









Efficient Multi-view Change Management in Agile Production Systems Engineering

Felix Rinker^{1,2}^a, Sebastian Kropatschek³^b, Thorsten Steuer³^c, Kristof Meixner^{1,2}^d,
Elmar Kiesling⁴^e, Arndt Lüder^{3,5}^f, Dietmar Winkler^{1,2}^g and Stefan Biffel^{1,3}^h

¹Institute of Information Systems Engineering, TU Wien, Vienna, Austria

²CDL for Security & Quality Improvement in the Production System Lifecycle, TU Wien, Vienna, Austria

³Center for Digital Production, Vienna, Austria

⁴Institute of Data, Process, and Knowledge Engineering, WU Wien, Vienna, Austria

⁵Institute of Ergonomics, Manufacturing Systems and Automation, OVGU, Magdeburg, Germany


Keywords: Industry 4.0, Change Management, Multi-view Modeling, Multi-aspect Information System, Production Systems Engineering.


Abstract: Agile Production Systems Engineering (PSE) is a complex, collaborative, and knowledge-intensive process. PSE requires expert knowledge from various disciplines and the integration of discipline-specific perspectives and workflows. This integration is a major challenge due to fragmented views on the production system with a difficult *a priori* coordination of changes. Hence, proper tracking and management of changes to heterogeneous engineering artifacts across disciplines is key for successful collaboration in such environments. This paper explores effective and efficient multi-view change management for PSE. Therefore, we elicit requirements for multi-view change management. We design the agile *Multi-view Change Management (MvCM)* workflow by adapting the well-established *Git Workflow with pull requests* with a multi-view coordination artifact to improve over traditional document-based change management in PSE. We design an information system architecture to automate MvCM workflow steps. We evaluate the MvCM workflow in the context of a welding robot work cell for car parts, using a typical set of changes. The findings indicate that the MvCM workflow is feasible, effective, and efficient for changes of production asset properties in agile PSE.


1 INTRODUCTION


Flexible production and shorter product and production development cycles are major goals of the Industry 4.0 (I4.0) vision (Galati and Bigliardi, 2019). Consequently, Production Systems Engineering (PSE) has to become increasingly agile, requiring engineers from several disciplines to work iteratively and in parallel (Eisenträger et al., 2018). PSE exhibits characteristics of knowledge-intensive processes (Di Ciccio et al., 2015), as design activities typically involve


considerable tacit knowledge. However, in agile settings, knowledge-intensive aspects are even more pronounced in PSE due to the dynamics coming from various domains after the completion of an activity. This setting results in an unpredictable flow of activities (dos Santos França et al., 2015). Effective and efficient PSE is covered in guidelines like the VDI 3695 (VDI, 2009) that discusses the maturity of PSE organizations. One aspect of the VDI 3695 is *Change Management (CM)* targeting the multidisciplinary nature of PSE. A method increasingly applied to support PSE maturity is the Product-Process-Resource (PPR) approach (Schleipen et al., 2015). It aims to organize the heterogeneous aspects of product designs, production process models, and production resources across disciplines. To understand the impact of collaborative design updates on production system properties (e.g., on performance, cost, and risk), changes across disciplines need to be managed


^a <https://orcid.org/0000-0002-6409-8639>


^b <https://orcid.org/0000-0002-9049-8038>


^c <https://orcid.org/0000-0002-4847-5182>

^d <https://orcid.org/0000-0001-7286-1393>

^e <https://orcid.org/0000-0002-7856-2113>

^f <https://orcid.org/0000-0001-6537-9742>

^g <https://orcid.org/0000-0002-4743-3124>

^h <https://orcid.org/0000-0002-3413-7780>

and integrated while supporting their heterogeneous perspectives (Passow and Passow, 2017). Views on a PPR properties (i) are often semantically similar, although the engineering artifacts or measurement units differ; and (ii) concern implicit domain knowledge that may be difficult to express. These dependencies introduce challenges for efficient Multi-view Change Management (MvCM) in agile PSE:

Challenge 1: Scattered and Heterogeneous Domain Knowledge. Scattered engineering views make it difficult to track changes' origin and manage them across stakeholder views (Wohlrab et al., 2020) properly. Consequently, MvCM requires comparison capabilities that go beyond best-practice text comparison for program source code (Toulmé, 2006).

