C2MV2: Consistency and Composition For Managing Variability in Multi-View Systems

Roberto E. Lopez-Herrejon
Systems Engineering and Automation
Johannes Kepler University Linz, Austria
roberto.lopez@jku.at

Alexander Egyed
Systems Engineering and Automation
Johannes Kepler University Linz, Austria
alexander.egyed@jku.at

***Abstract*—C2MV2 is an ongoing FP7-People Intra-European Marie Curie Fellowship project that runs for two years. The driving goal of the project is to apply and extend work on incremental consistency management to Software Product Lines that are developed with compositional approaches.**

***Keywords*-Software Product Lines; Safe Composition; Feature Orientation; Product Line Evolution**

I. Introduction

Software Product Lines (SPLs) [1]–[3] are families of systems that share common functionality but also have variations tailored for distinct needs. In a SPL, each member product thus implements a different combination of *features* – increments in program functionality [4]. The success of a SPL lies at the effective management and realization of its *variability*, defined as the capacity of software artifacts to vary [5]. Extensive research and practice has been documented that corroborates the significant benefits of applying SPL practices both in academia and industry [2], [3], [6].

Technology advances and demands for complex and varied applications have made reliable SPL techniques a necessity to tackle such demands. Most SPL methodologies integrate variability into software artefacts which contain both the common and variable parts. Depending on the features required, the unneeded variable parts are removed to build the system. As the number of variable parts increases so does the complexity of integrating them correctly. This complexity can potentially grow exponentially as all features and their interactions must be considered across all artefacts.

Emerging SPL techniques advocate encapsulation of variable parts across multiple views into separated features, which are composed according to client needs. With the compositional approach, the complexity is reduced as only those variable parts that can be composed together need to be considered simultaneously. Compositional SPL techniques allow separate development of variable parts by encapsulating them. However, this capability poses consistency problems as changing a variable part without considering the other related parts may introduce errors. This is compounded when variable parts are described through multiple and distinct models (views).

Common mainstream modelling techniques advocate the use of different yet related models to represent the different stakeholders' needs - a practice known as *Multi-View Modeling (MVM)* [7], [8]. MVM technologies provide different alternatives for handling consistency but have focused on one-of-a-kind multi-view systems without any variability considerations. Leveraging their use in compositional SPL approaches does pose a major challenge: how to extend them for handling consistency in and amongst the encapsulated multi-view variable parts while considering their composition. This is precisely the problem tackled by our project.

II. Project Overview

The C2MV2 [1] project started in August 2010 and will run until July 2012. The funding is approximately 220,000 Euros provided by the Seventh Framework Programme (FP7)[2], under the *People* umbrella. Under the Marie Curie programme there exist several funding schemes. In particular, the Intra-European Fellowships (IEF) are offered individually to a guest researcher (first author) that is hosted in a well-established research group, in our case the Systems Engineering and Automation Institute (SEA) – headed by the second author – with the purposes of advancing the fellow's academic career.

III. Project Background

In this section, we provide the basic background information on the research topics our projects builds upon.

A. Compositional SPL Technologies

Emergent compositional SPL technologies have been the subject of intensive research in recent years [9], [10]. An example of these technologies is *Aspect Oriented Software Development (AOSD)* [10]–[12]. AOSD techniques have proven effective to modularize concerns (features) on artefacts such as code, models, and requirements [10]. However, until recently, incipient work on consistency has been proposed in *Aspect Oriented Modelling (AOM) [13]*.

Another prominent approach in this category is F*eature Oriented Software Development (FOSD)* [1], [14]. FOSD provides formalisms, methods, languages and tools for building variable, customizable and extensible software. FOSD has been successfully used in several case studies [15], [16].

[1]Project website http://www.sea.uni-linz.ac.at/c2mv2/

[2]A funding scheme of the European Comission http://cordis.europa.eu/fp7/home_en.html

FOSD advocates modularizing features, increments in program functionality [4], as the systems building blocks. At the heart of FOSD is a feature algebra that drives the (de)composition of software artefacts [1], [17], [18]. A feature module contains all the software artefacts, or parts thereof, required for implementing the feature. In other words, feature modules capture the multiple views of a feature.

Current realizations of FOSD [9], [19] work under the assumption that either feature artefacts are derived from other artefacts (e.g. via compilation) or are by default manually kept consistent. This assumption may work for the artefact types (source code, make scripts, grammars, equations, XML, and state machines) FOSD has primarily focused on, but it is a limitation as FOSD expands to other activities such as analysis and design that typically employ modelling languages such as UML. Incipient research on modelling and FOSD has been conducted [20]–[23], it works by extending FOSD composition operator to UML models.

