The Journal of Systems and Software 91 (2014) 3–23

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

An overview of Dynamic Software Product Line architectures and techniques: Observations from research and industry

Rafael Capilla[a,*], Jan Bosch[b], Pablo Trinidad[c], Antonio Ruiz-Cortés[c], Mike Hinchey[d]

[a] Rey Juan Carlos University, Madrid, Spain
[b] Chalmers University of Technology, Gothenburg, Sweden
[c] University of Seville, Seville, Spain
[d] Lero – The Irish Software Engineering Research Centre, Limerick, Ireland

ARTICLE INFO

Article history:
Received 17 November 2012
Received in revised form
16 December 2013
Accepted 23 December 2013
Available online 8 January 2014

Keywords:
Dynamic Software Product Lines
Dynamic variability
Software architecture
Feature models

ABSTRACT

Over the last two decades, software product lines have been used successfully in industry for building families of systems of related products, maximizing reuse, and exploiting their variable and configurable options. In a changing world, modern software demands more and more adaptive features, many of them performed dynamically, and the requirements on the software architecture to support adaptation capabilities of systems are increasing in importance. Today, many embedded system families and application domains such as ecosystems, service-based applications, and self-adaptive systems demand runtime capabilities for flexible adaptation, reconfiguration, and post-deployment activities. However, as traditional software product line architectures fail to provide mechanisms for runtime adaptation and behavior of products, there is a shift toward designing more dynamic software architectures and building more adaptable software able to handle autonomous decision-making, according to varying conditions. Recent development approaches such as Dynamic Software Product Lines (DSPLs) attempt to face the challenges of the dynamic conditions of such systems but the state of these solution architectures is still immature. In order to provide a more comprehensive treatment of DSPL models and their solution architectures, in this research work we provide an overview of the state of the art and current techniques that, partially, attempt to face the many challenges of runtime variability mechanisms in the context of Dynamic Software Product Lines. We also provide an integrated view of the challenges and solutions that are necessary to support runtime variability mechanisms in DSPL models and software architectures.

1. Introduction

Nowadays, many organizations have adopted a Software Product Line (SPL) development approach for building their own product portfolio with higher quality and at lower cost. The reduction in time-to-market conditions and the launching of new products or releases more quickly is one of the key drivers for the adoption of an SPL strategy. However, as more and more systems require the adaptation to different context conditions or working under better quality conditions, a number of challenges have emerged that static or conventional SPL approaches cannot provide. Companies producing large-scale software and embedded systems that require highly configurable options, many of them configured by end-users at post-deployment time, are facing new challenges to adapt conventional SPLs to more dynamic approaches able to handle runtime concerns. For more than twenty years, companies have launched successful product lines to build software products faster and cheaper with higher quality. Several experiences in the product line hall of fame (Van der Linden et al., 2007) have demonstrated the benefits of the adoption of a product line approach.

Today, software-intensive embedded system families demand highly configurable and adaptable mechanisms many of them managed at runtime. Much of these systems are designed using software architectures that include runtime mechanisms for adaptation purposes, but stringent quality and business requirements drive the motivation for dynamic extensions, runtime reconfiguration, optimized performance, and autonomous behavior, amongst others. For instance, mobile and service-based applications have grown exponentially, affecting a large number of users that use software and devices that demand runtime post-configuration facilities. Customizable and context-aware services drive the current trend of cloud and SOA-based systems for selecting the most suitable service located anywhere, and context-ware properties play a role to adjust the service to new context conditions. Furthermore, binding services dynamically, demands rebinding and multiple binding capabilities by introducing variants for the dynamic selection of services, as this feature is not supported by conventional product lines. Combined approaches mixing SPL and SOA (Gomaa

* Corresponding author. Tel.: +34 91 4888119.
E-mail address: rafael.capilla@urjc.es (R. Capilla).

0164-1212/$ – see front matter © 2014 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.12.038

and Hashimoto, 2011) deal with the challenges to incorporate runtime mechanisms addressing context-awareness properties of services where family members need to evolve after deployment. In addition, the variety of self-adaptive, self-management, and autonomous systems (e.g., robots, unmanned vehicles) also require autonomous behavior and automatic decision-making, often based on a set of system options that can be activated and deactivated according to varying context conditions or user preferences.

During the past 15 years, traditional SPL approaches (Bosch, 2000; Clements and Northrop, 2001; Pohl et al., 2005) have successfully addressed the development of system families from a common architecture, maximizing reuse, and exploiting the variability of systems to produce cheaper and higher-quality products in less time. However, in a changing world, the increasing need for adaptive software demands autonomous behavior and self-management properties bringing new challenges for dynamic adaptation in system families. As variability becomes more dynamic, there is a clear to move to recent development approaches like Dynamic Software Product Lines – DSPLs (Hallsteinsen et al., 2008; Hinchey et al., 2012; Bencomo et al., 2012) as an emerging paradigm to handle variability at runtime and at any time.

