Change Request Management in Model-Driven Engineering of Industrial Automation Software

Heng-You Lin[1], Seppo Sierla[1], Nikolaos Papakonstantinou[2], Anatoly Shalyto[4], Valeriy Vyatkin[1,3]
[1]Department of Electrical Engineering and Automation, Aalto University, Finland
heng-you.lin@aalto.fi, seppo.sierla@aalto.fi, vyatkin@ieee.org
[2]VTT Technical Research Centre of Finland, Espoo, Finland, nikolaos.papakonstantinou@vtt.fi
[3]Department of Computer Science, Electrical and Space Engineering, Lulea Tekniska Universitet, Sweden
[4] ITMO University, St. Petersburg, Russia, shalyto@mail.ifmo.ru

***Abstract*—Change request management and Model Driven Engineering (MDE) are two key concepts for industrial automation software in today's competitive and fast changing environment. However, although there exist frameworks on general change management, they do not exploit the capabilities of MDE. This paper proposes a workflow to combine these two technologies, enabling the engineer responsible for the change to quickly and efficiently create analysis on the impact of the change, as well as the feasibility of a proposed solution. This result of the analysis is presented with concrete SysML model diagrams to better support decision making for a change request. One advantage of performing change management in SysML is the possibility to automate parts of the change analysis process; one step is automated as a proof-of-concept.**

Keywords—Change request management; model-driven engineering; SysML; Industrial automation software;

I. INTRODUCTION

One major motivation for model-driven engineering (MDE) with System Modelling Language (SysML) is to increase reusability of software components thus contributing to flexibility of systems. This is achieved by storing each model element and the relationships between them in a model repository, so that they can be used in several models representing different aspects of configurations of the system. This approach ensures the consistency between models and delivers integrity of data, giving SysML a fundamental advantage as compared to document-based design approach over the lifecycle of the system, which involves activities such as change impact assessment, reuse and development of variants. However, the majority of research on SysML is focused on product development without specifically exploiting the technology's potential advantages to support various lifecycle management activities, especially change management. SysML-based MDE approaches that explicitly aim at addressing some aspects of change management include change tracking approaches that are specifically suited to MDE [1], a modelling approach for analysing interdisciplinary change influences [2] and management of variants in a software product line [3]. One procedure for handling change requests in SysML-based MDE is presented in [4], involving an iteration through all development phases starting from requirements analysis; such an approach is in contrast to industrially established change request management procedures that are not widely accepted in today's Product Lifecycle Management (PLM) systems which have proven their efficiency in industrial engineering change management problems, e.g. [5]. However, there is still lack of specific guidelines to help engineers in creation of engineering change requests (ECR) exploiting the potential benefits brought by SysML modelling. Thus the research goal of this paper is to propose a change request management workflow for a system developed using SysML-based MDE approach that enables better decision making and impact analysis for a proposed engineering change.

This paper is structured as follows. Section 2 reviews research in SysML-based MDE and in change request management. Section 3 proposes a procedure for change request management that exploits the information available in a SysML model. Section 4 demonstrates the procedure with a case study. Section 5 presents a proof-of-concept that exploits the SysML model to automate one step in the change analysis workflow. Section 6 concludes the paper.

II. LITERATURE REVIEW

Several SysML based MDE approaches have the potential to support the management of engineering changes during the system lifecycle, if they are integrated to a MDE change management process. Work towards automating the generation of PLC software from SysML models [6-8] can significantly reduce the engineering effort for the change after the SysML model has been updated. Work on creating simulation models from SysML models [9, 10] can support the development of an early capability for detailed assessment of change impact through simulation. The industrial applicability of SysML-based MDE approaches for managing product safety certification [11, 12] would benefit from a systematic change management capability to ensure that the safety certification is valid after the changes.

