Towards a Framework for the Analysis of Multi-Product Lines in the Automotive Domain

Shaukat Ali
Simula Research Laboratory
Oslo, Norway
shaukat@simula.no

Paolo Arcaini
National Institute of Informatics
Tokyo, Japan
arcaini@nii.ac.jp

Ichiro Hasuo
National Institute of Informatics
Tokyo, Japan
i.hasuo@acm.org

Fuyuki Ishikawa
National Institute of Informatics
Tokyo, Japan
f-ishikawa@nii.ac.jp

Nian-Ze Lee
National Taiwan University
Taipei, Taiwan
nianzelee@gmail.com

ABSTRACT

Safety analyses in the automotive domain (in particular automated driving) present unprecedented challenges due to its complexity and tight integration with the physical environment. Given the diversity in the types of cars, potentially unlimited number of possible environmental and driving conditions, it is crucial to devise a systematic way of managing variability in hazards, driving and environmental conditions in individual cars, families of cars, and families of families of cars to facilitate analyses efficiently. To this end, we present our ongoing work in a research project that focuses on devising a model-based reasoning framework for systematically managing hazards in the automotive domain and supporting safety analyses (e.g., falsification).

CCS CONCEPTS

• **Computing methodologies → Model verification and validation**; • **Software and its engineering → Software product lines**;

KEYWORDS

Product Lines, Simulink, Falsification, Automotive domain, Hazard analysis

ACM Reference Format:
Shaukat Ali, Paolo Arcaini, Ichiro Hasuo, Fuyuki Ishikawa, and Nian-Ze Lee. 2019. Towards a Framework for the Analysis of Multi-Product Lines in the Automotive Domain. In *13th International Workshop on Variability Modelling of Software-Intensive Systems (VAMOS '19), February 6–8, 2019, Leuven, Belgium.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3302333.3302345

1 INTRODUCTION

In the automotive domain, cars, families of cars, and even families of families of cars, share common features, hazards, environmental conditions, and driving conditions. Thus, there is a need of a systematic way to manage commonalities and variability to support safety analyses (e.g., falsification) systematically not only for one car but also for families of cars, and families of families of cars efficiently. This becomes even more important in the field of automated driving.

The main application domain of the ERATO MMSD project [12] is the automotive one. To this end, we have established a collaboration with an industrial partner in the automotive domain about safety, and a collaboration with a group in Waterloo working on automated driving[1]. Discussing with our industrial partner and the Waterloo group, we realized that the systems we have to consider are Product Lines (PL), as they are affected by different sources of variability.

In the context of automated driving, the Waterloo group identified different kinds of variability that are relevant for the domain (8 documents in [9]), for example, related to the road type (e.g., local road or freeway), road surface (e.g., bituminous surface or Portland cement concrete surface), and the environment (e.g., Beaufort scale of wind condition) [9]. Moreover, based on discussions with both partners, we found two other sources of variability:

- variability in the functionality of families of cars, e.g., the type of braking system varies in different car models;
- variability in the built-in safety features in families of cars, e.g., a feature to automatically control a car on a slope may exist or not in a car model.

Therefore, we have to deal with a *multi-product line* (multi-PL) [13] where variability occurs in individual cars, families of cars, relationships across families, environmental conditions, and driving conditions. Note that, on purpose, we model the environment as part of the multi-PL, as this facilitates the design of the analysis techniques we propose.

When considering safety in the automotive domain, current verification strategies usually consider one product (car) at a time in a given environmental condition (e.g., on a given road with a given weather condition).[2] However, this way of analysis does not allow to easily understand whether a feature is particularly relevant (either positively or negatively) for a safety concern. Having such

[1] https://www.autonomoose.net/
[2] In a Simulink model, this means that all parameters' values are fixed.

kind of knowledge would be extremely important for designers who could estimate which is the impact of a particular feature on the overall safety. Indeed, cars from the same family or across different families share commonalities that can help manage safety hazards systematically. For example, a hazard may be less probable (under certain environmental conditions) for a set of cars sharing functional and specialized features with the same configurations. Consider, for example, the Emergency Brake Assist (EBA) feature. If the designers observe that, whenever EBA is present, the probability of accidents is significantly reduced, they can estimate the importance of the feature.

