technical background, business processes, and capabilities. These levels of target statuses range from A (lowest maturity) to D (highest maturity) and have descriptions which features characterise them. In the following, we will refer to specific maturity levels and describe how to achieve them for relevant topics.

Model Architecture Types

In industrial PSE practice, we can observe three main architecture approaches establishing the technical background of model transformation in engineering organizations: Manual Model Transformation (MMT), Tool Suite with input/output Model Transformation (TS-MT), and Multi-view Model Transformation (MvMT).

MMT concerns the manual transfer of engineering data/models between engineering tools [12, 27]. This approach requires only capabilities for exchanging data files and organizational rules to coordinate the data exchange. Proper timing of exchange in the PSE process and sufficient model transformation capabilities are essential. Benefits are low effort for setup and operation for simple settings. However, support for traceability is limited the risk and effort increase more than linear for larger, more complex

[5] ATL: https://www.eclipse.org/atl.

[6] MOF: https://www.omg.org/mof.

settings. According to the guideline VDI 3695 [4], MMT provides, at best, sparse integration between the data sets of two disciplines.

The more advanced approach TS-MT is based on tool suites (e.g., Siemens COMOS[7]) supporting the integrated engineering of a subset of the relevant PSE disciplines [11]. TS-MT offers much support for dedicated use cases. However, TS-MT requires high effort for the initial setup as well as for the adaption to new views and lacks flexibility beyond dedicated use cases. According to the guideline VDI 3695 [4], TS-MT partly integrates some disciplines very well, while disciplines outside the selected scope of the tool suite are not well integrated and, in general, hard to integrate.

MvMT covers all relevant PSE disciplines, however, without dedicated tracing capabilities. MvMT systems are based on a common semantic understanding of all data relevant within an engineering organization and enable automated PSE data propagation along the complete engineering tool chain of the PSE organization [7, 28, 29]. Considering the requirements of the VDI 3695 [4] for PSE projects, MvMT fully integrates disciplines with reasonable incremental effort. This characteristic leads to a common model for a sufficiently complete scope of PSE stakeholders. However, change propagation with MvMT does not necessarily imply the traceability of changes.

In this paper, we build on these traditional approaches for model transformation to evaluate traceability capabilities (cf. Sect. 6.3).

Domain-Specific Modelling Technologies

In PSE, DSLs are essential for facilitating model-based engineering in specialised domains and implementing the Industry 4.0 vision for complex data-driven use cases, such as smart production [30]. The VDI 3695-3 [4] guideline on description languages (DL) specifically focuses on this aspect. Level A (lowest maturity) requires the usage of structured description languages. To achieve level VDI 3695 DL-D (highest maturity), additionally identical facts have to be always described equally and the mapping of languages to semantics has to be consistent.

However, PSE processes are often optimised for *intra-disciplinary* activities lacking the common view perspective [31]. This issue limits the effectiveness and efficiency of *interdisciplinary* knowledge exchange and favours domain-specific approaches. Expressing the dependencies within production systems requires comprehensive models as the different disciplines and their concepts are inherently linked [32].

In software engineering, efforts have been made to introduce code pipelines that automate building and deployment processes [33] and could be adapted to the PSE context.

Several initiatives have developed industry standards that have been increasingly used to model and specify domain-specific contexts: For instance, modelling languages are Systems Modelling Language (SysML)[8] and AutomationML (AML)[9] for engineering data exchange, the Business Process Model and Notation (BPMN)[10] for generic process description in use cases, or Simulink[11] for control and signal processing. However, data exchange in PSE settings still concerns the exchange of documents [34] with highly heterogeneous data formats, hindering seamless model integration and transformation. Integration on multiple levels is required to reap the benefits of digitization in the manufacturing domain: Hildebrandt et al. [35] showcase meta-models described in AML and with domain-specific ECLASS[12] attributes for a mechatronic model.

The BaSys 4.0 project[13] investigated a run-time middleware for Industry 4.0 (I4.0) components [36] and assumes the modelling of stakeholder views as I4.0 Asset Administration Shell (AAS) I4.0 Asset Administration Shell (AAS) [37], a digital representation of the I4.0 component. However, the BaSys 4.0 project did not consider how to derive the required I4.0 ASS specifications from multi-view engineering artefacts.

Schleipen et al. [38] introduced a common integrated model to PSE, the *PPR* model, which is based on the three main aspects of a production system. These are the *product* with its properties, the *process* that produces this product, and the *resource* that executes production processes. The *Formalised Process Description (FPD)* [39] represents these aspects in a technology-agnostic way by defining a graphical notation and a data model for the functional view on a production system. However, the FPD does not provide functionality to formulate consistency constraints, like the maximal weight of a set of production resources.

Meixner et al. [18] introduced the PPR-DSL, a machine-readable and technology-agnostic DSL for PSE modelling, to represent PPR aspects and constraints between these aspects. Furthermore, they developed a mapping of the constraint syntax to recursive Structured Query Language (SQL) queries [40] to execute them on relational database systems, a well-established technology in PSE. In this paper,

[7] https://www.siemens.com/comos.

[8] SysML: https://www.sysml.org.

[9] AutomationML: https://www.automationml.org.

[10] BPMN: https://www.bpmn.org.

[11] Simulink: https://www.mathworks.com/simulink.

[12] ECLASS: https://www.eclass.eu.

[13] BaSys 4.0: https://www.basys40.de/.

we build on the PPR-DSL [18] to express the functional view in PSE. We extend it to describe dependencies for enabling traceability of value changes across discipline-specific views. Early in PSE, this view models the requirements towards a production system (cf. Sect. 4).

To conclude, comprehensive and systematic integration of all relevant system aspects is of high importance to enable innovative use cases around Industry 4.0 and CPPSs. For this reason, we build in this paper on established modelling frameworks, such as EMF Compare and the MOF specification, with domain-specific multi-view capabilities [7]. Current approaches focus on the integration of run-time processes, while our work focuses on integrating model in the engineering phase. To ensure traceability during the engineering phase, we investigate solution approaches from software engineering, such as continuous integration, regarding their suitability for the PSE context.

Model-based DevOps for Traceability

Traceability is an essential feature for multi-disciplinary PSE [41, 42] approaches. To achieve traceability, collaborating engineering work groups need to agree on boundary objects [43]. These boundary objects act as links for tracing across gaps between different disciplines and provide essential foundations for configuration management. Guideline VDI 3695-2 [4] concerns configuration management in PSE and defines four maturity levels from A (lowest maturity) to D (highest maturity). The overall aim is to move towards coordinated configuration management for multi-view models, instead of basic isolated discipline-specific configuration. These multi-view models should incorporate all relevant disciplines and require reference models for each discipline and the connection across disciplines to achieve maturity level CM-D in the VDI 3695-2 [4]. Such examples might include mechanical and electrical plans, software code, and respective configurations for the automation and simulation purposes (cf. Sect. 4).

In software engineering, DevOps [14] focuses on enabling continuous integration to achieve, among other goals, traceability: agile approaches, such as the Git workflow[14], and orchestration and automation software (e.g., Docker[15], Ansible[16], Chef[17]) improve the quality and traceability of software development processes. Furthermore, configuration management in continuous integration environments is based on configuration files (often text) that provide details about infrastructure and parameters to trace different versions. Both approaches, model-based engineering and the DevOps movement, have established methodologies and tools for supporting traceable workflows and increasing software quality. However, until now, these approaches have been focused on code-related, text-based artefacts and not on models Garcia et al. [33] extend the concept of continuous integration and present a model-based DevOps approach with continuous development tools. Wortmann et al. [30] survey the state of art in modelling languages in industrial contexts. As one contribution, they propose a vision of combining the model-based approach with DevOps technologies to support Industry 4.0 developments and point out relevant research directions. In this paper, we build on the traceability concepts in the guideline VDI 3695-2 [4] and the model-based DevOps vision in [30] to design a traceable multi-view model transformation approach.

In this paper, we extend our previous work [7] by the use case *Position-and-Screw Robot Cell* (cf. Sect. 4). From this use case, we design an integrated engineering view represented in the PPR-DSL [18]. The resulting multi-view engineering graph is the foundation (i) to model a SUM and (ii) to specify model transformation configurations that provide an agile model transformation pipeline (cf. Sect. 5).

Research Questions and Approach

In this paper, we follow the *Design Science* methodology [44] to address the main research question asking *what approach facilitates traceable multi-view model transformations in agile PSE*. To this end, we investigate how to improve the representation and transformation of multi-disciplinary knowledge for tracing changes to PSE assets in engineering artefacts, achieving VDI 3695-2 maturity level CM-D [4] (cf. Sect. 2). Figure 2 shows the Design Science methodology for this work instantiated as research steps and contributions.

Step 1, *Environment Analysis*, concerned the investigation of the results from a domain analysis [16] (cf. Sect. 4). This domain analysis was conducted on 80 types of robot-based screwing processes in the automotive industry, which is representative for discrete manufacturing for a product portfolio with high variability. The investigation resulted in (i) an abstracted description of the stakeholders, their tasks, views, typical engineering artefacts, and data integration concerns. Based on the use case (ii), *requirements* were derived on multi-disciplinary knowledge representation and integration for PSE change management and traceability from changed engineering artefacts to an integrated data model.

Step 2, Design/Build (cf. Fig. 2), derived the following research questions.

RQ1. Process Design. *What process enables traceable multi-view model transformation workflows in agile PSE?*

[14] Git Workflow: https://www.git-scm.com/about/distributed.

[15] https://www.docker.com.

[16] https://www.ansible.com.

[17] www.chef.io.

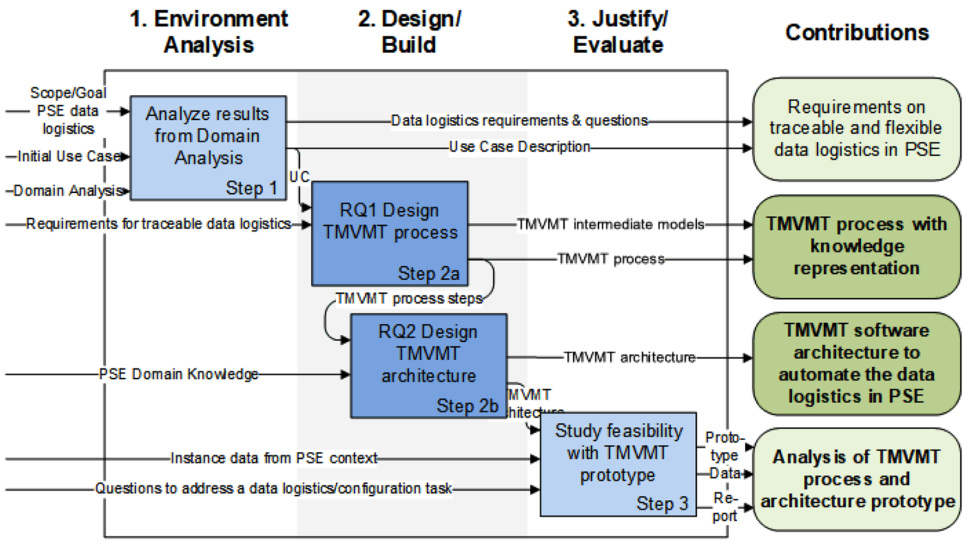

1. Environment Analysis
2. Design/ Build
3. Justify/ Evaluate
Contributions
Scope/Goal PSE data logistics
Initial Use Case
Domain Analysis
Requirements for traceable data logistics
Analyze results from Domain Analysis Step 1
UC
Data logistics requirements & questions
Use Case Description
Requirements on traceable and flexible data logistics in PSE
RQ1 Design TMVMT process Step 2a
TMVMT intermediate models
TMVMT process
TMVMT process with knowledge representation
TMVMT process steps
RQ2 Design TMVMT architecture Step 2b
PSE Domain Knowledge
TMVMT architecture
TMVMT software architecture to automate the data logistics in PSE
Study feasibility with TMVMT prototype Step 3
Instance data from PSE context
Questions to address a data logistics/configuration task
Proto-type
Data
Re-port
Analysis of TMVMT process and architecture prototype

Fig. 2 Research steps, methods, and contributions (in IDEF0 notation [45])

Domain experts exchange heterogeneous data in multiple iterations and horizontally across disciplines. This context can lead to changes coming from any of these disciplines. Domain experts require traceable transformation workflows for avoiding inconsistencies, errors, or data silos. To address RQ1, we designed the TMvMT process (cf. Sect. 5.1) with a focus on traceability (i) to define a multi-view engineering graph, (ii) to configure a multi-view modelling environment, and (iii) to execute the data integration pipeline for propagating shared data, provided from engineering artefacts, to an integrated model and to data consumers (cf. Fig. 7).

