Towards Managing Variability in the Safety Design of an Automotive Hall Effect Sensor

Dimitri van Landuyt, Steven Op de Beeck, Aram Hovsepyan, Sam Michiels, Wouter Joosen, Sven Meynckens, Gjalt de Jong, Olivier Barais, Mathieu Acher

Towards Managing Variability in the Safety Design of an Automotive Hall Effect Sensor

Dimitri Van Landuyt, Steven Op de beeck, Aram Hovsepyan, Sam Michiels, Wouter Joosen
iMinds-DistriNet
KU Leuven, Belgium
first.last@cs.kuleuven.be

Sven Meynckens
Gjalt de Jong
Melexis NV | ArchWorks
svm@melexis.be
gjalt@acm.org

Olivier Barais
Mathieu Acher
IRISA / Inria
University of Rennes 1, France
first.last@irisa.fr

ABSTRACT

This paper discusses the merits and challenges of adopting software product line engineering (SPLE) as the main development process for an automotive Hall Effect sensor. This versatile component is integrated into a number of automotive applications with varying safety requirements (e.g., windshield wipers and brake pedals).

This paper provides a detailed explanation as to why the process of safety assessment and verification of the Hall Effect sensor is currently cumbersome and repetitive: it must be repeated entirely for every automotive application in which the sensor is to be used. In addition, no support is given to the engineer to select and configure the appropriate safety solutions and to explain the safety implications of his decisions.

To address these problems, we present a tailored SPLE-based approach that combines model-driven development with advanced model composition techniques for applying and reasoning about specific safety solutions. In addition, we provide insights about how this approach can reduce the overall complexity, improve reusability, and facilitate safety assessment of the Hall Effect sensor.

Categories and Subject Descriptors

D.2.10 [**Software Engineering**]: Design—*Methodologies, Representation*; D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific architectures*; D.2.13 [**Software Engineering**]: Reusable Software—*Reusable libraries; Reuse models*

Keywords

Software product line engineering, hardware/software co-design, Safety patterns, automotive, ASIL validation

1. INTRODUCTION

Many software-based systems and products are designed to be *versatile*, i.e. they solve different problems for different customers. To accomplish this, the development addresses the superset of the requirements of sufficiently large groups of customers. Versatility is a common —yet often implicit— architectural driver in the development of software-based systems and products because it makes production economically feasible: servicing more customers with a single product increases the return on investment of the development costs, and benefits from economies of scale [13].

As a downside however, it shifts some of the responsibility and complexity to the deployment and integration stages of development. In many cases, it is left to the customer to customize the product to his needs as part of a system integration project and this requires a deeper understanding of the product and its many possible configurations.

The above applies to the industry case that we have analyzed in the context of a collaborative research project called MERgE [5]. This case involves the design of an integrated Hall Effect sensor used in automotive systems. This product by Melexis, a market leader in automotive sensors, is called Triaxis® MLX90365 [11]. A Hall Effect sensor is a transducer that varies its output voltage in response to a magnetic field, allowing sensors to calculate angles in two- or three-dimensional spaces. This product is currently integrated in a wide range of automotive applications with varying safety requirements, ranging from windshield wipers, to brake or gas pedals. On average, recent cars may have 10 such sensors, and the MLX90365 Hall Effect sensor is currently being used for around 30 different automotive applications. Economy of scale is realized by mass-producing a one-size-fits-all product and the burden of configuration and integration is placed on the integration engineer or the customer (typically an automobile constructor).

Melexis currently applies a hardware-software *co-design* [20] development methodology. Although this is an effective approach to deliver versatile products to the market, this practice suffers from the following shortcomings:

1. Safety is a key concern that affects both the development process and the end product. The main hardware- and software-related design decisions are strongly motivated and affected by safety solutions (e.g. architectural patterns and tactics for safety). In the current practice, safety complicates matters in two ways. Firstly, as safety solutions have a profound impact on the end product, it is hard to reason about them in isolation (for example to validate their soundness) and to explore alternatives. Secondly, safety verification and assessment in practice is cumbersome, repetitive and therefore costly as (i) the notion of safety is strongly based on that of a *safety hazard,* which is very specific to the automotive application at hand and (ii) the versatile nature of the application defines a large configuration space which must in principle be explored in full (all possible combinations of configuration parameters) to assess the effects

of different parameter combinations on safety.

