Considering Variability in a System Family's Architecture During COTS Evaluation[1]

Nelufar Ulfat-Bunyadi, Erik Kamsties, and Klaus Pohl

Software Systems Engineering, ICB, University of Duisburg-Essen,
Schützenbahn 70, 45117 Essen, Germany
{ulfat-bunyadi, kamsties, pohl}@sse.uni-essen.de

Abstract. COTS (commercial off-the-shelf) component designers and developers often envision different usage contexts for their component and, therefore, provide it with adaptation possibilities. These adaptation possibilities are especially important when considering system families. System family engineering is currently an emerging discipline. Variability is a core property of system families which allows deriving different customer-specific applications from a core artifact base. A system family's core artifact base may also be populated with COTS components. These COTS components then need to support the system family's variability, i.e. they have to offer the possibility to adapt them to different customer-specific applications. Through their adaptation possibilities COTS components are able to meet this requirement. During COTS evaluation, a system family's requirements and architecture need to be taken into account. Variability is inherent in both. That is, the question is how to evaluate COTS with regard to variable features. In this paper, we describe variability in architecture in more detail and point out how this variability needs to be reflected in COTS evaluation criteria. The contribution is an extension of 'traditional' COTS evaluation criteria in order to consider a system family's variability.

1 Introduction

The idea of reuse is fundamental to emerging disciplines such as component-based software engineering and system family engineering. During system family engineering, not only executable components are reused, but also development artifacts such as requirements and test cases. COTS components may also be considered for reuse in the context of system families either during domain engineering or during application engineering. Domain engineering is concerned with the development of artifacts that are shared among the applications of the system family. Application engineering, on

[1] This work has been funded by the BMBF Verbundprojekt CAFÉ „From Concept to Application in System Family Engineering" (Förderkennzeichen 01 IS 002 C), the European ITEA Project ip02009 FAMILIES „FAct-based Maturity through Institutionalisation Lessons learned and Involved Exploration of System-family engineering" Eureka Σ! 2023 Programme, and the DFG-Project PO607/1-1 PRIME "Prozessintegration von Modellierungsarbeitsplätzen".

X. Franch and D. Port (Eds.): ICCBSS 2005, LNCS 3412, pp. 223 – 235, 2005.

the other hand, is concerned with the development of system family applications through selection and configuration of shared artifacts (developed during domain engineering) and addition of application-specific extensions. If COTS components are used during application engineering, they are considered for use in a single application which resembles traditional software engineering using COTS components. In this case, the component has to be adaptable to the needs of the application. If COTS components are used during domain engineering, they are considered for use as core artifacts. Such a component must be adaptable to needs of several customer-specific applications. In this case, greater emphasis is placed on the component's ability to support the system family's variability. In this paper, we concentrate on this case.

During COTS evaluation for a system family, a COTS candidate component has to come up to three kinds of expectations: (a) it has to fulfill system family requirements, (b) it has to be integratable into the system family architecture, and (c) it has to support the variability inherent in system family requirements and architecture (cf. [16]).

Variability is defined as the ability to change and customize a system (cf. [18]). For this purpose, variation points are provided on different abstraction levels during system family development (i.e., for example, in the requirements specification, in the architecture description, in the source code). Each variation point offers the possibility to select a variant or to choose between variants. The point in time when this decision has to be made is referred to as the binding time of the variation point. Examples of binding times are compilation, linking, and installation time.

In an earlier publication, we have surveyed current approaches for COTS evaluation, for example, [1, 7, 10, 12, 13, 14, 15, 17, 19, 21] and did not find any support for variability (cf. [16]). Therefore, we have developed a new approach: the CoVAR process (Component Selection considering Variability, Architectural concerns, and Requirements). In this paper, we only present one part of the process that helps answering the question how a system family's variability (especially in the architecture) should be considered during COTS evaluation. Thereby, the focus lies on pointing out which kind of variability may be expected from a COTS component. The contribution of this paper is an extension of 'traditional' COTS evaluation criteria in order to consider a system family's variability. For more details on the CoVAR process, refer to the paper of Pohl and Reuys [16] and Chapter 8 in [14].

In the following, we firstly describe variability in a system family architecture, different ways to realize this variability, and its impact on COTS evaluation (Section 2). Afterwards, we shortly describe variability in requirements (Section 3). Note, that we focus on functional and quality requirements and do not consider other (strategic) aspects (e.g. stability / reputation of COTS vendors). Finally, we conclude with a summary in Section 4.

2 Variability in the System Family Architecture

Architectural design represents the first activity towards realizing requirements. This is true for single system development as well as system family engineering. As a first step, quality attribute requirements are prioritized in order to identify the most important attributes that the final architecture shall exhibit. These quality attributes must