Challenge 2: Insufficient Multi-view Change Management. The use of change management in PSE has matured within individual disciplines. However, mechanisms for multi-view change management across disciplines are currently not well understood. PSE, typically organized along several disciplines, requires engineers of different disciplines and organizations to cooperate, making change management much more difficult than for a single discipline (VDI, 2009).

To tackle these challenges, we raise the research question: *What process and information system design can improve the effectiveness and efficiency of multi-view change management in agile PSE?*

Aim. We follow the *Design Science* methodology to address the research question aiming at improving the effectiveness and efficiency of multi-view change management in PSE by (i) adapting and evaluating the Git workflow with pull request (Krusche et al., 2016); (ii) building on a Multi-Domain Engineering Graph (MDEG) as *coordination artifact* (Biffl et al., 2021; Rinker et al., 2021a) for change management.

Remainder of This Paper. Section 2 summarizes related work. Section 3 introduces an illustrative use case and requirements for efficient MvCM in agile PSE. Section 4 introduces the MvCM process and techniques to automate MvCM activities. Section 5 reports on the results of a feasibility study on a conceptual MvCM prototype. The requirements and data for the study come from use cases and domain experts from automotive manufacturing. Section 6 discusses the research results and limitations. Section 7 concludes and raises directions for future research.

2 RELATED WORK

This section summarizes related work on PSE, on the management of knowledge-intensive processes, and on change management in Software Engineering.

Multi-view System Modeling in PSE. In PSE, information is encapsulated in discipline- and tool-specific artifacts (Strahilov and Hämmerle, 2017). The VDI 3695 guideline (VDI, 2009) describes maturity levels for improving the capabilities of PSE organizations. In particular, it describes change management regarding the iterative exchange of engineering models/data, from single-discipline management (maturity level B) towards tool-assisted management across several disciplines (maturity level D). Change management concerns version/change management of systems engineering artifacts.

The I4.0 vision requires seamless and traceable information exchange across disciplines (Biffl et al., 2017) Model-based Systems Engineering (MBSE) provides the basis for domain-specific modeling and setting up cooperative processes (Huldt and Stenius, 2019). I4.0 assets (Heidel et al., 2017) represent the three main aspects of PSE: (i) *products* with their properties, (ii) *processes* producing the products, and (iii) *resources* that execute production processes. Such assets can be defined for the I4.0 Asset Administration Shell (AAS) (Plattform Industrie 4.0, 2020) using PPR modeling (Schleipen et al., 2015). PPR-based multi-view engineering networks seem promising to coordinate changes in artifacts and models, using markers according to a coordination policy (Biffl et al., 2021; Rinker et al., 2021a).

Knowledge-intensive Processes. A key characteristic of I4.0 is to support highly configurable products with a low overhead for production system re-configuration (Gilchrist, 2016). Therefore, agile PSE and production require agile and flexible workflows.

Agile PSE exhibits characteristics of Knowledge-Intensive Processes (KIPs) (Di Ciccio et al., 2015) as PSE processes are not tightly framed and do not follow predefined, deterministic workflows. Instead, their execution depends heavily on knowledge-intensive decision-making by experts coming from multiple disciplines. To manage KIPs, it is necessary to focus not only on tasks and workflows but to capture a variety of interrelated elements (Di Ciccio et al., 2015). In the agile PSE context, these elements include data, actions, rules, processes, production resources, and experts from various domains along all phases of the PSE life cycle.

Change Management in Agile Software Engineering. In agile software engineering, the *Git Workflow With Pull Requests* (Krusche et al., 2016) is a best practice for quality assurance and coordination of changes to source code, resulting from the parallel work of several contributors in a distributed project.

This workflow consists of three phases: (i) *scoping of source code management* by defining main and

feature branches as context for the change management workflow, including explicit dependencies of code elements; (ii) *pull requests* by contributors to trigger merging changes from feature branches into the main branch, defining the changescope to assess; and (iii) a *review process* for assessing the changeset regarding its impact on the main branch, including dependencies to other code parts. In particular, this concerns parallel changes, which may warrant considering the changesets of several pull requests that concern related/dependent code.

The Git Workflow with Pull Requests has been successfully used for source code, with text-based identification of changes. Popular code management platforms, such as GitHub¹ or Bitbucket², implement this workflow. Code reviewing systems, such as Gerrit³, also incorporate the *Pull Request* concept. While this approach could be useful in agile PSE, the following limitations prevent its direct application: PSE requires change management for changesets in heterogeneous multi-model, multi-domain engineering artifacts, not homogeneous source code.