2. Reusability. A consequence of the hardware-software co-design approach is that the resulting implementation artifacts are of a platform-specific nature, which makes them hard to reuse across different hardware platforms. However, since Melexis already adopts a hardware-based product-line approach, there is large potential for (i) implementation-level reuse, (ii) reuse of architectural safety patterns, and (iii) consolidation and reuse of expert knowledge over product variants.

3. Variability. The focus on designing a highly versatile component is in essence the materialization of variability as a key development concern. The core problem with the current practice is caused by the fact that the complexity associated to variability profoundly affects the end product, and this complexity is addressed very late in the development process (at integration/configuration time). This is complicated further by the observation that three fundamentally different types of variability affect the Hall Effect sensor: (i) variability in the functionality, (ii) variability in the different safety solutions, (iii) variability across the hardware-software boundary. The current state of practice mainly addresses the first type [15, 4]. We face the challenge of handling two additional forms of variability that also are highly interrelated.

In this paper, we report on our detailed analysis of these concerns and the interplay between them in the context of this industry case. In a second step, we propose a tailored approach that combines Software Product Line Engineering (SPLE) with Model-driven Development (MDD) and advanced model-based concern composition techniques for model-based instantiation of safety patterns. Finally, we provide insights about how our proposal can reduce the overall complexity, improve the reusability, and facilitate the safety assessment of the Hall Effect sensor. In comparison to related work [15, 4], these are complementary insights about the impact of safety in a variability-intensive industrial context.

This paper is structured as follows. Section 2 highlights the results of our in-depth concern analysis exercise on the existing product and current development practices. Section 3 distills the overall motivation for investigating and exploring alternative development methods and techniques. Then, Section 4 presents the proposed development approach, and Section 5 provides our insights and arguments why we expect it to be beneficial to Melexis. Section 6 presents related work and Section 7 concludes the paper.

2. CONCERN ANALYSIS

We have performed a detailed analysis of the different concerns that affect the current design of the MLX90365 Hall Effect Sensor. This was conducted as follows: firstly, we have studied the existing technical specifications and design documents. Secondly, we have conducted a number of ATAM-style [2] workshops with Melexis engineers to obtain a precise understanding of the current product and the different concerns playing a role, how they interact and which essential trade-offs have been made between these concerns. In the following sections, we discuss the main results of this activity.

2.1 Safety

Given its integration in automotive applications that potentially lead to hazards endangering human life, MLX90365 is a safety-critical building block. The *Functional Safety for Road Vehicles* standard (ISO 26262 [16, 14]) is the applicable safety standard in the Automotive domain, defining a risk classification scheme called the Automotive Safety Integrity Level (ASIL). As with other safety assessment standards (e.g. IEC-61508 [9]), the assessment of hazard factors is based on the relative impact of hazardous effects related to a system (Severity), and the relative likelihoods of the hazard manifesting those effects (Exposure and Controllability). ASIL defines five Risk Levels, ranging from ASIL D, the strictest level, to QM (Quality Management), representing applications in which the safety hazards are tolerable and do not need to be governed by ISO 26262. The remaining levels (ASIL C, B and A) represent intermediate degrees of hazard.

In the design of the MLX90365 Hall Effect sensor, the safety concern has had a clear impact on:

1. The **development process**: ISO 26262 prescribes a number of development-related practices, e.g. imposing clear traceability between requirements and code, imposing documentation quality, but also introducing safety verification and assessment activities (involving safety testing and expert review).