1.1. Target audience and summary of contributions

The target audience of this work are researchers in the software architecture and product line engineering fields and also those professional SPL designers and engineers who need to migrate from a conventional SPL to a DSPL or develop software products that demand dynamic variability mechanisms from scratch. Therefore, we elaborate a list of major challenges and solutions that cover the full spectrum of a DSPL. Consequently, we provide our observations from the trenches from research and industry about the current state-of-the-art of the DSPL technologies, and more specifically the need for runtime variability mechanisms. As we have observed that there are partial solutions that work in isolation, one of the key contributions of this work is a framework that encompasses the following:

1. the required DSPL properties consisting of runtime variability support, multiple and dynamic binding times of products, and a way to model context properties,
2. the organizational changes a DSPL should have compared to conventional SPLs, and
3. the suggested solutions for each DSPL challenge we describe as relevant pieces required for launching a DSPL.

These relevant pieces or suggested solutions go from runtime variability mechanisms for context adaptation purposes to optimization mechanisms aimed to provide the best or the optimal solution in a given context.

The remainder of this paper as follows. Section 2 provides the related work regarding DSPLs. In Section 3 we characterize DSPL elements and processes in a framework. In Section 4 we outline the role and challenges of runtime variability mechanisms and other related issues of DSPL research that we will address in the remainder of the paper. Section 5 describes the set of technical solutions necessary to implement and use dynamic variability. In Section 6 we outline examples of use of the proposed solutions in various application domains. Section 7 summarizes the discussion the major results derived from this research and Section 8 draws the conclusions and future work.

2. Related work

The current limitations of today's SPL models rely on its inability to change the structural variability at runtime, provide the dynamic selection of variants, or handle the activation and deactivation of system features dynamically and/or autonomously. Because the development of runtime reconfigurable assets is still innovative and not fully investigated in the SPL area (Gomaa and Hussein, 2004; Lee and Kang, 2006; Schmid and Kröher, 2009), there is an important need to support the dynamic properties of systems and post-deployment capabilities as a new promising research and development area using DSPL models.

At present, emerging efforts that suggest the use of DSPL-based models focus on the implementation of runtime variability mechanisms and domain-specific languages for reconfiguring software system options (e.g., Cetina et al., 2009a uses PervML, a domain-specific language for pervasive systems for reconfiguring a smart home using a DSPL), in particular for specific application domains such as smart and autonomic homes, robots, mobile software, or service-based systems, amongst others. For instance, Cetina et al. (2010) address the challenges of triggering runtime reconfigurations and understanding the effects of such reconfiguration when prototyping a DSPL. The authors describe the case of a Smart Hotel and how Dynamic Software Product Lines can assist a system to determine the steps during reconfiguration operations at runtime, such as the activation and deactivation of system features dynamically and based on varying context conditions. Hallsteinsen et al. (2008) discuss the role of DSPLs in emerging domains, such as ubiquitous computing, robotics, and life-support services, amongst others. Because it is impossible to predict all the expected variability in a product line, Dynamic Software Product Lines should be able to produce adaptable software where runtime variations can be managed in a controlled manner.

Many software applications exhibit random behavior at runtime, and the service computing area is another promising research field where DSPLs have a niche to exploit runtime variability mechanisms for the dynamic selection of services, often based on changing QoS (Quality-of-Service) properties. Several authors (Hallsteinsen et al., 2009; Gomaa and Hashimoto, 2011; Istoan et al., 2009) highlight the importance of using a DSPL for modeling and implementing service-based systems by encoding dynamic variability as a decision model for the selection of the services at the latest binding. However, as the computational complexity during variant selection may be a drawback for performance in certain systems that have strong real-time requirements, a DSPL should be able to handle the necessary adaptations and current reconfiguration tasks after the original deployment. For instance, the Service Component Architecture (SCA) architecture style, which reconciles SOA and Component-Based Software Engineering (CBSE), and SCA platforms can be used to assemble assets dynamically in a running system (Parra et al., 2009). Bencomo et al. (2010) address the research theme "*when to adapt and how to adapt?*" in order to meet the demand of postponement of decisions on software adaptations required by dynamic environments and users.

Dynamic product lines are aimed also at binding features dynamically and to supporting multiple binding times and one or more variability mechanisms (Abbas et al., 2011). A DSPL nicely integrates the adaptation of assets and products dynamically in changing contexts, which helps products to evolve autonomously when the environment changes and provides self-adaptive and optimized reconfiguration. Additionally, a DSPL may exploit knowledge and context profiling as a learning capability for autonomic product evolution by enhancing self-adaptation (Abbas et al., 2011). However, in the era of post-deployment evolution (Malek et al., 2012), where embedded systems can change and be deployed several times, DSPLs offer a solution for software-intensive and embedded system families as they provide dynamic variability mechanisms. There are many embedded systems and application domains where runtime variability can play a key role in order to support the "autonomic" condition (Kephart and Chess, 2003) and