In this paper, we build on a PPR based multi-view engineering graph as integrated *multi-view coordination artifact*, to coordinate multi-view change management in agile PSE following the *Git Workflow With Pull Requests* by mapping key concepts from software engineering to PSE.

3 USE CASE AND REQUIREMENTS

This section describes the use case *laser welding with a robot* and derives requirements from a domain analysis at an industry partner from car manufacturing. Fig. 1 illustrates the production process *laser welding* automated by a *robot* with a *welding head* (Kropatschek et al., 2021).

Multi-view Change Scenario 1: *Change propagation between semantically similar properties.* Engineers work on semantically similar asset properties. For instance, the welding head temperature properties in the process expert and the operator views (cf. Fig. 1, violet circles with letter *Y*) are considered similar. However, the properties might be represented in different formats or units, e.g., degrees C or K, in artifacts, e.g., a specification, plan, or log file. Therefore, a change of the welding head temperature by the process expert is a candidate for efficient propagation and

notification of the Operator.

Therefore, stakeholders require means to efficiently mark property changes (cf. Fig. 1, red diamonds) to semantically similar properties (cf. Fig. 1, dependencies represented as violet circles). The propagation typically occurs between stakeholder views of a single PPR asset, e.g., welding speed, but could also occur between property views of different PPR assets. If there is a transformation function to represent the semantic similarity relationship, the transformation can (i) notify the owner of the dependent property of the required change; and (ii) provide a suitable property value for a valid update. efficient

Multi-view Change Scenario 2: *Multi-view re-validation of properties after changes.* Engineers change asset properties that relate to other properties that then require re-validation. For example, welding speed, temperature, and performance are (i) related among each other and (ii) across stakeholder views (cf. Fig. 1). For instance, the process expert, automation engineer, and operator work with these properties, but in different method and technology contexts.

Hence, stakeholders require means to efficiently identify asset properties that need re-validation after such changes (cf. Fig. 1, red diamond markers). Fig. 1 shows these multi-lateral change dependencies as circles in orange with a letter. For instance, the letter *F* represents the dependency between welding speed, temperature, performance, and seam quality. On a property change, related properties could be marked (cf. Fig. 1, yellow diamond markers). These change markers can be used to indicate (i) a required re-validation to the property's "owner"; and (ii) possible change conflicts if several markers are present, e.g., the *welding robot's PE.welding performance*. Change conflicts may occur if stakeholders change asset properties with dependencies in parallel, in particular, if they span over engineering artifacts in several views.

For instance, the operator's change to the *welding speed* (cf. Fig. 1) needs to be re-validated and released by the quality manager, the process expert, and the automation engineer. The quality manager needs to approve the achieved quality, the process expert has to check dependencies to connected processes, and the automation engineer may have to adjust a program to address new speed tolerances.

Requirements for Multi-view Change Management. Based on the use case, we identified the following requirements (Rx) for an efficient MvCM approach to achieve maturity level D (VDI, 2009) regarding changes to PPR asset properties. *R1. Multi-view configuration management.* The approach shall represent domain knowledge for multi-view configuration management of PPR assets and properties. *R2.*