A significant number of publications in software engineering address change requests e.g. [13-15], but in mechatronic applications, interdisciplinary dependencies need to be taken into account [2, 16, 17]. Comprehensive support for managing change orders is available in PDM/PLM (Product Data/Lifecycle Management) systems, which are able to cover the full scope of change impacts including redesign, updating documentation, procurement of new parts from the supply chain and/or rescheduling manufacturing operations [5, 18-20]. However, each of these methods has a step such as "investigate problem" or "assess impact" for which there is no more detailed

methodological support available. This is a shortcoming of PLM systems [21], which are document based systems that use metadata as opposed to being fully model-based [22], so they do not have complete information of the dependencies between different aspects of the product, such as the relationship between structural and behavioural models of the product [23]. In order to bridge the gap between MDE and PLM, a UML-based approach is presented in [24] and more recently in [25], with SysML identified as a topic of further research. The partially model-based nature of PLM systems may be overcome by the STEP standard, but the highly complex work on mapping these standards to SysML has not reported progress since 2010 [26].

III. Proposed Workflow

In this section, we present a change request management workflow for a product that has been designed using MDE with SysML. In particular, our work builds on previous work [5, 18, 19], with special focus on addressing the crucial steps of the change request process. The proposed workflow achieves this goal by providing a guideline for the change engineer to better identify the impact of changes and to assess the feasibility of a proposed solution with the goal of enabling the developer to produce concrete and formal deliverables to the Change Review Board (CRB). The latter uses this information to make decisions on whether to accept and when to execute the change request. Since the goal of this workflow is to generate the necessary information to enable informed decision making, the workload of this phase should not be heavy, and the internals of the model elements may be incomplete and addressed later in the change implementation phase. An additional benefit of this proposed approach is that the proposed design in SysML can be directly taken as a starting point for implementation after the change request is approved. The proposed workflow has the following two steps: Technical Review and Impact Analysis (Fig. 1).

The first step in the proposed workflow, "Technical Review", has the following actions:

1. Based on the Problem Report, the change engineer identifies the existing functional and non-functional SysML requirements that need to be modified or created (e.g. increase the throughput of product or material flow through the automated production systems).
2. Based on these newly created/changed requirements, the engineer, responsible for change implementation, identifies the affected existing model elements that have a "satisfy" or "trace" relationship with the said requirement and determines the type and instances of the affected model elements, thus pointing to where a potential solution is likely to be found (structural elements correspond to new hardware, behaviour elements suggest software fixes);
3. All other model elements that have either "allocate", "aggregation", "Satisfy" and "trace" relationship with the model elements, identified in the previous step, are also marked for possible impact by the change;
4. The final action is to build/design the "Model for Proposed Solution", starting from the new requirements resulting from the action one, then adding both structural and behaviour model elements with their dependency (satisfy, allocate, trace etc.). The output result of this "Technical Review" step should include a complete model of the proposed solution and the extract of the impacted model elements from the existing model. These two models, shown as the object nodes following the "Technical Step" in Fig. 1, are the input material for the next, "Impact Analysis" step.