2. The **end product**: From a software engineering point of view, "*architectural concerns with safety are almost identical to those for availability, which is also about recovering from failures. Tactics for safety, then, overlap with those for availability for a large degree.*" [2]. Two examples of architectural safety patterns that have been instantiated in the design of MLX90365 are:

- **Watchdog** (similar to the *heartbeat*-tactic [2]): a watchdog is a hardware-based countdown timer that continually decreases a variable. If the variable reaches zero, the watchdog resets the system (hard reset). When the software is operating normally, it continually sends out watchdog "ticks", which causes the variable to be reset to its maximum value. As long as the software runs normally, the watchdog will not intervene.

- **Fail-safe mode** (*graceful degradation* tactic [2]): when unrecoverable errors are detected (e.g. impossible input values), the system will go into fail-safe mode. Fail-safe mode is reported to the output: the system is reporting the error instead of outputting unreliable measurements, which might cause unpredictable behavior. **Debouncing** is a tactic commonly used in combination with this pattern: to avoid going into fail-safe mode on the basis of outliers and occasional errors and exceptions, a tactic called *debouncing* is applied, meaning that certain errors are ignored until they occur systematically.

2.2 Reuse

The MLX90365 Hall Effect sensor is not a stand-alone product, but is in itself part of a family of related products (cf. [12]). Being of MLX heritage, some elements of MLX90365 are the result of practical reuse, in terms of design but mostly implementation. Reuse is accomplished at the code level, i.e. code fragment and/or modular functions have been copy-pasted from different code bases. This

has been documented in the technical specification: e.g., *"It comes directly from 90360 f/w – reuse, coded in C."* (annotated in the documentation of a number of supporting functions).

2.3 Variability

We have identified three different types of variability that affected the design of the MLX90365 Hall Effect sensor. We adopt the distinction between internal and external variability [13]:

Variability due to versatility.

As discussed, the MLX90365 Hall Effect sensor is designed to be highly versatile, i.e. it must serve the purposes for a wide range of different automotive applications. To realize this degree of versatility, a configuration interface is provided in the form of a set of writable EEPROM parameters that have to be set at integration time, either by a Melexis engineer or by the automobile constructor himself. The functionality of the Hall Effect sensor depends highly on these specific parameters. To address this form of external variability, in total 54 EEPROM parameters are provided to the customer. For example[1], they provide the means to configure:

- The axes according to which the sensor must be calculate angles (either (X, Y), (X, Z), or (Y, Z)), and the rotation of the coordinate base.
- Different applications require different types of digital transformations being applied to the acquired angles. For example, for a brake pedal, the range between $[10 - 15]$ will only apply light braking, while the range between $[15 - 25]$ will apply heavy braking, and the range $[25 - 30]$ will apply emergency braking.

Realization-driven variability for safety.

Apart from these functionality-related configurations, the EEPROM parameters provide the means to set safety-related configuration parameters. Examples include: debouncing parameters, ROM checksums, optionality parameters, sanity intervals, etc.

These variables realize a form of internal variability as they directly influence the safety mechanisms at play and thus indirectly affect the ASIL safety level that can be attained. For example, in automotive applications that are less safety-critical (e.g. windshield wiper), some safety features can be disabled or configured to be very liberal (e.g. setting the debouncing threshold to an unrealistic value) to practically disable them. This is a form of internal variability that relates to how a specific safety level can be realized. There are in practice different strategies to realize compliance to a certain ASIL level (different combinations of safety patterns), and making the appropriate decision is left to the Melexis integration engineer with expert knowledge on the matter.

Variability across HW/SW boundary.

Another form of internal variability is related to the embedded nature of the Hall Effect sensor. As the hardware platform is designed and refined, so too will these decisions affect the implementation of the software. One example is the inclusion of the necessary hardware components to implement for example a Watchdog in products that require safety levels to which the Watchdog pattern contributes. Another example is related to the necessity of implementing a safety pattern involving continuously testing the RAM, which depends highly on the likelihood of RAM corruptions (bit flips), and is a specific characteristic of the selected hardware.