¹<https://github.com>

²<https://bitbucket.org>

³<https://www.gerritcodereview.com>

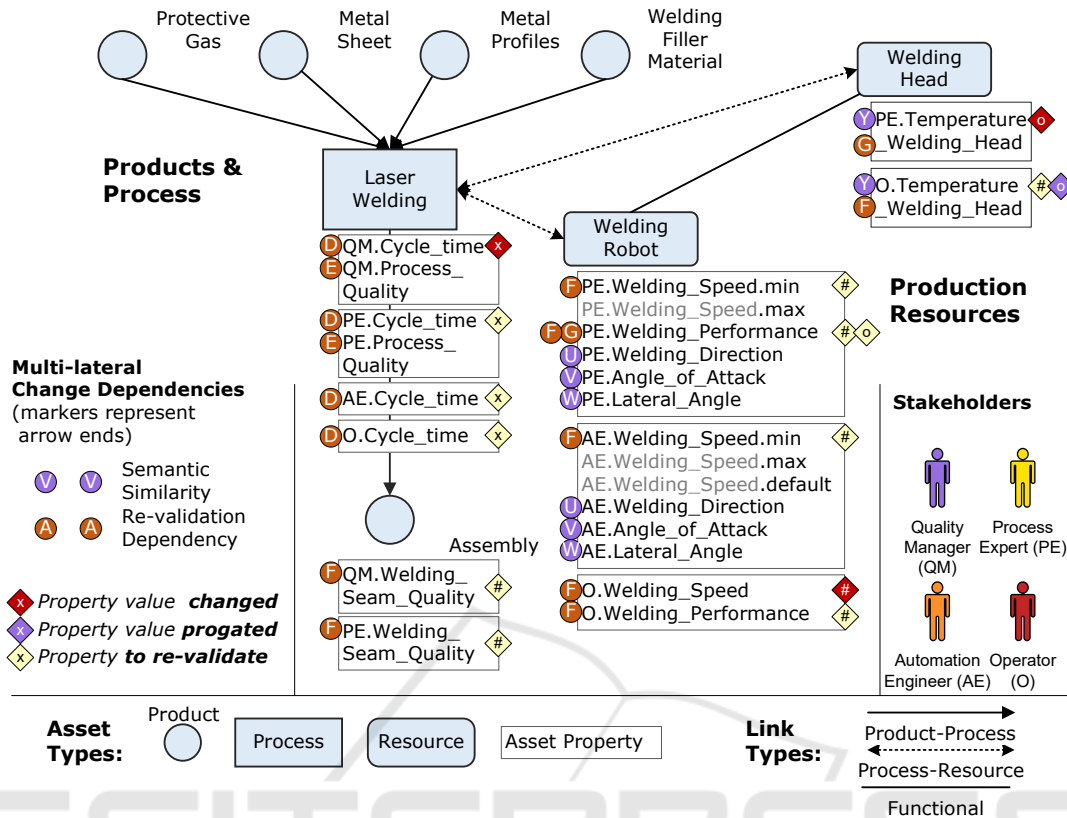


Figure 1: Stakeholder view properties for (i) the *Laser Welding* process (and products); and (ii) the *Robot* and *Welding Head* as main resources of a work cell, based on the Formalised Process Description (FPD) notation (Biffel et al., 2021).

Change tracing. The approach shall facilitate tracing change requests and changes in engineering artifacts across stakeholder views. *R3. Change coordination.* The approach shall facilitate representing and manipulating the changestate of PPR assets and properties as a basis for defining and executing change coordination policies. In particular, change propagation for semantically similar properties, and change re-validation analysis, to efficiently determine the potential impact of a change on dependent PPR properties. *R4. Efficient multi-view change management process.* The approach shall provide an efficient process for change management across PSE stakeholder views, building on domain knowledge represented according to requirements R1 to R3.

4 MULTI-VIEW CHANGE MANAGEMENT

This section introduces the MvCM workflow and the MvCM system design to automate workflow tasks. We adapted the *Code Review Workflow* introduced in (Krusche et al., 2016) from the software engineering

domain to equivalent phases in multi-view management in PSE.

MvCM Phase 0: Team Workspace Setup. Before the MvCM workflow can be conducted, the team workspace, which is comparable to a Git repository, needs to be set up.

Setting up the team workspace concerns: (i) Define the local concepts of each discipline; (ii) Negotiate Common Concepts (CCs) between the disciplines; (iii) Instantiate a Multi-Domain Engineering Graph (MDEG) using Common Concepts (CCs); and (iv) Define semantic links and dependencies between properties in the graph (Rinker et al., 2021a).

The MDEG contains the multi-view domain knowledge required for MvCM as a foundation for efficient analysis in a graph database. To facilitate describing dependencies among a group of properties, we adapt the PPR Asset Network (PAN) meta model (Biffel et al., 2021) to represent multi-lateral dependencies (cf. Fig. 1, dependency *F*), going beyond bi-lateral dependencies, between PPR asset properties. Further, we define structured properties, such as *welding_speed.min*, to represent dependencies on property *welding_speed*, including property substructures, such as required, minimal, or maximal welding

