Feature Identification for Engineering Model Variants in Cyber-Physical Production Systems Engineering

Kristof Meixner
Christian Doppler Lab CDL-SQI, ISE
Technische Universität Wien, Austria
kristof.meixner@tuwien.ac.at

Rick Rabiser
LIT CPS Lab
Johannes Kepler Univ. Linz, Austria
rick.rabiser@jku.at

Stefan Biffl
Inst. of Information Systems Eng.
Technische Universität Wien, Austria
stefan.biffl@tuwien.ac.at

ABSTRACT

In *Cyber-Physical Production System (CPPS)* engineering, *Assembly Sequence (AS)* models of products are primary engineering artifacts. Product variants are often designed as *Product-Process-Resource (PPR)* AS models that are initiated with *clone-and-own* approaches and by the manual derivation of shared features. This paper introduces the *PPR Feature Candidate Identification (PPR-FCI)* approach for identifying features from PPR AS models of product variants. From these features our approach derives a superimposed PPR that describes design options for engineers planning the CPPS. The approach is based on existing feature extraction research which we adapted to the scope of PPR models in CPPS engineering. Based on a real-world product line, we evaluate our PPR-FCI approach for feasibility and usefulness by comparing our automated approach to the traditional manual approach with domain experts. Initial findings show that the approach can identify relevant features from PPR AS models and domain experts found the results useful. However, further research is required to improve the PPR-FCI approach regarding the optimization of PPR Assembly Sequence models.

CCS CONCEPTS

• **Software and its engineering → Software product lines**.

KEYWORDS

Feature Extraction, Product Lines, Cyber-Physical Production System (CPPS), Product-Process-Resource (PPR)

ACM Reference Format:
Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2020. Feature Identification for Engineering Model Variants in Cyber-Physical Production Systems Engineering. In *Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS '20), February 5–7, 2020, Magdeburg, Germany*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3377024.3377043

1 INTRODUCTION

Cyber-Physical Production Systems (CPPSs) are software-intensive systems that utilize modern *ICT* for smartly interacting with their physical environment to produce a large variety of products, such as automated car manufacturing plants. CPPSs have attracted considerable research interest recently [9, 10]. CPPS engineering involves several engineering disciplines, like mechanical, electrical, and software engineering, and related artifacts [1, 6]. The complexity and heterogeneity of such engineering environments makes knowledge exchange, e.g., on product properties, essential but challenging.

For our industry partner, an Austrian design and system integration enterprise for automated high-performance manufacturing machines, *Assembly Sequences (ASs)* are crucial knowledge artifacts that define production step sequences transforming *input* to *output products*. The model-based *Product-Process-Resource (PPR)* approach [13] is capable of representing ASs, by allowing to model products (e.g., gear boxes), the corresponding processes to create them (e.g., screwing or gluing), and resources that provide skills for these production processes (e.g., a screwdriver or a glue gun).

To model the requirements for a specific CPPS, engineers follow a *clone-and-own* approach to create ASs for product variants representing *product*, *process*, and *resource* variability of the CPPS. Today, engineers, typically use COTS spreadsheet tools to design ASs, manually identify their commonalities by comparing them, and transfer the commonalities to large component matrices. These component matrices are input to planning CPPS modules that create assembly units of the products. However, manually identifying features in ASs and component matrices is considered error-prone, hard to reproduce, and costly. A systematic, reproducible, and (semi-)automated approach to identify and model *Feature Candidates (FCs)* would thus be a significant improvement and support engineers in reusing (parts of) *Product-Process-Resource Assembly Sequences (PPR ASs)* for designing additional product variants.

In earlier work, we proposed the *superimposed PPR model* [9]. In this paper, we investigate how technology-independent, model-based methods for *Feature Extraction (FE)* can be utilized in the context of PPR modeling to support CPPS engineers in systematically defining PPR model variants. We present a reproducible (semi-) automated four-step *PPR Feature Candidate Identification* approach to identify feature candidates from a set of PPR ASs. We build on the *Feature Candidate Identification* method for source-code-based UML models [14] and adapt it to PPR models for identifying candidates to build a superimposed model from them. This provides an initial variability model to the engineers that represents the commonalities and variability of the Assembly Sequences.

