Products as Product Lines

Grady H. Campbell, Jr.
domain-specific.com
Annandale, VA, USA

***Abstract*—An argument is made that a product line approach can be beneficial for even single product development, particularly for products that support customers whose needs are likely to change over time. Furthermore, product line techniques for identifying requirements and design alternatives and documenting rationale for decisions would benefit any product development effort.**

***Index Terms*—methodology, diversity, change, variability, customization.**

I. Introduction

Product lines are an approach for achieving reuse within a set of similar products. The motivation for a product line approach however is much more significant than avoiding redundant effort through reuse. It enables an enterprise to focus and leverage its resources towards a goal of effectively addressing the needs of a coherent but diverse and changing market. The theory of product lines is that, given suitable conditions, products will be less costly to build, sustain, and evolve, and yet be customized to each customer's specific needs. In the ultimate vision, a product line and its associated market will coevolve to the mutual benefit of the product line enterprise and its customers.

The argument to be made here is that there is a valid business case for adopting a product line approach not only for a market consisting of multiple similar products but also for a market that consists of only a single product. The basis for this is the view that a product that is susceptible to change will exist in multiple similar versions delivered serially over its life. This is no less a characterization of a product line than when there is a need for multiple distinct products that are delivered concurrently to one or more customers. Whether to represent the means for building these versions in a product line production capability is a business decision on how to make best use of organizational resources. The rationale for doing so remains that a product line approach will greatly reduce the cost of building subsequent product versions because the nature and locality of differences in the product across versions can be anticipated and managed. A weaker case can also be made that product line techniques will have value in the development of even some single-version products.

II. Background

The usual challenge for single product development is that the customer's needs are initially poorly understood such that the problem changes even as a solution is being developed. Furthermore, critical aspects of a solution may require experimentation to choose among alternative resolutions. Any well-understood problem will have numerous potential solutions, differing to a greater or lesser degree from whatever solution is finally delivered. Furthermore, a customer's needs are likely to change over time so that the problem itself, once adequately understood, can still not be viewed as static for long.

The reality is that a software solution is normally the result of a long chain of considering and resolving the different aspects of a solution, even as understanding of the problem changes, until an acceptable problem-solution pair is achieved. Product line techniques for identifying requirements alternatives and for deferring choices among alternative implementations lead to greater flexibility, a better understanding of the ultimate solution, and less risk of schedule compression late in development. The same techniques reduce the difficulty of rederiving a new solution as the problem changes over time.

The context for this discussion is the Domain-specific Engineering (DsE) methodology [1] which is a reformulation and refinement of the Synthesis product line methodology [2]. DsE defines a family of product line approaches. A DsE process (Fig. 1) encompasses a process adoption process, a

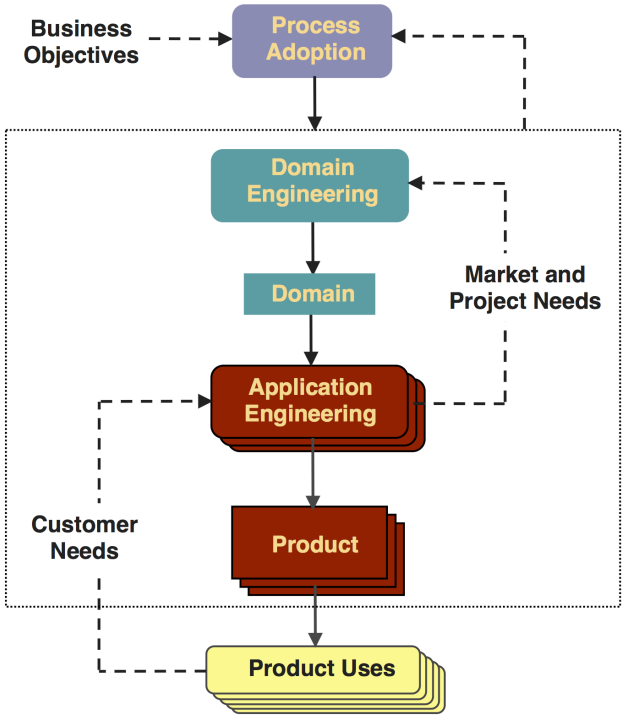

Fig. 1. A DsE Process

domain engineering process, and an application engineering process, all customized as part of process adoption to the needs of the adopting enterprise.

The focus of DsE is a family of similar products conceived to satisfy the envisioned current and future needs of a targeted coherent market of potential customers. The concept of a product family is a generalization of the concept defined by Dijkstra of a program family. A product consists of all of the artifacts of an application engineering effort, including requirements and design specifications, the program (software implementation), hardware, documentation, and validation and delivery materials. A DsE domain consists of the representation of a product family and an associated process and environment for deriving customized instance products.