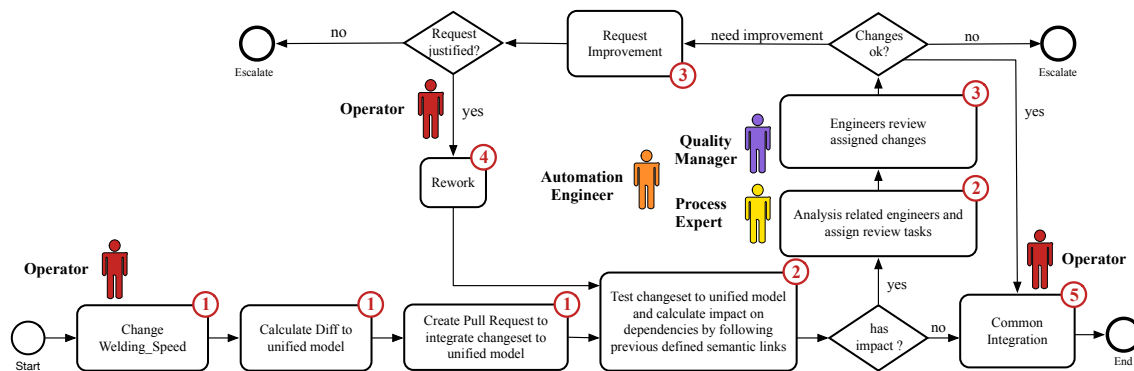


Figure 2: Multi-view Change Management (MvCM) workflow tasks and decisions, based on (Krusche et al., 2016).

speed. Fig. 2 depicts the MvCM workflow to address changes in a MDEG.

MvCM Phase 1: Local Preparation. In this phase, an engineer changes a property in a local, domain-specific view. The local view is comparable to a *fork* in the Git workflow in software engineering. For instance, Figure 1 shows the property *Welding_Speed* of the operator view (O) to be changed, e.g., from 10 to 20 mm/s.

To initiate the integration of the local view changes to the common view, the engineer creates a *Pull Request*, which includes the origin and target models and the set of change candidates. For instance, the property *O.Welding_Speed* in the common view has to be changed to 20. The *Pull Request* is placed into the main repository that holds the common view, assigning the local view engineer as initial reviewer.

MvCM Phase 2: Multidisciplinary Change Analysis. In this phase, the *Semantic Analyzer* service analyzes the impact of the changeset on dependent properties following the previously defined semantic links in the MDEG. For instance, semantic links occur if properties are semantically equal, requiring the propagation of a property value, or if a semantic relation requires the re-validation of a property value by an engineer of the corresponding discipline. If the changeset has no impact on other properties, the next phase is *Common Integration*. If the changeset concerns other properties, the *Semantic Analyzer* service marks these properties (i) *changed* due to an update of a stakeholder artifact (cf. Fig. 1, red diamonds); (ii) *changed* due to change propagation between semantically similar properties (cf. Fig. 1, violet diamond); or (iii) *to re-validate* due to a change dependency to a changed property (cf. Fig. 1, yellow diamonds). These marked changes are assigned for review to engineers of the disciplines concerning marked properties, e.g., the *Quality Manager*, *Process Expert*, and *Automation Engineer*.

MvCM Phase 3: Multidisciplinary Examination.

In this phase, the assigned engineers review the multi-view changes. Review activities concern (i) the validation of a changed property value due to a value propagation initialized by the origin pull request changeset or (ii) the re-validation of a changed property value due to a calculation, dependent on the origin change set values. These changes can be (i) marked as correct, which leads to the phase *Common Integration*, (ii) marked as incorrect, which leads to the task *Request Improvement*, or (iii) declined, which terminates the pull request process and escalates the workflow that requires the involved disciplines to discuss solution options.

MvCM Phase 4: Local Rework. This phase is initialized by the *Request Improvement* as a result of the *Multidisciplinary Examination* phase. A *Request Improvement* is filed, if the initial changeset results in implausible values in other disciplines, e.g., after a change by the *Operator* the calculated *QM.Welding_Seam_Quality* is not in the range of the required quality. The engineer, who submitted the original change in our case the *Operator*, decides whether (i) she can address the improvement request or (ii) rework is impossible, and the pull request needs to be escalated to project management. If rework is possible, the engineer updates the open pull request and triggers phase 2 *Multidisciplinary Change Analysis*.

MvCM Phase 5: Common Integration. In this phase, the pull request applicant integrates the changeset into the common view. The changeset consists of a list of model differences, i.e., deltas of the original modification in the local view and the propagated and calculated values. These model deltas are displayed for a final review and consistency checking, following the *Git Workflow* staging step. If the list of model differences is feasible, the pull request applicant commits the changes to the common view repository and adds a change description. Finally, the

common view version is incremented, similar to the *Git Workflow* using a semantic versioning number.