B. Safe Composition

An important trait of SPL is that not all feature combinations yield correct and meaningful software products. Depending on the problem domain, selecting a feature may require the selection of other features; conversely, selecting a feature may exclude or prevent the selection of other features.

Feature Models are the de facto mechanism to model the commonality and variability of SPL and there exists a significant body of work on their formal analysis [24], [25]. At a finer granularity, *Safe Composition* is the guarantee that all programs which are members of a product line are indeed type safe (i.e. absent of references to undefined elements) [26]. It works by including the constraints that composed programs should meet (e.g. single introduction of a class member) in addition to the domain constraints. It is important to stress that Safe Composition eliminates the need to individually check every single program that can be composed, which even for small SPL is impractical. Even though, it focuses on source code, the same principles and techniques could be applied to other artefacts.

C. Consistency Checking

A crucial demand of MVM systems is consistency checking to describe and preserve the semantic relationships amongst the elements of the different views. Consistency checking works by evaluating consistency rules on models to verify if they meet the semantic constraints. As the size of models increase, so does the time taken to verify them. A recent trend in consistency checking is work on incremental approaches which react to changes and evaluate only those rules and on only those model elements (previously identified by profiling) that can potentially cause an inconsistency. Among those incremental approaches is work by Egyed et al. [27]–[29] and Blanc et al. [30], [31].

D. Multi-Dimensional Separation of Concerns (MDSoC)

MDSoC argues that stakeholders concerns should be encapsulated across all dimensions (views) simultaneously and subsequently composed to build an entire system [32], [33]. A key novel insight is to consider model changes as increments (decrements) in the functionality expressed in the models. MDSoC can thus be regarded as a foundation of SPL compositional technologies because both propose to build complex multi-dimensional systems by assembling less complex modules in a disciplined, flexible and scalable way. Such modules are increments in functionality of multi-dimensional systems.

IV. Project Contributions

The novelty of the project lies on exploiting the synergies between: Multi-View Modelling (MVM) and Multi-Dimensional Separation of Concerns (MDSoC). Both advocate that software development involves multiple stakeholders with different needs that require different views or perspectives from the same system. The incremental and multi-view (multi-dimension) standpoint on software development that MVM and MDSoC take suggests far-reaching research opportunities on leveraging and extending work from MVM incremental consistency checking to complement compositional SPL technologies.

As a first step, it is imperative to place SPL work into a consistency context. A criterion to categorize consistency is the number of model kinds involved: *Intra-Model consistency* considers a single model kind, and *Inter-Model consistency* considers more than one model kind [34]. A second criterion is the level at which consistency checking is performed [35]. Consistency is considered at the *Domain Engineering* [3] level when it deals with the consistency of the entire product line and at the *Application Engineering* [3] level when it focuses on the consistency of a single system member of product line.

Using this consistency framework we now describe where current work on compositional SPL fits in and where the contributions of the proposed work are expected. The reference consistency framework is shown in Figure 1, where our expected contributions are shaded.

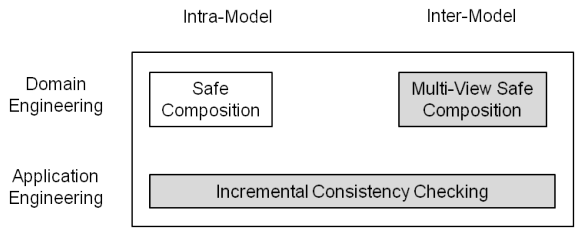

Fig. 1: Consistency Framework and Proposed Research

It is important to stress that standard consistency checking techniques have been conceived and applied to single systems without any variability considerations. In the consistency perspective of SPL, these techniques can be considered at the Application Engineering level because they deal with a single product and can be applied without distinction to one or more model kinds. To the best of our knowledge, this is a research venue that remains largely unexplored. In this regard, our project will study how to adapt and apply

incremental consistency checking at Application Engineering level to support Intra and Inter-Model consistency (bottom of Figure 1).

The key challenge is understanding and exploiting the interplay between feature composition and consistency checking. Furthermore, our work aims at elevating the role of consistency checking to Domain Engineering such that inconsistencies are detected and dealt with while realizing the assets of the SPL, and thus reducing any negative impact they may have at later development stages.