We evaluated the *PPR Feature Candidate Identification* approach in a real-world context, a CPPS recently planned by our industry partner by (a) testing *feasibility* with a use case from the industry partner, (b) comparing the PPR-FCI approach with the traditional manual feature extraction approach, and (c) collecting feedback on its *usefulness* from CPPS engineering domain experts.

VaMoS '20, February 5–7, 2020, Magdeburg, Germany

ACM ISBN 978-1-4503-7501-6/20/02…$15.00
https://doi.org/10.1145/3377024.3377043

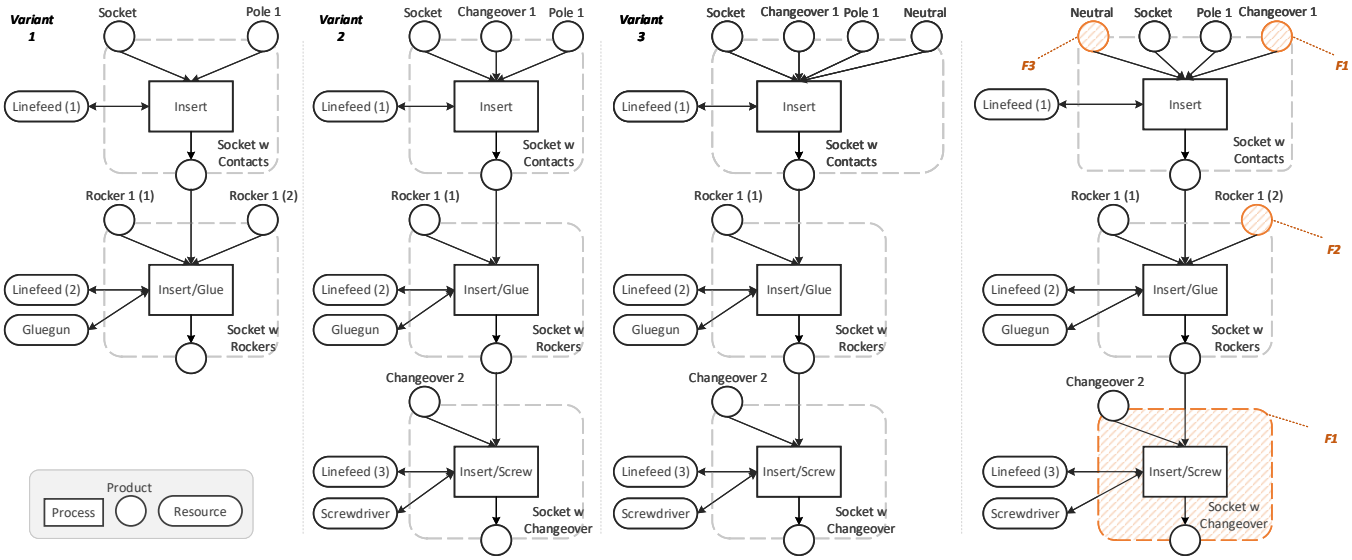

Figure 1: Assembly Sequences of three rocker switch variants and the resulting superimposed model on the rightmost side.

2 RELATED WORK

Manually analyzing and modeling the variability of existing systems is a tedious task. *Feature Extraction and Identification* methods have thus been proposed to automate this activity. They support revealing variability in artifacts like source code or design models.

Cruz et al. [4] investigated various feature location techniques, revealed that textual approaches are of growing interest and evaluated three textual information retrieval methods. Such methods aim at large (natural language) text representations rather than structured text or models.

Most feature extraction methods focus on a particular type of model or on source code written in a particular programming language. For example, Ryssel et al. [11, 12] introduced an approach for automated *feature identification in function-block models*, which are used to program embedded systems, to foster their reuse. Carbonnel et al. [2] investigated two mathematical frameworks for knowledge discovery, i.e., *Formal Concept Analysis* and *Pattern Structures*, to extract complex variability information from relations of product descriptions.

Some feature extraction methods are able to deal with multiple artifacts (types). For instance, Martinez et al. [8] described the tool-supported, generic, and extensible *BUT4Reuse* approach that can extract variability information from a variety of software artifacts.