Multi-view Change System Design. Based on the elicited requirements (cf. Section 3) we infer the need for a lightweight and easy-to-manage system design that incorporates low-code solutions (Bucacioni et al., 2022). The system design is inspired by the techniques of the well-established Eclipse Modeling Framework (EMF) for advanced meta-modeling and model comparison and merging operations. However, EMF is closely coupled to Eclipse⁴, which hinders a custom-tailored integration into other system architectures (Batory and Altayan, 2020). Also, the Git⁵ change tracking system and workflow is the foundation for the *multi-view model change management* system.

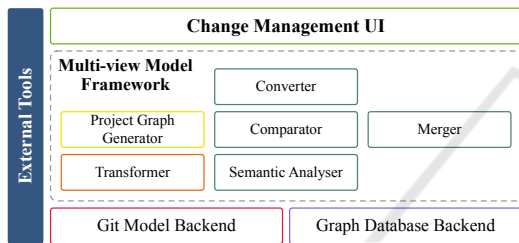


Figure 3: MvCM System Design, based on MvMF (Rinker et al., 2021b).

Fig. 3 depicts our design of the *Multi-view Change System* architecture building on the Multi-view Modeling Framework (MvMF) implementation (Rinker et al., 2021b). The system consists of a *Git Model Backend*, *Graph Database Backend*, a Multi-view Modeling Framework adapted by the *Semantic Analyzer* and *Change Management UI*.

5 EVALUATION IN A FEASIBILITY STUDY

This section reports on the evaluation of the *MvCM* approach. The evaluation environment consists of a MDEG, i.e., a PAN (Biffel et al., 2021), for the *laser welding* process (cf. Section 3). A Neo4J⁶ instance can be found online.⁷ The case focuses on three PPR assets and 30 to 40 properties from four stakeholder views, 12 change dependencies, and a set of three changes to asset properties. The input data for the

⁴Eclipse IDE: <https://www.eclipse.org>

⁵Git SCM: <https://git-scm.com/>

⁶Neo4J: <https://neo4j.com>

⁷Graph instance: https://github.com/tuw-qse/use-cases/tree/main/laser_welding

MvCM process are XML files from the use case, including quality dependencies from an FMEA tool⁸. The symbols in the diamond-shaped change markers (cf. Fig. 1) represent a property changeset. The assets, properties, and links in the PAN provide the basis to specify graph queries answering questions concerning the change workflow, e.g., *which properties depend on a changed property?* The following Cypher⁹ query retrieves a PAN sub-graph for marking PAN elements that depend on a changed PAN element (cf. Fig. 1, yellow diamonds).

```
MATCH (a:Attribute)
      [:has_PPRDependency *]
      (b:Attribute)
WHERE a.name="O.Welding_Speed" AND
      a.ChangeState="Changed"
SET b.ChangeState="To Validate"
```

Listing 1: Cypher query for marking dependent properties of a changed property for re-validation.

MvCM Efficiency in Comparison to Document-based Change Management. The conceptual process comparison estimates the effort for each *MvCM* phase, analyzing for which scenarios *MvCM* is likely to be more efficient than the traditional approach.

Phase 0. Team Workspace Setup. The engineering organization in the study context operates on maturity level CM-B, with engineering toolchains within the disciplines and data exchange among the disciplines. Modeling the PAN took only a small extra effort, as the data can be efficiently derived input data.

In the study context, the PAN of the robot cell can be defined as an output artifact of the data exchange. For a typical robot cell, defining the PAN requires up to three workdays. If the PAN is filled automatically, it should only take seconds. For the traditional approach, this task is not required.

Phases 1. Local Preparation and 2. Multidisciplinary Change Analysis. For the *MvCM* approach, the conceptual *MvCM* information system enables the analyzing of PPR asset property value updates. The conceptual *Semantic Analyzer* service uses Neo4J for the change impact evaluation and sets change markers based on the dependency definitions coming from Phase 1 (cf. Listing 1). Therefore, this analysis can be conducted efficiently after each change to the team workspace, taking a few seconds, to raise the awareness of the involved domain experts.

For the traditional approach, change impact analysis on the level of PPR asset properties is time-consuming and error-prone as the domain experts

⁸APIS FMEA: <https://www.apis-iq.com>

⁹Cypher language: <https://www.opencypher.org>

have to consider the impact of their local changes to related disciplines. This is often tacit knowledge or can only be decided by all involved domain experts. In the study context, domain experts from the involved disciplines have to come together in a time-consuming meeting to identify the changes in the use case.