RQ2. Architecture Design. *What software architecture enables a traceable model transformation workflow in agile PSE?* Efficient model transformation requires an architecture that is compatible with typical PSE system landscapes. To address RQ2, we designed the TMvMT software architecture (cf. Sect. 5.2) to automate major parts of the TMvMT process. We extend our previous architecture design [7] (i) with capabilities to describe multi-view engineering graphs and (ii) with extended rule engine functionalities to enable attribute value traceability.

Step 3, Justify/Evaluate (cf. Fig. 2), aims at demonstrating the feasibility of the TMvMT approach with the illustrative use case *Position-and-Screw Robot Cell* (cf. Sect. 4). We analyse the TMvMT process results, e.g., integration of a local view to a common view PSE data model, for improving the traceability of changes. Further, we aim at better understanding benefits and limitations of the TMvMT approach in comparison to traditional approaches. Therefore, we conducted the TMvMT process to instantiate a TMvMT pipeline. We evaluated to what extent the TMvMT process and prototype fulfil the requirements for traceability and the effort for conducting the TMvMT process. We compared the results to traditional alternative approaches in PSE: (i) *Manual Transformation* between engineering artefacts, without a common view [12] (cf. Sect. 4); (ii) a *Tool Suite* with a limited common view [11]; and (iii) our previous work of the *MvMT* [7] with a common unified view, but without traceability concerns.

Engineering Use Case and Requirements

This section describes the illustrative use case *Position-and-Screw Robot Cell* from automotive manufacturing [16], and requirements for traceable multi-view model transformation workflows.

Position-and-Screw Robot Cell

The use case *Position-and-Screw Robot Cell* is based on a domain analysis [16] from discrete automotive manufacturing, i.e., the production of car parts and cars. The production lines used in automotive manufacturing utilise industrial robots in mounting units, e.g., to position and fasten screws. A typical car production plant holds around 200 to 300 of such robot cells.

For illustrative purposes, we assume a scenario with two robot cells, one for *Positioning* and one for *Screwing* the required car screws (cf. Fig. 3). The purpose of this production process is to mount a dashboard to a car body. The process consists of two steps: The first robot cell carries out the correct positioning of the dashboard and the screws in the car body. Then, the second robot cell fastens the screws and measures the result. A major challenge in planning the production process is integrating and coordinating the different discipline-specific engineering artefacts, which are iteratively affected by updates of PSE design decision outcomes.

Stakeholders and their views. Figure 3 shows selected stakeholders and their partial views on the overall production system design. The PSE process starts with functional system planning, followed by detailed mechanical and automation engineering.

For conciseness, Fig. 3 shows the *Quality Engineer* as a proxy for the combined views of the *Product Designer* and the *Functional Planner*. The *Product Designer* is responsible for the design of the product, such as a car part, and has to consider customer and technical requirements. *Functional Planners* take up the product design and requirements from the previous step and develop a conceptual production system design to produce the product variants required by the customer. Therefore, they define the input and output products together with the production system and quality attributes, such as cycle time and screwing torque. Additionally, they specify the main CPPS resources, which are required for the production processes.

In the detailed design phase, detail engineers select and integrate concrete system components to ensure the feasibility of the conceptual production system design. While real-world engineering scenarios involve 15+ different views, we focus on detailed mechanical, electrical, and automation engineering.

Detailed *mechanical engineering* concerns concrete mechanical system parameters and arguments, adding mechanical characteristics to processes and resources. For example, the *Mechanical Engineer* would add the mechanical property *torque* to the abstract electric screwdriver.

Common concepts concern, e.g., the property *torque* that is assigned to the process and related to the *torque* of the electric screwdriver. The values of such an attributes can be propagated to the other attribute. In Fig. 3, these dependencies are represented as orange relations. Furthermore, CPPS subresources, such as two drives, provide subfunctions for the main CPPS resources.

Detailed *electrical engineering* concerns the wiring to supply energy and information to production system resources. For instance, the *Electrical Engineer* specifies CPPS subresources, such as the transformer, the robot controller, and the screwdriver controller. The electric layout also defines network details regarding high and low power supply and the fieldbus network.

Detailed *automation engineering* builds on the aforementioned artefacts to design configurations and programs that automate the behaviour of the production system. Examples for such artefacts are, e.g., available resources, conceptual process design, and input from mechanical/electrical engineering. Many of the technical details for system components usually stem from existing in-house technologies or third-party vendor catalogues.

The columns in Fig. 3 categorise common concepts based on the PPR notation [39]: *Products & Processes* and CPPS *Resources*, detailed as Main CPPS Resources, CPPS SubResources, and Automation Resources. Furthermore, *Plant Networks* provide information and electrical power supply, and topological information. *Engineering Artefacts* represent stakeholder documents, which contain actual engineering data values according to stakeholder views.

Requirements for Traceable Multi-view Model Transformation

From the domain analysis of the use case [16] and the VDI 3695-2 [4] maturity status level CM-D for configuration management for PSE, we derived the following *requirements (Rx)* for traceable and agile model transformation workflows.

R1. Multi-view modelling capabilities. The PSE process needs to support multiple stakeholder views and their artefacts (cf. Fig. 3). Therefore, stakeholders should be able to define local concepts in discipline-specific design views and models that can be mapped to common concepts. The transformation workflow shall support and preserve these multiple stakeholder views regarding engineering artefacts and the integrated model as a foundation for tracing back model changes.

R2. Distributed process synchronization. The engineering disciplines have to synchronise and discuss changes to designs to gain a common view of updated information. This synchronization capability is the foundation to check for inconsistencies (a) in the common view, e.g., inconsistent changes to several values that depend on each other, or (b) between stakeholder views, i.e., inconsistent values of one common concept in different stakeholder views. The transformation workflow shall provide capabilities for synchronising distributed engineering processes sufficiently in the parallel and iterative development of different production system parts.

R3. Traceable model change representation. Heterogeneous artefacts are common in PSE: Discipline-specific concepts and representation formats describe different views. The transformation workflow shall consider: (i) a common view that bridges these concerns and provides a comprehensive understanding of dependencies and links of the system model; (ii) the representation of change dependencies; (iii) the capability to trace back changes from the integrated model to its sources, such as a change to a model element, e.g., a property value in a local model view, which can be directly mapped or semantically linked to the integrated model; and (iv) a description language for defining a common view and the dependencies.

R4. Version representation/management. The traceable model transformation shall represent and facilitate managing versions of engineering models and artefacts as required for parallel and iterative collaboration of several engineering disciplines. For example, the VDI 3695-2 guideline [4] requires capabilities for resetting a model to a historical state if changes lead to an invalid production system

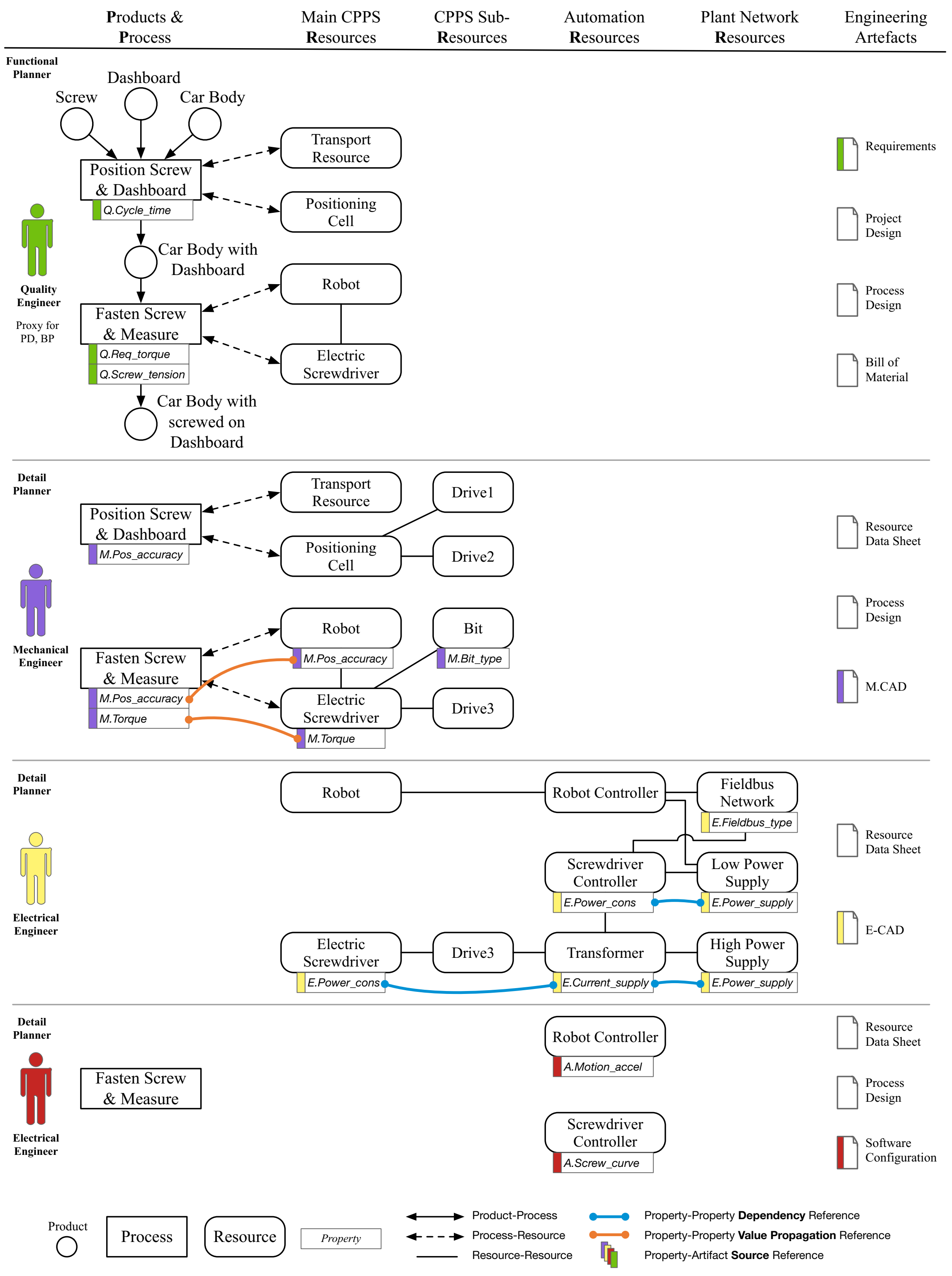


Fig. 3 Use case *Position-and-Screw Robot Cell*: stakeholder views, concepts, and artefacts

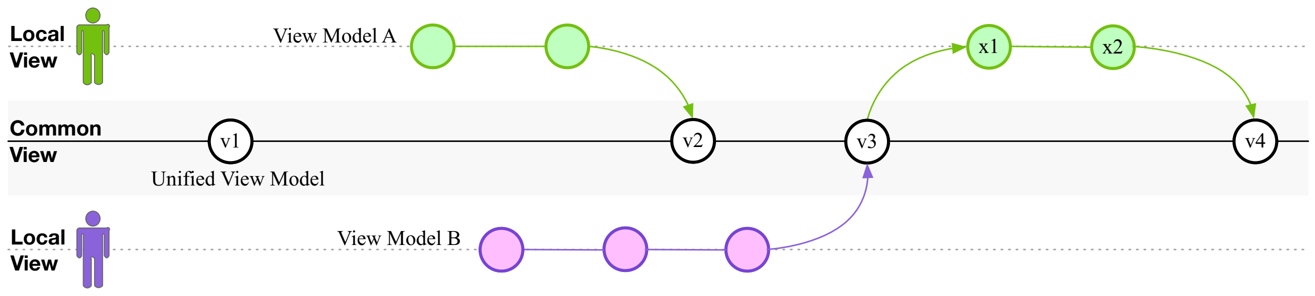


Fig. 4 Agile multi-disciplinary artefact synchronization based on the SUM [26] and on Git workflow concepts

configuration. The transformation workflow shall provide capabilities for version representation/management to conduct configuration management in PSE.