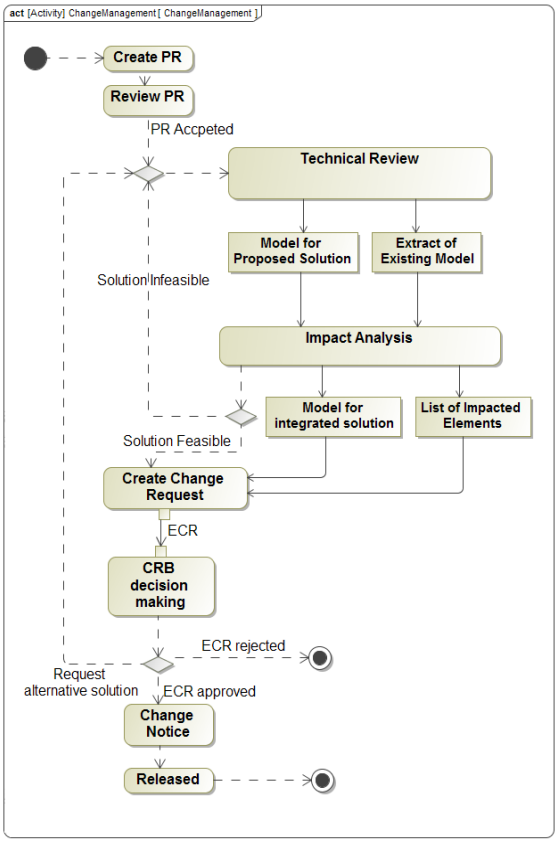

Figure 1 Proposed change management workflow

In the next step, "Impact Analysis" allows the change engineer to determine the impact of the change after the model of the proposed solution is integrated into the existing model. The "change engineer" performs this integration manually, replacing the model elements, as well as their roles in any existing dependency relationships. This helps in analysing the feasibility of the proposed solution, as well as in identifying any conflicting requirements or other model elements. The "Impact Analysis" activity has three actions:

1. Identify existing elements that are to be replaced, and for each dependency relationship these elements have with other elements in the existing model, reconnect them to the corresponding elements in the proposed solution model;
2. Integrate the new model elements introduced in the proposed solution and create necessary new relationships between them and the existing elements ("Model for Integrated Solution" in Fig. 1);

3. Lastly create a list of dependency relationships and model elements, including functional and non-functional requirements that are affected by the change (i.e. elements and dependencies that are to be replaced or deleted), which will be included directly into the ECR ("List of impacted elements" in Fig. 1).

After these three actions, if a relationship cannot be reconnected, for example a new block cannot be aggregated into an existing block, standing higher up in the structural hierarchy, or if there are conflicting requirements, the proposed solution is considered to be infeasible. The solution and the list of dependencies that could not be reconnected are sent back to the previous step "Technical Review". If all dependency relationships can be successfully connected and all new elements can be integrated to the system model successfully, the proposed solution is considered to be feasible. The integrated model for the proposed new product model and the list of affected model elements are then passed on to complement the ECR document.

After these deliverables and the ECR are sent to the CRB, if the CRB identifies that the impact of the change is potentially major and that it wants more detailed information on the implementation of the proposed solution, it can request a new iteration of technical review and impact analysis to obtain a more detailed proposed design.

IV. Case Study

In this section, we present a case study of the proposed workflow in a change scenario applied to the well-studied FESTO MPS workstations [12, 27-29] consisting of the Distribution, Testing, Processing and the Handling stations forming a laboratory scale production line (Fig. 2). A SysML model for the current configuration of the production line was built. In this configuration, the system exhibits sequential behaviour, meaning that each station can only work with one workpiece at one time. We now assume a new problem report has been approved indicating a solution is needed to increase the throughput of the processing station, which has been observed to be a bottleneck in the system throughput of the production line.

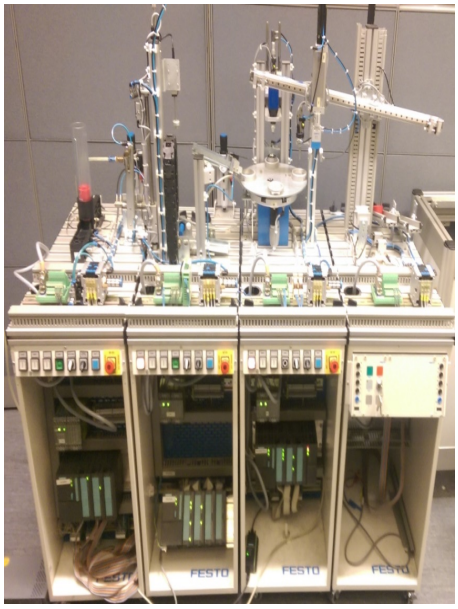

Figure 2 FESTO MPS workstations

A. Technical Review

The actions 1-4 listed in section III will be performed.

1. We first identify that the relevant existing non-functional requirement that needs to be addressed is "ThroughputProcessingLow" (Fig. 3).

2. The model element that has the "satisfy" dependency relationship with it is the activity "ProcessingSequential" (Fig. 3) which describes the sequential behaviour that the processing station currently exhibits. This suggests that a proposed solution may be found in modifying the control software of the system.