To address this challenge, our work will extend on current approaches to Safe Composition. Within our consistency context framework, work on Safe Composition can be regarded as an example of Intra-Model and Domain Engineering consistency because it focuses on a single artefact (source code) and considers the composition constraints of the entire product line as expressed in a feature model with its additional domain constraints. The extensions proposed will include constraints from several models simultaneously thus providing Inter-Model consistency. We refer to this extension as Multi-View Safe Composition (top of Figure 1).

In summary, despite the extensive and successful research on SPL there is still a lack of consistency checking support in SPL techniques; consistency is either ignored, assumed, or at best manually verified and enforced. This is the problem the proposed work addresses by leveraging and extending the vast research on consistency checking as described.

V. Research Objectives

The overall goal of this project is to develop, support with tools, and evaluate techniques for providing flexible and efficient consistency checking for SPL development. Because of the reduced complexity of variability management in compositional approaches, our work will leverage, extend, and ally recent developments in consistency checking for these SPL approaches. The concrete research objects are described and briefly explained.

A. Develop extensions to Safe Composition of diverse artefacts.

Safe Composition research has focused on source code artefacts, however techniques such as that proposed in [26] can be mapped to non-code artefacts. By adding constraints from other artefacts it is possible to consider multiple artefacts simultaneously. We refer to this as Multi-View Safe Composition. We have recently shown how to extend Safe Composition for checking consistency in basic UML models [36], and highlighted its importance and applicability in software architectures [37]. A key issue is if the underlying logic foundations (feature models and SAT solvers [38]) scale when constraints (e.g. written as OCL rules) from multiple artefacts are considered.

B. Develop techniques that ally compositional SPL approaches with incremental consistency checking.

Compositional SPL works by incrementally adding (composing) feature functionality across multiple artefacts. Incremental consistency checking could be applied in synchronization with feature composition with two potential benefits: improved performance to check consistency, and a lightweight non-formal empirical validation that composition is implemented consistently across all the artefacts kinds.

C. Devise and apply an assessment plan to evaluate usability and usefulness of the techniques proposed.

For this objectives, several case studies of different sizes and domains will be considered. They will be drawn from examples publicly available from projects websites and those developed by our group and our collaborators. They will be used to evaluate performance, applicability, and usability of the tool support.

VI. Project Relevance

Two themes relevant to the CSMR community are maintenance and evolution. Both activities can be supported with effective and efficient consistency checking to verify that important semantic and syntactic properties (expressed as consistency rules) are met. Additionally, maintenance and evolution of SPL are topics of increasing interest to the community, for instance [39]–[41]. Thus our work on SPL extensions to consistency checking can potentially contribute to the ongoing research on those topics.

VII. Related Projects and Works

Past European Eureka-ITEA projects ESAPS, CAFE, and FAMILIES built and consolidated a large community of researchers, and paved the way for European leadership in SPL research and practice [3], [6], [42]. On a similar token the MODELWARE project helped cement the prominent presence of Europe in Model-Driven Development research and practice [43]. FP6 project MODELPLEX has extended this work to tackle Software Engineering challenges of building complex systems with Model-Driven Engineering approaches [44]. Along the same lines, project AMPLE researched the use of Model-Driven Development and Aspect Orientation as two emerging techniques that can be integrated for SPL development [10]. The proposed research aims to complement and build upon these extensive and successful European efforts.

Model composition has been an active research area, of particular relevance is the work from MODELWARE project by Bezivin et al. proposes using canonical model operators as general mechanisms to represent and understand model composition [45]; and the work from MODELPLEX project by Herrmann et al. which gives an algebraic view of model composition operators and identify some of the desirable properties such as consistency preserving but it is not considered from a SPL perspective [46].

Acknowledgment

This research is partially sponsored by the Austrian FWF under agreement P21321-N15 and and Marie Curie Actions - Intra-European Fellowship (IEF) project number 254965.

REFERENCES

[1] D. S. Batory, J. N. Sarvela, and A. Rauschmayer, "Scaling step-wise refinement," *IEEE Trans. Software Eng.*, vol. 30, no. 6, pp. 355–371, 2004.

[2] K. Czarnecki and U. Eisenecker, *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.

[3] K. Pohl, G. Bockle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, 2005.

[4] P. Zave, "Faq sheet on feature interaction," http://www.research.att.com/ pamela/faq.html.

[5] M. Svahnberg, J. van Gurp, and J. Bosch, "A taxonomy of variability realization techniques," *Softw., Pract. Exper.*, vol. 35, no. 8, pp. 705–754, 2005.

[6] F. J. van d. Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer, 2007.