Traceable Multi-view Model Transformation

To address the requirements for traceable model transformation workflows (cf. Sect. 4.2), we propose in this section (i) the TMvMT process to combine stakeholder view models into an integrated engineering model and to configure a multi-view modelling environment and (ii) the TMvMT software architecture to automate the TMvMT process in a multi-view modelling environment.

Traceable Multi-view Model Transformation Process

The collaborative and parallel nature of PSE requires a common understanding and agreement on boundary objects [43], e.g., building on an integrated engineering graph. Furthermore, traceability in the parallel working environment requires flexible model and data exchange. To achieve such a flexible multi-view data integration and exchange, we build on the Multi-view Model Transformation (MvMT) workflow [7], which is explained in the following subsection.

Multi-view Model Transformation Workflow

Multi-view model transformation requires the synchronization of multiple disciplines and collaborative workflows. These advanced capabilities are required to address more complex goals, such as traceability or documentation, e.g., for digital twins, predictive maintenance, or retrofitting tasks.

Established process models for such collaborative approaches are, for instance, defined workflows for source code management. Inspired by the agile development movement, Git supports an agile distributed non-linear workflow, initially developed for software engineering. However, while Git is well suited for text-based change detection and tracing, it lacks capabilities for advanced analysis on a semantic level, which is required for tracing model changes (cf. also [22]).

The MvMT workflow [7] is based on the SUM architecture [26] and on the Git workflow. In our case, the SUM is represented by a *unified view model* that explains how different discipline-specific views and overlapping model components are mapped into a common view. For illustrative purposes, Fig. 4 depicts the interaction of two stakeholder views with the *unified view model*, consisting of the tasks: (1) integration of *View Model A* into the *unified view model* (cf. label *v2* in Fig. 4); (2) integration of *View Model B* into the *unified view model* (cf. label *v3* in Fig. 4); and (3) export of the modified *View Model A* from the *unified view model* (cf. label *x1* in Fig. 4). While engineers work on their particular local views, changes that concern the common view are incorporated into the *unified view model*.

However, conducting the MvMT workflow requires first the definition of a *unified view model* (cf. Fig. 4) considering all relevant views. Therefore, additional steps are required to facilitate tracing changes back to the local engineering views. These TMvMT process steps are described in the following subsection.

Traceable Multi-view Model Transformation Method

To address RQ1 (cf. Sect. 3), Fig. 5 illustrates the TMvMT process that consists of the following three steps to prepare inputs for MvMT pipelines: (1) the definition of a multi-view engineering graph as a foundation for (2) the configuration of the multi-view modelling environment; and (3) the execution of the data integration pipeline.

Step 1: Definition of multi-view engineering graph. To achieve traceability and a holistic overview of engineering artefacts, an integrated engineering view is required (cf. Fig. 9). Therefore, a specialised domain expert, the *data curator*, guides the engineers of the different disciplines in the process of defining a *multi-view engineering graph*. This graph provides a common understanding of concepts, their relations between each others, mappings between different concepts, and dependencies. The definition can be separated into the following abstraction levels:

Concepts. Domain experts collect local concepts of their domains from relevant engineering artefacts in their particular

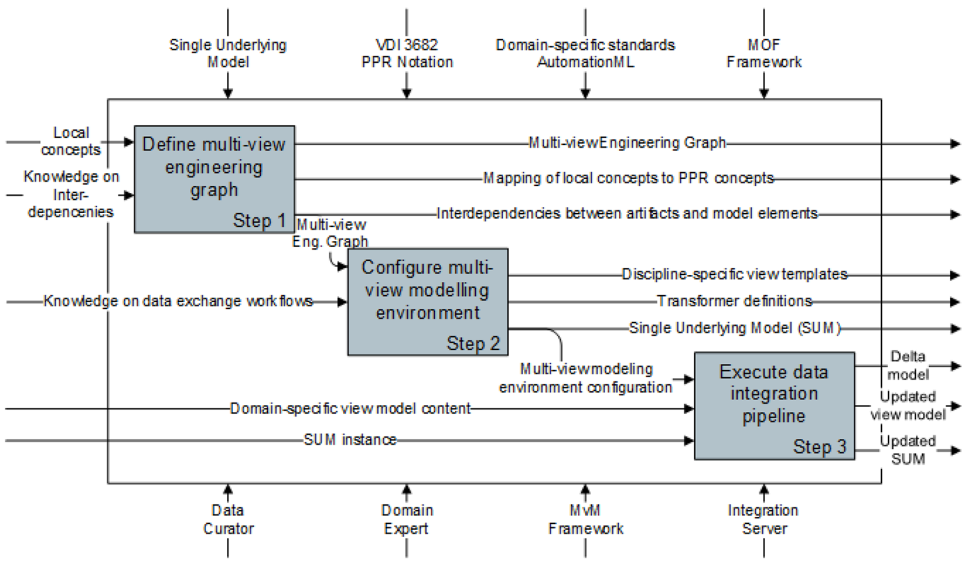

Single Underlying Model
VDI 3682 PPR Notation
Domain-specific standards AutomationML
MOF Framework
Local concepts
Knowledge on Inter-depencencies
Define multi-view engineering graph
Step 1
Multi-view Engineering Graph
Mapping of local concepts to PPR concepts
Interdependencies between artifacts and model elements
Multi-view Eng. Graph
Knowledge on data exchange work flows
Configure multi-view modelling environment
Step 2
Discipline-specific view templates
Transformer definitions
Single Underlying Model (SUM)
Multi-view modeling environment configuration
Execute data integration pipeline
Step 3
Delta model
Updated view model
Updated SUM
Domain-specific view model content
SUM instance
Data Curator
Domain Expert
MvM Framework
Integration Server

Fig. 5 TMvMT process steps (in IDEF0 notation [45])

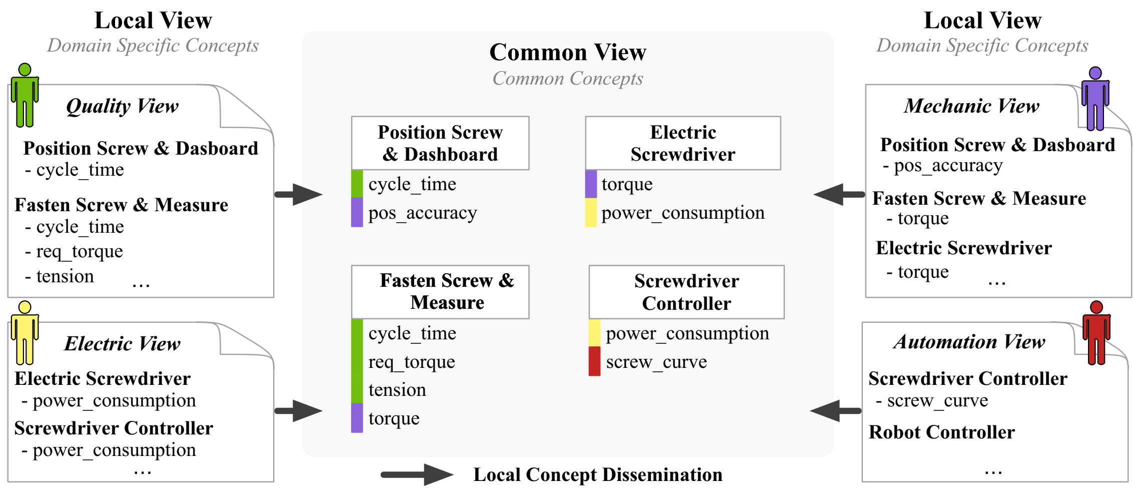


Fig. 6 Exemplary mapping of selected local concepts in stakeholder views (cf. Fig. 3) to a common view, based on the Common Concept Glossary (CCG) approach [31]

Local View. Examples are in the *Quality View* the *required torque* (cf. Fig. 6 top-left) and in the *Mechanical View* the *torque* (cf. Fig. 6 top-right).

Common Concepts. The engineers map the local concepts to Common Concepts (CCs) following the CCG approach [31] depicted in Fig. 6. For instance, for the process *Fasten Screw & Measure*, the domain-specific attributes *torque* in the view of the quality engineer and the mechanical view are mapped in the *Common View* to the process *Fasten Screw & Measure* attribute *torque* (cf. Figs. 3 and 6). The result is a list of mappings between the concepts of different domain-specific views and the common view.

Links. There are two main purposes for links: (i) They specify from which engineering artefacts (e.g., documents) values originate, and (ii) change propagation of dependent values, if the source element of an element is changed, the target element should also be changed. The data curator, with the support from the domain experts links, e.g., the local mechanical *torque* attribute to the corresponding mechanical artefact source such as a M-CAD drawing (cf. Fig. 3). For this task, a linking language is needed to represent semantic dependencies. Therefore, for our purposes, we will utilise a semantic linking language based on the *RefSemantic* concept [46] following the URI schema RFC3986[18].

[18] RFC3986: https://datatracker.ietf.org/doc/html/rfc3986.

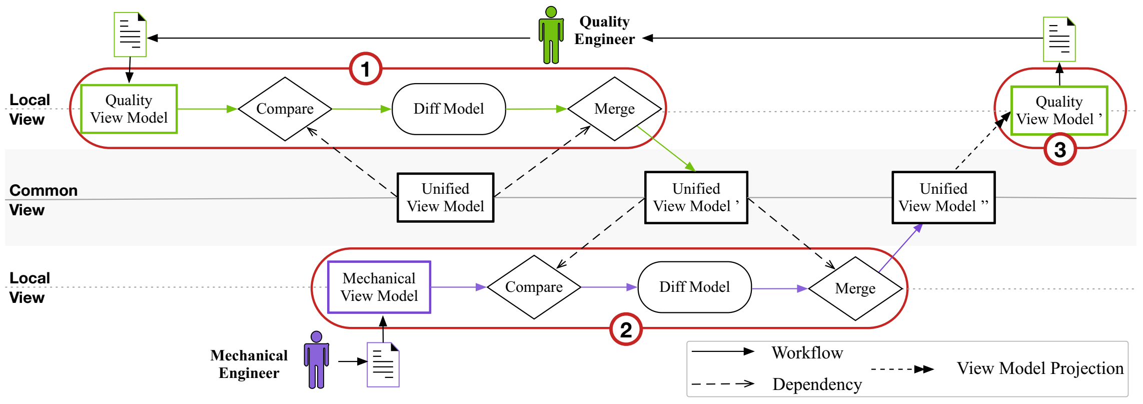


Fig. 7 Model transformation workflow for combining tool artefacts in an SUM [26] and the Git workflow. (1) Integration of the *Quality View Model*, (2) Integration of the *Mechanical View Model*, and (3) Export of the modified the *Quality View Model*, based on [7]

Engineering Graph Template. Conceptual planners, responsible for the plant floor design, create a first production system design that initiates the initial engineering graph. Therefore, they use the previously defined CCs by categorising them according to the PPR aspects: product, process, and resource.