[1]These are strongly simplified and abstracted examples.

3. MOTIVATION

We have identified a number of essential problems with the current state of practice that was characterized in the previous section. These problems are elaborated below.

Safety engineering.

In the current practice, safety assessment and verification (i.e. ASIL) are cumbersome, repetitive and therefore costly development activities. This has a number of causes:

- Safety verification is based heavily on the notion of a *safety hazard* that in turn depends on the nature of the automotive application for which the Hall Effect sensor is used. For example, a windshield wiper is less safety-critical than a brake pedal because less safety hazards are associated to a faulty windshield wiper. As a consequence, safety assessment and verification has to be re-executed whenever the Hall Effect sensor is used in a different application, and in many cases, this entails repeating earlier efforts.
- In addition, safety verification activities always start from scratch. However, as many products in the same product family share a large number of architectural decisions and design elements, there is large potential for reuse of verification effort. Because the current development methodology does not involve reuse of existing safety solutions (e.g. safety patterns), nor reuse of architectural decisions (architectural knowledge related to when to pick which safety pattern), these activities involve repeating earlier-spent and costly verification efforts.
- The versatile nature of the Hall Effect sensor implies that the entire configuration space (all potential combinations of configuration parameters) has to be explored to attain a certain degree of confidence, even though some solutions will in practice be inactive (e.g. safety mechanisms such as debouncing). Due to the versatile nature of the Hall Effect sensor, the safety solutions provided in the end product must comply for the strictest validation requirements. In addition, safety verification must include showing that there is freedom of interference between the inactive and the active safety mechanisms.

Product-line reuse.

The existing product line setting described in Section 2.2 has a lot of potential for reuse. However, reuse of specific elements of a single product is difficult in the current state of practice:

- **Implementation-level reuse** is difficult due to the hardware-specific nature of development on the one

hand, and due to typical maintainability problems associated to the current state of practice of copy-pasting code fragments and/or individual functions on the other hand.

- **Safety patterns**. As mentioned above, the current methods and techniques do not allow reusing architectural patterns related to safety. More so, current safety solutions are not described as separate, reusable patterns, but are implemented directly in the Hall Effect sensor. The safety engineer is responsible for instantiating these patterns correctly, but he is provided limited tool support for the task at hand.

- **Consolidation and reuse of expert knowledge**. Most of the architectural knowledge about key decisions related to safety and other concerns remains implicit. As a result, the knowledge about (i) when to use which pattern, (ii) the costs associated to specific safety patterns, (iii) the implications of selecting a pattern on ASIL assessment, and (iv) which patterns interaction (either positively or negatively) remains unrecorded. Nonetheless, this is critical information to create and maintain a successful product line-based offering.

Variability management.

In Section 2.3, we introduced three types of variability that affect the design of the Hall Effect sensor.

- *Variability due to versatility* is currently covered very late in the development process (i.e. at configuration/integration time), at the cost of dragging along features (that are potentially unnecessary for the targeted automotive application and require unnecessary verification and assessment effort) throughout the development life-cycle.

- *Realization-driven variability for safety* is not supported in current practice. Safety decisions are made early on in the development process and motivated by implicit expert knowledge, while alternative solutions are disregarded.

- Finally, *variability across the HW/SW boundary* is not supported in current practice. More specifically, the design of the hardware platform is decided in an initial phase of the development process, and thereafter, these hardware decisions serve as fixed technical constraints for the software design activities.

4. APPROACH

Figure 1 depicts the proposed development approach, which is tailored for the design of the MLX90365 Hall Effect sensor. Our approach is aligned with the generic SPLE method by Pohl et al. [13]. The practical realization of this approach is currently ongoing work in the context of the MERgE research project [5].

The *Domain Engineering* activities (presented at the top of the figure) are:

- **Domain Analysis** involves studying the requirements for multiple instances of the Hall Effect sensor. Some of the results of this activity have been highlighted in Section 2. This activity leads to the definition of the feature model for external variability, which in our ongoing implementation efforts is created using FAMILIAR [1], a domain-specific language for performing feature model elaboration, reasoning, and manipulation (e.g., composition).

