HyVar

Scalable Hybrid Variability for Distributed Evolving Software Systems

Thomas Brox Røst[1](✉), Christoph Seidl[2], Ingrid Chieh Yu[3], Ferruccio Damiani[4], Einar Broch Johnsen[3], and Cristina Chesta[5]

[1] Atbrox AS, Trondheim, Norway
thomas@atbrox.com
[2] Technische Universität Braunschweig, Braunschweig, Germany
c.seidl@tu-braunschweig.de
[3] Universitetet i Oslo, Oslo, Norway
{ingridcy,einarj}@ifi.uio.no
[4] Università di Torino, Turin, Italy
damiani@di.unito.it
[5] Santer Reply SpA, Turin, Italy
c.chesta@reply.it

Abstract. The HyVar project (www.hyvar-project.eu/) proposes a development framework for continuous and individualized evolution of distributed software applications running on remote devices in heterogeneous environments, focusing on the automotive domain. The framework combines variability modeling and software reuse from software product lines with formal methods and software upgrades and can be integrated in existing software development processes. HyVar's objectives are: (O1) To develop a Domain Specific Variability Language (DSVL) and tool chain to support software variability for highly distributed applications; (O2) to develop a cloud infrastructure that exploits software variability as described in the DSVL to track the software configurations deployed on remote devices and to enable (i) the collection of data from the devices to monitor their behavior; and (ii) secure and efficient customized updates; (O3) to develop a technology for over-the-air updates of distributed applications, which enables continuous software evolution after deployment on complex remote devices that incorporate a system of systems; and (O4) to test HyVar's approach as described in the above objectives in an industry-led demonstrator to assess in quantifiable ways its benefits. The end of the project is approaching and we are close to reaching all the objectives. In this paper, we present the integrated tool chain, which combines formal reuse through software product lines with commonly used industrial practices, and supports the development and deployment of individualized software adaptations. We also describe the main benefits for the stakeholders involved.

The HyVar project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 644298.

Z. Á Mann and V. Stolz (Eds.): ESOCC 2017, CCIS 824, pp. 159–163, 2018.
https://doi.org/10.1007/978-3-319-79090-9_12

Keywords: Software engineering · Software maintenance · Software evolution
Software product lines · Variability models · Distributed software
Over-the-air updates · Data intensive systems · Internet of things
Cloud computing

1 Motivation and Approach

1.1 Software Evolution in the Automotive Domain

Evolution is a well-known problem in any software product family of non-trivial complexity. Over the lifespan of a product line, new features are added, old features are deprecated, and separate code branches may be created to deal with spin-off products. As the code and customer base grow, the possible feature combinations used by various product instances soon become unwieldy.

In the automotive domain, all the possible variants of a car (make, model, base equipment, extras etc.) can create a situation with a combinatorial explosion of feature combinations. This poses a challenge when maintaining the supporting software. Not only new features must be catered for but also all previous feature combinations for products that are still on the market and under support agreements. This can easily lead to bad software development practices, such as "clone and own", where code is copied between repository branches and modified in isolation.

Regarding deployment of software updates in the automotive scenario, how does one ensure that a given car gets an update that is customized to its particular feature combination? Moreover, how can we also take, e.g., the driving characteristics of the car's owner into account when updating the car's software? Using traditional software development techniques, it is easy to end up with a software repository with lots of duplicated and overlapping code, where making any substantial change carries the risk of unforeseen down-the-line implications.

1.2 HyVar Solution

Within the HyVar project, we take a two-fold approach towards tackling this problem, focusing on both the *development* and *deployment* aspect of maintaining complex software products.

On the *development* side, we have created a DSVL and other tools that use software product line (SPL) modeling techniques for a structured approach towards handling feature combinations [1–3]. These tools can either be adopted for development from scratch for new projects or be applied to existing projects with low adoption efforts. Moreover, our analysis tools can be applied to a project that is already suffering from previous clone-and-own development so that differences and similarities of cloned products are automatically extracted and modeled [4]. This greatly simplifies the task of cleaning up from years of bad development practices.

On the *deployment* side, we introduce a cloud-based tool chain that complements our development tools [5]. The tool chain functions in an automotive context, keeping tabs on a large number of connected devices and their feature configurations. A re-configuration component detects whenever a software update must be applied, either due to changes on the device (pull action) or changes on the software side (push action). Instead of taking the naïve approach of pushing the same monolithic update to each device, the tool chain creates a customized update for each device and only does the full recompile when it is deemed necessary. This greatly reduces the complexity and bandwidth required to do over-the-air updates for a device fleet. Also, as part of the supporting toolkit, we can do a static analysis of the resources required for running our tool chain on the cloud. For a realistic traffic pattern model, this removes a lot of the guesswork otherwise necessary for cloud infrastructure cost estimation.

2 Application and Benefits

The approach has been applied to an automotive domain scenario, namely the dynamic reconfiguration of car software based on context, where we addressed the following issues: *(i)* To develop a software product line by allowing developers to derive a new product from an existing one; *(ii)* To reduce the risk in distributed software development projects by developing several software product lines with distributed teams keeping track of the dependencies; *(iii)* To support personalized deployment from the cloud; and *(iv)* To derive a software product line from existing products. For the stakeholders in our automotive scenario (e.g., car manufacturers and automotive software developers), there are several benefits associated with using our tool chain. Some of these benefits are outlined below, grouped according to issues (i)–(iv).

2.1 Software Product Line Development Using the HyVar Tool Chain

We experienced the following benefits by developing a software product line realizing the emergency call service for both the European and Russian market, exploiting the commonalities and ensuring compliance with the respective standards.