Ziadi et al. [14] introduced the three-step (semi-automated) *Feature Candidate Identification (FCI)* approach to extract features from source code of software product variants. The approach identifies feature candidates in Unified Modeling Language (UML) models. We build on this FCI approach and adapt it to the context of PPR modeling. Ziadi et al. [14] achieved the feature identification by (a) reverse-engineering source code into a UML class model for each software product variant, (b) decomposing the UML models to create their particular *Construction Primitives (CPRs)*, atomic pieces that represent building blocks of UML models, and identifying features using their FCI algorithm, and, finally, (c) manually pruning irrelevant candidates or adding missed features. Thus, the FCI approach makes use of CPRs that can be considered a technology-agnostic Domain-Specific Language (DSL) for UML models, which we pursue to re-design for PPR models.

3 ROCKER SWITCH PRODUCT LINE

We illustrate our Feature Candidate Identification approach with the running example of a *rocker switch*. *Rocker switches* have one or more rockers[1] for controlling devices like light-switches or sunblinds. Basically a *rocker switch* has a socket with several contacts wired to the electrical devices, the energy source, and the ground, with a corresponding number of rockers and, to activate them, a varying number of switch panels. They can be realized, e.g., as single-pole-single-throw, double-pole changeover, or four-way switches, depending how many device functions a single *rocker switch* controls at once. Depending on the type of switch the order of how the parts are assembled is essential. In Figure 1, the first three diagrams from the left illustrate parts of the Assembly Sequences of three *rocker switch* variants as PPR AS models.

Our industry partner plans a CPPS for manufacturing 12 *rocker switches* variants, each requiring up to 60 steps to assemble the final product resulting in more than 600 Assembly Sequence segments to design, manage, and maintain. Before planning the actual CPPS, the basic engineers have to gain insights on the sequence of manufacturing steps, i.e., the Assembly Sequence, of a particular product variant, often by disassembling the products into their assembly units. Today, our industry partner creates and manages the ASs in spreadsheets, using a *clone-and-own* approach to re-build the particular AS for each variant and manually derive the commonalities and variability of the products. However, the industry partner explores designing a tool that utilizes PPR models for reducing effort and risk in the CPPS engineering process.

[1] Example of a rocker switch: https://commons.wikimedia.org/wiki/File:3977.jpg

```
Base = {
  process('insert','socket w contacts')
  input2output('socket','socket w contacts')
  input2output('pole 1','socket w contacts')
  resource('linefeed (1)','insert','socket')
  resource('linefeed (1)','insert','pole1')
  process('insert/glue','socket w rockers')
  input2output('rocker 1 (1)','socket w rockers')
  input2output('socket w contacts','socket  w rockers')
  resource('linefeed (2)','insert/glue','rocker1 (1)')
  resource('linefeed (2)','insert/glue','socket w contacts')
  resource('gluegun','insert/glue','rocker 1 (1)')
  resource('gluegun','insert/glue','socket w contacts')
}, F1 = {
  input2output('changeover1','socket w contacts')
  resource('linefeed (1)','insert','changeover 1')
  process('insert/screw','socket w changeover')
  input2output('changeover 2','socket w changeover')
  input2output('socket w rockers','socket w changeover')
  resource('linefeed (3)','insert/screw','changeover 2')
  resource('screwdriver','insert/screw','changeover 2')
  resource('linefeed (3)','insert/screw','socket w rockers')
  resource('screwdriver','insert/screw','socket w rockers')
}, F2 = {
  input2output('rocker 1 (2)','socket w rockers')
  resource('linefeed (2)','insert/glue','rocker 1 (2)')
  resource('gluegun','insert/glue','rocker 1 (2)')
}, F3 = {
  input2output('neutral','socket w contacts')
  resource('linefeed (1)','insert','neutral')
}
```

Listing 1: CPRs from Figure 1, clustered to a set of features.

4 FEATURE CANDIDATE IDENTIFICATION FOR PPR ASSEMBLY SEQUENCE MODELS

This section introduces the steps of our *PPR Feature Candidate Identification (PPR-FCI)* approach. Building on the approach by Ziadi et al. [14], our approach follows four steps described below.

4.1 PPR-FCI 1: Design PPR AS models

Domain experts design or collect a set of PPR AS models as part of their usual design tasks in basic engineering (see Figure 1, three models on the left).

