

Keynote Outline – Challenges for testing in software product lines

Author: CR/AEA3-Fr – Georg Grütter,
georg.gruetter@de.bosch.com
+49 69 7909-514
Date: 2006-06-06

The importance of SW for Bosch products has been and will be continuously growing. Bosch manufactures hundreds of thousands of electronic control units per year for the automotive and other industries worldwide. The number of software developers Bosch employs worldwide rivals that of major players in the software industry. The majority of software intensive products are developed in software product lines.

A lot of Boschs software is safety relevant, like e. g. the ESP, engine and airbag controller software. It is also continuously growing more complex. The software for engine control units typically contain up to 840.000 lines of code (before compile time and including the feature superset for a high number of variants), processes the input of up to 25 sensors, controls up to 31 actuators, has up to 5200 features and about 7200 calibration parameters. Some of the Bosch product lines yield up to 1000 products in one year.

Since 2000, Bosch successfully applies the Software Product Line Practices Framework (SPLP) of the SEI in various business units. The new value motronic product line of the Gasoline Systems business unit has been successfully introduced in to the market. The same is true for the new AB10 generation of airbag controllers produced by the Automotive Electronics division. The development of both these product lines has been guided by the SPLP Framework from the outset. A thorough scoping combined with a solid business case as well a good system and software architecture have been the deciding success factors for both product lines.

In the light of the growing complexity and the safety criticality, Bosch regards testing as one of the major challenges for its future software development. The software product line testing community has addressed many of the technical challenges and proposed processes and practices during the last few years. In our experience, design for testability is both the most fundamental and best understood of those. However, there are a number of important nontechnical challenges for product line development that have, so far, not been addressed. These are:

1. Meeting the product developers' quality expectations for core assets
2. Establishing testing as a practice well regarded by managers and developers
3. Controlling the growth of variability
4. Making design decisions for testability

In [1], Barry Boehm argues that in the software industry, there is currently no good understanding of the business impact, that software design decisions have. We might add that this is also true for testing. For the software product line paradigm, being business centric, it is paramount. With business impact we refer to everything that has an impact on the achievement of the business goals of an organization. For a company trying to achieve the business goal „Quality leadership“ e. g., a design decision to apply the latest and immature technology potentially has a negative business impact. It is our hypothesis, that the lack of

understanding this business impact is the root cause for the last three challenges not being solved yet. But first things first.

Meeting the product developers' quality expectations for core assets

One of the promises of product lines is increased quality. This is a very appealing argument for companies producing safety critical software products. The assumption behind the quality promise is, that a component, if reused very often, will mature in the process of reusing it. However, this assumption is usually quickly forgotten. What remains is: „With product lines, we get better quality than before“. Furthermore, in the market of safety critical applications there really is no margin for „maturing in the field“ – „quality by quantities“ does not work here.

The expectations in management and project development are high and so is the pressure on the core asset developers. The common expectation is, that core assets will meet higher quality standards than before product line development from the outset. Additionally, the core asset developers are expected to fill up the core asset base with a complete set of core assets early in order to keep the time-to-market promise. This, certainly, doesn't make things easier.

In our experience, when product developers find out that the quality promise is not kept, the faith in product lines can be seriously shaken pretty quick. Product developers tend to revert to the old ways of clone and own and the whole idea of product line may die.

All this puts a high pressure on testing. Although quality cannot be „tested into“ core assets, the testers must make sure that even less defects slip through to product development than before. At the same time, variability in the core assets makes testing more difficult.

To deal with these challenges, we need to convey a realistic and clear picture of what the effects of product line development are and when they will take place. We see two, complementary means to achieve this. The means is to create a mind share in the organization. Have key persons temporarily participate in the work of other teams. That is, a core asset developer works with the product development team and vice versa, a manager spends a day with the developers to get a feel. The second means is to measure and try to quantify the experiences made during this exchange.

Establishing testing as well regarded practice by management and developers

The problem is not exactly new. The only surprising thing is that it is still around. Testing is an inherently destructive task. Most developers prefer creating software to trying to break it. That is why software developing organizations operate test centers. Management often views testing as unproductive and does not give any or not enough incentives for proper testing. When the pressure is on, the decision „Do we start working on the next feature our customer wanted or do we finish testing the ones already realized?“ is easily made. There is an inherent vicious circle to this: the more pressure there is, the less testing is done, the less stable core assets are, the more rework is needed, the less productivity there is, the more pressure there is, and so on.

Before product line testing can lift off, we need to break that vicious circle. We need to acknowledge that, although inherently destructive, testing actually creates value. Automated

suites of tests are valuable, reusable core assets. As with other development efforts for product lines, a rather huge up front investment into testing is necessary. The problem is this: we are usually not able to specify the positive business impact of testing and tests. How exactly does testing help us to attain our business goals and how could we weigh this against the costs. The costs, however, are clear, which makes it hard to break the vicious circle.

Controlling the growth of variability

The major source of complexity within product lines stems from the number of variants. Variation points are often a source of faults. Testing all variants of core assets a priori is usually impossible for all but the simple cases. So far, research in product line testing has focused on making variant testing more effective and efficient.

We propose to additionally investigate, how we can identify or reject those variants, that have a negative overall business impact. We'd need to be able to specify the business impact of those design decisions and tradeoffs, that would need to be altered for those new variants. We would need to be able to specify, how much additional testing effort a variant would cause. In our experience it is very hard to reject requests for unprofitable variants without a justifiable estimate of their negative business impact. The impact on testing and the overall health of the product line is obvious.

Making design decisions for testability

In the automotive embedded software domain, the dominating qualities of software are currently reliability and resource consumption. It has been recognized that for product lines to fulfill their promise of increased quality, decreased cost and shorter time to market, testability needs to be added to this list. Design for testability is, what is needed.

A lot of appropriate design decisions are known, such as „Separate variant from invariant code“ or „Separate controller software from information processing software“. In our experience, the impact of design decisions for testability on the system architecture can be substantial. Changing the architecture of a system might affect its reliability in the short run.

Hence, a precondition for these design decisions to be put into practice is, that we understand and communicate their business impact. Barry Boehm stated in [1]: „The links between value and software design are tight but still not understood well enough today. [...] The connection between technical parameters and value creation are understood vaguely, if at all.“. The questions to be answered are: „How will design for testability impact other qualities?“ „How much more effective and efficient will testing be when design decision x is implemented?“.

Without being able to answer these questions, design decisions for testability will likely not be implemented and the testing challenge will remain. Obviously, further research in this area is needed.

Conclusion

The product line paradigm is business centric. In order to achieve economical success, we need to quantitatively understand the business value of the technical decisions that we can make. The absence of such knowledge constitutes a serious problem. This is especially true for testing, which is not yet widely viewed as creating value in the first place.

So we need to find adequate models. George Box said „All models are wrong, some are useful“. We need to focus on the useful models that help us to understand and communicate what we do in product line testing.

The Personal Software Process (PSP), e. g., has demonstrated how building and applying models can be put to good use. We should also learn how other engineering disciplines align themselves to business goals. But in the end, we need to measure, measure, measure, build and validate our own models which might be different for each company and domain.

From all this also stems a requirement for the future technically oriented research in our field: it should quantitatively describe the possible business impact of its results. Although laborious, controlled software engineering experiments will help to verify and communicate the utility of the proposed new testing processes and practices.

References

[1] Barry W. Boehm, Kevin J. Sullivan, „Software Economics“, University of Southern California and University of Virginia, 1999