Therefore, our position is that considering the variability for various safety analyses may have some advantages [19]. First of all, it would permit to better understand whether in the multi-PL some feature is more/less important for the safety property under consideration. Moreover, it could also permit to achieve a better scalability, as commonalities in the different products could be exploited in the safety analysis.

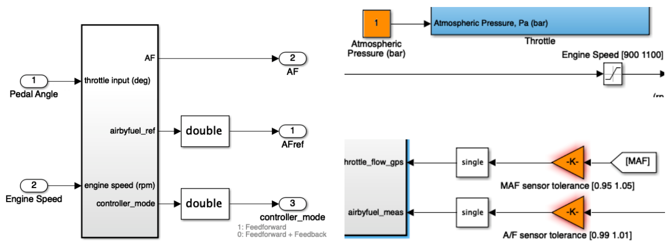
A common means for reasoning about the safety of hybrid systems (typical of the automotive domain) is by using Simulink[3] models[4]. As their formal verification is not feasible, current research focuses on *falsification* [15, 22]; given a model $\mathcal{M}$ and a specification φ (a signal temporal logic (STL) formula), falsification tries to find an input signal u making the formula false, i.e., $\mathcal{M}(u) \not\models \varphi$.

Verification of multi-PLs have been studied in the context of delta-programming and feature-based programming [7, 20]. However, to the best of our knowledge, the analysis (by falsification) of multi-PLs of hybrid models has not been considered so far.

In this paper, we describe our long-term project to raise the verification/validation of hybrid models at the level of multi-PL. In particular, we devise three research directions:

(1) falsification of the entire multi-PL. The challenge here is to search a falsifying input in the whole search space of the multi-PL. Indeed, it could be that a safety property is guaranteed by a given product of the multi-PL (e.g., a car model A having a safety feature S), but it is easily falsified in another product (e.g., another model B without S).
(2) even if the model under consideration does not falsify a given safety property, this does not mean that it can be considered completely safe. In falsification, the notion of *robustness* says how close we are to falsify the property when executing a given input u, i.e., the higher the robustness, the safer is the system in this simulation. Therefore, we consider robustness as a proxy of safety guarantee. Knowing which features are crucial in obtaining high/low robustness is extremely important, as this can be used to support design/business decisions (i.e., whether to invest or not on a safety feature that is costly but that provides good guarantees in terms of safety). We plan to build an approach based on robustness evaluation that is able to assign “safety scores” to features.
(3) On the top of the results of the previous phase, we also plan to devise a querying framework in which the designer can pose “safety questions”, as, for example, “which is (are) the most important feature(s) for guaranteeing safety?”.

[3]Simulink™ is a registered trademark of the MathWorks Inc.
[4]As a matter of fact, Simulink is one of the formalisms used in the “Applied Verification for Continuous and Hybrid Systems” competition [11].



(a) Black box view (b) Excerpt of detailed view

Figure 1: Abstract Fuel Control System (taken from [14])

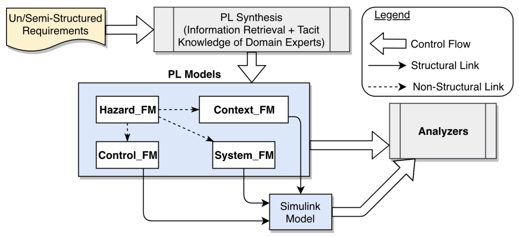


Figure 2: Overview of the Framework

The paper is structured as follows. Sect. 2 introduces the running example we use in the paper. Sect. 3 presents the proposed approach, Sect. 4 discusses some related work, and Sect. 5 concludes the paper identifying the main research challenges.