4.2 PPR-FCI 2: Derive construction primitives

An expert or algorithm derives the *Construction Primitives* [14] from the set of PPR models coming from Step 1. For the context of CPPS engineering, we designed *PPR Construction Primitives*, atomic elements of a PPR AS model (see online material[2] and Listing 1, lines 2, 3, and 5). Therefore, we designed an initial PPR DSL that provides these construction primitives in a machine-readable format. Following the three aspects of the PPR approach, the main concepts that the DSL represents are *processes*, *resources*, and relations between *products* describing how products are connected.

4.3 PPR-FCI 3: Identify PPR feature candidates

The PPR-FCI algorithm then identifies feature candidates from the set of PPR construction primitives (see PPR-FCI 2). Following the initial idea of Ziadi et al. [14] to cluster the construction primitives to a set of *Feature Candidates*, we build on their FCI algorithm (see the steps in Algorithm 1).

Algorithm 1: PPR-FCI algorithm based on Ziadi et al. [14]

Input: A set PPR AS models $AllM = \{M_1, .., M_n\}$, where each M_x is a set of PPR CPRs
Output: FC, the set of feature candidates of the models

1 $FC = \emptyset$;
2 $R = \{i, cp_j | \exists M_i \in AllM \exists cp_j \in M_i\}$;
3 **while** $R \neq \emptyset$ **do**
4 $\quad mfcp := mostFrequentCP(R)$;
5 $\quad models := \{M_i | M_i \in AllM \wedge mfcp \in M_i\}$;
6 $\quad f := \bigcap_{M_i \in models} M_i \cap R$;
7 $\quad R := R \setminus f$;
8 $\quad FC := FC \cup \{f\}$;
9 **end**
10 **return** FC

The PPR-FCI algorithm initially holds an empty set F for the PPR feature candidates and a set R of all construction primitives of the PPR model variants (see Algorithm 1, lines 1 and 2). In our case, these are the construction primitive sets of the first three PPR model variants of Figure 1. The algorithm then iterates over the set R until it is empty, conducting the following steps.

(1) the algorithm first identifies in each iteration the most frequent PPR construction primitives $mfcp$ of sets in $AllM$. (2) The PPR construction primitives of the models containing the $mfcp$ are united in a set $models$. (3) From this combination, the PPR feature candidates f, which is a set of the most frequent construction primitives in the $models$ set, is extracted and united with the construction primitives remaining in R. (4) The PPR construction primitives of f are removed from R to ensure that they are not used in another feature candidate. (5) Finally, the set f is united as subset with the PPR feature candidate set FC, representing the sets of construction primitives clustered into distinct features.

We developed the *CPR2FC prototype* (see online material), a simple Java-based implementation for executing the algorithm on PPR models represented as text-based construction primitive files, returning a list of feature candidates.

For the rocker switch variants shown in Figure 1, the result of this process is the list of features presented in Listing 1 with a *Base* feature that is used in all three assembly sequences variants and three optional features, *F1* to *F3*. The listing shows, e.g., that the *Neutral* contact is only part of feature *F3*, while feature *F1* includes the *Changeover 1* and the complete process step with *Changeover 2* marked with the hatched pattern in Figure 1.

4.4 PPR-FCI 4: Build superimposed PPR model

In this step, the approach builds a superimposed PPR model [9] from the set of identified feature candidates coming from PPR-FCI Step 3. We, therefore, developed the *superimposed PPR model* algorithm (see online material) that constructs a graph of the superimposed PPR model (see rightmost model in Figure 1) from the set of PPR

[2]Further material, like the DSL Definition and the Java prototypes, is available online: https://github.com/tuw-qse/ppr-fci

feature candidates identified (see Listing 1). To include the representation of the features, differently colorized nodes replace the feature annotations in the resulting superimposed model.

To evaluate the construction step of the superimposed model, we developed the Java-based *FC2SI* (see online material) prototype. This prototype executes the SI PPR algorithm on the feature candidates identified in the previous step, facilitates *JGraphT*[3] to build the graph, and exports it to the popular *GraphViz DOT* [4] notation, which tools like, e.g., *WebGraphViz*[5] can easily visualize.