Phases 3. Multidisciplinary Examination, 4. Local Rework, and 5. Common Integration. The review requires an overview of changes and their impacts to accept or reject a pull request and identify follow-up tasks to address the change impact. All properties with change dependencies receive markers to guide the review on these changes (cf. Fig. 1, yellow diamonds). To inform planning a focused review, the PAN provides an integrated view to identify which stakeholders are required to discuss a specific set of changes and to decide on reworking and integrating the changes, or to reject the pull request.

For the traditional approach, all potentially involved stakeholders have to meet to determine which views could be affected by a change. Therefore, the review meetings tend to become large and inefficient. Further, in a large meeting, it is easy for stakeholders to overlook a dependency in their view, leading to the risk of late and costly rework.

6 DISCUSSION

The MvCM process and system design introduced in this paper facilitate dynamic cross-domain collaboration of experts in KIPs (Di Ciccio et al., 2015) that are typical in agile PSE. Specifically, we investigated how the MvCM approach can improve multi-view change management by tracking and organizing parallel changes by several stakeholders. The evaluation in the context of a typical industrial joining process showed the MvCM process to be feasible, effective, and efficient (cf. Section 5).

The traditional document-based approach works well for treating isolated, sequential changes in small, co-located project settings. However, distributed or large projects in agile PSE, require fast, precise, and automated analysis of changes on the level of model property values, considering dependencies between individual domains, i.e., maturity level CM-D (cf. Section 2). Therefore, we expect the MvCM approach to make up the one-time effort required for modeling the PAN within a typical large automation project by reducing the effort for recurring change impact analysis and avoidable rework due to uncoordinated changes.

The results go beyond the state of the art in

the area of integrating data from and coordinating multi-disciplinary changes in PSE (Kattner et al., 2019; Meixner et al., 2021), in particular MvCM revalidation, (i) by defining a sufficiently fine-grained MDEG for analyzing changes of property values, (ii) by considering multi-lateral dependencies, rather than just bi-lateral dependencies between PPR asset properties, (iii) by efficiently describing semantic constraints among a group of properties, and (iv) by demonstrating the feasibility based on data from changes to PSE artifacts.

Limitations. The following limitations require further investigation. *Feasibility study.* The study focused on a use case abstracted from a large PSE company. This may introduce bias due to the specific selection of the production process, stakeholder views, or individual preferences of the domain experts. To overcome these limitations, we plan case studies in a wider variety of application contexts. *Change Cases and Workflow.* The selected change cases may introduce bias due to the specific selection of the data sets and assumptions of parallel changes. Further, the research focused on the *Git Workflow with Pull Requests*, a best practice in software engineering, while there is a wide variety of other engineering change management approaches. Therefore, we plan to explore further change cases and change management approaches to compare their benefits and limitations in diverse application settings.

7 CONCLUSION AND FUTURE WORK

This paper investigated effective and efficient Multi-view Change Management (MvCM) capabilities to integrate discipline-specific aspects in Production Systems Engineering (PSE) into a unified view. The MvCM process definition and evaluation builds on the Software Engineering best-practice process *Git Workflow with pull request* (Krusche et al., 2016). The MvCM process and system design provide capabilities for tracking the origin of changes and properly managing changes in parallel PSE to achieve multi-view maturity levels according to the PSE guideline VDI 3695-3 (VDI, 2009).

Results of the feasibility on a typical scope of changes in PSE to a robot work cell for welding car parts indicate the MvCM process to be feasible, effective regarding requirements derived from the PSE guideline VDI 3695, and efficient in comparison to traditional document-based change coordination.

Future Work. We plan to investigate approaches that are semantically more expressive than Neo4J, such

as Semantic Web technologies and domain-specific model management options, to represent and query larger change management use cases.

ACKNOWLEDGEMENT

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital & Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) via “Austrian Competence Center for Digital Production” (CDP) under contract nr. 881843. This work has received funding from the Teaming.AI project in the European Union’s Horizon 2020 research and innovation program under grant agreement No 95740.