2 RUNNING EXAMPLE

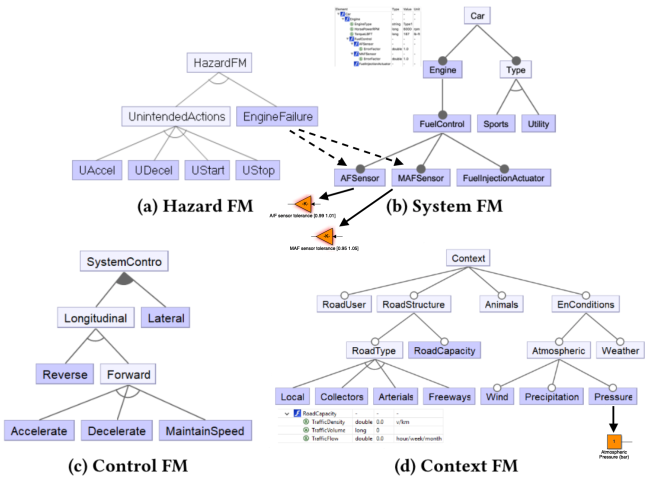
To illustrate our research plan, we use the *Abstract Fuel Control System* model proposed in [14] as falsification benchmark. Since the model is publicly available, it is suitable for demonstration purposes, differently from the models of our industrial partner. Moreover, although the model only represents a part of the car, it still contains some variability that allows to explain our ideas.

The model is shown in Fig. 1. As shown in the black-box view in Fig. 1a, the model takes as *input signals* the pedal angle and engine speed, that range in intervals [0,61.2) and [900,1000]. The model outputs the signal air-fuel ratio (AF), which influences fuel efficiency and car performance. The value is expected to be close to a reference value *AFref* (this is the safety property we check).

The model contains some *parameters* (in orange in the detailed view in Fig. 1b). For example, the *A/F sensor tolerance* and the *MAF sensor tolerance* specify the precision of the sensors’ measurements, while another parameter specifies the atmospheric pressure. Assigning values to all these parameters means selecting one particular product (i.e., one particular model of AFC system) and testing it in a particular environmental condition.

3 PROPOSED FRAMEWORK

Our proposed framework has two main components (see Fig. 2), i.e., *Models* and *Analyzers*. We aim to develop a set of multi-product line models (see Sect. 3.1) through the Product Line (PL) Synthesis process. The process is semi-automatic: it mines variabilities/commonalities in hazards and their relationships with features



(a) Hazard FM (b) System FM (c) Control FM (d) Context FM

Figure 3: Feature models

of cars, system control, and context (e.g., environment), using information retrieval from textual requirements, manuals, logs, incident reports of hazards (images, audios, or videos). The process also improves the models with the tacit knowledge of automotive domain experts. The developed models will be a set of feature models (see Fig. 2 and Sect. 3.1). These feature models will have relationships among them (shown as non-structural links in Fig. 2) and with underlying Simulink models (shown as structural links in Fig. 2) used for safety analyses.

As said in Sect. 3.2, analyzers take as input the feature models and perform different types of analyses: we want to perform falsification at the level of multi-PL, derive "safety scores" for features, and use these scores for reasoning about the multi-PL safety.

3.1 Modeling Solution

We propose to use feature models as modeling solution as abstraction layer on top of the Simulink models (Fig. 1). However, other notations for modeling variability will be used, if required. To explain our ideas, we use the example of multi-PL shown in Fig. 3.

One key problem is how to build the initial feature models (PL Models in Figs. 2 and 3). Based on our understanding of the current practice, we found that variability is present in various documents such as in car specifications and requirements. Therefore, we plan to devise a technique to (semi)-automatically extract features and constraints from car specifications, hazard requirements, etc. We conducted a preliminary experiment by manually inspecting the specifications of a family of cars available online. We observed that it is possible to build the model starting from these documents. A similar experiment on a document describing hazards in the automotive domain [10] showed that, also in this context, it is possible to represent variability. Regarding the context, variability is described in the documents produced by the Waterloo group about automated driving [9]. We plan to apply standard techniques from information retrieval (e.g., thesauri) to give a standard representation of input documents (specifications, requirements) and then extract feature models from these. To build feature models, some techniques are already available (e.g., [4]), but require an even more structured form for the input documents.

