Contents lists available at ScienceDirect

Science of Computer Programming

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

Defining variability in activity diagrams and Petri nets[☆]

André Heuer[a,*], Vanessa Stricker[a], Christof J. Budnik[c], Sascha Konrad[d], Kim Lauenroth[b], Klaus Pohl[a]

[a] *Paluno - The Ruhr Institute of Software Technology, University of Duisburg-Essen, Gerlingstr. 16, 45127 Essen, Germany*
[b] *adesso AG, Stockholmer Allee 24, 44269 Dortmund, Germany*
[c] *Siemens Corporate Research, 755 College Road East, Princeton, NJ 08540, USA*
[d] *Plainsboro, NJ, USA*

A R T I C L E I N F O

Article history:
Received 16 February 2011
Received in revised form 16 May 2012
Accepted 12 June 2012
Available online 23 June 2012

Keywords:
Variability
Variability in activity diagrams
Documenting variability
Quality assurance for product line engineering

A B S T R A C T

Control flow models, such as UML activity diagrams or Petri nets, are widely accepted modeling languages used to support quality assurance activities in single system engineering as well as software product line (SPL) engineering. Quality assurance in product line engineering is a challenging task since a defect in a domain artifact may affect several products of the product line. Thus, proper quality assurance approaches need to pay special attention to the product line variability. Automation is essential to support quality assurance approaches. A prerequisite for automation is a profound formalization of the underlying control flow models and, in the context of SPLs, of the variability therein.

In this paper, we propose a formal syntax and semantics for defining variability in Petri nets. We use these extended Petri nets as a foundation to formally define variability in UML activity diagrams; UML activity diagrams serve as a basis for several testing techniques in product line engineering. We illustrate the contribution of such a formalization to assurance activities in product line engineering by describing its usage in three application examples.

1. Introduction

Performing quality assurance (QA) activities as early as possible is important for any development project [2] but especially in product line engineering (PLE) since defects in a domain artifact can affect several products of the product line [3]. In PLE two development processes are differentiated: domain engineering and application engineering [4].

During domain engineering, the domain artifacts of the product line are developed. These domain artifacts contain variability, i.e., they contain modification possibilities to address different customer needs [4]. The variability enables the reuse of domain artifacts during application engineering.

QA is conducted in both processes. In domain engineering, the quality of domain (or reusable) artifacts (e.g., requirements, design, or implementation artifacts) has to be assured. In application engineering, the quality of each derived product of the product line has to be assured. It is commonly agreed that QA in domain engineering requires special attention, since a defect in a domain artifact can affect several products of a product line and can lead to high costs for defect correction [3,4].

Variability in domain artifacts increases complexity since a variable domain artifact represents not only a single artifact, but a set of artifacts (e.g., a variable requirement represents a set of requirements) [5]. Due to this complexity, QA for PLE is a

[☆] This paper was partially funded by the DFG grants PO 607/2-1 IST-SPL and PO 607/4-1 KOPI and is an extended version of our paper presented in Heuer et al. (2010) [1].
[*] Corresponding author.
E-mail address: andre.heuer@paluno.uni-due.de (A. Heuer).

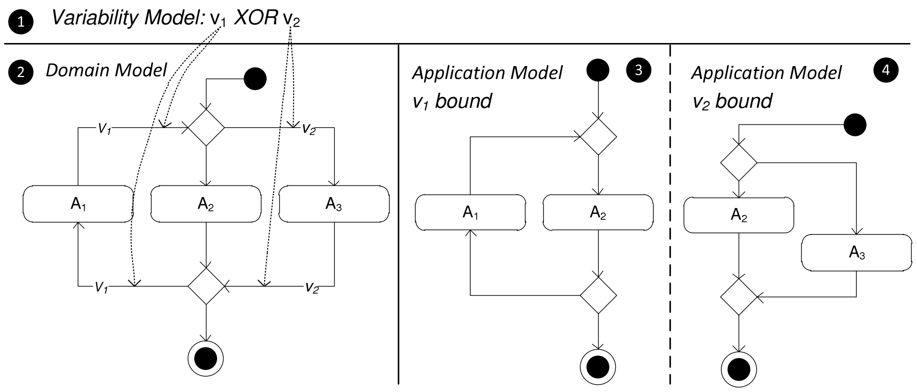
1 Variability Model: v1 XOR v2
2 Domain Model
v1
v2
A1
A2
A3
v1
v2
Application Model v1 bound
3
A1
A2
Application Model v2 bound
4
A2
A3

