Component-Based Product Line Development:
The KobrA Approach

Colin Atkinson, Joachim Bayer, and Dirk Muthig

Fraunhofer Institute for Experimental Software Engineering (IESE), Sauerwiesen 6, D-67661 Kaiserslautern, Germany
{atkinson, bayer, muthig}@iese.fhg.de

Keywords: Product lines, components, component based software engineering, frameworks

Abstract: The product line and component-based approaches to software engineering both hold the potential to significantly increase the level of reuse in industrial software development and maintenance. They also have complementary strengths, since they address the problem of reuse at opposite ends of the granularity spectrum; product line development essentially supports “reuse in the large” while component based development supports “reuse in the small.” This paper describes a method, KobrA, that cleanly integrates the two paradigms into a systematic, unified approach to software development and maintenance. Key synergies resulting from this integration include support for the rapid and flexible instantiation of system variants, and the provision of methodological support for component-based framework development.

1. INTRODUCTION

The potential advantages of a product line approach to the development, maintenance, and deployment of software have long been recognized. Instead of continually “reinventing the wheel,” or incorporating parts of “old” systems in an ad hoc manner, organizations following a product line approach can consolidate their key software assets within a high-quality, reusable software core, and concentrate their resources on adapting this core to meet the changing needs of customers.

Despite its promise, however, systematic product line development still remains the exception rather than the rule, and its potential remains largely
unfulfilled. A major reason is that traditional software implementation technologies do not really provide the mechanisms needed to support the rapid and cost-effective adaptation of implemented code in a way required by a genuine product line approach. As a result, existing product line approaches have been forced to concentrate on the earlier activities in the software life cycle, and thus often appear to developers to be somewhat divorced from the "real" business of coding. There is also an erroneous perception that product line development is incompatible with "regular" development of single systems. Many developers therefore feel the adoption of a product line approach would force them to discard their current single-system practices with which they feel comfortable.

The advent of component-based software engineering changes this situation by making available mechanisms that enable software elements, right down to the binary level, to be rapidly and efficiently assembled into new applications. This allows the basic tenet of product line development to be applied at all phases and levels of software development, and to work with software in all its different forms, including binary forms. The ability of component-based software engineering to support the interoperation of binary code modules is one of the key characteristics that distinguishes it from the object-oriented paradigm (upon which it is based). Components, therefore, provide the perfect foundation for the practical application of product line development.

The benefits are not just one-way, however. Component-based software engineering also stands to gain significantly from product line ideas. It is to be expected that component-based systems within a given domain, or created by a given organization, will share many similarities, and in particular, will use many of the same components. The variabilities between systems in a family will thus likely revolve around a relatively small number of critical components. Therefore, rather than assemble every system in the family from scratch, it makes sense to build so-called frameworks that "hard-wire" the common aspects of the family and allow the variable components to be "plugged in" as and when needed. Although the value of such frameworks has been recognized for some time, however, their creation and maintenance is still something of a black art, lacking concrete methodological support. The techniques and ideas of product line development are the ideal foundation for the provision of such methodological support.

In short, the product-line and component-based approaches to software development seem to have complementary strengths. They both represent powerful techniques for supporting reuse, but essentially at the opposite ends of the granularity spectrum. Components provide a technology for "reuse in the small" while product line development represents an approach for "reuse