Updated Engineering Graph. This initial design is then filled in and updated by the other domain experts, who specify open and/or updated domain-specific attribute values. This information is required to reflect sources and interdependencies between different discipline model elements and attributes in the graph. The multi-view engineering graph resulting from this step is the basis to generate the unified and discipline-specific view models.

Step 2: Configuration of the multi-view model transformation environment.

The setup of an executable TMvMT environment requires (i) a *unified view model*, acting as SUM; (ii) d*iscipline-specific view models*; and (iii) *data integration workflow descriptions* between the disciplines.

The basis for the *unified view model* is the engineering graph created in Step 1. However, the graph needs to be adapted to be used in the data integration workflow. The project generator of the MvMT framework supports the generation of the *unified view model* based on the updated engineering graph. The *discipline-specific view models* will be also generated with the generator from the engineering graph. These templates provide the scope for the view-specific local concepts needed to describe the discipline-specific data points. For the *date integration workflow descriptions*, (i) process descriptions of the data flow required as well as, (ii) data and model capabilities, such as *text-to-model* and *model-to-model* operations. Furthermore, the data curator designs model data exchange flows between the discipline-specific data sources. To automate the data exchange flows, the data curator creates transformer definitions, describing the mapping of the discipline-specific model concepts to the concept in the unified model.

Step 3: Execution of multi-view model transformation pipeline.

Based on the multi-view model transformation configuration coming from Step 2, the transformation pipeline is executed to perform the required model transformation workflows. Figure 7 depicts an exemplary TMvMT workflow that consists of three steps, Step 3.1 to Step 3.3.

Step 3.1 Integration of the Quality View Model. The *Quality Engineer* starts the model transformation workflow by editing an artefact in a discipline-specific tool, *Tool A*, (cf. Fig. 7, upper left-hand side). The engineer wants to integrate the modelling information into the unified view model (cf. Fig. 7, Common View lane). First, the artefact is exported from the discipline-specific *Tool A* into an export format, e.g., a Extensible Markup Language (XML) or Comma Separated Value (CSV) file. Then, a transformer transforms the *Tool-A*-specific format into the *Quality View Model*, respectively, the previously defined discipline-specific template. This populated template is then compared to the SUM (cf. Fig. 7, Common View lane) to detect differences between the two versions based on the changes. Changes can include new elements and the modification of elements, e.g., the change of a property value. The result of this step is a list of changes, which can be reviewed by the *Quality Engineer*. A core advantage of this task is to allow the engineer to specify which changes to accept or decline. Based on this finalised list, the changes are merged into the unified model, creating a new version. The changes are then available for all stakeholders of the workflow when accessing the new unified model version (cf. Figs. 4 and 7).

Step 3.2 Integration of the Mechanical View Model. The tool-specific data of a discipline-specific tool, *Tool B*, have to be transformed into the discipline-specific *Mechanical View Model* using the corresponding template. Then, the transformed structure is compared to the new unified model version. Different to Step 3.1, this unified model version

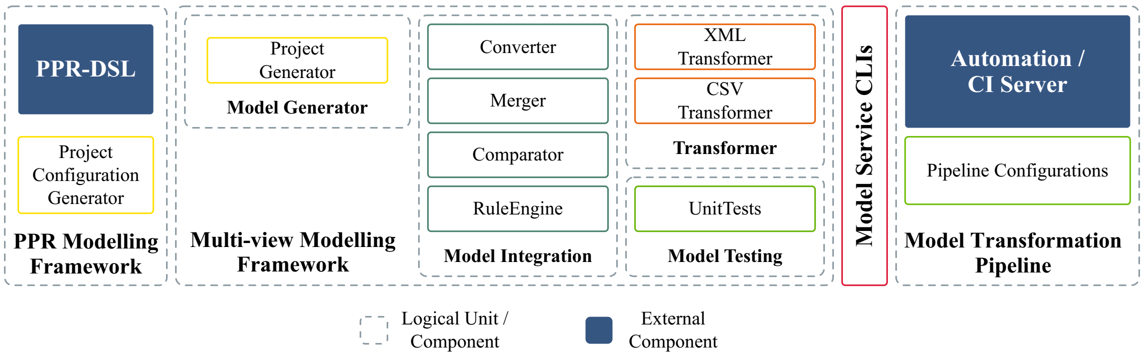


Fig. 8 Architectural system design of the TMvMT *Pipeline System* and its components

incorporates the previous changes to the *Quality View Model*. Similar to Step 3.1, the changes are calculated resulting in a list of changes. The *Mechanical Engineer* can select or reject changes and merge the model data to the unified model version creating an updated model.

Step 3.3 Extraction of the Modified Quality View Model. Based on the unified model version, created in Step 3.2, the *Quality Engineer* can extract the most recent unified model version, which incorporates the changes coming from both the *Quality Engineer* and the *Mechanical Engineer*. The local view of the *Quality View Model* can be generated from this updated unified model version, enabling the *Quality Engineer* to access the data in her discipline-specific *Tool A*.

Conducting these TMvMT workflows requires the following capabilities in an architecture: (i) the capturing and support of the distributed working process; and (ii) advanced model comparison approaches that extend text-based diffs to model-based change analysis.

Traceable Multi-view Model Transformation Architecture

To automate TMvMT process, this section describes an architectural system design for a TMvMT pipeline for PSE. Figure 8 shows the proposed architecture of the TMvMT pipeline system that contains, from left to right, three main components: (i) the PPR Modelling Framework (PPRMF), (ii) the Multi-view Modelling Framework (MvMF), and (iii) the Model Service Command Line Interfaces (CLIs) and (iv) the Model Transformation Pipeline (MTP) component.

In [7], we developed the MvMF to provide the multi-view modelling work process capabilities. The MvMF is motivated by the Eclipse Modelling Framework (EMF), a meta-modelling framework that offers comparing and merging functionality [47] for integrating heterogeneous models. However, EMF is tightly coupled with Eclipse[19] and has complex interdependencies. These issues make the EMF hard to use for users without model-driven software engineering expertise or set up the EMF in custom software solutions [48]. This shortcoming is a major drawback in the PSE application context.

From the domain analysis (cf. Sect. 4) and previous work [6], we learned that accessibility and understandability of modelling and model integration processes are major concerns for engineers. Approaches in industry, such as low code, have been devised [49, 50] to reduce setup and configuration effort for domain experts, who are not familiar with software engineering or model engineering techniques. Therefore, we applied the principles from the EMF to our use case by designing a light-weight Service-Oriented Architecture (SOA) to enable model engineering and transformation in PSE with little setup and configuration effort. In the following, we will explain the different components in more detail:

The **PPR Modelling Framework (PPRMF)** (cf. Fig. 8) provides the PPR-DSL and the *Project Configuration Generator*. The PPR-DSL is an external component that supports the modelling of PPR networks by providing a text-based definition language as well as parsing and validation features [18]. To enable traceability, we extended the PPR-DSL model and the prototypical DSL framework with semantic linking features. This extension allows to describe property value sources, dependencies, and propagation of attribute values to other properties (cf. Sect. 6 and Fig. 10).

We newly introduced the concept of *Relations*. There are three types of *Relations*: (i) *Source Relations*, specifying the origin of a value, (ii) *Dependency Relations*, that provide a linking between the source value and the dependent value, and (iii) *Propagation Relations*, values in one concept that need to be propagated to other concepts. Relations can have the following attributes: type (source, dependency, or propagation), semantic (reference key for the engineering artefact), and reference (attribute name in the engineering artefact or target attribute).

The *Project Configuration Generator* uses the specified PPR network and generates a project configuration. This configuration consists of common concepts, partial discipline-specific

[19] Eclipse: https://www.eclipse.org.

views and attributes, and relations between attributes to source elements. Furthermore, it contains an initial graph network to generate the unified view and local view templates. This configuration subsequently serves as an input for the *Project Generator* (cf. Sect. 6 and Fig. 11).

The **Multi-view Modelling Framework (MvMF)** uses the SUM approach as a metamodel according to Layer 2 of the MOF architecture to preserve local views while providing mappings to a unified view. The framework implements model operations including: *Model-to-Text* and *Model-to-Model* transformation, model *Comparison* and *Merging*, model *Injection*, and model *Validation*. The *MvMF* includes four components:

Model Generator. The *Project Generator* (cf. Fig. 8) constructs the SUM template and corresponding discipline-specific views required for the MvMF for a particular project. For this reason, discipline-specific knowledge and hierarchies stemming from custom tools must be externalised and encoded in a common computer-readable format (e.g., YAML[20]), the project generator configuration.

Model Integration. The *Model Integration* components (cf. Fig. 8) support the multi-view model integration workflow and consist of four model operation services: The *Converter* restructures a view-specific model into a SUM compliant-structure to provide the required input for the *Comparator* in the next step. In addition, the service can also retrieve a view-specific model from the SUM, if required by an engineer. The *Comparator* derives the delta model by comparing two models and calculating the differences. The *Merger* merges changes from the comparator into the SUM. The *RuleEngine* enables traceability between element mappings. Semantic links between elements (cf. Fig. 10) provide the foundation for automated change propagation. For instance, if the torque value in the mechanical view is semantically equal to the torque value in the quality view, a change in one view will be propagated to the other view.

Transformer. The *CSVTransformer* and *XMLTransformer* provide *Text-to-Model* transformation to import and export the tool-specific artefacts. The transformer has to be configured for an engineering artefact using a custom object mapping language (cf. Fig. 12).

Model Testing. *UnitTests* check the consistency and quality, e.g., of the model data. In the development phase, these tests can check new configuration workflows [51].

The *Model Service Command Line Interfaces (CLIs)* component facilitates access to MvMF services via CLIs. This CLI service can be used to define a workflow by combining several services in a shell script. Furthermore, this enables a flexible configuration of MvMF services in DevOps automation tasks, like build pipelines on a continuous integration server.

The *Model Transformation Pipeline* provides means for workflow definition descriptions. The different MvMF services can be orchestrated through these pipeline configuration using a domain-specific language. New pipelines can easily be defined and deployed, by providing an additional pipeline configuration, e.g., generation of a report based on SUM for management.

[20] YAML: https://yaml.org/.

Evaluation with a Feasibility Study

This section demonstrates the feasibility of the *Traceable Multi-view Model Transformation (TMvMT) approach* with the illustrative use case *Position-and-Screw Robot Cell* (cf. Sect. 4). We conducted the TMvMT process (cf. Sect. 5.1) to instantiate a traceable model transformation pipeline and an integrated model for the use case. Based on results of the feasibility study, we evaluate the TMvMT approach regarding requirements for traceability in comparison to three traditional alternative approaches (cf. Sect. 2.2): (i) *Manual Model Transformation* between engineering artefacts, without a Single Underlying Model (SUM) (cf. Sect. 4); (ii) a *Tool Suite with a limited SUM*; and (iii) Multi-view Model Transformation with an SUM, but without traceability concerns.

