Abstract

We would like to understand the interests of our target audience. Please register at **www.softmetaware.com/whitepapers.html** to provide us with some information about yourself, and to obtain access to the full content of all SoftMetaWare white papers.

Best Practices for Component-Based Development and Model-Driven Architecture

Version 1.0
28 August 2003

Jorn Bettin
jorn.bettin@softmetaware.com
http://www.softmetaware.com

Abstract

Although component-based development provides the concepts and mechanisms to manage complexity in large software development efforts, most teams in practice still have to come to grips with the basics. Inexperienced teams perceive the introduction of additional interfaces and rigorously enforced subsystem boundaries as avoidable overhead. Especially when a system started out small, and initially a less rigorous approach to system design seemed to work, resistance to change is to be expected. A model-driven approach to component specification and generative techniques can help to simplify the process for component-based development, and at the same time can enforce standardised implementation of design patterns and ensure adherence to architectural guidelines. This paper summarises lessons from several projects where a model-driven approach and component based development was implemented to address productivity problems. The important issues are surprisingly non-technical and require paying careful attention to team structure and project organisation.

1. Introduction

The emergence and rapid rise in popularity in the software development community of agile methodologies is a good indication that methodologies perceived as being heavy are going out of fashion. If heavier methodologies were the flavour of the day in the late nineties, it does not mean that the majority of organisations actually ever fully em-braced these methodologies wholeheartedly and followed them to the letter. On the contrary, it is much more an admission that high-ceremony approaches are only practical for the development of life-critical applications, where the associated costs of a heavy process can be justified and absorbed under the heading of quality assurance. For development of more mundane business application software, any process that involves a high degree of ceremony is incompatible with the push of the markets for lower software development costs.

The complexities of today's heterogeneous systems and the exploding number of implementation technologies have only exacerbated the situation. Organisations using mainstream implementation technologies that support component-based development, such as J2EE compliant technologies or the Microsoft DotNET framework, are finding it hard to implement truly component-based systems, and dependency management—which should be at the heart of any enterprise application architecture—is a much neglected topic. A quote from a senior software architect in a 1000+ person software organisation: *Lack of dependency management is the biggest problem in our organisation - this is not purely technical, it's a mindset thing*.

In this paper we would like to present ideas and approaches that are of practical relevance in helping organisations with the steep learning curve associated with component-based development. A critical element in the suggested approach is the pragmatic use of today's MDATM tools. We acknowledge the shortcomings of the current tools, and then proceed by showing how to build a custom-tailored process that automates the tedious and repetitive tasks involved in implementing against standards such as J2EE.

This paper provides guidance on combining "heavier" methodologies for software product line development with the use of MDA™ tools and critical elements of agile methodologies to create robust and reliable processes for infrastructure development, and highly agile processes for application development.