5 EVALUATION AND DISCUSSION

Our industry partner provided the use case data (see Section 4.1) as a spreadsheet. Due to non-disclosure agreements the authors obfuscated the particular evaluation set data provided in this work.

To evaluate the feasibility, we compared our PPR-FCI approach to the traditional manual approach, together with two senior domain experts of our industry partner. One expert was a planning engineer with longstanding experience in planning product variants and CPPS, the other was an expert for CPPS configuration. In *PPR-FCI 1*, one of the authors created the PPR models from the *rocker switch* evaluation set in correspondence with the industry partner and discussed their validity with the domain experts. The models in Figure 1 represent a small subset of the evaluation set. Guided by the domain experts, the researchers analyzed valid features from the set of PPR AS variants in the study context. In *PPR-FCI 2*, the researchers translated the model variants to the construction primitives in the DSL. We conducted this step manually, but plan to develop a suitable supporting tool as part of our future work. *Before starting PPR-FCI 3*, domain experts, guided by researchers, manually created a superimposed PPR model to create an empirical evaluation baseline for the study context. In *PPR-FCI 3*, we ran the *CPR2FC prototype* on the construction primitives to identify the feature candidates. In *PPR-FCI 4*, the researchers ran the *FC2SI* prototype to create a superimposed PPR model from the feature candidates, and compared it to the manually created model.

In comparison to the traditional manual approach, our approach (a) *effectively* found the same features that domain experts extracted, (b) was *reproducible*, which does not leave the feature identification to the skills of the domain expert, and (c) *reduced errors* that occur in the manual approach, in particular, due to copy-and-adapt errors in the large spreadsheets that are hard to verify [3].

The researchers interviewed the domain experts on the perceived usability and *usefulness* [5] of the results of the PPR-FCI approach, in particular the superimposed PPR model. Overall the domain experts found the approach (a) *usable*, specifically the FCI prototype making the process reproducible and more efficient, stating that *"this approach could save us a lot of time and prevent many* copy & paste *errors"* and (b) *useful*, in particular the superimposed PPR AS model as it facilitates and eases the analysis and discussion of which product variants can be produced on a CPPS. One expert commented that *"the [superimposed] model allows us to better discuss requirements and issues of variants with colleagues and customers."*

Discussion of Results. In this work, we aimed at extracting features from existing PPR models to build a basic PPR variability model that systematically captures the variability of products, production processes, and related resources. We found the suitable definition of construction primitives for a particular model type to be the key enabler for using the FCI algorithm [14] in the PPR domain. Therefore, model information, such as relations to attribute lists, needs to be deconstructed to multiple primitives describing single instances, as a prerequisite to the correct algorithmic extraction of feature candidates. Fortunately, we were able to identify promising primitive concepts in PPR models. Our evaluation results confirmed the feasibility of the adaptation of the FCI approach to the context of PPR models. We, furthermore, investigated to what extent the PPR-FCI approach supports CPPS engineers together with domain experts from our industry partner. Domain experts appreciate the variability representation in a type of model they know and understand, which would advocate for visual model enrichment approaches [7, 9]. However, we also found that dependencies in the superimposed model were often not "visible enough" impeding reasoning on these models and that it misses functionality, e.g., formulas, that domain experts use regularly in spreadsheets.

Limitations. Several tasks of the evaluation were conducted by the researchers, which we recognize as a threat to validity. However, to mitigate this threat, during these phases the authors were in close contact with domain experts at the industry partner to gather feedback. Further, the empirical study was limited to a set of product variants and domain experts of one industry partner. However, our industry partner claimed that the rocker switch sample is *"representative for a considerable range of industrial products including those with a relatively large share of variability."*

6 CONCLUSION AND FUTURE WORK

Product-Process-Resource Assembly Sequence models of product variants manufactured by a CPPS are primary engineering artifacts, often created using *clone-and-own* approaches. In this paper, we investigated which existing feature extraction approaches can be adapted to PPR models and introduced a four-step approach for *PPR Feature Candidate Identification* in PPR model variants based on the FCI approach Ziadi et al. [14]. We constructed a DSL for PPR models representing construction primitives and created a superimposed model from the identified candidates to support practitioners. We evaluated our approach based on the example of a set of *rocker switch* variants from an industry partner and showed that the approach is able to identify features relevant to domain experts.