[7] B. Nuseibeh, J. Kramer, and A. Finkelstein, "A framework for expressing the relationships between multiple views in requirements specification," *IEEE Trans. Software Eng.*, vol. 20, no. 10, pp. 760–773, 1994.

[8] A. Finkelstein, D. M. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh, "Inconsistency handling in multperspective specifications," *IEEE Trans. Software Eng.*, vol. 20, no. 8, pp. 569–578, 1994.

[9] D. Batory, "AHEAD Tool Suite," 2010, http://www.cs.utexas.edu/users/schwartz/ATS.html.

[10] "Aspect-oriented model-driven product line engineering (ample)," 2009, http://www.ample-project.net/.

[11] M. Mezini and K. Ostermann, "Variability Management with Feature-Oriented Programming and Aspects," in *Proceedings of the International Symposium on Foundations of Software Engineering (FSE)*, 2004, pp. 127–136.

[12] I. Groher and M. Völter, "Using aspects to model product line variability," in *SPLC (2)*, S. Thiel and K. Pohl, Eds. Lero Int. Science Centre, University of Limerick, Ireland, 2008, pp. 89–95.

[13] J. Kienzle, W. A. Abed, and J. Klein, "Aspect-oriented multi-view modeling," in *AOSD*, K. J. Sullivan, Ed. ACM, 2009, pp. 87–98.

[14] R. E. Lopez-Herrejon, "Understanding Feature Modularity," Ph.D. dissertation, Department of Computer Sciences, The University of Texas at Austin, 2006.

[15] S. Trujillo, D. S. Batory, and O. Díaz, "Feature oriented model driven development: A case study for portlets," in *ICSE*. IEEE Computer Society, 2007, pp. 44–53.

[16] D. Batory and S. O'Malley, "The Design and Implementation of Hierarchical Software Systems with Reusable Components," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 1, no. 4, pp. 355–398, 1992.

[17] R. E. Lopez-Herrejon, D. S. Batory, and C. Lengauer, "A disciplined approach to aspect composition," in *PEPM*, J. Hatcliff and F. Tip, Eds. ACM, 2006, pp. 68–77.

[18] D. S. Batory, "Using modern mathematics as an fosd modeling language," in *GPCE*, Y. Smaragdakis and J. G. Siek, Eds. ACM, 2008, pp. 35–44.

[19] S. Apel, C. Kästner, and C. Lengauer, "Featurehouse: Language-independent, automated software composition," in *ICSE*. IEEE, 2009, pp. 221–231.

[20] R. E. Lopez-Herrejon, "Language and uml support for features: Two research challenges," in *VaMoS*, ser. Lero Technical Report, K. Pohl, P. Heymans, K. C. Kang, and A. Metzger, Eds., vol. 2007-01, 2007, pp. 97–100.

[21] S. Umapathy, "Extension of UML models to Support Feature Modularization of Software Product Lines," Master's thesis, Computing Laboratory, University of Oxford, 2007.

[22] R. E. Lopez-Herrejon, "Models, features and algebras - an exploratory study of model composition and software product lines," in *ICSOFT (SE/MUSE/GSDCA)*, J. Cordeiro, B. Shishkov, A. Ranchordas, and M. Helfert, Eds. INSTICC Press, 2008, pp. 293–296.

[23] R. E. Lopez-Herrejon and J. E. Rivera, "Realizing feature oriented software development with equational logic: An exploratory study," in *JISBD*, A. Vallecillo and G. Sagardui, Eds., 2009, pp. 269–274.

[24] D. Benavides, S. Segura, and A. R. Cortés, "Automated analysis of feature models 20 years later: A literature review," *Inf. Syst.*, vol. 35, no. 6, pp. 615–636, 2010.

[25] M. Mendonça, A. Wasowski, K. Czarnecki, and D. D. Cowan, "Efficient compilation techniques for large scale feature models," in *GPCE*, Y. Smaragdakis and J. G. Siek, Eds. ACM, 2008, pp. 13–22.

[26] S. Thaker, D. S. Batory, D. Kitchin, and W. R. Cook, "Safe composition of product lines," in *GPCE*, C. Consel and J. L. Lawall, Eds. ACM, 2007, pp. 95–104.

[27] A. Egyed, "Fixing inconsistencies in uml design models," in *ICSE '07: Proceedings of the 29th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 292–301.

[28] ——, "Instant consistency checking for the uml," in *ICSE*, L. J. Osterweil, H. D. Rombach, and M. L. Soffa, Eds. ACM, 2006, pp. 381–390.

[29] A. Egyed, E. Letier, and A. Finkelstein, "Generating and evaluating choices for fixing inconsistencies in uml design models," in *ASE*. IEEE, 2008, pp. 99–108.

