On the Evaluation and Improvement of Feature-based Configuration Techniques in Software Product Lines

A. Hubaux
Advisor: P. Heymans
PReCISE Research Centre
Faculty of Computer Science
University of Namur
{ahu, phe}@info.fundp.ac.be

Abstract

Our work builds upon previous research on software product lines and formal analysis of feature diagrams carried out since several years at the University of Namur. This PhD thesis aims at evaluating and improving existing feature-based configuration techniques to ease their uptake by practitioners and their integration into the software product line engineering process. The affordability and scalability of the delivered languages, methods and tools are major concerns. Evaluation will take place in the context of an open source development project.

1 Introduction

Software Product Line Engineering (SPLE) [21] is the software engineering paradigm institutionalising reuse throughout product development. Central to the SPLE paradigm is the modelling and management of variability [26]. An increasingly popular family of languages for variability management is the one of Feature Diagrams (FD) [17]. FD languages come in various flavours, a large part of which have been surveyed, theoretically compared and formalized in [24, 25]. They offer constructs to capture commonalities and variabilities, and to represent dependencies between features. Ultimately, they drive product configuration and determine the allowed combinations of features in the SPL.

Considering a set of requirements and an SPL specification, the configuration task refers to, in our case, the process of completely specifying and validating a product given an FD containing unresolved variation points. In a realistic project, the configuration process is a lengthy undertaking piloting product engineering. It may take up to several months and involve different stakeholders with various concerns [22]. This, together with the large size of industrial applications, calls for ways (1) to prescribe the configuration task and (2) to distribute it among stakeholders [5, 13, 3].

In this PhD thesis, we intend to evaluate and improve FD-based configuration techniques to bring them closer to the needs of practitioners and to ease product development. Special attention will be given to the affordability and scalability of the delivered languages, methods and tools. The evaluation and validation of our results will chiefly take place in the context of open source projects.

The remainder of the paper is structured as follows. Sec. 2 describes the state of the art of this research. Sec. 3 presents the objectives of the thesis. Sec. 4 formulates its two main research questions, Sec. 5 outlines the research method and Sec. 6 discusses the related papers already published by the first author.

2 State of the art

As previously mentioned, the implementation of the configuration process and the configuration itself are labour-intensive activities. Several techniques based on FDs exist, the most common being briefly discussed below.

Unguided configuration. In unguided configuration, the FD is configured without following a well defined approach. Usually, the features required by a product are merely selected in a top-down fashion.

Staged configuration. In staged-configuration [6], the product configuration is performed in stages, each stage eliminating configuration choices in the FD. The elimination is achieved through a series of successive specialisation stages targeting specific parts of the model and strictly reducing the available variability until the final configuration is reached. The need to work with stages to “emphasise precision in the descriptions of partially designed programs” has already
been advocated by Parnas in [19]. Different dimensions can be associated to the configuration stages, e.g. the times of the product lifecycle, the target systems or subsystems of the configured product, and the roles of the parties performing the configuration.

**Multi-level staged configuration.** Multi-level staged configuration (MLSC) extends staged configuration by adding configuration levels, where each level is represented by an FD linked to the other level’s FDs through inter-level links [5]. The configuration process is decomposed in two iterative steps: (1) manual configuration of the level \( l_i \) FD and (2) automated specialisation of the level \( l_{i+1} \) FD based on the \( l_i \) configuration and the inter-level links. By using levels and stages, one modularises FDs into coherent feature sets. Thereby, one increases the reuse potential of FDs, the relevance of stages/levels to stakeholders and enhances the scalability of FD-based configuration techniques. Levels also allow the progressive addition of variation points and, thereby, follow the principle of stepwise refinement [10].

**Probabilistic configuration.** Probabilistic feature models (PFM) [7] augment traditional FDs with soft constraints and legal joint probability distributions (JPD). Soft constraints specify conditional probabilities on the selection of features and should be obeyed, on average, by all product configurations. JPD assign a probability to each possible configuration of the SPL. The combination of soft constraints and JPD allows, for instance, to infer recommended or default configuration choices as the configuration process progresses. The approach proposed by Czarnecki et al. [7] also comes with a mining algorithm supporting the generation of JPD and soft constraints.

**Dynamic configuration.** The recent line of research on dynamic SPL (e.g. [18]) is shaping a new way of tackling feature modelling and configuration. The fundamental distinction between static and dynamic configuration techniques is that the latter considers the evolution of the product configuration at runtime. In dynamic configuration, not only the static constraints of the FD must hold, but also the dynamic constraints determining the allowed transitions between the product configurations. In contrast to static configuration, both the context evolution and the stakeholder actions entail automated reconfigurations of the running product. To deal with context evolution, some directly use FDs to model context [14], some annotate FDs with contextual information [9] and some extend FD languages [12].

The configuration of a product is only the initial step of its lifetime. Product configuration needs to be considered in the more general process of configuration management (CM) [1] whose purpose is to control product elaboration and change. To a larger extent, the whole SPL should be endowed with an adapted CM system. However, genuine CM is most often overlooked, incomplete or ad hoc in SPL practices, mostly because of the task complexity [2]. And too little effort is dedicated to the integration of FDs as basic reasoning units for CM.

