NEGLECTED ASPECTS OF PRODUCT FAMILY DEVELOPMENT

Grady Campbell
Prosperity Heights Software
GradyCampbell@domain-specific.com

Most discussions of product family development fail to give adequate weight to several aspects that are important in actual practice. This position paper discusses the significance of these aspects.

Based on 15 years experience defining and using product family practices for industry and government needs, there are four aspects of product family development that are usually not given sufficient weight in discussions of the topic. These four are process improvement, product family requirements, process engineering, and whole-product support. There are certainly other aspects, related for example to the practical use of commercial tools or assets, that are more poorly understood or supported but the need for any complete product family methodology to account in some way for these four aspects deserves greater attention.

The Domain-specific Engineering™ methodology (<http://www.domain-specific.com>) provides a comprehensive framework for both these and the more familiar aspects of product family development. DsE is oriented toward organizations that have traditionally built custom software to order but that recognize the potential advantages of a product family capability for targeting the anticipated diverse and changing needs of a specific market. The following discussions reflect the way these aspects are accommodated within DsE.

PROCESS IMPROVEMENT

Software development today is essentially a tool-supported craft industry. This means that productivity and quality are highly dependent on the expertise and skills of individual developers. To mitigate the risks that this entails, many organizations have extensive efforts to standardize and improve their software practices, most based on models such as the SEI's Capability Maturity Model® arising out of previous work on continuous quality improvement in manufacturing.

The relevance of these efforts to product family development is twofold. First, the impact of poor or inconsistent software practices is magnified in a product family, either through wider propagation of errors or through proliferation of many similar assets that differ for inconsequential reasons. Furthermore, a manageable product family requires a degree of standardization that is difficult to achieve without a significant level of consensus on software practices (and associated representations) that are most appropriate to targeted products. The larger impact of inadequate practices increases the importance (but also the value) of an effective process improvement effort for an organization adopting a product family approach. To be effective, product family practices need to reflect minimally the

---

™ Domain-specific Engineering is a trademark of Prosperity Heights Software.

same criteria by which an organization would improve its more traditional software practices.

Second, product family development practices extend conventional improvement criteria and add criteria that an organization must address to improve its performance. Extensions to conventional criteria relate to ensuring that effective software practices are applied across the activities and assets of product family development as they would be in a conventional product development process. For example, this ensures that configuration management practices are applied for the control of product family materials (such as reusable assets) as well as to individual product materials. Additions to conventional criteria relate to issues of business strategy and management, raw materials and assets, and organizational and technical infrastructure that are unique to product family development.

A further distinction concerns the proper scope of an improvement effort, related to achieving results of greatest benefit to each business. Although a conventional improvement effort can extend over a large multi-market organization, as long as similar software practices are in use, differing business objectives and market prospects heavily influence improvement priorities related to product families. Even within a single organization, the needs of different businesses may justify different approaches to product family development. To account for these differences, process improvement in the context of product family development must provide criteria that an organization can use to determine how best to apply product family practices to its needs. There are several different levels of capability, with associated levels of cost, benefit, and risk, that an organization can decide to target for its approach to product family development.

PRODUCT FAMILY REQUIREMENTS

There is broad agreement that a viable product family depends on a stable architecture that supports the common and diverse needs of the market to be served by the product family. What has not been generally discussed is that a stable architecture depends on stable requirements. If requirements can change in unpredictable fashion, there is no basis for assuming that an architecture will be stable. It is essential that an understanding of commonality and variability start with the requirements that motivate the products in a family. A product family architecture cannot be more restrictive than the corresponding product family requirements permit. Whether explicitly stated or not, a product family architecture is a response to the anticipated requirements for a set of similar products. If these requirements are not explicitly defined, there is no precise basis for determining whether the primary purpose of the family can be achieved, that is to provide a sound basis for a customized solution to each new problem presented to it.

Product family requirements differ from the requirements of a single product in the same way that the design and implementation of a product family differ from those of a single product. That is, the requirements of a product family must establish how the requirements for different products are similar. Similarity encompasses some aspects that are common and others that differ. The differences among products' requirements represent the essential factors that motivate a need for multiple products. These factors,

when applied consistently across all elements of a product family, are the basis for a streamlined decision-based application engineering process.