3. We identify all the functional and non-functional requirements that have a "satisfy" or "trace" dependency to this activity, since these requirements set the basic criteria in determining the feasibility of the proposed solution later in the Impact Analysis step. In Fig. 3, we therefore include the functional requirement "ProcessingOfMaterial" and its aggregated sub-requirements, which specify the functionality the proposed solution must satisfy. We also include the "PLCPlatform" requirement, which indicates that this requirement may no longer be needed in the new model based on the being the client in a "trace" relationship with the "ProcessingSequential" activity. The "allocate" relationships help us to also identify the "PS_PLC_Code" block representing the software that implements the current sequential behaviour and the PLC block "S7-300PLC" the software is deployed to. These two elements, as well as the Platform (electrical) dimension and the "ProcessingStation" block itself are also potential impacted model element and thus needs to be included in the impacted model, so when determine the feasibility of the proposed solution, they are also considered.

4. The final action in the Technical Review step is to propose a solution to address the problem and build a SysML model for this solution for further analysis. The obvious first step is to create a new non-functional requirement representing the higher throughput "ThroughputProcessingHigh" and refined it further if necessary. In action 1, we have identified the throughput requirement to be satisfied by an activity in the existing model. Therefore, a proposal is investigated in which the throughput is increased by changing the behaviour (captured by the activity) of the processing station. We therefore include an additional requirement that is refined from "ThroughputProcessingHigh", demanding the new system to exhibit parallel, pipeline behaviour. The new requirement is named "ParallelBehaviour". The next step in building the proposed solution model is to construct a new activity, which satisfies both the new throughput