- **Domain Design** concerns defining reusable architectural models that can be composed to generate specific solutions. We create these models using SysML [17], an OMG standard for systems engineering. In addition, the internal variability model is created which defines the alternative safety solutions and their effect on the ASIL level.

- **Domain Realization** involves creating implementation artifacts on a per-feature basis.

- **Domain Verification** involves the creation of reusable tests and ASIL verification inputs on a per-feature basis.

The *Application Engineering* activities (presented at the bottom of the figure) are:

- **Product Configuration** involves tailoring the product to the specific requirements of the automotive application (e.g. setting the axes) on the basis of the external feature model. This is supported by FAMILIAR [1].

- **Solution Generation** involves generating alternative solutions that comply to the product configuration by combining the alternatives for realization defined in the internal feature model. These architectural variants are generated by instantiating the safety patterns into the base model. This is done with the Pattern Instantiation tool, which is part of the Thales Melody Advance software [6].

- **Analysis and Selection** involves comparing the different architectural candidates and selecting the most suitable variant.

- **Code Generation** deals with creating an implementation by composing the feature implementations that correspond to the selected architectural variant.

- **Safety Assessment and Verification** involves applying the composed tests, and extended test activities, which for example lead to ASIL validation of the entire end product.

To realize the entire approach, we are applying kCVL, an implementation of CVL [18]. kCVL is able to manage the different artifacts (feature models, SysML models, safety patterns) and supports variant derivation. For more details, please refer to `http://barais.github.io/splc2014merge/`

5. DISCUSSION

In this section, we provide insights on how the key principles behind this process are expected to improve the current state-of-practice.

Safety is supported explicitly. The internal variability model documents safety solutions and provides explicit support on how to attain specific safety levels. As such the

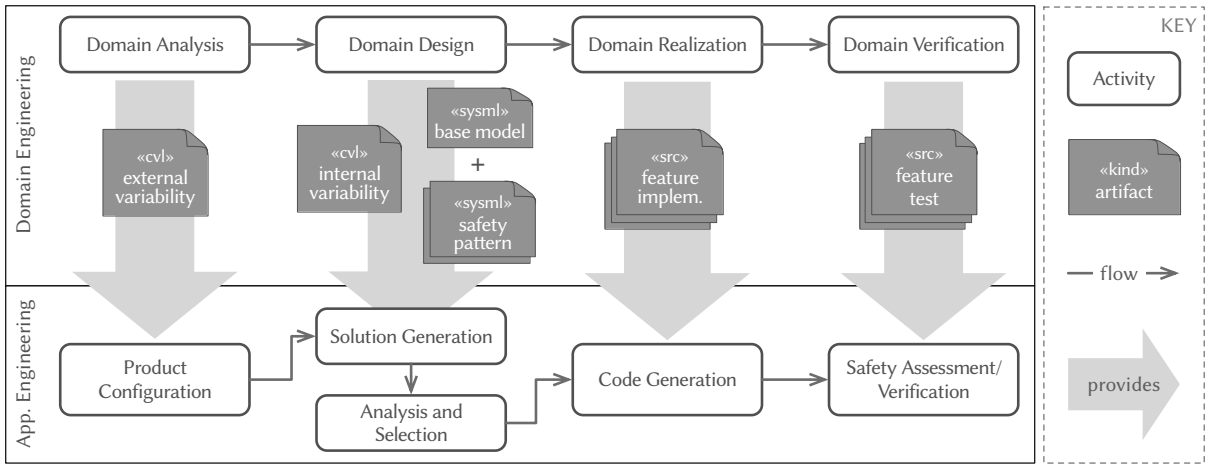

Figure 1: Our SPLE-based approach for applying and reasoning about specific safety solutions.

expert knowledge related to safety solutions is consolidated in this feature model. By creating tests on a per-feature basis, safety and verification efforts can partially be reused. The focus shifts from verifying an entire, composed model to verifying individual models (base models and safety patterns) and proving the correctness of the composed result.

