# FPLA: A Modeling Framework for Describing Flexible Software Product Line Architecture*

Jennifer Pérez, Jessica Díaz, Juan Garbajosa, Agustín Yagüe

## 1 Introduction

Nowadays, Software Product Line (SPL) engineering [1] has been widely-adopted in software development due to the significant improvements that has provided, such as reducing cost and time-to-market and providing flexibility to respond to planned changes [2]. SPL takes advantage of common features among the products of a family through the systematic reuse of the core-assets and the effective management of variabilities across the products. SPL features are realized at the architectural level in product-line architecture (PLA) models. Therefore, suitable modeling and specification techniques are required to model variability. In fact, architectural variability modeling has become a challenge for SPLE due to the fact that PLA modeling requires not only modeling variability at the level of the external architecture configuration (see [3,4] literature reviews), but also at the level of internal specification of components [5]. In addition, PLA modeling requires preserving the traceability between features and PLAs. Finally, it is important to take into account that PLA modeling should guide architects in modeling the PLA core assets and variability, and in deriving the customized products. To deal with these needs, we present in this demonstration the **FPLA Modeling Framework.**

FPLA Modeling Framework offers tool support for (i) modeling PLAs in a graphical way by providing mechanisms to specify the external and internal variability and the different artifacts of the PLA following a view model, (ii) documenting the design decision driving the PLA solution, (iii) tracing PLAs to features allowing change impact analysis, (iv) configuring a specific product architecture (PA), and (v) translating PL specifications into code (automatically) by guarantying the traceability between PLA and code. FPLA has been used for modeling the PLAs from representative software exemplars [6] to industrial projects, such as OPTIMETER [7] which has been involved in different developments of two ITEA projects (IMPONET and NEMO&CODED) focused on Smart Grids. Specifically, to illustrate the use of FPLA, this demonstration shows several snapshots of one of the developments of OPTIMETER (see Fig. 1). OPTIMETER consisted of the development of a SPL for metering management systems in electric power networks [7]. Metering management

* INNOSEP (TIN200913849), IMPONET (ITEA 2 09030, TSI-02400-2010-103), iSSF (IPT-430000-2010-038), NEMO&CODED (ITEA2 08022, IDI-20110864) and ENERGOS (CEN-20091048).

systems capture and manage meter data from a large number of distributed energy resources, load these data in a database, support data querying and processing, and provide these data to other systems for billing, forecasting or purchasing.

## 2 FPLA Modeling Framework

FPLA supports Model-Driven Development (MDD) of SPLs by modeling feature models (adopting feature-oriented analysis), PLAs, traceability from features to PLA through architectural design decisions (ADD), and PA configurations; and also by generating AspectJ code from models. FPLA modeling is supported by a 4+1 view model that consists of the Core, Variability, Derivation, Product, and PLAK (Product-Line Architectural Knowledge) views. Fig. 1 shows the models and views of OPTIMETER. The feature model describes the features of OPTIMETER (see A). The core view (see B) allows configuring the common PLA for all the products of the SPL. This view is supported by components, such as the *Database Manager*, and **Plastic Partial Components** (PPCs) [8], such as the *Data Querying*. PPCs allows specifying the internal variation of components, in such a way that part of its behavior corresponds to the core functionality of a SPL and part of its behavior is specific of a product or set of products from that SPL. The variability view (see C) supports modeling: the **external variability** by adding and removing attachments among components, and the **internal variability** of PPCs. The PPCs *DataQuerying* and *Data Loader* implement the variability for the different data storing technologies. The variability of a PPC is specified using *variability points* (see the AVP *clustering)* which hook fragments of code to the PPC known as *variants* (see the variants *HadoopMAP/REDUCE* and *RAC)*. Finally, *weavings* specify where and when to extend PPCs through the use of variants (see C). The PLAK view (see D) allows one to trace features to PLA through design decisions and to analyze the impact of changes. The derivation and product views (see E) allow architects to select the variants of a specific product of the SPL and to generate its PA. In this case, two specific products were configured and their code skeletons were automatically generated by automatically composing the required source-code from external sources. FPLA is a plug-in of Eclipse that has been developed by using the infrastructure provided by the Eclipse Modeling Framework for the modeling support, and the Epsilon Generation Language of the Epsilon Generative Modeling Technologies research project for the model-to-code transformation. A complete video demonstration is available at https://syst.eui.upm.es/FPLA/examples.