PROCESS ENGINEERING

Many discussions of product family development start with a presumption that the application engineering process will have the form of a traditional software development process with activities such as requirements analysis, design, and implementation. Although such an approach is feasible, and is in fact characteristic of lower levels of product family capability, it imposes significant constraints on the achievable benefits of a product family approach. A primary argument for product families is the potential for increased productivity in application engineering but the only way to really improve this productivity is to reconceive the software development process to eliminate traditional activities and the associated overt levels of complexity.

Although an organization can decide to adopt an application engineering process having a traditional form, this choice ought to result from an analysis of whether this is the best choice given that organization's needs and constraints. It should not be chosen as if this were the only possibility. The primary tradeoff in this decision is the costs and difficulty of introducing the use of a significantly different process into an organization versus the much greater improvement in productivity, quality, and flexibility that such a transition could produce.

Another reason to give more attention to process engineering is that this is the proper context in which to consider how and to what degree application engineering should be automated. Given that current commercial tools tend to poorly support or even obstruct product family practices, an organization may decide to adopt a much more limited application engineering process in order to continue using existing tools. However, while reverting to entirely manual techniques is too costly, more advanced product family approaches can provide much greater levels of productivity if an organization is willing to forgo the use of conventional commercial tools and can afford to invest a relatively modest level of effort in custom development of an appropriate application engineering infrastructure. This can result in the equivalent of a rapid prototyping capability for products that can be much more quickly refined to meet specific needs than is possible in a conventional development effort.

WHOLE-PRODUCT SUPPORT

Most discussions of product family development seem to focus exclusively on exploiting the potential for code reuse. This is understandable but the complete automation of product implementation would result in only a modest reduction in the time and cost of product development if efforts associated with documentation and testing were not also reduced. The true potential for product family development is to address all aspects of a complete product, not just code but also associated documentation and testing.

Documentation associated with a product must be targeted to different audiences who need to understand the product from different perspectives. The fact that a product has been derived from a product family may, some day, eliminate the need for some of the

intermediate forms of documentation that are produced today to show the results of development as it progresses. For example, in some cases, documentation of the product family's design and implementation may suffice as a substitute for documenting the specific design and implementation of each derived product. However, outward-directed documentation will still be needed to serve the same purposes as it does today. Users need to know how to interact with the product to perform their work, customers need to have descriptions that demonstrate how the product responds to their needs, and maintainers (in cases where the product is to be maintained apart from the product family) need to have explanations of how the product is put together and how it behaves. The means of creating customized documentation based on a product family are similar to those for code: the factors that motivate users to ask for different products indicate the differences that must be possible to represent in the various associated documents. By creating abstract representations of needed documents that accommodate derivation of alternate content based on these factors, a capability can be created for the derivation of consistently tailored versions of all required documents based on customer-specific resolutions of these factors.

Testing serves two purposes: validation that the application engineer has properly specified the required product consistent with the assumptions of commonality and variability underlying the product family and verification that the derivation of the product from the product family has occurred correctly. Validation can be streamlined if it can be established that all producible products differ strictly in accordance with the factors that motivate users to request different products. Any product that cannot be adequately characterized as conforming to the commonality assumptions of the product family and as having only differences that can be expressed in terms of those factors would then be recognized as either not belonging to the product family or as requiring that the definition of the product family be extended accordingly.

Verification is much more complex to streamline. Ultimately, product families offer a significant opportunity for applying formal verification methods with the prospect of much greater leverage than can be obtained when applied to single products. Until such methods are available, there is an even greater dependence on effective testing practices within a product family context. Although there is reasonable experience to suggest that a product derived from a product family will contain fewer errors and omissions than a handcrafted product, there is always the possibility of previously undetected errors appearing in a newly derived product. The only way to mitigate this risk is a rigorous acceptance testing regime of the same scope and depth as would be applied to a product developed conventionally, with the expectation however that this will expose fewer problems in a derived product and require less rework to correct.

Given this need, the best way to streamline verification is again to represent test scenarios and associated materials in a product family form, that is providing the means to create test scenarios and materials that are tailored according to the same customer-oriented factors as corresponding code and documentation. The result then is a capability to rapidly derive testing scripts and materials (including test documentation) that are tailored to verifying that a particular product has been derived without any foreseeable errors. This can of course be augmented with additional testing but the resulting total effort cost is greatly reduced.