Fig. 1. Examples of domain and variability models.

challenging task since QA approaches from single system engineering cannot simply be applied to domain artifacts without adaptation [6].

To minimize the time, costs, and effort for QA, a high degree of automation is desirable. This requires a formal syntax and semantics of the underlying test models to enable an algorithmic as well as automated analysis and usage of these models. In PLE, the formal syntax and semantics have to be extended to consider variability.

1.1. Challenges of quality assurance in software product line testing

Many QA approaches use control flow models as test models (cf., e.g., [2,7]) and thus the challenge is to include variability into the formal syntax and semantics of these control flow models. The control flow of a system can be modeled, for example, by control flow graphs or flowcharts (cf. [2]). However, these models do not support concurrency, which is a highly relevant property of today's software systems. UML activity diagrams [8] are widely accepted by academia and industry as a standard for modeling concurrent control flows in single system engineering (cf., e.g., [7]) and in PLE (cf., e.g., [9,10]).

Fig. 1, for example, shows an activity diagram that contains variability (cf. ❷ in Fig. 1).[1] A variant is a representation of a particular instance of a variable item in the real world or a variable property of such an item [4]. To document that an edge (and related activities) is variable and thus selectable for a derived product, the edges of the activity diagram are related to variants of the variability model. Thus, the variability model states that either variant v_1 or v_2 has to be part of a product. For example, if variant v_1 is selected, only the related edges become part of an application activity diagram and the edges related to variant v_2 are deleted (❸). To illustrate why approaches from single system engineering cannot be applied to activity diagrams that contain variability, we assume an automated approach for deriving test cases based on a statement by the test cases at least once. If such an algorithm is executed on the domain model shown in Fig. 1, a resulting test case scenario would be the flow $A_2 \rightarrow A_1 \rightarrow A_3$. Taking only the activity diagram (❷) into account (without the variability model ❶), this test case scenario is acceptable, since the activity diagram allows this scenario. However, if the variability model is taken into account, the proposed path is not possible, since the selection of edges contradicts the variability model. According to the variability model either v_1 or v_2 has to be selected.

Thus, approaches from single system development cannot be directly used for QA of software product lines. The only valid approach for applying QA techniques from single system engineering is the derivation of sample products of the product line [3]. A derived sample product no longer contains variability and can be checked with single system techniques. However, such an approach creates a high effort if the quality of the entire product line shall be ensured, since all possible products of the product line would have to be derived as sample product [14]. To avoid such a high effort, the adaptation of single system techniques or the development of new techniques is necessary to support comprehensive QA in domain engineering [13].

In order to allow such an adaptation it is necessary to understand how variability in test models, i.e., activity diagrams has to be interpreted. In particular, we need to understand how the binding of variability affects the activity diagrams in application engineering. This is not only relevant for QA techniques applied in application engineering but also needs to be known in domain engineering in order to analyze how variability affects the QA techniques prior to providing automation for these techniques.

1.2. Contribution of the paper

Throughout this paper, we will show three examples for those QA techniques, that directly benefit from our proposed approach: the first application example supports the commonality testing strategy (cf. [3]), by using the approach to find

[1] In Section 3.2, different approaches for modeling variability in activity diagrams will be presented. For orthogonal modeling of variability in domain artifacts, different models can be used such as feature models [11] or the orthogonal variability model [4]. These and other documentation forms of variability can be transformed into a Boolean expression [12,13]. In order to keep our approach independent from a particular variability modeling approach, we use a variability model which uses Boolean variables to represent variants and Boolean expressions for their dependencies (see ❶ in Fig. 1).