There are four feature models (see Figs. 2 and 3): Hazard_FM, Context_FM, System_FM, and Control_FM. Features in these feature models may have *non-structural links* between them (shown as dashed arrows in Figs. 2 and 3). These links are the logical connections between features that are relevant for hazards being analyzed and may be known or unknown at the time of analysis. If the non-structural links are known, they may have been identified based on specifications, based on the available data (e.g., operational data, testing data), or specified by a stakeholder (e.g., developer). However, in this work we assume that these non-structural links are not known, and one of the aims of the analyses described in the next section is indeed to discover them.

Instead, the *structural links* (shown as concrete arrows in Fig. 2 and 3) connect features from all the feature models, except Hazard_FM, with the configuration parameters (representing variability) in the underlying Simulink model. Modeling variability in Simulink models is studied in the literature (e.g., in [21]) and also implemented in tools, e.g., pure::variants. From our industrial partner, we already have some Simulink models that have parameters that can be seen as configurations points (as the sensors' tolerance and the atmospheric pressure in the running case study). Therefore, among the different ways that can be used to model variability in Simulink models [21], we now consider *data variability* that comes from the values assigned to *constant blocks* (orange squares in Fig. 1) and *gain blocks* (orange triangles in Fig. 1); however, also other means for expressing variability [21], as *model variant blocks* or *variant subsystems*, will be considered if needed.

Hazard_FM captures variability in hazards and their relationships. An example of such a feature model is shown in Fig. 3 (a). This feature model captures two hazards, i.e., hazards related to *UnintendedActions* and *EngineFailure*. The *UnintendedActions* hazard is due to unintended acceleration, deceleration, start, or stop of the car shown as four concrete features in Fig. 3 (a). Note that *EngineFailure* has some non-structural links with the other feature models that, however, we do not know in advance and we want to discover with the analysis described in Sect. 3.2.2.

Context_FM captures features of the context of car, including the environment. Note that we extend the classical definition of feature [6] also to the environment (instead of considering it part of the input space). We claim that this can help in devising and reasoning about different testing/simulation conditions; the multi-PL, therefore, represents both the product line of cars and the variability of the environment. An example of this feature model is shown in Fig. 3 (d). The context of a car includes *RoadUser*, *RoadStructure*, *Animal*, and Environmental Conditions (*EnvConditions*). The example is created based on the detailed documentation available at [9]. The *pressure* feature is connected through a structural link with the *atmospheric pressure* parameter of the Simulink model (for the sake of space, we only report the Simulink constant block in orange).

System_FM captures variability in features of families of cars. Fig. 3 (b) shows an example of such a feature model that shows that, for instance, *Engine* is a mandatory feature of a car. The engine further has the *FuelControl* feature that consists of two sensors (i.e., *AFsensor* and *MAFSensor*), and one actuator (i.e., *FuelInjectionActuator*). Some features can have attributes, e.g., both sensors have attribute *ErrorFactor* representing the error in sensor measurements. The *EngineFailure* from Hazard_FM is linked (with non-structural links) to the *AFsensor* and *MAFSensor* features signifying that engine failure hazard is related to the accuracy of the sensors (also these links will be discovered with the proposed analyses techniques). These two same features are connected through structural links with two parameters of the Simulink model (two gain blocks).

Control_FM captures variability in system control by a driver to control a car (e.g., acceleration, torque). Fig. 3 (c) shows an example of Control_FM that captures various control features (classified into *Longitudinal* and *Lateral*). Note that this example is created based on the documentation provided in [9].

There are two keys problems in establishing structural and non-structural links. The structural links (black full arrows in Figs. 2 and 3) can be established based on the available specifications, or with the help of developers. Another problem is how to extract the non-structural links between the Hazard_FM and the Control_FM, Context_FM, and System_FM (black dashed arrows in Figs. 2 and 3) describing how a given hazard is affected by the features of the context, the system, and the control. Our proposal is to assign safety scores to features by performing different simulations and recording their robustness, as explained in Sect. 3.2.2.