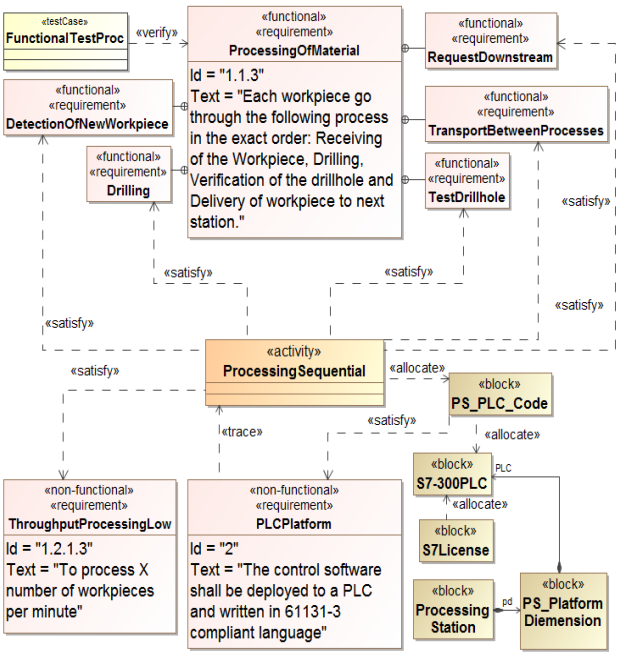

Figure 3 Affected elements of the existing model

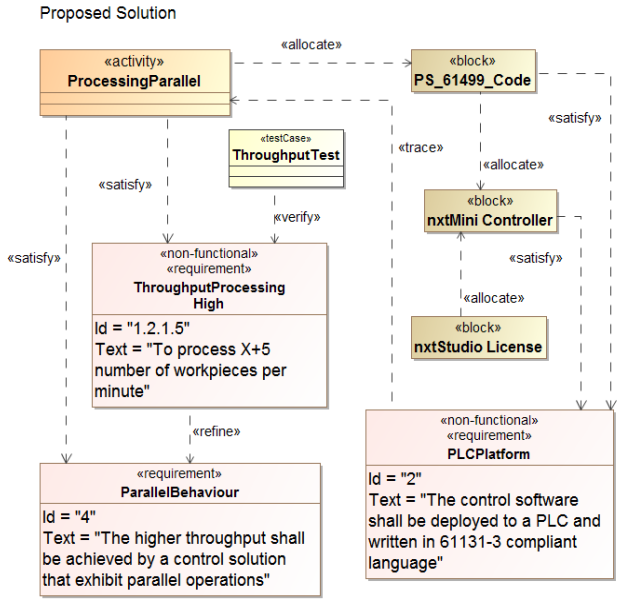

Figure 4 Model for Proposed Solution

requirement and also the parallel behaviour requirement. The activity should also satisfy the functional requirements as listed in the previous step, in preparation for validation in the next step's Impact Analysis. Once the new activity has been created, we identify that the new activity can be implemented by exploiting an existing software component written in IEC61499 language and thus we then allocate the activity to a "PS_61499_Code" block, which in turn is allocated to an IEC61499 compliant "nxtMini_Controller". At this stage, we do not yet develop the actual implementation or create a complete detailed model for the proposed solution - keeping to the goal of enabling quick ECR decision making, the granularity and the completeness of this model for the proposed solution is only developed to a level which enables us to perform analysis on the feasibility of this solution in the next step.

B. *Impact Analysis*

In this step, we take the two models created in the previous step and try to integrate the proposed solution into the existing model to determine its feasibility. We achieve this goal by making sure that all the existing dependency relationships held by the model element targeted by the change, as presented in the "Extract of the Existing Model", can be reconnected to the corresponding elements in the proposed solution model. By the end of this step, the output material should include a model of the integrated solution model representing the updated Processing Station with pipeline/parallel behaviour, and also a list of affected model elements with additional information for each of them whether they are added, removed or replaced.

The actions A-C listed in section III are performed.

A. In the first action of Impact Analysis, we identify the old throughput requirement "ThroughputProcessingLow" as to be replaced by the new requirement "ThroughputProcessingHigh". The activity "ProcessingSequential" is to be replaced by "ProcessingParallel" (Fig. 3 and 4). Also the existing non-functional requirement PLCPlatform will be removed. We then try to reconnect all the "satisfy" relationships the obsolete activity "ProcessingSequential" held to the replacing activity "ProcessingParallel", in the process checking that the new activity does indeed satisfy these relationships.

B. The next action is to integrate the remaining model elements from the proposed solution model, including the"PS_61499_Code", "nxtMini Controller" and "nxtStudio License" by adding them to the existing model. We also realize that there is no activity that is allocated to "PS_PLC_Code" and "S7-300PLC" obsolete they are removed from the model. In the process we reconnect the aggregation relationship "PLC" between the old "S7-300" and the Platform (electrical) dimension block to the new "nxtMini Controller" indicating a replacement of the controller. The completed integrated model is shown in Fig. 5 and we have observed that all the existing "satisfy", "allocate" and "aggregate" relationships have been reconnected, suggesting that the proposed change to the existing system model is feasible and is an adequate solution to the new throughput requirement.

C. The last task of building a list of model elements to be added, removed or replaced is then easily carried by simply observing the difference between the integrated model and the existing model of the current system.