Future Work includes extending a model-based design tool towards a PPR tool to create models and identify common features. Further research is also required to improve the algorithm in terms of stability to refactorings. Finally, we plan to extend the empirical evaluation to a larger setting including several product types.

ACKNOWLEDGMENTS

The financial support by the Austrian Federal Ministry for Digital, Business and Enterprise and the National Foundation for Research, Technology and Development is gratefully acknowledged. We gratefully acknowledge the support of Philip Liszt, who developed a prototype of the adapted FCI algorithm.

[3] JGraphT: https://jgrapht.org
[4] GraphViz: https://graphviz.org
[5] WebGraphViz: http://webgraphviz.com/

REFERENCES

[1] Stefan Biffl, Detlef Gerhard, and Arndt Lùder. 2017. Introduction to the Multi-Disciplinary Engineering for Cyber-Physical Production Systems. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Springer, 1–24.

[2] Jessie Carbonnel, Marianne Huchard, and Clémentine Nebut. 2019. Towards complex product line variability modelling: Mining relationships from non-boolean descriptions. *Journal of Systems and Software* 156 (2019), 341–360.

[3] Chris Chambers and Chris Scaffidi. 2010. Struggling to Excel: A Field Study of Challenges Faced by Spreadsheet Users. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*. 187–194.

[4] Daniel Cruz, Eduardo Figueiredo, and Jabier Martinez. 2019. A Literature Review and Comparison of Three Feature Location Techniques Using ArgoUML-SPL. In *Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems (VAMOS '19)*. ACM, New York, NY, USA, 16:1–16:10.

[5] Fred D. Davis. 1989. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly* 13, 3 (1989), 319–340.

[6] Volkan Gunes, Steffen Peter, Tony Givargis, and Frank Vahid. 2014. A survey on concepts, applications, and challenges in cyber-physical systems. *KSII Transactions on Internet & Information Systems* 8, 12 (2014).

[7] Sascha Lity, Sophia Nahrendorf, Thomas Thüm, Christoph Seidl, and Ina Schaefer. 2018. 175% Modeling for Product-Line Evolution of Domain Artifacts. In *Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS 2018, Madrid, Spain, February 7-9, 2018*, Rafael Capilla, Malte Lochau, and Lidia Fuentes (Eds.). ACM, 27–34.

[8] Jabier Martinez, Tewfik Ziadi, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2017. Bottom-up Technologies for Reuse: Automated Extractive Adoption of Software Product Lines. In *Proceedings of the 39th International Conference on Software Engineering Companion (ICSE-C '17)*. IEEE Press, Piscataway, NJ, USA, 67–70.

[9] Kristof Meixner, Rick Rabiser, and Stefan Biffl. 2019. Towards Modeling Variability of Products, Processes and Resources in Cyber-Physical Production Systems Engineering. In *Proceedings of the 23rd International Systems and Software Product Line Conference, SPLC 2019, Volume B, Paris, France, September 9-13, 2019*. ACM, 68:1–68:8.

[10] László Monostori. 2014. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP* 17 (2014), 9–13.

[11] Uwe Ryssel, Joern Ploennigs, and Klaus Kabitzsch. 2010. Automatic Variation-point Identification in Function-block-based Models. In *Proceedings of the Ninth International Conference on Generative Programming and Component Engineering (GPCE '10)*. ACM, New York, NY, USA, 23–32.

[12] Uwe Ryssel, Joern Ploennigs, and Klaus Kabitzsch. 2012. Automatic library migration for the generation of hardware-in-the-loop models. *Science of Computer Programming* 77, 2 (2012), 83 – 95. Special Issue on Automatic Program Generation for Embedded Systems.

[13] Miriam Schleipen, Arndt Lüder, Olaf Sauer, Holger Flatt, and Jürgen Jasperneite. 2015. Requirements and concept for Plug-and-Work. *at-Automatisierungstechnik* 63, 10 (2015), 801–820.

[14] Tewfik Ziadi, Luz Frias, Marcos Aurélio Almeida da Silva, and Mikal Ziane. 2012. Feature Identification from the Source Code of Product Variants. In *2012 16th European Conference on Software Maintenance and Reengineering*. 417–422.