3.2 Applications

The aim of our project is to exploit variability modeling for safety analyses. Exploiting the multi-PL models defined in Sect. 3.1, we plan to perform falsification in different ways. In Sect. 3.2.1, we explain how we plan to extend the falsification problem from a single product to the whole multi-PL. Sect. 3.2.2, instead, describes our proposal of understating the influence of each feature on the safety and building the non-structural links of Figs. 2 and 3. Finally, Sect. 3.2.3 illustrates the querying framework we devised for reasoning on the safety of the multi-PL. In the following, we assume that one particular hazard h of Hazard_FM has been selected and that all the described analyses techniques are performed for h.

3.2.1 *Falsification of the multi-PL.*

The first aim of our project will be to extend the classical falsification problem for one given product to the falsification of a product family. The idea is that some products are *safer* than others and some environmental conditions can be causing (or favouring) effects for the hazard under consideration. Therefore, it could be that some products are more difficult to falsify than others. For example, if a car model A contains the ABS feature and a model B does not, it can be easier to find an input falsifying a safety property related to aquaplaning in B rather than in A; and, of course, to falsify the safety property, the road must be wet. Note that, in general, understanding which combination of features is more likely to be falsified is not easy.

As already said in Sect. 3.1, such variability is encoded as constant values for the parameters of the Simulink model. A naïve solution could be to turn these parameters in inputs and perform the falsification over the whole solution space. However, this approach would not scale. Our position is that describing and fixing the variability before falsification has some advantages:

- it permits to impose constraints among the parameters, so that invalid products are not experimented. For example, in our running example, the better sensors with high precision are only available on sport cars;
- it allows to discretize the input space of the variable parameters, thus achieving scalability;
- it permits to reason on commonalities and differences among the products and so to drive the search in a *smart* way.

Table 1: Robustness values

Product	Robu.	Time (s)
p_1 = {Pressure=1, Sports, MAFSensor=1%, AFsensor=1%}	0.37	171
p_2 = {Pressure=1, Utility, MAFSensor=3%, AFsensor=3%}	-0.002	46

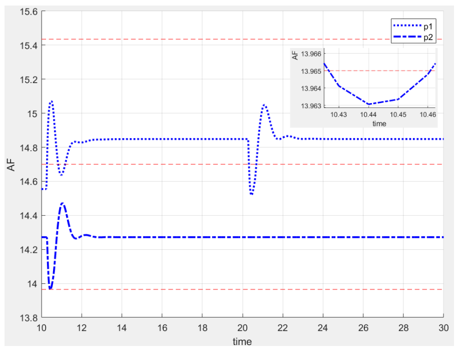


Figure 4: Output signals of p_1 and p_2 (the snapshot shows the falsified part of p_2 signal)

Example. Let's consider the running example AFC. We want to falsify the STL property $\square_{[10,30]} \frac{|AF-AFref|}{AFref} < 0.05$ (with $AFref = 14.7$) stating that the A/F ratio must remain close (±5%) to the reference AFref in the time interval [10, 30]. Table 1 shows two products, the robustness values we obtained by running the falsification tool Breach[5] with the default limit of 100 simulations, and the execution time. Product p_1 represents the AFC of a sport car having precise sensors with 1% of error tolerance. Product p_2, instead, represents the AFC of an utility car whose sensors are less precise (3% of error tolerance). In both cases, the environmental setting is the same, i.e., pressure is 1 bar.[6] Fig. 4 reports the output signals of the two experiments. The figure also reports (as red lines) the boundaries in which the property is satisfied and the reference value. We observe that having good sensors increases the safety, as we have not been able to falsify the STL property for p_1 (we reached a minimum value of robustness of 0.37 in 171 seconds). On the other hand, with less precise sensors, the AFC is less safe, as demonstrated by the fact that the property has been falsified for p_2 (in 46 secs, we found a robustness value of -0.002).

3.2.2 *Understanding Feature Influence on Hazards.*

The approach in Sect. 3.2.1 tries to efficiently find a falsifying input over the whole solution space. Although this is extremely important for understanding whether a safety problem is present in the system, it does not completely allow to understand which feature is more important

[5]https://github.com/decyphir/breach

[6]For conciseness, we do not report other features constituting the product but that do not affect the Simulink model.

for guaranteeing safety. Even if the safety property is not falsified, the system can not be considered completely safe, as there could be some system configuration that is *close* to falsify the property; therefore, robustness can be considered as a proxy for measuring the confidence on the safety of a given system configuration.