Evaluation Context

Based on the use case *Position-and-Screw Robot Cell* (cf. Sect. 4), Fig. 9 illustrates the combined view of the separate stakeholder views for the process *Fasten Screw & Measure* shown in Fig. 3, as a result of conducting the TMvMT process. We will refer to *Electric Screwdriver* as one example of a common concept, integrating the mechanical and the electrical view through semantic links. The mechanical property torque is extracted from M-CAD engineering artefacts. The electric property power consumption is extracted from E-CAD engineering artefacts. We will showcase the traceability functionality through the two types of property linkings: source reference, from where a value originates from, dependency reference and propagation reference to change a dependent value based on a source value.

Feasibility Study

This section investigates the feasibility of the TMvMT approach by instantiating a model transformation pipeline of the use case. Full versions of the discussed excerpts of the multi-view engineering graph and the configurations, as well as as set of input and output files of an example pipeline execution can be accessed on this online repository[21]. Binaries of the the prototype that are used to execute the pipeline can be found in the Bin folder.

[21] TMvMT Resources Repository: https://github.com/tuw-qse/tmvmt-resources.

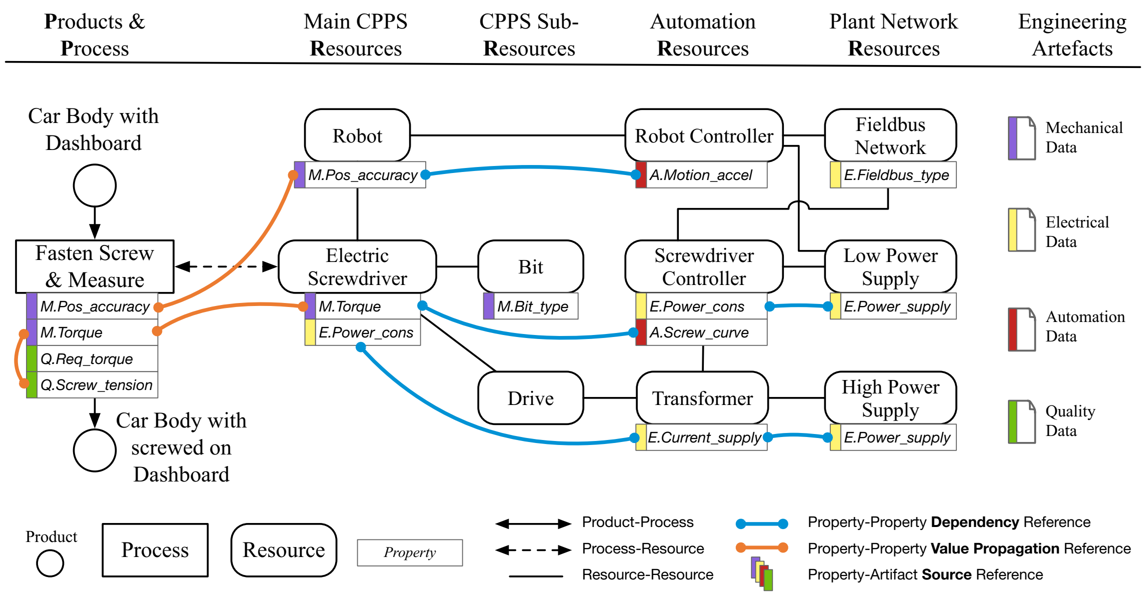


Fig. 9 Multi-view engineering graph of the process *Fasten Screw & Measure* based on the the use case *Position and Screw*: with change dependency trace links, in an adapted VDI 3682 notation, based on [17]

Step 1 of the TMvMT process (cf. Sect. 5.1) defines the multi-view engineering graph.

Figure 9 shows the conceptual view of such an engineering graph. To achieve traceability, the data curator needs to add relationships between concept attributes to the graph. As described previously, there are three different types of such relations: blue lines indicate *Dependency*; orange lines *Value Propagation* and *Source References*.

Figure 10 shows an excerpt of a common concept *Electric Screwdriver* in the engineering graph defined in the PPR-DSL (cf. screw-dashboard.dsl[22], depicting different views and the source artefact. Attribute names (Line 4) are prefixed by the view name to organise them later on into different views in Step 2. For example, the attribute *mechanical. torque* has two relations (lines 4-15). The first relation of *type source* describes a *reference* to another attribute in the Process hierarchy . In this case, the element is located in an XML document, and can be retrieved using an *XQuery*[23] term, defined in the *reference* property. The *propagate* reference describes how an attribute refers to another attribute in the engineering graph, which can be automatically updated during the change process. Similarly, a second attribute *power_consumption* is defined for the *electrical* view (lines 14–19).

Step 2 configures the model transformation workflow environment. Therefore, first, the PPR engineering graph needs to be translated by the *Project Configuration Generator* (cf. Table 1) into the *project configuration*. A project configuration defines the discipline-specific view models to guide the transformation of local concepts into corresponding concepts in the SUM template. Figure 11 shows a simplified example project configuration, specified in a simple structured markup language, Yet Another Markup Language (YAML).

The discipline-specific view models are specified in lines 5-8 and transform the local concepts into concept descriptions in the SUM definition. Our example concept *ElectricScrewdriver* is specified (line 12) and the concept mappings according to the model views (PPR, mechanical, electrical) are defined (lines 14, 16, and 31). Attributes are put under the respective view, based on the view prefix from Step 1. In our case, the attribute *torque* with prefix *mechanical* in the engineering graph (Fig. 10, line 4) matches the defined mechanical-view attribute *torque* in the project configuration (Fig. 11, line 18). Each view provides further tool-specific attributes. As a result, the SUM configuration contains all relevant views, concepts, attributes, and reference links in a list (cf. generator-config.yml[24]). The *AML Project Generator* (cf. Table 1) is used to build the AML SUM model and local view models templates[25]).

[22] screw-dashboard.dsl: https://raw.githubusercontent.com/tuw-qse/tmvmt-resources/main/uc-screw_dashboard/generate/project-config-pprdsl/input/screw-dashboard.dsl.

[23] XQuery: https://www.w3.org/XML/Query.

[24] generator-config.yml: https://raw.githubusercontent.com/tuw-qse/tmvmt-resources/main/uc-screw_dashboard/generate/project-config-pprdsl/output/generator-config.yml.

[25] Generated Model Templates: https://github.com/tuw-qse/tmvmt-resources/tree/main/uc-screw_dashboard/generate/project/output.

As a next task, the data curator has to define *data transformation workflows*. These discipline-specific workflows either deliver view-specific data to the SUM or export view-specific data from the SUM. Figure 12 shows a simplified example of a transformer configuration, in the modelling language YAML as an example (cf. mechanical-import-config.yml[26]. The *AML Transformer* (cf. Table 1) represents transformer implementations for AML and for Text-to-Model and Model-to-Text transformations. For *Text-to-Model* imports, the data artefact usually is an export from a discipline-specific tool. The data curator defines object mappings (line 3) between the particular data artefacts and the local view model. Line 4 indicates the expression for which element type in an XML documents the mapping is applicable (cf. mechanical-import-data.xml[27]). Specifically, these mappings describe which artefact elements map to which AML concept, in the example a particular *systemUnitClassPath* (line 5). Conditions can further detail the source element (line 8), specifying that the XML element attribute *@DescrEN*= needs to have the value *ElectricScrewdriver*.

To further prepare the workflow, the data curator has to set up a *pipeline configuration* file. Figure 13 shows an excerpt of the pipeline configuration using the domain-specific definition language of the Jenkins server with several steps. Additionally, the workflow can be defined as shell scripts.

An workflow setup with the input and output artefacts is available in the online repository[28]. First, each view-specific transformer is defined, which takes as input the respective view template, transformer configuration, input data, and the file path to the output file. The outputs are the particular view templates populated with the data values from the input data that come from the engineering artefacts. Then, the view model is converted, and inputs to this step are the local view model and the SUM.

Then, the local view model is translated into the SUM structure using the *CAEX Converter* (cf. Table 1). The *CAEX Comparer* (cf. Table 1) compares this SUM-structured view model with the contents of the SUM resulting in a diff-model that contains the computed changes of the comparison (cf. mechanical-view-compare-result.json[29]). Figure 14 show an excerpt of the diff-model that containing the detected changed value of the attribute *torque* . Also, a new link element is detected that describes the hierarchy dependency between the *ElectricScrewdriver* and the *Bit*. Engineers can investigate the changes in the diff-model and accept or reject them (cf. line 4). For complex changes that affect several disciplines, an additional multi-view change management approach can be applied [52]. Based on their input, the *CAEX Merger* processes the diff-model and applies the changes to the SUM instance.

The *CAEX RuleEngine* evaluates the semantic links to propagate the changes across the local view models defined in the project configuration. The current implementation of the TMvMT approach focuses on tracing new model elements and attribute value modifications, as these are the most frequent sources of change.

Step 3 executes the multi-view model transformation workflow. The data curator uploads the project configuration, defined in Step 1, to the Jenkins server. Data updates in a shared data repository trigger the execution of the associated view pipelines. The data curator can inspect partial results, such as the input data, configuration data, converted view models, and the diff model, to validate the correct execution of the pipeline.

Figure 15 shows the defined stages of the pipeline, consisting of a *tool installation* and *general setup* of the pipeline environment. After this task, the project's *model transformers* are initially generated, and three *view transformations* for the different disciplines are executed.

First, the required modelling operation services (provided as jar-files) are defined in the *tools* section. In consecutive stages, the model transformations for the different views are executed with their specific configurations. The data curator can easily modify the pipeline steps in the Continuous Integration server and review the implications. Jenkins executes the model transformations providing feedback on every step's success (or failure) and writes the resulting models to the respective locations. The feedback can further be visualised in an issue tracker or reporting system.

Furthermore, another advantage compared to the manual model transformation process is the multi-view modelling environment. In the manual process, the addition of views can lead to breaking workflows or errors due to the *point-to-point* update flows. According to the TMvMT process, new views are incorporated to the multi-view engineering graph (cf. Sect. 5.1, Step 1). The next step guarantees that the semantic links and dependencies are generated. Based on the feasibility study, we evaluate the capabilities of the TMvMT approach in the following section.

Implementation and modelling technologies. Table 1 shows the mappings between modelling concepts and implementation technologies in the feasibility study. Command line versions of the implementations used in the feasibility

[26] mechanical-import-config.yml: https://raw.githubusercontent.com/tuw-qse/tmvmt-resources/main/uc-screw_dashboard/transform/input/mechanical-import-config.yml.

[27] mechanical-import-data.xml: https://raw.githubusercontent.com/tuw-qse/tmvmt-resources/main/uc-screw_dashboard/transform/input/mechanical-import-data.xml.

[28] https://github.com/tuw-qse/tmvmt-resources/tree/main/uc-screw_dashboard.

[29] mechanical-view_compare-result.json: https://raw.githubusercontent.com/tuw-qse/tmvmt-resources/main/uc-screw_dashboard/integrate/compare/output/mechanical-view_compare-result.json.