Reuse is accomplished at multiple levels. At the level of architectural models, which describe either the main functionality for angle measurement (the *base model*), or specific architectural safety patterns. As part of the Model-Driven nature of the approach, these are then composed using advanced model transformation techniques and composition mechanisms as part of the *Solution Generation* activity. As corresponding implementation and tests are structured on a per-feature fashion, different variants of the firmware respectively the test suites for that firmware can be generated. As a direct result of this tailored approach, traceability links can easily be maintained and documented.

Variability is managed up front by adopting a top-down SPLE-based approach. External variability is kept apart from internal variability by introducing two distinct variability models. Maintaining this distinction is essential as the former represents configuration options offered to the customer, while the latter represents architectural alternatives in the safety solution space. Unnecessary features (those that are not selected during product configuration) will not be present in the generated firmware and will not be taken into account during safety assessment and verification. In addition, the generated firmware is a priori compliant to the constraints set by the feature models, and so, unrealistic or impossible variants are excluded very early on in the development process. Finally, by having a streamlined approach, the generation of traceability documentation will require less manual effort and thus be easier (a consequence of feature decomposition applied already at the early stages of development and maintained thereafter) and this in turn facilitates ASIL assessment.

6. RELATED WORK

Safety and SPLE. The issues raised by safety assessment are apparent in many software-intensive systems. In an SPLE context, the process of assessing and checking safety properties has to be applied to each (potential) product. The variability (or versatility) of a product (or the management of related similar products) substantially increases the complexity. In practice, the amount of possible products is exponential to the number of variation points (parameters, features, options, etc).

At the theoretical level, numerous techniques have been developed to efficiently check a SPL [19, 10]. These techniques are based on testing, type checking, model checking, or theorem proving; they can verify the whole set of product or only a subset, etc. With many sources of variability, our context precludes a full checking of every possible product. Moreover the safety properties are difficult to properly formalize, an assumption of many research works.

From an industrial perspective, few papers report on their experience in managing safety in an SPLE context [15, 4]. Schulze et al. [15] demonstrated that safety-related artifacts can be treated like other artifacts and presented a comprehensive model-based tool. We are following the same direction by linking feature models with safety modeling artifacts. Yet two additional, technical solutions have to be developed and integrated for (1) applying safety patterns (as part of the derivation process) and (2) managing *multiple* feature models (see below). As emphasized in the paper, the former activity is central to the success of a product line approach and requires further work (e.g., to compare applications of different safety patterns and then determine the most suitable).

Multi-variability. We observe multiple sources of variability in the industrial context. Pohl et al. proposed to distinguish internal variability from external variability [13]. Hubaux et al. reviewed the different concerns addressed in feature modeling [8]. Two kinds of variability caught our attention: hardware variability and realization-driven variability. Another noticeable observation is that the different sources of variability are intimately related. We are investigating compositional SPL techniques to manage this complexity [7, 3].

7. CONCLUSION

This paper has presented an industrial case —the design of a Hall Effect sensor— in which we combine techniques

of Software Product Line Engineering (SPLE), Model-driven Development (MDD) and advanced model composition mechanisms to model and instantiate architectural safety patterns. Specifically, we presented a detailed concern analysis with a focus on safety, reusability and variability. In addition, we presented a tailored development methodology for the Hall Effect sensor, of which we expect that it will simplify the end product, and make safety verification easier and less costly.

One of the main merits of this case is that it is a rather atypical example of an industrial system suited for SPLE, yet that it provides clear indications of the added value of adopting these techniques. In addition, the presented approach is effectively being applied on next-generation products and actively influences the business and product roadmap of Melexis.