### REFERENCES


1. Pohl K., Böckle G., van der Linder F.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag (2005)
2. Schmid, K., Verlage, M.: The economic impact of product line adoption and evolution. IEEE Softw. 19(4) pp. 50–57 (2002)
3. Schaefer I., et al.: Software diversity: state of the art and perspectives. International Journal on Software Tools for Technology Transfer 14(5), Springer-Verlag pp. 477-495 (2012)

4. Chen, L., Ali Babar, M., Ali, N.: Variability management in software product lines: a systematic review. In Proc. of the Int. Software Product Line Conference, pp. 81-90 (2009)
5. Bachmann F., Bass L.: Managing variability in software architectures. In Proc. of the symposium on Software reusability. New York, NY, USA: ACM, pp. 126–132. (2001)
6. Díaz, J., Pérez, J., Garbajosa, J., & Wolf, A: Change impact analysis in product-line architectures. In Software Architecture, volume 6903 of LNCS, Springer pp. 114-129 (2011).
7. Díaz, J, Pérez J, Garbajosa J, Yagüe A.: Change-Impact driven Agile Architecting. In Proc. of the 46th Hawaii Int. Conference on System Sciences (HICSS '13), IEEE (2013)
8. Pérez, J., Díaz, J., Soria, C. C., Garbajosa, J.: Plastic partial components: A solution to support variability in architectural components. In Proc. WICSA/ECSA 2009, IEEE (2009)

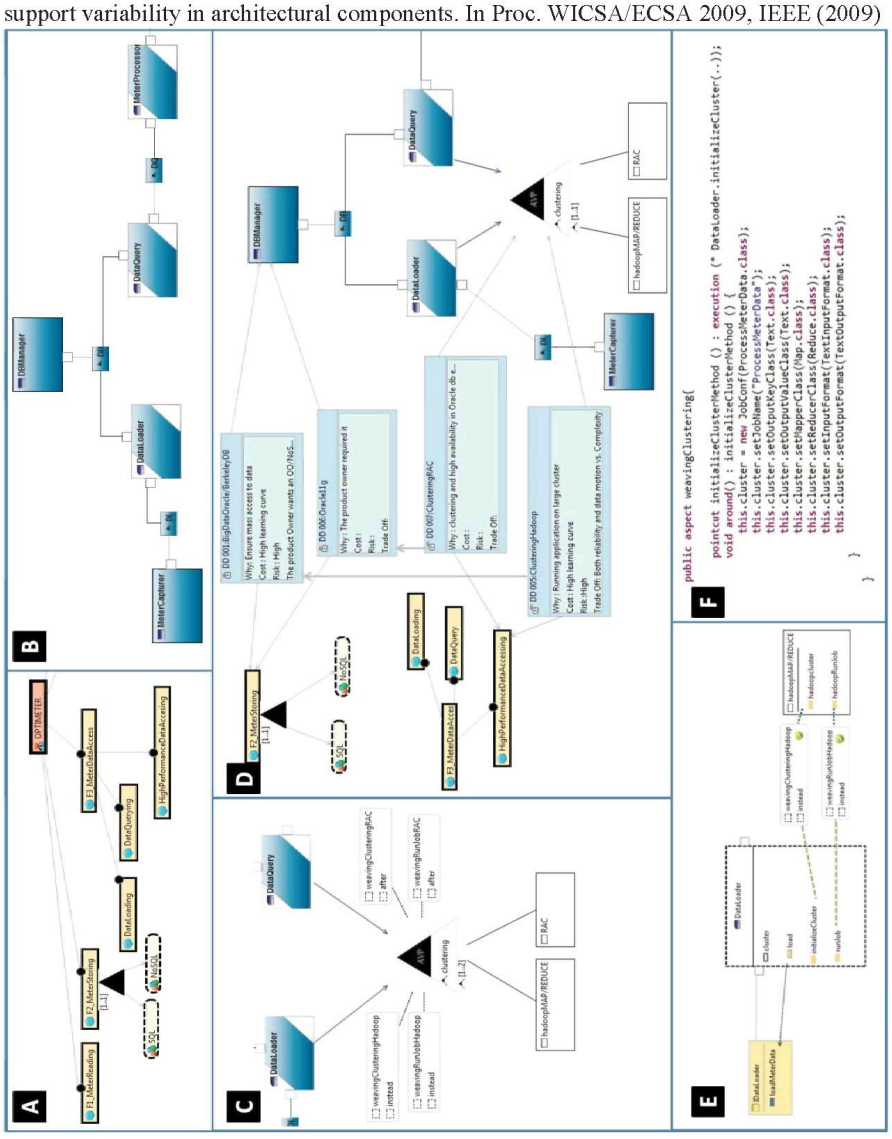


**Fig. 1.** FPLA Modeling Framework – OPTIMETER project