The Journal of Systems and Software 103 (2015) 392–411

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications

Gustavo G. Pascual [a,*], Roberto E. Lopez-Herrejon [b], Mónica Pinto [a], Lidia Fuentes [a], Alexander Egyed [b]

[a] Department of Languages and Computer Science, University of Málaga, 29071 Málaga, Spain
[b] Institute for Systems Engineering and Automation, Johannes Kepler University Linz, Austria

ARTICLE INFO

Article history:
Received 1 December 2013
Revised 11 December 2014
Accepted 17 December 2014
Available online 3 January 2015

Keywords:
DSPL
Dynamic reconfiguration
Evolutionary algorithms

ABSTRACT

Mobile applications require dynamic reconfiguration services (DRS) to self-adapt their behavior to the context changes (e.g., scarcity of resources). Dynamic Software Product Lines (DSPL) are a well-accepted approach to manage runtime variability, by means of late binding the variation points at runtime. During the system's execution, the DRS deploys different configurations to satisfy the changing requirements according to a multiobjective criterion (e.g., insufficient battery level, requested quality of service). Search-based software engineering and, in particular, multiobjective evolutionary algorithms (MOEAs), can generate valid configurations of a DSPL at runtime. Several approaches use MOEAs to generate optimum configurations of a Software Product Line, but none of them consider DSPLs for mobile devices. In this paper, we explore the use of MOEAs to generate at runtime optimum configurations of the DSPL according to different criteria. The optimization problem is formalized in terms of a Feature Model (FM), a variability model. We evaluate six existing MOEAs by applying them to 12 different FMs, optimizing three different objectives (usability, battery consumption and memory footprint). The results are discussed according to the particular requirements of a DRS for mobile applications, showing that PAES and NSGA-II are the most suitable algorithms for mobile environments.

1. Introduction

Mobile applications demand runtime reconfiguration services that make it possible for them to self-adapt their behavior to the continual contextual changes that occur in their environment (e.g., scarcity of available resources) (Brataas et al., 2011; Capilla et al., 2014; Floch et al., 2013; Mizouni et al., 2014). In some applications, the reconfiguration can be made to maintain a certain quality of service (QoS) the user requires, in others the reconfiguration can be made to offer a personalized service to the user such as location-based services, and even to provide the user's personal suggestions based on the recognized activity and context (Mizouni et al., 2014). For instance, the battery level of the mobile device may be a critical decision parameter in changing the behavior of an application if the goal of this reconfiguration is to extend the lifespan of the battery and hence the device connectivity (Mizouni et al., 2014).

One accepted approach to manage the runtime variability of applications is the Dynamic Software Product Line (DSPL) approach. DSPLs produce software capable of adapting to changes, by means of binding the variation points at runtime (Hallsteinsen et al., 2008). This requires to model the elements that could be adapted dynamically as dynamic variation points and to generate, at runtime, the different variants of the DSPL.

A *runtime configuration* is the set of values assigned to the dynamic variation points, defining a member of the dynamic SPL. If a change in the execution context is detected, then a reconfiguration service should generate a new runtime configuration adapted to the new context. Therefore, the reconfiguration service should be continuously generating optimum *runtime configurations* adapted to the changing context. But, which and how many objectives should be considered in the generation of optimum configurations? In the case of mobile applications, multiple objectives should be taken into account like, loss of network connectivity, drastic increase or reduction of the available resources (e.g., battery, memory, CPU) or the user preferences about quality of service (QoS). For instance, if a user wants to save battery life in a mobile phone application, the reconfiguration service should generate a configuration with a low battery consumption while trying to keep the quality of service as high as possible.

Harman et al. (2014) show how Search-Based Software Engineering (SBSE) has been successfully applied by different approaches to SPLs. In this paper, we demonstrate that SBSE, and in particular

* Corresponding author. Tel.: +34665372568.
E-mail addresses: gustavo@lcc.uma.es (G.G. Pascual), roberto.lopez@jku.at (R.E. Lopez-Herrejon), pinto@lcc.uma.es (M. Pinto), lff@lcc.uma.es (L. Fuentes), alexander.egyed@jku.at (A. Egyed).

http://dx.doi.org/10.1016/j.jss.2014.12.041
0164-1212/

multiobjective evolutionary algorithms (MOEAs), can be used to solve the problem of generating a valid configuration of a DSPL at runtime. Runtime configurations of a DSPL are generated from a variability model, which specifies the common and the variable elements of the dynamic product line. Most of DSPL approaches use Feature Models (FMs) as the de facto standard to specify the commonalities and variabilities of the product line in terms of *features* and *constraints* between them (Cetina et al., 2008; Dinkelaker et al., 2010; Rosenmüller et al., 2011; Trinidad et al., 2007; White et al., 2007). In this case, runtime configurations are defined in terms of features, and are known as dynamic feature model configurations. This means that the set of valid configurations that can be deployed during the execution of the application is then determined by the FM. Therefore, the MOEA needs the FM that models the dynamic variation points. With an FM available at runtime, an MOEA can generate valid variants of the application adapted to the context changes.