Future work will focus on empirically validating the proposed approach, by assessing how this alternative development methodology improves the current practice, in terms of key performance indicators (KPIs) currently employed by Melexis, such as the required developer effort and effective reuse of expert knowledge and solutions over time. In addition, we will investigate broadening the scope from software design to system design (SW/HW), in order to address the most challenging type of variability manifesting itself in this industry case: variability across the hardware/software boundary.

Acknowledgements. The presented research is partially funded by the Research Fund KU Leuven and the Flemish agency for Innovation by Science and Technology (IWT 120085). The research activities were conducted in the context of ITEA2-MERgE (Multi-Concerns Interactions System Engineering, ITEA2 11011), a European collaborative project with a focus on safety and security [5].

8. REFERENCES

[1] Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert B. France. Familiar: A domain-specific language for large scale management of feature models. *Science of Computer Programming (SCP)*, 78(6):657–681, 2013.

[2] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Professional, 3rd edition, 2012.

[3] Jan Bosch. Toward compositional software product lines. *IEEE Software*, 27(3):29–34, 2010.

[4] Rosana T. V. Braga, Onofre Trindade, Jr., Kalinka R. L. J. Castelo Branco, and Jaejoon Lee. Incorporating certification in feature modelling of an unmanned aerial vehicle product line. In *Proceedings of the 16th International Software Product Line Conference - Volume 1*, SPLC '12, pages 249–258, New York, NY, USA, 2012. ACM.

[5] The MERgE consortium. Merge: Multi-concerns interactions system engineering. `http://www.merge-project.eu/`.

[6] Thales Group. Melody advance system modeler: Pattern instantiation tool, soon to be opensourced. `https://www.thalesgroup.com/en`, 2014.

[7] Arnaud Hubaux, Mathieu Acher, Thein Than Tun, Patrick Heymans, Philippe Collet, and Philippe Lahire. Separating concerns in feature models: Retrospective and support for multi-views. In *Domain Engineering, Product Lines, Languages, and Conceptual Models*, pages 3–28. 2013.

[8] Arnaud Hubaux, Thein Than Tun, and Patrick Heymans. Separation of concerns in feature diagram languages: A systematic survey. *ACM Comput. Surv.*, 45(4):51, 2013.

[9] International Electrotechnical Commission (IEC) Subcommittee 65A: Industrial-process measurement, control and automation – Systems aspects. Functional safety. `http://www.iec.ch/functionalsafety/`.

[10] Tomoji Kishi and Natsuko Noda. Formal verification and software product lines. *Commun. ACM*, 49(12):73–77, 2006.

[11] Melexis NV. MLX90365 Hall Effect Sensor IC. Product Datasheet. `http://melexis.com/Assets/MLX90365-Datasheet-6163.aspx`, Sept 2013.

[12] Melexis NV. Triaxis product overview. `http://melexis.com/Hall-Effect-Sensor-ICs/Triaxis%C2%AE-Hall-ICs/Triaxis-760.aspx`, Sept 2013.

[13] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, 2005.

[14] SAE International. J2980 - Considerations for ISO 26262 ASIL Hazard Classification. `http://www.sae.org/works/documentHome.do?docID=J2980&inputPage=wIpSdOcDeTaIlS&comtID=TEVEFS`, 2011.

[15] Michael Schulze, Jan Mauersberger, and Danilo Beuche. Functional safety and variability: Can it be brought together? In *17th International Software Product Line Conference*, pages 236–243, 2013.

[16] The International Organization for Standardization (ISO). Road vehicles – functional safety (iso 26262). `http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=43464`, November 2011. Part 1–10.

[17] The Object Management Group (OMG. SysML: Systems Modeling Language. `http://www.omgsysml.org/`.

[18] The Object Management Group (OMG). The Common Variability Language (CVL). `http://www.omgwiki.org/variability/doku.php`, 4 2014.

[19] Thomas Thüm, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake. A classification and survey of analysis strategies for software product lines. *ACM Computing Surveys*, 2014. to appear.

[20] Wayne Wolf and Jorgen Staunstrup. *Hardware/Software Co-Design: Principles and Practice*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.