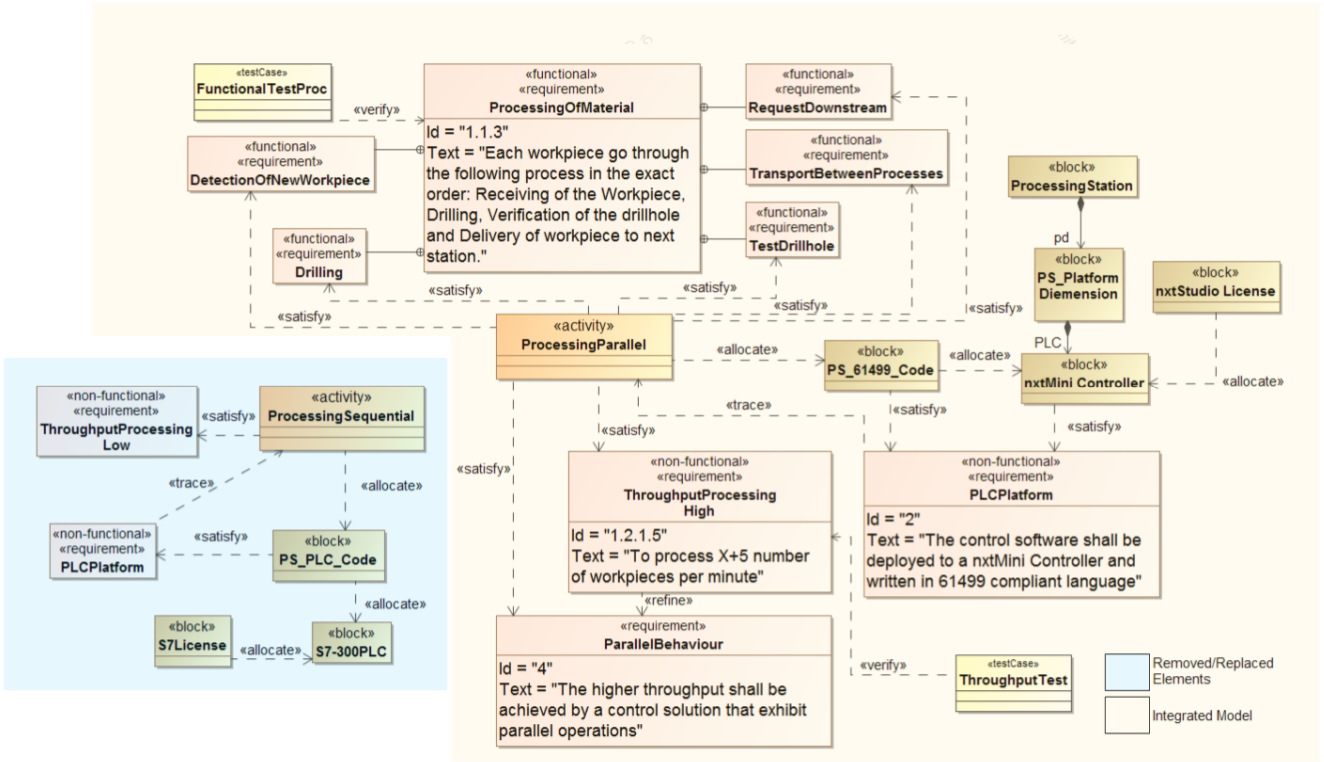

Figure 5 Integrated Model and model elements to be removed/replaced

V. Proof of Concept

One major benefit of performing change management in the SysML environment is the possibility to exploit the machine readable model representation to partially automate the change analysis workflow. As a proof of concept, the step 3 of the technical review is automated with a Java tool that has been developed for this purpose. The tool parses the XMI XML file of the SysML model of the system in Fig. 2, which includes the excerpt in Fig. 3. The tool identifies the abstractions linked to the element to be modified, which is the "ProcessingSequential" activity in Fig. 3, filters them and keeps only the <<satisfy>> and <<trace>> relationships. It then locates the model elements at the other end of these relationships and presents them to the user; a screenshot is presented in Fig. 6.

Java XML processing technology was chosen instead of some OMG technologies. For example, the MOF Model-To-Text Transformation Language (MOFM2T) is a standardized method for extracting relevant information from a SysML model and presenting it in a human readable form, but it does not fully support modification of the SysML model [30]. Model to model transformation languages such as ATL and QVT [31] were not used since the change management workflow defined in this paper performs model updates rather than model transformations.

VI. Conclusion

This paper presents a proposed workflow to exploit design information that is created by MDE process in the context of change order management. We have also illustrated a case study showing how the workflow is applied to a change scenario for a laboratory scale production line.

Several of the actions listed in section III can be automated by tools that read the XMI representation of the model; as a proof-of-concept, step 2 of the technical review was automated, and the outputs are both in human-readable forms (Fig. 6) as well as in the tool memory to support further work to automate the rest of the process, resulting in a partially automated change order process that is currently not possible in change order management approaches that do not exploit MDE.