Fig. 10 RefSemantic in the attributes of the resource electric screwdriver

```
Resource "ElectricScrewdriver": {
    name: "ElectricScrewdriver",
    children: ["Bit", "Drive3", "Robot"],
    mechanical.torque: {
        relations: [{   type: "source",
                        semantic: "MechExport",
                        reference: "@torque"
                    },
                    {   type: "propagate",
                        semantic: "SUMProject",
                        reference: "${FastenScrewMeasure}.mechanical@torque"
                    }]
    }
    electrical.power_consumption: {
        relations: [{   type: "source",
                        semantic: "ElectricExport",
                        reference: "@power-consumption"
                    }]
    }
}
```

Fig. 11 Project definition for a position-and-screw robot cell

```
projectDefinition:
  projectName: "Position-and-Screw"
  projectID: "1cd7ed82-9219-4489-8477-3ca91fff57b9"

viewDefinitions:
  - view: "PPR"
  - view: "Mechanical"
  - view: "Electrical"

conceptMappings:
  ...
  - concept: "ElectricScrewdriver"
    views:
      - view: "PPR"
          ...
      - view: "Mechanical"
        attributes:
          - attribute: "torque"
            dataType: "Number"
            value: "0.0"
            unit: "Nm"
            relations:
              - relation: "source"
                fileSemantic: "MechExport"
                reference: "@torque"
              - relation: "propagate"
                mimeType: "aml"
                fileSemantic: "SUMProject"
                reference: "${FastenScrewMeasure}.mechanical@torque"

      - view: "Electrical"
        ...
```

study can be found in the online repository[30]. The implementation extends the MvMF [7], using the industrial engineering data exchange standard AutomationML (AML) [53] to define the SUM. Furthermore, we use and extend the PPR-DSL [18], to model the concrete data model of the multi-view engineering graph. The *Model-to-Model* transformation is conducted by the *CAEX Converter*, which converts view models into the SUM structure to enable model comparison. The conversion can also transform the SUM

[30] TMvMT Implementation: https://github.com/tuw-qse/tmvmt-resources/tree/main/bin.

Fig. 12 Transformer configuration for the electric screwdriver

```
uriScheme: xml

objectMapping:
    - expression: "Assembly"
      systemUnitClassPath:
  ↪  "AML2MechanicalViewSystemUnitClassLib/ElectricScrewdriver"
      listType: "MechExport"
      condition:
        - "./@DescrEN='ElectricScrewdriver'"
```

structure back into a specific view model. In this case, the view-specific data are extracted from the SUM.

The *CAEX Comparer* implementation for model comparison is based on the internal hierarchical structure of AML files, i.e., the Computer-Aided Engineering eXchange (CAEX) structure. Our *CAEX Comparer*, similar to *EMF Compare*, computes the comparison and diff analysis of the model based on element attribute content rather than on a textual representation. The service compares the converted view model in the SUM structure to the currently instantiated SUM and generates a list of model differences. This delta list can be reviewed to either accept or reject single changes. The *CAEX Merger* merges the reviewed list of changes into the SUM to generate an updated version. After the merge, *CAEX Rule Engine* propagates changes based on defined rules. This can happen if model elements or attributes have references, which indicate semantic similarity, to view updates. Subsequently, unit tests on different stages conduct model validation. Automating the improved model transformation workflow requires a flexible method to link the transformations to a pipeline sequence. For this purpose, we chose for our prototypical implementation *Jenkins* as an automation/CI server to combine the engineering with DevOps and execute the model transformation workflow.

Evaluation of Traceability Capabilities

This section evaluates the fulfilment of the traceability requirements (cf. Sect. 4), expected extra effort, and complexity for establishing traceability in PSE.

Evaluation of traceability requirements. We evaluate the TMvMT approach regarding requirements for traceability and effort with traditional alternative approaches in PSE (cf. model architecture types in Sect. 2.2): (i) Manual Model Transformation (MMT) between engineering artefacts, without a *Single Underlying Model (SUM)* [12] (cf. Sect. 4); (ii) a *Tool Suite with a limited SUM (TS-MT)* [11]; and (iii) Multi-view Model Transformation with a SUM, but without specific traceability concerns [7]. In the evaluation, we illustrate these alternative approaches with application examples from PSE practice, e.g., concrete implementations of the approaches in commercial or custom tools at a PSE organization.

A main advantage of a model transformation pipeline as described in [7] compared to the other approaches is the separation of concerns. For example, understanding local model elements or mapping and transforming local to common concepts lead to a higher number of simpler transformers when compared to the TS-MT approach. Therefore, these transformers are easier to reuse and require only limited knowledge for their adaptation, allowing the data curator to share the work load with local domain experts, e.g., an electrical domain expert. However, change propagation with MvMT does not necessarily imply the traceability of changes.

Traceable Multi-view Model Transformation (TMvMT) in PSE (cf. Sect. 5) extends the MvMT approach with a traceability concern: Changes to model elements, in particular to property values, are traced back to the sources of change in a discipline. This is needed to act as a foundation for auditable and verifiable configuration management in parallel and iterative PSE as required to realise the Industry 4.0 vision.

Table 2 summarises the fulfilment of the requirements *Rx*, introduced in Sect. 4, on a 5-point *Likert* scale (++, +, o, -, −−), where ++/−− indicate very high/low capabilities for the TMvMT approach and alternative *transformation workflow* approaches (cf. Sect. 2.2).

R1. Multi-view modelling capabilities. MMT is rated very low due to point-to-point transformation without an integrated model. TS-MT is rated high as transformers map stakeholder views in engineering artefacts to an integrated model. MvMT is rated very high as the approach explicitly represents stakeholder views both in the engineering artefacts and in the integrated model. TMvMT is rated very high as the approach explicitly represents and preserves stakeholder views both in the engineering artefacts and in the integrated model.

R2. Distributed process synchronization. MMT is rated very low due to very limited synchronization capabilities via document-based exchange, triggered manually by domain experts, insufficient representation of configuration dependencies as mainly tacit domain expert knowledge,

Fig. 13 Jenkins pipeline configuration for pipeline stages

```
pipeline {
  tools {
    jdk 'openjdk11'
  }
  stage('Project Generation') {
    steps {
      configFileProvider([configFile(fileId: '115b',
        targetLocation: '${GEN_IN}/aml-gen-config.yml')]) {}
      sh 'java -jar aml-class-gen.jar -c ${GEN_IN}/aml-gen-config.yml
        -t ${GEN_IN}/usedLibs.aml -o ${GEN_OUT}'
    }
  }
  stage('Next stage') { steps { ... } }
}
```

and no systematic process support for parallel and iterative development. TS-MT is rated average as the tool suite can act on changes for a limited set of engineering disciplines. However, it does not consider dependencies between data elements, as discipline-specific views are not represented in the integrated model. These issues puts the burden of identifying relevant changes on the user or on hard-coded scripts. Dependencies are hard-coded in importer scripts making the propagation of changes inflexible and hard to adapt for a domain expert. MvMT is rated high as the representation of change views per discipline and change dependencies provides the foundation for automated change propagation and flexible analysis and adaptation. TMvMT is rated very high as it goes beyond the MvMT approach by representing dependencies and states for each asset element as a foundation for synchronising agile PSE processes.

R3. Traceable model change representation. MMT is rated very low due to the missing common model, possibly incompatible description languages of the engineering artefacts, and the missing representation of change dependencies. These features make the capability to trace back a change to its source depend on the knowledge of the involved domain experts rather than on a systematic approach. TS-MT is rated low as the description languages are compatible only for a limited scope of stakeholder views that are very hard to extend. Further, the collection of changes to model elements that may come from several disciplines is very difficult to trace back to the respective source. It requires to analyse log files, which are not visible to the normal user, taking high effort for the data curator. MvMT is rated average as the integrated model can represent the full scope of stakeholder views and change dependencies. However, it does not consider version numbers and considers only stakeholder roles but no individuals. The latter may submit conflicting changes that are hard and error-prone to trace back to individual stakeholders within a discipline. TMvMT is rated high as it goes beyond the MvMT approach by considering change states for asset elements and both stakeholder roles and individuals.

R4. Version representation/management. MMT is rated very low due to typically an event/timestamp-based sequence of changes that provide only fragile version management capabilities. TS-MT is rated average as version management is possible for model elements, but only in limited scope of stakeholder views that is very hard to extend. MvMT is rated average as the approach does not consider version numbers of asset elements. TMvMT is rated high as it goes beyond the MvMT approach by supporting version numbers for internal elements and concepts.

Effort for model transformation. PSE domain experts will only consider changing their approach to model transformation, if the effort not too high. The baseline with which new approaches are compared is the traditional approach, in this case manual model transformation. Therefore, we compare the expected effort for the model transformation alternatives. MMT is rated high due to the very low effort for setup, taking into account the high incremental effort for operation, often leading to infrequent model updates and technical debt [6]. TS-MT is rated average due to the high effort required for the first setup of the common data model and system architecture. Further, the extension of the tool suite with new stakeholder views is very costly, requiring the involvement and approval of tool suite consultants. MvMT is rated high as the first setup of the common data model and system architecture requires high effort. During operation effort is reduced, and the reuse of artefacts, methods, and configurations facilitates the efficient inclusion of new stakeholder views and engineering tools. TMvMT is rated high as it has similar effort characteristics as the MvMT approach. Considering traceability in the TMvMT is a one-time cost, as it is incurred while designing the multi-view engineering graph.

Traceability Factors. In the feasibility study context, we identified as factors that are likely to influence the quality of model integration and traceability effectiveness and effort in agile PSE: (i) the scope of the model, e.g., one work cell or several work cells in a potentially large work line; (ii) the

Fig. 14 Example comparison result of the resource electric screwdriver in the mechanical view to the SUM

```json
{
    "type" : "AttributeModify",
    "changeKind" : "CHANGE",
    "accepted" : false,
    "leftParent" : {
      "type" : "Attribute",
      "id" : null,
      "name" : "torque",
      "parentElement" : {
        "type" : "Element",
        "id" : "e1d6dd5a-6024-4977-9591-bbd8bad87a33",
        "name" : "MechanicalView"
      }
    },
    "rightParent" : {
      "type" : "Attribute",
      "id" : null,
      "name" : "torque",
      "parentElement" : {
        "type" : "Element",
        "id" : "e1d6dd5a-6024-4977-9591-bbd8bad87a33",
        "name" : "MechanicalView"
      }
    },
    "property" : {
      "type" : "Property",
      "name" : "value"
    },
    "oldValue" : "0.0",
    "newValue" : "5"
  }, {
    "type" : "ElementChange",
    "changeKind" : "ADD",
    "accepted" : false,
    "leftParent" : {
      "type" : "Element",
      "id" : "e1d6dd5a-6024-4977-9591-bbd8bad87a33",
      "name" : "MechanicalView"
    },
    "rightParent" : {
      "type" : "Element",
      "id" : "e1d6dd5a-6024-4977-9591-bbd8bad87a33",
      "name" : "MechanicalView"
    },
    "value" : {
      "type" : "Link",
      "id" : "10a562f5-4a1d-449a-b50e-3caa2c442b4b",
      "name" : "ElectricScrewdriver.MechanicalView:MechanicalView_toChild-
      Bit.MechanicalView:MechanicalView_toMother",
      "refPartnerSideA" : "e1d6dd5a-6024-4977-9591-bbd8bad87a33:MechanicalView_toChild",
      "refPartnerSideB" : "4538eebf-132b-4552-ad54-d92e9c775cd4:MechanicalView_toMother"
    }
  }
```

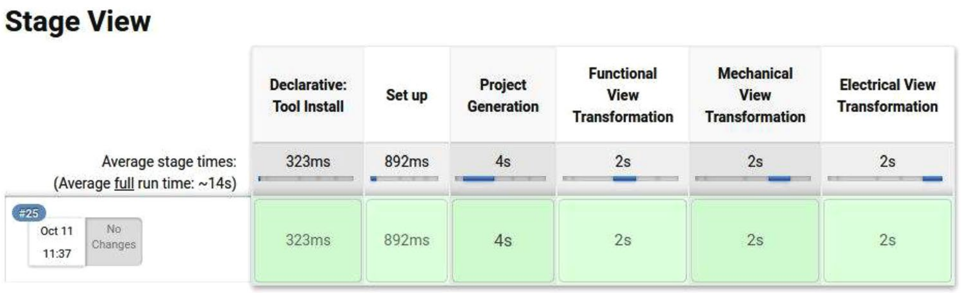


Fig. 15 Jenkins Build Pipeline with six steps, based on [7]

Stage View

	Declarative: Tool Install	Set up	Project Generation	Functional View Transformation	Mechanical View Transformation	Electrical View Transformation
Average stage times: (Average full run time: ~14s)	323ms	892ms	4s	2s	2s	2s
#25 Oct 11 11:37 No Changes	323ms	892ms	4s	2s	2s	2s

number and complexity of stakeholder views that have to be integrated, e.g., disciplines working in parallel, engineering artefact types, and tool export formats; and (iii) input data quality, reflecting technical debt [6] input data that may accumulate in the course of a project and across projects due to reuse of data models and instances. While these factors

Table 1 TMvMT process step–modelling concept–technology mapping

TMvMT Process	Modelling Concept	Technology
Step 1	Modelling Framework	MvMF, PPR-DSL
Step 2	Meta-Model Design	ProjConfigGen, AML ProjGen
Step 3	Text2M2Text Transformation	AML Transformer
Step 3	M2M Transformation	CAEX Converter
Step 3	Model Comparison	CAEX Comparer
Step 3	Model Merge	CAEX Merger
Step 3	Model Injection	CAEX RuleEngine

had a limited impact in the feasibility study, they should be considered when applying the TMvMT approach in larger settings.