Knowing the relation between the presence/absence of a given feature and the obtained robustness, can be helpful for taking design or business decisions; e.g., the company can decide to invest on a given car feature because it gives a big return in terms of safety.

To discover these relationships, we plan to apply search-based techniques for finding different system configurations and measuring their robustness. The search could be guided by different coverage criteria of the variability space, but also coverage criteria on the output, as in [18]. At the end of the search, we should have a representative set of system configurations from which to extract "safety scores" for the features. To extract these scores, different statistical approaches (e.g., regression analysis) will be experimented.

Note that, as outcome of this phase, we will be able to establish the non-structural links in Figs. 2 and 3 that show how the features in the different feature models are related w.r.t. a given hazard. In the running example, for instance, we could discover that there are non-structural links between the EngineFailure feature in Hazard_FM and the AFSensor and MAFSensor features in System_FM.

Existing approaches. Note that there already exist some approaches that could be used for our aims, as the sensitivity analysis in Matlab, or the search on the parameters of Breach. However, such approaches suffer from two drawbacks. First, they do not scale with big systems having a lot of parameters. Second, they do not have a way for describing constraints among the parameters and so they may execute also invalid configurations (e.g., low quality sensors on sport cars) that need to be discarded at the end. As already observed in Sect. 3.2.1, reasoning at the level of multi-PL allows to reason on the commonalities and differences of the different products, and so it possibly allows to select a subset of *meaningful* products.

3.2.3 Querying. We here aim at building a query framework to be used for reasoninig on the multi-PL. The underlying knowledge will come from the results of the approach presented in Sect. 3.2.2.

We aim at answering the following types of questions:

- Which cars are more exposed to a given hazard?
- Which hazards is a given car more exposed to?
- Which environmental conditions make crashes more likely?

In this phase, we also plan to build a model that would allow the stakeholders to reason on the economic impact of each feature. Assigning to each feature an economic cost and a "safety score", we can calculate the *return on investment.* For example, we could discover that the high quality sensor of our running example, although very costly, provides high guarantees in terms of safety.

4 RELATED WORK

The literature on multi-product lines is vast and it also comprises different similar notions. For example, recently the notion of multi-SPL has been used to model and reason on the full sets of packages of the Gentoo Linux distribution [17]. We refer the reader to [13] for a survey of the area. We here mainly relate our approach to alternative approaches for validation in the automotive domain, and with verification approaches for SPLs.

Testing Advanced Driver Assistance Systems (ADASs). Works on testing ADASs are related to us, e.g., by Abdessalem et al. [1–3]. The first two works [1, 2] focus on finding the most critical scenarios and regions for testing ADASs and consider environment attributes such as road conditions and weather conditions. Another work [3] focuses on finding feature interactions that can lead to failures in autonomous cars. These works focus on testing one product at a time. With our framework, instead, we aim to exploit commonalities across families of products to support safety analyses not only for one car but families of cars.

Variability in Matlab/Simulink. Variability in Simulink has been studied in the literature (e.g., in [21]). Such variability modeling is also supported by tools, e.g., pure::variants. Kolassa et al. [16] compared two variability concepts implemented in Matlab/Simulink with a controlled experiment with developers at Volkswagen AG. The results found that both compared approaches were feasible options for the developers. With our modeling solution, we aim to provide an abstraction layer on top of the variability in Simulink models to manage the system complexity. Moreover, with such abstraction layer, we aim to provide support for various analyses on multi-product lines at a higher level of abstraction, and a user does not need to go into detailed Simulink models for their analyses.