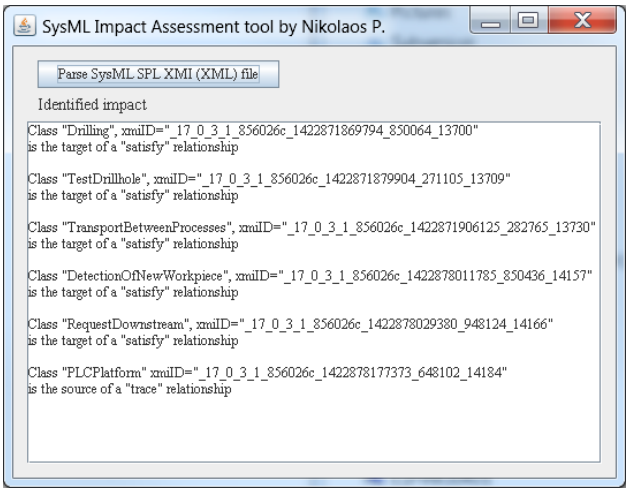

Figure 6 Screenshot of Java tool that automates step 3 of the technical review process

Several directions of further work are possible and planned. Firstly, a more comprehensive automation of the change analysis process, with user-friendly interfaces, is pursued. With SysML, it is also possible to quantify the impact of the change by recording the number and type of model elements that are affected. Another important direction of research is definition of model design patterns for industrial automation applications engineering, among which the change pattern, describing online or offline change of software and hardware and their mutual impacts, which can be represented in form of rules. We also plan to investigate mathematical formulation of the change feasibility check and application of the proposed method on a large-scale industrial grade case study.

REFERENCES

[1] M. Koegel, M. Herrmannsdoerfer, L. Yang, J. Helming, and J. David, "Comparing State- and Operation-Based Change Tracking on Models," in *Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International*, 2010, pp. 163-172.

[2] K. Kernschmidt and B. Vogel-Heuser, "An interdisciplinary SysML based modeling approach for analyzing change influences in production plants to support the engineering," in *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, 2013, pp. 1113-1118.

[3] N. Papakonstantinou and S. Sierla, "Generating an Object Oriented IEC 61131-3 software product line architecture from SysML," in *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, 2013, pp. 1-8.

[4] H. Hoffmann, "Streamlining the development of complex systems through model-based systems engineering," in *Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st*, 2012, pp. 6E6-1-6E6-8.

[5] W.-H. Wu, L.-C. Fang, T.-H. Lin, S.-C. Yeh, and C.-F. Ho, "A Novel CMII-Based Engineering Change Management Framework: An Example in Taiwan's Motorcycle Industry," *IEEE Transactions on Engineering Management,* vol. 59(3), pp. 494-505, 2012.

[6] M. Obermeier, S. Braun, and B. Vogel-Heuser, "A Model Driven Approach on Object Oriented PLC Programming for Manufacturing Systems with regard to Usability," *IEEE Transactions on Industrial Informatics* vol. PP(99), pp. 1-1, 2014.

[7] K. Thramboulidis and G. Frey, "An MDD process for IEC 61131-based industrial automation systems," in *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*, 2011, pp. 1-8.

[8] B. Vogel-Heuser, D. Schütz, T. Frank, and C. Legat, "Model-driven engineering of Manufacturing Automation Software Projects – A SysML-based approach," *Mechatronics,* vol. 24(7), pp. 883-897, 2014.

[9] S. Berrani, A. Hammad, and H. Mountassir, "Mapping SysML to modelica to validate wireless sensor networks non-functional requirements," in *Programming and Systems (ISPS), 2013 11th International Symposium on*, 2013, pp. 177-186.

[10] G. D. Kapos, V. Dalakas, A. Tsadimas, M. Nikolaidou, and D. Anagnostopoulos, "Model-based system engineering using SysML: Deriving executable simulation models with QVT," in *Systems Conference (SysCon), 2014 8th Annual IEEE*, 2014, pp. 531-538.

[11] M. Sabetzadeh, S. Nejati, L. Briand, and A. E. Mills, "Using SysML for Modeling of Safety-Critical Software-Hardware Interfaces: Guidelines and Industry Experience," in *High-Assurance Systems Engineering (HASE), 2011 IEEE 13th International Symposium on*, 2011, pp. 193-201.

[12] K. Thramboulidis and A. Buda, "3+1 SysML view model for IEC61499 Function Block control systems," in *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, 2010, pp. 175-180.