Discussion

This section discusses the research results and limitations regarding the research questions introduced in Sect. 3. In our previous work [7], we presented the MvMT workflow as an improved model transformation method compared to manual model integration. The presented approach is based on the Git workflow and multi-view modelling. The aim is to facilitate the traceable mapping and incorporation of different view models to a SUM. The continuation of this work was now applied to the *Position and Screw* use case and we extended the PPR-DSL implementation to define and represent the three different relation types.

RQ1. Process Design. *What process enables traceable multi-view model transformation workflows in agile PSE?* To address RQ1 and the requirements for traceability (cf. Sect. 4), we proposed in Sect. 5.1 the TMvMT process. This process aims to extend the MvMT method to enable the design and operation of a customisable and traceable multi-view model transformation workflow.

The TMvMT process consists of three steps: (1) Define a multi-view engineering graph to achieve a holistic and common view on engineering concepts. This is achieved through negotiating common concepts, defining semantic references to describe the origin of attribute values and mappings within the network. (2) Configure multi-view modelling workflows to setup a flexible environment. Traceability is enabled through the automatic generation of the SUM and local view models derived from the engineering graph. (3) Execute data integration pipeline. In this step, the distributed model operations—compare, diff, and merge—ease the model integration process and facilitate reviewing and tracing partial results. For example, input and output models can be viewed to validate mapping results. Coupled with the model transformation pipeline, these model transformation services can be flexibly orchestrated and automatically executed.

Based on the feasibility study regarding the traceability requirements, the comparison of the TMvMT results showed clear improvements over the traditional alternative model transformation approaches (cf. Table 2), in particular regarding traceable model change representation, version representation, and distributed process synchronization. The study results indicate that the TMvMT approach provides a sound foundation for PSE domain experts to define multi-view models in a traceable way. This foundation gives way for an evaluation of the approach in a broader context regarding its usability and scalability various PSE scenarios of different size and complexity.

RQ2. Architecture Design. *What software architecture enables a traceable model transformation workflow in agile PSE?* A major goal of Development and IT Operations (DevOps) and MDE architectures is to increase productivity through automation and orchestration of processes. For Model-Driven Engineering (MDE), this includes the automated generation of models and code, while DevOps focuses on automated integration and testing. Although the Eclipse Modelling Framework (EMF) provides rich functionality for MDE, it would have introduced too much complexity to our context. For this reason, we decided to reuse our custom-built Multi-view Modelling Framework (MvMF) [7], incorporating the main MDE principles, while keeping overall configuration effort lower.

To address RQ2, we extended the MvMF [7] by designing a TMvMT software architecture to automate the TMvMT process and to facilitate tracing changes to attribute values. The TMvMT process requires a modelling capability to design the engineering graph. Domain-Specific Languages (DSLs) are established means to provide such capability for domain-specific contexts such as PSE. A well-established modelling concept in the PSE domain is PPR; thus, we decided to reuse PPR-DSL [18]. In the feasibility study, we showed how to automate the TMvMT workflow, using defined build pipelines in Jenkins. The project definition can be reused for other projects or adapted to changing needs. PSE engineers also benefit from the pipeline configuration, which allows an adaptation to different contexts, i.e., use cases in the multi-disciplinary PSE domain.

Table 2 Traceability requirements fulfilment with TMvMT and alternative approaches, using a 5-point Likert scale (++, +, ○, −, −), where ++ indicates very high capability, and − very low capability

Req. Rx/Model Transform. Approach	MMT	TS-MT	MvMT	TMvMT
R1. Multi-view modelling capabilities	−	+	++	++
R2. Distributed process synchronization	−	○	+	++
R3. Traceable model change representation	−	−	○	+
R4. Version representation/management	−	○	○	+
Effort for model transformation	+	○	+	+

Our research results go beyond the state of the art in model-based software engineering by showcasing an industrial use case from the PSE domain and the feasibility of a domain-specific model-based DevOps approach.

On the other hand, our research results exceed the state of the art in model integration for PSE: (i) by addressing the integration of multi-view stakeholder models in PSE; (ii) by focusing on model-based analyses of changes, rather than text-based analyses that do not work well in a PSE context with heterogeneous engineering artefacts; (iii) by providing the modular, configurable TMvMT approach, building on the EMF concept with different technologies that are suitable for an application in the PSE context; (iv) by providing the PSE research community with a modular Continuous Integration (CI) approach that works with a variety of artefact types; and (v) by demonstrating the feasibility of conducting the TMvMT process to integrate a multi-view model for a typical PSE scope of a robot work cell in automotive production.

Limitations. The feasibility study focused on a use case derived from projects at large PSE companies in automotive industry. This may introduce bias due to the specific selection of stakeholder views and alternative approaches considered, as well as the roles or individual preferences of the domain experts. To overcome these limitations, we plan case studies in a wider variety of application contexts.

The current implementation aims at models described in the industry standard AML [53]. However, we are aware that this is also limiting the applicability of our approach and plan to propose extensions. Implementing TMvMT pipelines using the proposed system architecture requires additional setup time and effort *a priori*. However, the integration effort is then managed through the pipeline and will, once set up, save time and complexity. Furthermore, validation of the effectiveness and usability of the TMvMT approach will require empirical studies with domain experts and their typical PSE artefacts.

Conclusion and Future Work

Lost changes, diverging local views, and repetitive manual integration tasks can lead to late design changes and, thus, costly errors and mitigation efforts. These issues potentially influence the process quality in PSE negatively. Identifying and resolving change conflicts in parallel engineering are essential to the success of agile PSE. To reduce the risk of late design changes, this paper aimed at improving capabilities for traceable multi-view model transformation for the configuration management of multi-disciplinary assets and dependencies according to VDI 3695-2 [4]. The synchronization of changes in distributed engineering on multi-view models depends on capabilities to trace changes to attribute values in PSE assets back to their sources.

This paper investigated the *Position-and-Screw Robot Cell* use case from automotive manufacturing [16] to identify traceability issues and requirements for multi-view modelling. To support multi-view changes in PSE, the TMvMT process and architecture provides required semantic model analysis capabilities. To this end, the architecture extends the architecture of the Multi-view Model Transformation (MvMT) [7]. The goal is to define traceable and flexible multi-view model transformation pipelines for building intermediate models and an integrated PSE model. A main advantage of the approach is its potential to define discipline-specific model transformations and integrate multiple view models to an SUM while updating corresponding views. Another advantage is the modular, configurable TMvMT approach, building on the EMF concept with different technologies that are suitable for the PSE context.

In a feasibility study, we evaluated the TMvMT approach in the scope of a robot work cell from automotive manufacturing. We implemented the TMvMT approach building on the AML standard [53] and automated it with a Continuous Integration (CI) system. Furthermore, we compared the traceability capabilities of the approach to three alternative model transformation approaches in PSE. The study results indicate that the TMvMT approach provides a sound foundation for PSE domain experts to define multi-view models in a traceable way.

Future Work. We plan to investigate different methods for supporting the construction and validation of the multi-view engineering graph. Semantic web approaches and method will be a starting point for this direction. Furthermore, we will experiment with different graphical representation forms and approaches such as low code. Additionally, we will explore possibilities to integrate the design model and data with the operational phase to cover the whole PSE lifecycle. Further, we plan to enable auditability of data traces for error detection. We will also investigate how the TMvMT approach will scale up to larger model sizes, more engineering disciplines, and larger sets of changes to model elements.

Author contributions Following the contributor roles taxonomy, we disclose the following contributions by the authors. FR—conceptualization, data curation, investigation, methodology, resources, software, validation, visualization, writing—original draft, and writing—review & editing; LW & KM—conceptualization, data curation, investigation, methodology, validation, visualization, writing—original draft, and writing—review & editing; DW—writing—review & editing, and funding acquisition; AL—investigation, validation, and funding acquisition; SB—conceptualization, data curation, investigation, methodology, supervision, validation, writing—original draft, writing—review & editing, and funding acquisition.

Funding Open access funding provided by TU Wien (TUW). This study was funded by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs, and the National Foundation for Research, Technology and Development. This study was funded by the Austrian Research Promotion Agency (FFG) via "Austrian Competence Center for Digital Production" (CDP) under Contract No. 881843.

Availability of data and materials https://github.com/tuw-qse/tmvmt-resources

Code availability https://github.com/tuw-qse/tmvmt-resources

Declarations

Conflict of interest Felix Rinker declares that he has no conflict of interest. Laura Waltersdorfer declares that she has no conflict of interest. Kristof Meixner declares that he has no conflict of interest. Dietmar Winkler declares that he has no conflict of interest. Arndt Lüder declares that he has no conflict of interest. Stefan Biffl declares that he has no conflict of interest.

Ethics approval This article does not contain any studies with human participants or animals performed by any of the authors.

Consent to participate This article does not contain any studies with human participants performed by any of the authors.

Consent for publication This article does not contain any third-party material that requires consent for publication.