REFERENCES

- Batory, D. S. and Altoyan, N. (2020). Aocl : A Pure-Java Constraint and Transformation Language for MDE. In *MODELSWARD 2020*, pages 319–327, Setúbal, Portugal. SCITEPRESS.
- Biffi, S., Lüder, A., and Gerhard, D., editors (2017). *Multi-Disciplinary Engineering for Cyber-Physical Production Systems, Data Models and Software Solutions for Handling Complex Engineering Projects*. Springer.
- Biffi, S., Musil, J., Musil, A., Meixner, K., Lüder, A., Rinker, F., Weys, D., and Winkler, D. (2021). An Industry 4.0 Asset-Based Coordination Artifact for Production Systems Engineering. In *23rd IEEE Int. Conf. on Business Informatics*. IEEE.
- Bucaioni, A., Cicchetti, A., and Ciccozzi, F. (2022). Modelling in low-code development: a multi-vocal systematic review. *Software and Systems Modeling*.
- Di Ciccio, C., Marrella, A., and Russo, A. (2015). Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches. *Journal on Data Semantics*, 4(1):29–57.
- dos Santos França, J. B., Netto, J. M., do ES Carvalho, J., Santoro, F. M., Baião, F. A., and Pimentel, M. (2015). KIPO: the knowledge-intensive process ontology. *Software & Systems Modeling*, 14(3):1127–1157.
- Eisenträger, M., Adler, S., Kennel, M., and Möser, S. (2018). Changeability in engineering. In *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–8.
- Galati, F. and Bigliardi, B. (2019). Industry 4.0: Emerging themes and future research avenues using a text mining approach. *Computers in Industry*, 109:100–113.
- Gilchrist, A. (2016). Introducing industry 4.0. In *Industry 4.0*, pages 195–215. Springer.
- Heidel, R., Hankel, M., Döbrich, U., and Hoffmeister, M. (2017). *Basiswissen RAMI 4.0: Referenzarchitekturmodell und Industrie 4.0-Komponente Industrie 4.0*. Beuth Verlag.
- Huldt, T. and Stenius, I. (2019). State-of-practice survey of model-based systems engineering. *Systems engineering*, 22(2):134–145.
- Kattner, N., Bauer, H., Basirati, M. R., Zou, M., Brandl, F., Vogel-Heuser, B., Böhm, M., Krcmar, H., Reinhart, G., and Lindemann, U. (2019). Inconsistency management in heterogeneous models. In *Proc. Design Society: Int. Conf. Eng. Design*, pages 3661–3670. Cambridge Univ.
- Kropatschek, S., Steuer, T., Kiesling, E., Meixner, K., Frühwirth, T., Sommer, P., Schachinger, D., and Biffi, S. (2021). Towards the representation of cross-domain quality knowledge for efficient data analytics. In *ETFA 2021*, pages 1–4.
- Krusche, S., Berisha, M., and Bruegge, B. (2016). Teaching code review management using branch based workflows. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 384–393.
- Meixner, K., Lüder, A., Herzog, J., Winkler, D., and Biffi, S. (2021). Patterns For Reuse In Production Systems Engineering. *International Journal of Software Engineering and Knowledge Engineering*, pages 1623–1659.
- Passow, H. J. and Passow, C. H. (2017). What competencies should undergraduate engineering programs emphasize? a systematic review. *Journal on Engineering Education*, 106(3):475–526.
- Plattform Industrie 4.0 (2020). Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01 Review). Standard, German BMWI. <https://bit.ly/37A002I>.
- Rinker, F., Meixner, K., Waltersdorfer, L., Winkler, D., Lüder, A., and Biffi, S. (2021a). Towards efficient generation of a multi-domain engineering graph with common concepts. In *ETFA 2021*, pages 1–4. IEEE.
- Rinker, F., Waltersdorfer, L., Meixner, K., Winkler, D., Lüder, A., and Biffi, S. (2021b). Continuous Integration in Multi-view Modeling: A Model Transformation Pipeline Architecture for Production Systems Engineering. In *MODELSWARD 2021*, pages 286–293. SCITEPRESS.
- Schleipen, M., Lüder, A., Sauer, O., Flatt, H., and Jasperneite, J. (2015). Requirements and concept for plug-and-work. *at-Automatisierungstechnik*, 63(10):801–820.
- Strahilov, A. and Hämmerle, H. (2017). Engineering workflow and software tool chains of automated production systems. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer.
- Toulmé, A. (2006). Presentation of EMF Compare Utility. In *Eclipse Modeling Symposium*, pages 1–8.
- VDI (2009). VDI Guideline 3695: Engineering of industrial plants - Evaluation and optimization. Standard, VDI-Verlag, Düsseldorf, DE.
- Wohlrab, R., Knauss, E., Steghöfer, J.-P., Maro, S., Anjorin, A., and Pelliccione, P. (2020). Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture. *Requirements Engineering*, 25(1):21–45.