[13] M. Linares-Vasquez, K. Hossen, D. Hoang, H. Kagdi, M. Gethers, and D. Poshyvanyk, "Triaging incoming change requests: Bug or commit history, or code authorship?," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, 2012, pp. 451-460.

[14] Z. Stojanov, "Discovering automation level of software change request process from qualitative empirical data," in *Applied Computational Intelligence and Informatics (SACI), 2011 6th IEEE International Symposium on*, 2011, pp. 51-56.

[15] M. Gethers, H. Kagdi, B. Dit, and D. Poshyvanyk, "An adaptive approach to impact analysis from change requests to source code," in *Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on*, 2011, pp. 540-543.

[16] Y.-M. Chen, W.-S. Shir, and C.-Y. Shen, "Distributed engineering change management for allied concurrent engineering," *International Journal of Computer Integrated Manufacturing,* vol. 15(2), pp. 127-151, 2002/01/01 2002.

[17] C.-H. Yang, V. Vyatkin, and C. Pang, "Model-Driven Development of Control Software for Distributed Automation: A Survey and an Approach," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 44(3), pp. 292-305, 2014.

[18] M. H. El-Jamal, "Requirements Change using Product Lifecycle Management for Manufacturing Processes in a Systems Engineering Context," in *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, 2008, pp. 1-5.

[19] J.-Y. Shiau and X. Li, "Product configuration for engineering change decision," in *Networking, Sensing and Control, 2009. ICNSC '09. International Conference on*, 2009, pp. 691-696.

[20] A. Shaout, M. Arora, and S. Awad, "Automotive software development and management," in *Computer Engineering Conference (ICENCO), 2010 International*, 2010, pp. 9-15.

[21] D. Habhouba, S. Cherkaoui, and A. Desrochers, "Decision-Making Assistance in Engineering-Change Management Process," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 41(3), pp. 344-349, 2011.

[22] H. Abid, P. Pernelle, C. Benamar, and D. Noterman, "A modelling approach of mechatronic products in PLM Systems," in *Control, Decision and Information Technologies (CoDIT), 2013 International Conference on*, 2013, pp. 714-718.

[23] L. Horvath and I. J. Rudas, "An Approach to Processing Product Changes During Product Model Based Engineering," in *System of Systems Engineering, 2007. SoSE '07. IEEE International Conference on*, 2007, pp. 1-6.

[24] Z. Wang, L. Wang, Y. Rui, and C. Li, "Research on PLM multidimensional data model," in *Technology and Innovation Conference, 2006. ITIC 2006. International*, 2006, pp. 1499-1504.

[25] A. Houssem, P. Philippe, N. Didier, C. J. Pierre, and B. Chokri, "Integration approach of mechatronics system in PLM systems," in *Automation and Computing (ICAC), 2013 19th International Conference on*, 2013, pp. 1-6.

[26] O. M. G. Inc. (2010). *SysML and AP233 Mapping Activity*. Available: http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-ap233:mapping_between_sysml_and_ap233

[27] M. Hirsch, "Systematic Design of Distributed Industrial Manufacturing Control Systems," PhD dissertation, Martin-Luther-Universität Halle-Wittenberg, 2010.

[28] M. Hirsch, C. Gerber, H. M. Hanisch, and V. Vyatkin, "Design and Implementation of Heterogeneous Distributed Controllers According to the IEC 61499 Standard - A Case Study," in *Industrial Informatics, 2007 5th IEEE International Conference on*, 2007, pp. 829-834.

[29] K. Thramboulidis, "Challenges in the development of Mechatronic systems: The Mechatronic Component," in *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, 2008, pp. 624-631.

[30] L. M. Rose, N. Matragkas, D. S. Kolovos, and R. F. Paige, "A feature model for model-to-text transformation languages," in *Modeling in Software Engineering (MISE), 2012 ICSE Workshop on*, 2012, pp. 57-63.

[31] K. Lano and S. Kolahdouz-Rahimi, "Model-Transformation Design Patterns," *IEEE Transactions on Software Engineering,* vol. 40(12), pp. 1224-1259, 2014.