Verification of SPLs. Different approaches have been proposed for the verification of SPLs, but not related to hybrid models and falsification. The approach in [7] tries to reduce the effort of verification of Java SPLs described using Delta-oriented programming. Specifications are written in the Java Modeling Language (JML) and verified with the KeY theorem prover. The approach considers specifications using the delta language and shows how proof-slicing and similarity-guided proof can be used to facilitate the verification. Differently from [7], our approach does not consider different specifications for different products, as the considered safety specification must hold for all the products. The approach in [20] verifies JML specifications of a Java SPL (specified in feature-oriented programming). The approach consists in transforming the Java program in a *metaproduct* describing the behavior of all the products and in a *metaspecification* describing the expected behavior of all products. The verification performed with the KeY theorem prover has shown to be much more efficient than the verification done at the product level. Our aim is similar, as we want to exploit variability description to achieve scalability in falsification. Other approaches, instead, perform classical model checking on the SPL model, by building comprehensive models of the whole SPL, as in [8] where *featured transition systems* are used. Beohar and Mousavi [5] propose an approach for testing SPLs using input-output featured transition system (IOFTS), an extension of featured transition systems that distinguish between input and output actions. The approach is similar to ours in using a behavioral model for describing the behavior of the SPL. However, we mainly focus on verification (through falsification), while they apply conformance testing based on the IOCO-theory.

5 CONCLUDING REMARKS

This paper describes our research plan to build a framework for reasoning about hazards in the automotive domain by considering the variability that arises from different sources. We identify the following as the major research challenges of our project.

Modeling. Our framework assumes that the variability in the environment, in the car, and in the car control are modeled in terms of feature models. However, these feature models must be built starting from requirements documents, car specifications, documents about hazards, that are written in natural language and do not have a defined structured. To semi-automatically derive the feature models (or at least some preliminary versions) from this plethora of documents, we plan to investigate informational retrieval (IR) techniques and collaborate with IR experts.

Structural linking. One challenge is how to establish the structural links (see Fig. 2) between features in the Control_FM, Context_FM, and System_FM feature models and underlying configuration parameters in the Simulink model. This linking could be partially discovered automatically (e.g., with name matching). In some cases, it would also require the domain knowledge of the designers who know what a parameter corresponds to. We need to devise new methods to help developers establishing such links.

Search. Both the falsification over the multi-PL (see Sect. 3.2.1) and the computation of the safety scores (see Sect. 3.2.2) require search over the products input space, select some of them to simulate, and record their robustness values. Selecting the minimum set of meaningful products will make the approach scalable and still efficient. To this aim, search-based approaches will be pursued, not only considering input coverage, but also output coverage [18].

Analysis. The approach in Sect. 3.2.2 aims at extracting "safety scores" from the set of generated input sequences and corresponding outputs. To this aim, we plan to exploit statistical (e.g., regression analysis) and machine learning approaches (e.g., classification methods). However, we need to investigate more around this area.

Results presentation. The results about the relation between features and hazard will have to be communicated to the stakeholders (e.g., our industrial partner) who can use them to take design/business decisions. At this stage, we devised the notion of safety scores and non-structural links between feature models; however, the informativeness of these artifacts will be studied and possibly more advanced ways of showing the results will be pursed.

Our research plan focuses on functional testing, as we aim to assess the safety of the car model w.r.t. hazards. However, we can also consider non-functional requirements, and adapt the devised framework for non-functional analysis. The multi-PL component will be similar, but the simulation model should be different. For example, instead of a Simulink model, we could have a queuing network describing the service response time of the system.

ACKNOWLEDGMENTS

S. Ali is supported by the Research Council of Norway (RCN) under MBT4CPS project (grant no. 240013/O70). The other authors are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST. We thank our industrial partner Mazda and Prof. Czarnecki for the fruitful discussions.

REFERENCES

[1] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter. Testing advanced driver assistance systems using multi-objective search and neural networks. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ASE 2016, pages 63–74, New York, NY, USA, 2016. ACM.

[2] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, pages 1016–1026, New York, NY, USA, 2018. ACM.

[3] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, pages 143–154, New York, NY, USA, 2018. ACM.

[4] M. Acher, A. Cleve, G. Perrouin, P. Heymans, C. Vanbeneden, P. Collet, and P. Lahire. On extracting feature models from product descriptions. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '12, pages 45–54, New York, NY, USA, 2012. ACM.

[5] H. Beohar and M. R. Mousavi. Input-output conformance testing for software product lines. *Journal of Logical and Algebraic Methods in Programming*, 85(6):1131–1153, 2016. NWPT 2013.