In this paper, we explore the use of MOEAs to generate at runtime the variants of the DSPL that fit the current execution context with regard to several optimization criteria such as battery consumption or usability. However, in order to be suitable for our reconfiguration service, the employed algorithms should satisfy several requirements such as:

1. *Fast enough execution time*. Reconfiguring the application should not harm excessively the user experience. Furthermore, since we are focusing on mobile devices, the response time of the optimization algorithm should be of few seconds.
2. *Generate only valid configurations*. While in different kinds of approaches (Sayyad et al., 2013) configurations which do not satisfy all the constraints can be useful, it is not appropriate for a reconfiguration service to deploy invalid configurations of the application. Therefore, the optimization algorithm should only return valid configurations (i.e., configurations which satisfy all the constraints).
3. *Multiobjective optimization*. Generally, it is necessary to generate configurations which are optimal regarding several criteria (e.g., battery consumption, usability).
4. *Support for DSPLs in mobile applications*. In DSPLs the number of variation points that need to be managed is usually much lower than in SPLs at design time. The reason is that in DSPLs only the variations points that can change at runtime need to be considered. Therefore, unlike design time SPLs, in DSPLs only a subset of the FM variation points are considered; the rest of them are fixed at design time. For instance, the variability model of the Linux kernel (Lotufo et al., 2010) contains more than 6000 features, but although some of these variation points can be decided at runtime (e.g., I/O scheduler, CPU frequency governor), many points are decided at design time because they depend on the hardware of the target device (e.g., CPU architecture, CPU model, virtualization support, cryptography hardware). Moreover, since our approach focuses on the development of mobile applications, the DSPLs managed in our approach would generally be even smaller than DSPLs for desktop applications.

In this sense, several algorithms have been defined which are able to obtain an optimal configuration of an FM according to a given optimization criteria (Benavides et al., 2010; Guo et al., 2011; Li et al., 2012; Sayyad et al., 2013; Soltani et al., 2012; White et al., 2009a, 2009b). However, none of them are suitable for reconfiguration in mobile devices mainly because they were proposed to optimize the configuration of Software Product Lines (SPLs), being designed to be used only at design time. Furthermore, only Sayyad et al. (2013) support multiobjective optimization (i.e., generating configuration which are optimal regarding different criteria simultaneously), but not for DSPLs.

Our experiments have been performed using six existing MOEAs algorithms, which have been applied to 12 FMs in order to optimize three objectives (usability, battery and memory). We have evaluated these algorithms according to the requirements imposed by a reconfiguration service for mobile applications and the results show that PAES and NSGA-II are the most suitable MOEAs to be used in mobile environments. They have the lowest execution time and, at the same time, they satisfy the rest of requirements.

Following the Introduction, the rest of the paper is organized as follows. The backgrounds to DSPLs and FMs are presented in Section 2. After this, the related work is discussed in Section 3 and the realization of our reconfiguration mechanism is described in Section 4. Then, the experimental setup and the evaluations results are presented in Sections 5 and 6 respectively. Finally, in Section 7 the evaluation results and threats to validity are discussed, while our conclusions and future work are described in Section 8.

2. Background

In this section we show the basics of DSPLs and FMs, which are used in our approach to reconfigure mobile applications at runtime.

2.1. Dynamic software product lines

An SPL is "a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way."[1] DSPLs redefine existing SPL engineering processes by moving them to runtime, with the goal of ensuring that system adaptations lead the system to a valid state. So, in SPLs the engineering processes are able to generate several systems of the same family at design time, but a DSPL is considered a single system able to adapt its behavior at runtime.

The variability model is the central artifact for both SPLs and DSPLs for formally specifying their commonalities and variabilities. The engineering processes of SPLs generate products by selecting concrete values for the variable characteristics specified in the variability model. This means that the SPL engineer binds the variation points at design time considering the requirements of the intended product. In contrast, in DSPLs the variability model describes the potential range of variations that can be produced at runtime for a single product, i.e., *the dynamic variation points* already defined in the introduction. Then, as the set of dynamic variation points drive system adaptation, they must be available to be consulted at runtime by a reconfiguration service. But these dynamic variation points must make reference to the system architectural components. So, in DSPLs the system architecture supports all the possible adaptations defined by the set of dynamic variation points (Hallsteinsen et al., 2008).

So, as part of a DSPL definition the engineer must identify: (i) the range of potential adaptations supported by the system in terms of architectural components; (ii) define an explicit representation of the valid configuration space of the system; (iii) the context changes that may trigger an adaptation, i.e., the criteria (which can have several objectives) to initiate a reconfiguration or Decision Making Process (DMP); and (iv) the set of possible reactions to context changes that should be supported by the system. However, the way these issues are implemented may differ greatly, as will be shown in Section 3.1.

Since for the majority of DSPLs the decision to initiate a reconfiguration is made autonomously by the system (not by a human), they are considered a good technology for developing self-adapting systems such as mobile applications. In this sense, most of DSPL approaches share some common capabilities and goals with the Autonomic Computing (AC) paradigm (IBM, 2005) such as the monitoring of the environment and the generation of successive configurations.

[1] http://www.sei.cmu.edu/productlines/