[30] X. Blanc, I. Mounier, A. Mougenot, and T. Mens, "Detecting model inconsistency through operation-based model construction," in *ICSE*, W. Schäfer, M. B. Dwyer, and V. Gruhn, Eds. ACM, 2008, pp. 511–520.

[31] X. Blanc, A. Mougenot, I. Mounier, and T. Mens, "Incremental detection of model inconsistencies based on model operations," in *CAiSE*, ser. Lecture Notes in Computer Science, P. van Eck, J. Gordijn, and R. Wieringa, Eds., vol. 5565. Springer, 2009, pp. 32–46.

[32] P. Tarr, H. Ossher, W. Harrison, and J. S. M. Sutton, "N Degrees of Separation: Multi-Dimensional Separation of Concerns," in *ICSE*, 1999, pp. 107–119.

[33] D. S. Batory, R. E. Lopez-Herrejon, and J.-P. Martin, "Generating product-lines of product-families," in *ASE*. IEEE Computer Society, 2002, pp. 81–92.

[34] G. de Fombelle, X. Blanc, L. Rioux, and M.-P. Gervais, "Finding a path to model consistency," in *ECMDA-FA*, ser. Lecture Notes in Computer Science, A. Rensink and J. Warmer, Eds., vol. 4066. Springer, 2006, pp. 101–112.

[35] K. Lauenroth and K. Pohl, "Dynamic consistency checking of domain requirements in product line engineering," in *RE*. IEEE Computer Society, 2008, pp. 193–202.

[36] R. E. Lopez-Herrejon and A. Egyed, "Detecting inconsistencies in multi-view models with variability," in *ECMFA*, ser. Lecture Notes in Computer Science, T. Kühne, B. Selic, M.-P. Gervais, and F. Terrier, Eds., vol. 6138. Springer, 2010, pp. 217–232.

[37] ——, "On the need of safe software product line architectures," in *ECSA*, ser. Lecture Notes in Computer Science, M. A. Babar and I. Gorton, Eds., vol. 6285. Springer, 2010, pp. 493–496.

[38] M. Huth and M. Ryan, *Logic in Computer Science. Modelling and Reasoning about systems*. Cambridge University Press, 2004.

[39] H. D. Rombach, "Design for maintenance - use of engineering principles and product line technology," in *CSMR*, A. Winter, R. Ferenc, and J. Knodel, Eds. IEEE, 2009, pp. 1–2.

[40] M. de Medeiros Ribeiro and P. Borba, "Improving guidance when restructuring variabilities in software product lines," in *CSMR*, A. Winter, R. Ferenc, and J. Knodel, Eds. IEEE, 2009, pp. 79–88.

[41] C. Nunes, U. Kulesza, C. Sant'Anna, I. Nunes, A. F. Garcia, and C. J. P. de Lucena, "Comparing stability of implementation techniques for multi-agent system product lines," in *CSMR*, A. Winter, R. Ferenc, and J. Knodel, Eds. IEEE, 2009, pp. 229–232.

[42] T. Käkölä and J. C. Dueñas, Eds., *Software Product Lines - Research Issues in Engineering and Management*. Springer, 2006.

[43] "Modelware project website," 2009, http://www.ample-project.net/.

[44] "Modelplex project website," 2009, http://www.modelplex.org/.

[45] J. Bézivin, S. Bouzitouna, M. D. D. Fabro, M.-P. Gervais, F. Jouault, D. S. Kolovos, I. Kurtev, and R. F. Paige, "A canonical scheme for model composition," in *ECMDA-FA*, 2006.

[46] C. Herrmann, H. Krahn, B. Rumpe, M. Schindler, and S. Völkel, "An algebraic view on the semantics of model composition," in *ECMDA-FA*, ser. Lecture Notes in Computer Science, D. H. Akehurst, R. Vogel, and R. F. Paige, Eds., vol. 4530. Springer, 2007, pp. 99–113.

[47] Y. Smaragdakis and J. G. Siek, Eds., *Generative Programming and Component Engineering, 7th International Conference, GPCE 2008, Nashville, TN, USA, October 19-23, 2008, Proceedings*. ACM, 2008.

[48] A. Winter, R. Ferenc, and J. Knodel, Eds., *13th European Conference on Software Maintenance and Reengineering, CSMR 2009, Architecture-Centric Maintenance of Large-SCale Software Systems, Kaiserslautern, Germany, 24-27 March 2009*. IEEE, 2009.