[6] T. Berger, D. Lettner, J. Rubin, P. Grünbacher, A. Silva, M. Becker, M. Chechik, and K. Czarnecki. What is a feature?: A qualitative study of features in industrial software product lines. In *Proceedings of the 19th International Conference on Software Product Line*, SPLC '15, pages 16–25, New York, NY, USA, 2015. ACM.

[7] D. Bruns, V. Klebanov, and I. Schaefer. Verification of software product lines with delta-oriented slicing. In *Proceedings of the 2010 International Conference on Formal Verification of Object-oriented Software*, FoVeOOS'10, pages 61–75, Berlin, Heidelberg, 2011. Springer-Verlag.

[8] A. Classen, P. Heymans, P.-Y. Schobbens, A. Legay, and J.-F. Raskin. Model checking lots of systems: Efficient verification of temporal properties in software product lines. In *Proc. of the 32Nd ACM/IEEE International Conference on Software Engineering - Vol. 1*, ICSE '10, pages 335–344, New York, NY, USA, 2010. ACM.

[9] K. Czarnecki. WISE drive: Requirements analysis framework for automated driving systems. Technical report, Waterloo Intelligent Systems Engineering Lab (WISE), University of Waterloo, 07 2018. https://uwaterloo.ca/waterloo-intelligent-systems-engineering-lab/projects/wise-drive-requirements-analysis-framework-automated-driving.

[10] S. Dobi, M. Gleirscher, M. Spichkova, and P. Struss. Model-based hazard and impact analysis. *CoRR*, abs/1512.02759, 2015.

[11] A. Dokhanchi, S. Yaghoubi, B. Hoxha, G. Fainekos, G. Ernst, Z. Zhang, P. Arcaini, I. Hasuo, and S. Sedwards. ARCH-COMP18 category report: Results on the falsification benchmarks. In G. Frehse, editor, *ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 54 of *EPiC Series in Computing*, pages 104–109. EasyChair, 2018.

[12] I. Hasuo. Metamathematics for systems design. *New Generation Computing*, 35(3):271–305, Jul 2017.

[13] G. Holl, P. Grünbacher, and R. Rabiser. A systematic review and an expert survey on capabilities supporting multi product lines. *Information and Software Technology*, 54(8):828 – 852, 2012. Special Issue: Voice of the Editorial Board.

[14] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts. Powertrain control verification benchmark. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 253–262, New York, NY, USA, 2014. ACM.

[15] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts. Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. *IEEE Control Systems Magazine*, 36(6):45–64, Dec 2016.

[16] C. Kolassa, H. Rendel, and B. Rumpe. Evaluation of variability concepts for simulink in the automotive domain. In *2015 48th Hawaii International Conference on System Sciences*, pages 5373–5382, Jan 2015.

[17] M. Lienhardt, F. Damiani, S. Donetti, and L. Paolini. Multi software product lines in the wild. In *Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems*, VAMOS 2018, pages 89–96, New York, NY, USA, 2018. ACM.

[18] R. Matinnejad, S. Nejati, L. C. Briand, and T. Bruckmann. Automated test suite generation for time-continuous Simulink models. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 595–606, New York, NY, USA, 2016. ACM.

[19] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake. A classification and survey of analysis strategies for software product lines. *ACM Comput. Surv.*, 47(1):6:1–6:45, June 2014.

[20] T. Thüm, I. Schaefer, S. Apel, and M. Hentschel. Family-based deductive verification of software product lines. In *Proceedings of the 11th International Conference on Generative Programming and Component Engineering*, GPCE '12, pages 11–20, New York, NY, USA, 2012. ACM.

[21] J. Weiland and P. Manhart. A classification of modeling variability in Simulink. In *Proc. of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems*, VaMoS '14, pages 7:1–7:8, New York, NY, USA, 2013. ACM.

[22] Z. Zhang, G. Ernst, S. Sedwards, P. Arcaini, and I. Hasuo. Two-layered falsification of hybrid systems guided by monte carlo tree search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2894–2905, Nov 2018.