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.

Model-Driven Software Development Teams

Building a Software Supply Chain
for Distributed Global Teams

Author: Jorn Bettin

Version 0.2

June 2004

Copyright © 2003, 2004 SoftMetaWare Ltd.

SoftMetaWare is a trademark of SoftMetaWare Ltd.

All other trademarks are the property of their respective owners.

SoftMetaWare

Revision History

Version	Author	Description
0.1	Jorn Bettin	Initial version, April 2004
0.2	Jorn Bettin	Update June 2004. Added section on Team Structure, Roles, and People. Added references.

1 Introduction

In a world where software development costs are becoming a growth-limiting factor, off-shoring software development activities to low-cost locations is a necessity, and distributed software product development is becoming the norm. As those who have been involved in software projects that span not only time zones but also culture and language barriers know, off-shoring is not risk-free, and even though savings on software development costs are achievable, time-to-market and development of the "right" product is far from guaranteed. To avoid the pitfalls, outsourcing service providers now offer configurations that include a mix of onshore and offshore resources of the service provider. This can help to reduce many of the basic communication challenges, but the efficiency of the end-to-end product development process is still determined by the quality of the software development process used.

The industry has learnt – the hard way – that *big design up-front* is not practical for fast-paced development high quality software. A collocated team that includes user representatives can use a highly agile process, and can benefit from an on-site customer. What risk-reduction techniques do you need to compensate for the lack of a collocated team that has direct access to the customer? Your off-shoring partner may have provided you with a number of on-shore resources that provide the link between your product managers, your domain experts and the offshore development resources. To be effective, at least some of the on-shore resources need to shuttle back and forth between locations. What impact does this have on costs, and more importantly, on time-to-market, and the ability to drive an iterative development cycle that includes regular validation of software-under-construction by customers?

In this article we examine different software product development models from the perspective of a *software vendor*, i.e. from the perspective of a company building and selling software product lines, and we evaluate the limitations and the applicable scenarios of each model.

1.1 The Motivation for Software Supply Chains

The software industry needs to follow in the footsteps of other more mature industries and move towards product development models that are highly flexible, where critical knowledge is embedded in highly automated processes, and where the switching costs involved in moving to different suppliers of commodity inputs are minimized.

Even the ideal scenario, where all these issues sketched above have been satisfactorily addressed, still contains risks that can not be ignored. Distributed software product development that relies on external suppliers – whether onshore or offshore – creates dependencies on external organizations and on external technologies/components. The trend towards Open Source infrastructure software shows that the industry only grudgingly accepts dependencies on software from external sources. Indeed, the Open Source paradigm offers a lot of advantages by spreading the costs of infrastructure development and by counter-acting the forces that lead to quasi-monopolies in the software infrastructure space.

The typical business software vendor who is forced to consider off-shoring of product development faces the big challenge of finding offshore partners who can be entrusted with intellectual property that is at the heart of a product line that needs to be built. Tacit knowledge built up over time in the offshore organization creates a truly monopoly-like dependency unless measures are taken that go well beyond the scope of established off-shoring techniques. Big vendors can solve this problem by buying the offshore organization or by building an offshore organization, but where does that leave financially weaker new incumbents? Even the big players can't control all risks. How stable is the political situation in the offshore location? Countries that seem politically stable today may not be stable in two years, not to mention potential changes in tax legislation and other legislation impacting foreign investors. Issues like these demand the construction of an agile organization, where critical intellectual property is not tied up in a specific geographical location or in the heads of individuals.

1.2 Fundamentals of software product family development

In developing software products and software product families it is important to distinguish between the development of products and the development of the underlying product platform, which may be shared across a number of products in a product family. This distinction is necessary for very practical reasons.

Firstly, concrete projects, such as the development of a specific product release must be completed by specific target dates. Release milestones are best kept immutable in order to avoid being perceived as one of the average software vendors that is not even able to manage a regular release cycle.

Secondly, the quality of the underlying product platform is of critical importance to the success of a highly automated model-driven approach, and it is not advisable to force compromises in the design of the product platform as a result of pressure exerted by fixed milestones in the product release cycle. It is better to make compromises in the development stream of a concrete product, and then to ensure that an upgraded product platform is retrofitted in a subsequent release. This is of course something that is only viable in a model-driven approach, where changes in framework interfaces and changes in the design patterns used in the implementation can be automatically propagated through the code base.

Thirdly, the necessary skills for product platform development differ significantly from the skills required to build product instances, which leads to a natural division of work. The goal of a well-designed software supply chain is to create an organizational model that implements a sensible partitioning of the end-to-end product development process. The parties involved usually extend beyond the boundaries of a single product vendor. External organizations may supply specific components, in the form of off-the-shelf products, and in the form of components custom-built according to specifications by the product vendor. Beyond that we also have to consider the use of Open Source components, which can substantially reduce the initial cost of providing a basic technology infrastructure on top of which product families can be built.

Proper implementation of an iterative approach in both product platform development and product development is a prerequisite for a smooth release cycle. It is a non-trivial task to ensure that the product development teams obtain real value from the product platform, i.e. to ensure that product platform development does not create an