References

1. Monostori L, Kádár B, Bauernhansl T, Kondoh S, Kumara S, Reinhart G, Sauer O, Schuh G, Sihn W, Ueda K. Cyber-physical systems in manufacturing. CIRP Ann. 2016;65(2):621–41. https://doi.org/10.1016/j.cirp.2016.06.005.
2. Vogel-Heuser B, Bauernhansl T, ten Hompel M (eds). Handbuch Industrie 4.0 Bd. 4. Allgemeine Grundlagen 2020;4:2. https://doi.org/10.1007/978-3-662-53254-6
3. Vogel-Heuser B, Böhm M, Brodeck F, Kugler K, Maasen S, Pantförder D, Zou M, Buchholz J, Bauer H, Brandl F. Interdisciplinary engineering of cyber-physical production systems: highlighting the benefits of a combined interdisciplinary modelling approach on the basis of an industrial case. Des Sci. 2020;6:5. https://doi.org/10.1017/dsj.2020.2.
4. VDI Guideline 3695: Engineering of Industrial Plants—Evaluation and Optimization. Beuth Verlag.
5. Buchmann T, Dotor A, Westfechtel B. Mod2-scm: A model-driven product line for software configuration management systems. Information and Software Technology 2013;55(3), 630–650. Special Issue on Software Reuse and Product Lines
6. Waltersdorfer L, Rinker F, Kathrein L, Biffl S. Experiences with technical debt and management strategies in production systems engineering. In: Proceedings of the 3rd International Conference on Technical Debt, 2020;pp. 41–50. ACM, New York, USA. https://doi.org/10.1145/3387906.3388627
7. Rinker F, Waltersdorfer L, Meixner K, Winkler D, Lüder A, Biffl S. Continuous Integration in Multi-view Modeling: A Model Transformation Pipeline Architecture for Production Systems Engineering. In: 9th Int. Conf. on Model-Driven Engineering and Software Development, 2021;pp. 286–293. https://doi.org/10.5220/0010309902860293
8. Oevermann J. Semantic PDF Segmentation for Legacy Documents in Technical Documentation. In: Proceedings of the 14th International Conference on Semantic Systems, SEMANTICS 2018, Vienna, Austria, September 10-13, 2018. Procedia Computer Science, 2018;vol. 137, pp. 55–65. Elsevier, Amsterdam, The Netherlands. https://doi.org/10.1016/j.procs.2018.09.006
9. Biffl S, Lüder A, Rinker F, Waltersdorfer L, Winkler D. Engineering data logistics for agile automation systems engineering. In: Security and Quality in Cyber-Physical Systems Engineering, 2019;pp. 187–225. Springer, Cham, Switzerland
10. Rinker F. Flexible Multi-aspect Model Integration for Cyber-Physical Production Systems Engineering. In: Krogstie, J., Ouyang, C., Ralyté, J. (eds.) Proceedings of the Doctoral Consortium Papers Presented at the 33rd International Conference on Advanced Information Systems Engineering (CAiSE 2021), Melbourne, Australia, June 28 - July 2, 2021. CEUR Workshop Proceedings, 2021;vol. 2906, pp. 31–40. CEUR-WS.org, Aachen, Germany. http://ceur-ws.org/Vol-2906/paper4.pdf
11. Florian Himmler MA. Data integration framework for heterogeneous system landscapes within the digital factory domain. Proc Eng. 2014;69:1138–43.
12. Strahilov A, Hämmerle H. Engineering workflow and software tool chains of automated production systems. In: Multi-Disciplinary Engineering for Cyber-Physical Production Systems, 2017;pp. 207–234. Springer, Cham, Switzerland
13. Brambilla M, Cabot J, Wimmer M. Model-driven software engineering in practice. Synth Lect Software Eng. 2017;3(1):1–207.

14. Ebert C, Gallardo G, Hernantes J, Serrano N. Devops. IEEE Software. 2016;33(3):94–100.
15. Feichtinger K, Meixner K, Rinker F, Koren I, Heinemann T, Holtmann J, Konersmann M, Michael J, Neumann E-M, Pfeiffer J, Rabiser R, Riebisch M, Schmid K. Industry voices on software engineering challenges in cyber-physical production systems engineering. In: 27th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2022:1–8.
16. Meixner K, Lüder A, Herzog J, Winkler D, Biffl S. Patterns for reuse in production systems engineering. Int J Software Eng Knowl Eng. 2021;2:2.
17. Biffl S, Musil J, Musil A, Meixner K, Lüder A, Rinker F, Weyns D, Winkler D. An Industry 4.0 Asset-based coordination artifact for production systems engineering. In: 23rd IEEE International Conference on Business Informatics. IEEE (in press), New York, USA 2021
18. Meixner K, Rinker F, Marcher H, Decker J, Biffl S. A domain-specific language for product-process-resource modeling. In: 26th IEEE International Conference on Emerging Technologies and Factory Automation. ETFA 2021, Västerås, Sweden, September 07–10, 2021. New York, USA: IEEE; 2021. p. 1–8.
19. Tolk A, Muguira JA. The levels of conceptual interoperability model. In: Proceedings of the 2003 Fall Simulation Interoperability Workshop, 2003;vol. 7, pp. 1–11. Citeseer.
20. Berre A-J, Elvesæter B, Figay N, Guglielmina C, Johnsen SG, Karlsen D, Knothe T, Lippe S. The ATHENA interoperability framework. In: Enterprise Interoperability II, pp. 569–580. Springer, Cham, Switzerland 2007.
21. Jouault F, Allilaire F, Bézivin J, Kurtev I. ATL: A model transformation tool. Sci Comput Program. 2008;72(1–2):31–9.
22. Toulmé A. Presentation of EMF Compare Utility. In: Eclipse Modeling Symposium, 2006;pp. 1–8.
23. Atkinson C, Burger E, Meier J, Reussner R, Winter A. Preface to the 1st Workshop on View-Oriented Software Engineering (VoSE). In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), 2019;pp. 370–370. IEEE, New York, USA. https://doi.org/10.1109/models-c.2019.00057. IEEE.
24. Feldmann S, Kernschmidt K, Wimmer M, Vogel-Heuser B. Managing Inter-Model Inconsistencies in Model-based Systems Engineering. Software Engineering 2020;2020. https://doi.org/10.18420/SE2020_69.
25. Atkinson C, Gerbig R, Kühne T. Comparing multi-level modeling approaches. In: MULTI@ MoDELS, 2014;pp. 53–61.
26. Tunjic C, Atkinson C. Synchronization of projective views on a single-underlying-model. In: Proceedings of the 2015 Joint MORSE/VAO Workshop on Model-Driven Robot Software Engineering and View-based Software-Engineering, 2015;pp. 55–58.
27. Pätzold K. In: Biffl, S., Lüder, A., Gerhard, D. (eds.) Product and Systems Engineering/CA* Tool Chains, 2017;pp. 27–62. Springer, Cham.
28. Bihani P, Drath R, Kadam A. Towards meaningful interoperability for heterogeneous engineering tools via automationml. In: 24th IEEE ETFA, 2019;pp. 1286–1290. https://doi.org/10.1109/ETFA.2019.8869532.
29. Lüder A, Baumann L, Behnert AK, Rinker F, Biffl S. Paving pathways for digitalization in engineering: common concepts in engineering chains. In: Proc. IEE Int. Conf. ETFA, 2020;vol. 1, pp. 1401–1404. IEEE, New York, USA. https://doi.org/10.1109/etfa46521.2020.9212009.
30. Wortmann A, Barais O, Combemale B, Wimmer M. Modeling languages in Industry 4.0: an extended systematic mapping study. Softw Syst Model. 2020;19(1):67–94. https://doi.org/10.1007/s10270-019-00757-6.
31. Rinker F, Waltersdorfer L, Meixner K, Biffl S. Towards support of global views on common concepts employing local views. In: 24th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2019, Zaragoza, Spain, September 2019;10-13, 2019, pp. 1686–1689. IEEE, New York, USA. https://doi.org/10.1109/etfa.2019.8869239.
32. Berardinelli L, Mazak A, Alt O, Wimmer M, Kappel G. Model-driven systems engineering: principles and application in the CPPS domain. In: Multi-disciplinary engineering for cyber-physical production systems. Cham: Springer; 2017. p. 261–99. https://doi.org/10.1007/978-3-319-56345-9_11.
33. Garcia J, Cabot J. Stepwise adoption of continuous delivery in model-driven engineering. In: International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment, 2018;pp. 19–32. Springer.
34. Ponsard C, Darquennes D, Ramon V, Deprez J-C. Assessment of EMF Model to Text Generation Strategies and Libraries in an Industrial Context. In: MODELSWARD, 2020;433–440 .
35. Hildebrandt C, Scholz A, Fay A, Schröder T, Hadlich T, Diedrich C, Dubovy M, Eck C, Wiegand R. Semantic modeling for collaboration and cooperation of systems in the production domain. In: 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2017;1–8. IEEE.
36. Wagner C, Grothoff J, Epple U, Drath R, Malakuti S, Grüner S, Hoffmeister M, Zimermann P. The role of the industry 4.0 asset administration shell and the digital twin during the life cycle of a plant. In: 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2017;1–8. IEEE.
37. Plattform Industrie 4.0, ZVEI: Part 1—The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01 Review). Standard, German BMWI (Nov. 2020). https://bit.ly/37A002I
38. Schleipen M, Lüder A, Sauer O, Flatt H, Jasperneite J. Requirements and concept for Plug-and-Work at-Automatisierungstechnik. 2015;63(10):801–20. https://doi.org/10.1515/auto-2015-0015.
39. VDI/VDE 3682: Formalised process descriptions. Beuth Verlag (2005)
40. Date CJ, Darwen H. A Guide to the SQL Standard, vol. 3. Addison-Wesley New York 1987.
41. Chopra AK, Dalpiaz F, Aydemir FB, Giorgini P, Mylopoulos J, Singh MP. Protos: Foundations for engineering innovative socio-technical systems. In: 2014 IEEE 22nd International Requirements Engineering Conference (RE), 2014;53–62. IEEE.
42. Wohlrab R, Knauss E, Steghöfer J-P, Maro S, Anjorin A, Pelliccione P. Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture. Requirements Eng. 2020;25(1):21–45. https://doi.org/10.1007/s00766-018-0306-1.
43. Wohlrab R. Living boundary objects to support agile inter-team coordination at scale. PhD thesis, Chalmers University of Technology 2020.
44. Wieringa RJ. Design science methodology for information systems and software engineering. Cham: Springer; 2014. p. 3–11. https://doi.org/10.1007/978-3-662-43839-8_1.
45. Presley A, Liles DH. The use of IDEF0 for the design and specification of methodologies. In: Proceedings of the 4th Industrial Engineering Research Conference 1995.
46. Lüder A, Pauly J, Rinker F, Biffl S. Data exchange logistics in engineering networks exploiting automated data integration. In: 24th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2019, Zaragoza, Spain, September 10-13, 2019:657–664. IEEE, New York, USA. https://doi.org/10.1109/ETFA.2019.8869352.
47. Bruneliere H, Perez JG, Wimmer M, Cabot J. Emf views: A view mechanism for integrating heterogeneous models. In: International Conference on Conceptual Modeling, 2015;317–325. Springer

48. Batory DS, Altoyan N. Aocl : A pure-java constraint and transformation language for MDE. In: Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD,, 2020;319–327. SCITEPRESS, Setúbal, Portugal. https://doi.org/10.5220/0008942803190327.
49. Sanchis R, García-Perales Ó, Fraile F, Poler R. Low-code as enabler of digital transformation in manufacturing industry. Appl Sci. 2020;10(1):12.
50. Tisi M, Mottu J, Kolovos DS, de Lara J, Guerra E, Ruscio DD, Pierantonio A, Wimmer M. Lowcomote: Training the next generation of experts in scalable low-code engineering platforms. CEUR Workshop Proceedings, vol. 2405, pp. 73–78. CEUR-WS.org, Aachen, Germany 2019.
51. Rinker F, Waltersdorfer L, Biffl S. Towards test-driven model development in production systems engineering. In: Proceedings of the 22nd International Conference on Enterprise Information Systems, ICEIS 2020, Prague, Czech Republic, May 5-7, 2020, Volume 1, 2020;213–219. SCITEPRESS, Setúbal, Portugal. https://doi.org/10.5220/0009425302130219.
52. Rinker F, Kropatschek S, Steuer T, Meixner, K, Kiesling E, Lüder A, Winkler D, Biffl S. Efficient multi-view change management in agile production systems engineering. In: Proceedings of the 24th International Conference on Enterprise Information Systems, ICEIS 2022, Online Streaming, April 25–27, 2022, Volume 2, 2022;134–141. SCITEPRESS, Setúbal, Portugal. https://doi.org/10.5220/0011074000003179.
53. IEC 62714:2014 Engineering data exchange format for use in industrial automation systems engineering—automation markup language. IEC. http://www.iec.ch