### 3 General objective

To date, the most generic and precisely specified FD-based product configuration technique is the MLSC from Czarnecki et al. [5]. The flexible decomposition markedly improves the scalability of FD-based engineering techniques and opens many new application and research paths. Open perspectives are, for instance, (1) the specification of level ordering, (2) the prevention of illegal configuration steps, (3) the management of multiple SPLs or (4) the reuse of existing FDs and their derived configurations. The advancement of MLSC and its applicability to real-world problems are our top priority objectives.

The growing demand for automated and on-the-fly product configuration, as experienced with our industrial partners [4], leads us to extend MLSC with concepts of probability and dynamic product configuration. To this end, we also intend to integrate all the obtained results in an existing tool suite supporting SPLs endowed with advanced automated reasoning capabilities. This tool should allow us to leverage the integration of MLSC in CM by providing a scalable solution along with a robust reasoning engine.

### 4 Research questions

The two main research questions addressed in this PhD thesis can be formulated as follows.

**RQ1** *What obstacles do we encounter when applying MLSC to real-world SPL configuration problems?* This question will notably address the applicability of the separation of variability modelling and resolution into stages. Questions like “does MLSC offer adequate modelling constructs?”, “does MLSC offer sufficient methodological support?” and “are the current automations appropriate and sufficient?” are the kind of questions that will have to be covered.

**RQ2** *How to improve support for MLSC and adjust it to CM?* Based on the results obtained in RQ1, we will investigate means to (1) improve and extend MLSC and (2) bind it with CM techniques.
5 Research method

Our research method consists of four tasks.

Task 1 State of the art. Although our focus will be on MLSC in RQ1, we will carefully survey a wider range of configuration approaches relevant to SPL engineering in SPLE in the hope that some will provide inspiration to answer RQ2. Therefore, we will not limit our investigations to the approaches currently used in SPL but will also consider solutions from other fields, most prominently artificial intelligence.

Task 2 Problem identification. Here, we aim to answer RQ1 empirically according to the highest standards in case study research. We will apply MLSC to one or more case studies that will be prepared, performed and analysed according to the research method described in [20].

Our major case study is PloneMeeting. PloneMeeting\(^1\) is part of the PloneGov\(^2\) international Open Source project intending to promote secure, collaborative, and evolutive eGovernment web applications. This Belgian government-initiated project offers advanced meeting management functionality to local and national authorities. Some smaller applications might be envisaged depending on progress, opportunities and results obtained with PloneMeeting.

Task 3 Improvement of MLSC. Based on the results of Task 2, RQ2 will be answered by making the necessary changes and additions to the MLSC FD language, method and tool support. More specifically:

- language improvements will build upon recent results [25];
- methodological improvements will take the form of reusable and composable “method fragments”. We also intend to populate an existing software engineering method base [23];
- tool support improvements will be achieved by contributing to an FD-based tool chain developed in cooperation with other researchers of the lab as well as research partners. The tool development effort will be minimized by reusing existing components as much as possible, most notably off-the-shelf solvers for reasoning support and a metaCASE [11] for model visualisation.

Task 4 Evaluation of MLSC. Evaluation of our improvements to MLSC will be achieved by applying the improved language, tool and method support to the same case study used to elicit the challenges, observe and assess the benefits.

These tasks should be orchestrated as follows.

Task 1 will take place essentially at the beginning of the project while the review of the state of the art will be continuously updated throughout the project with the most recent advances.

Tasks 2-4 will be performed sequentially, in several iterations. Each iteration will focus on a specific part of the case study or on a specific aspect of MLSC. Working iteratively will allow us to reduce the risk of the doctoral research: (1) each iteration will act as a pilot for the next one; (2) each iteration will expedite the production of a self-contained deliverable.

6 Current achievements

In previous work [8], we introduced the idea of using SPL principles to engineer the PloneGov project. Our conclusion showed a number of organisational and technical problems that had to be tackled such as handling the distributed developers and managing the already existing variability. In [16], we focused on the identification and modelling of the variability in PloneMeeting. Since no variability model formed existed, the variation points had to be reverse engineered, which resulted in separate FDs for the different concerns we identified. The results we subsequently obtained are four modelling challenges identified during the reverse engineering of PloneMeeting [15].

The workarounds we proposed to tackle these challenges still have to be systematically applied to concrete cases and properly assessed. These papers are primary contributions to Task 2.

Although the cardinality-based FD language supporting MLSC received various formal semantics, the MLSC process never received one. In [3], we presented a semantics for MLSC that builds on our earlier work on formal FD semantics [25] to which it adds the concepts of level and configuration path. We notably discovered some important properties that an MLSC process should possess and that a configuration tool should guarantee. This contribution builds upon the work carried out in Task 1 and Task 2, and is a first step towards the completion of Task 3.

While there exists several SPL processes for static systems, there has been less focus on dynamically adaptive systems. In [4], we observed the limitations related to domain engineering in SPL and identified what fundamental concepts must be rethought in order to achieve SPL for dynamically adaptive systems. Although not centered on FDs, this paper raises several research questions to consider when dealing with the dynamic reconfiguration of FDs.
7 Acknowledgments

This work is sponsored by the Interuniversity Attraction Poles Programme of the Belgian State of Belgian Science Policy under the MoVES project.

References