The existing product can be used as it is. Using the DSVL and delta modeling techniques to transform statecharts and/or source code, it is possible to start from an existing product.

New features are fully implemented and recorded in statecharts. The configuration differences between product branches are highly visible and explicit so that they are much easier to communicate within the company. Moreover, the executable programs can be created directly from the statechart editor.

Living models. As new features are both modeled and built from statecharts, there is a direct connection between the model and the final product.

Reduced code duplication and development time. As code is generated from models, the amount of code duplication is reduced. The copy/paste approach towards software development is no longer needed.

2.2 Reducing Risk in Distributed Software Development Projects

Our initial demonstrator involved a single software product line. We then extended it into a more complex system, including three software product lines with interdependencies. This yielded additional benefits.

All interfaces are defined through MSPL feature model interfaces. This encourages both better encapsulation and a more structured approach towards feature interfaces and simplifies the distributed development.

Independent software product line. By the use of feature model interfaces of the HyVar tool chain, the new functionality can be planned and constructed independently from the rest of the software product line.

Feature encapsulation helps evolution. Better encapsulation makes it explicit which parts of the software system can be changed without introducing errors in the existing functionality.

Early detection of specification errors. Using feature model interfaces, it is possible to guarantee that only intended configurations can be created.

2.3 Personalized Deployment from the Cloud

In our final demonstrator we enabled software updates that take the driving style and preferences of individual drivers into account.

Cloud-based infrastructure. One of the major benefits of using cloud services is that you only pay for the resources you use. This means that there is no need for an upfront data center investment and that the costs scale along with the number of users as the company grows. For customers who are wary of using public clouds, private cloud installations are also possible.

Simulation model of cloud resource requirements. With the traffic data collected by a car manufacturer stakeholder, it is possible to simulate the resources needed for the tool chain cloud infrastructure. This makes it possible to estimate the costs required for deploying the tool chain for a given fleet of cars even if there are peak periods.

Context changes can be reflected in the software configuration. Using validity formulas, context constraints and the HyVarRec reconfigurator, the software can be customized for a highly specific environment.

2.4 Derivation of an SPL from Existing Products

We have developed a variability mining methodology that provides the benefits listed below. Although the methodology is compatible with the HyVar tool chain, the HyVar demonstrator is not suitable for evaluating the methodology. Therefore, we have evaluated it by considering another case study in the automotive domain.

Automated analysis of existing software products. The differences and similarities of existing, cloned products can be analyzed almost completely automatically. From this, it is possible to generate feature models, mappings and delta modules that later can be used when adding new features or spinning off new product lines. The effort compared to doing this manually is reduced greatly.

Generated software product line elements. From the results of the analyses, the variability mining also generates suitable elements, such as delta modules, a technical feature model and, if needed, even an entire suitable delta language, e.g., for elements written in domain-specific languages. This overall greatly reduces the manual effort to set up an SPL from cloned variants.

Controlled restructuring. The process is not fully autonomous, meaning that developers have a lot of control over things such as feature naming and how the mined products should be restructured. They can also guide the restructuring through doing an iterative feedback/adaption process with the variability mining technology.

Increased abstraction between features and code. As the feature models are refined, so is the abstraction to the underlying code. This has long-term benefits in terms of both planning, discussing and working with a product as a combination of evolved features rather than just lines of code.

3 Conclusion

We have presented an innovative solution to the software reuse problem, integrating SPL engineering principles with existing tools and commonly used industrial practices. The HyVar approach supports the development and deployment of individualized software adaptations and realizes the concept of hybrid variability. The methodology and tool chain has been applied in a scenario from the automotive domain, and seems promising also for other emerging scenarios, such as Internet of Things (IoT) and Cyber-Physical Systems (CPS), characterized by a huge number of remote devices, each of which has its own hardware configuration, runs a customizable distributed software application and needs to evolve in order to fix or prevent misbehavior, to adapt to environmental changes, accomplish new regulations, satisfy new user requests or meet new market expectations.

References

1. Chesta, C., Damiani, F., Dobriakova, L., Guernieri, M., Martini, S., Nieke, M., Rodrigues, V., Schuster, S.: A toolchain for delta-oriented modeling of software product lines. In: Margaria, T., Steffen, B. (eds.) ISoLA 2016. LNCS, vol. 9953, pp. 497–511. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47169-3_40
2. Nieke, N., Engel, G., Seidl. C.: DarwinSPL: an integrated tool suite for modeling evolving context-aware software product lines. In: ter Beek, M.H., Siegmund, N., Schaefer, I. (eds.) Proceedings of the Eleventh International Workshop on Variability Modelling of Software-intensive Systems (VAMOS 2017), pp. 92–99. ACM (2017). https://doi.org/10.1145/3023956.3023962
3. Damiani, F., Lienhardt, M., Paolini, L.: A formal model for multi SPLs. In: Dastani, M., Sirjani, M. (eds.) FSEN 2017. LNCS, vol. 10522, pp. 67–83. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68972-2_5
4. Wille, D., Schulze, S., Seidl, C., Schaefer, I.: Custom-tailored variability mining for block-based languages. In: Proceedings of the International Conference on Software Analysis, Evolution, and Reengineering (SANER 2016). IEEE (2016). https://doi.org/10.1109/saner.2016.13
5. Mauro, J., Nieke,, N., Seidl, C., Chieh Yu, I.: Context aware reconfiguration in software product lines. In: Schaefer, I., Alves, V., de Almeida, E.S. (eds.) Proceedings of the Tenth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 2016), pp. 41–48. ACM (2016). https://doi.org/10.1145/2866614.2866620