The linchpin in the DsE approach is a decision model which defines the criteria, expressed informally in variability assumptions, by which products can be customized. Decisions correspond to differences in customer needs and engineering tradeoffs that must be resolved to derive a particular product. An application engineering project derives a product by specifying and validating the set of decisions that will result in the product that is the best fit to the customer's needs. Implicit to DsE is that only products that are instances of the represented product family are derivable; any product needing capabilities that fall outside this scope requires either revision of the domain or construction in whole or part by other means.

III. The Duality of Diversity and Change

There seems to be a perception that a product line approach is viable and even applicable only when there is a need to build multiple distinct products. The position we take is that this is an unnecessarily narrow view of product lines and that it is beneficial to apply product line practices in the context of single products.

Dijkstra argued, in 1972, that the development of any program should be viewed as the derivation of an instance of a family of similar programs [3]. He defined a family as a set of either alternative solutions to the same problem or similar solutions to similar problems. The latter definition fits the principal way in which product lines have been considered; we will focus on the case of the former definition. For that perspective, Dijkstra argued that to adequately understand and have confidence in the solution to a problem the developer needed to understand the choices that had led to that solution as opposed to other somehow different solutions. Furthermore, he suggested that changes to that solution should be derived not by directly modifying an existing solution but instead by revisiting the choices that led to that solution and rederiving a different one of the alternative solutions.

In conceiving the Synthesis approach, the primary motivation was clearly to achieve a streamlined means of building multiple similar products for different customers constituting a coherent market. However, its conception was also motivated by developer experiences not only of having to repeatedly build similar products for different customers but also with constantly creating and then having to modify solutions as requirements understanding changed during the development effort. This concern was reinforced by experiences in modifying existing products without information about prior design tradeoffs and rationale and having to rediscover dependencies and implications before making changes that could inadvertently undermine the coherence and integrity of the design upon which the original product was based.

Another basis for our position is the observation that there is no way to distinguish objectively between two products built for different purposes (diversity) and two products built serially as different solutions to the same (changing) problem. There is no question, in that it happens often in common practice, that an existing program can be modified either to accommodate changes in the needs of its existing customer or to create a new product that accommodates the differences that characterize the needs of a different customer. Similarly, if we have a product line production capability, it is implicit that we can use it to build both new and revised products.

As an example, consider a notional sensor net type of product. The initial product, consisting of hardware and software, is a network of motion detectors. A subsequent product is needed that includes both motion detectors and infrared or heat sensors. The question is whether the new product is a replacement of the existing product due to changing needs or is it a new product targeting a different use? There is no way to know this either from the descriptions of the existing and new products or from contemplating how the new product is derived. In either case, the new product could be derived from a ground-up application engineering effort or it could start with and modify the specifications and implementation of the existing product.

We can conclude from this that a product line can feasibly support both diverse and changing needs and in fact there is no reason technically to distinguish at all between diversity and change as the motivation for building a new product. Clearly then we can use a product line approach to build the alternate potential versions of a single product.

So given that a business's perceived market entails only a single product (whether consisting of only one or of many customers) and a product line approach can support iterative building of developmental and production versions of that single product, how would developing a product line capability differ for the purpose of building only the initial and subsequent versions of that one product?

From the perspective of supporting change, we have argued that there is no technical basis for distinguishing between building a modified product in response to changed needs and building a new product to address the needs of a different customer. A product line capability inherently supports both and offers the same advantages for both: the ability to more systematically explore a coherently bounded problem-solution space so as to efficiently determine a best

fit of product to needs and the ability to rapidly modify that fit when needs change.

A technical difference with a product line approach for only a single product, as opposed to one for multiple products concurrently for customers having differing needs, is that we have a more limited basis for making variability assumptions about the ways in which we will need to be able to customize the product. In the case of only a single product, we have to make variability assumptions based entirely on "predicting" how customer needs will differ in the future. However, the duality of diversity and change applies here as well. Even with the basis that a diverse market gives us for identifying differing needs from the start, we do not limit our thinking to only what customers in that market need today. Leverage comes from imagining ways in which those needs "might" change in the future and representing that vision in the design of the product family. This does not mean that we have to fully implement every imagined alternative future need but only that we will establish a framework within which we will know how and where in the representation of the product family to implement each such alternative should it later arise as a recognized need of some customer.

In the simpler case of a single product, we do not have to worry about fully implementing any unneeded alternatives immediately – we only have to implement what we think the customer needs right away. The major exception to this of course is that we most likely do not know absolutely what the customer's actual needs are when we start or how best to resolve associated engineering tradeoffs. In areas of uncertainty about these, we may need to accommodate some alternatives immediately so as to maintain the conceptual integrity of the product as our understanding changes and avoid falling back into the trap of changing the product haphazardly as the actual needs (and all intervening partial clarifications) reveal themselves to us.

IV. Build a Product or a Product Line?

Whether to actually adopt a product line approach is a business decision, essentially whether to minimize current costs or to invest for ease of change and future capability. This is not a simple judgement in that even within the context of a product line approach, the level of investment to be made and the consequent potential payoff can vary greatly depending on choices related to market scope, timeframe, and extent of automation. From the perspective of a product line investment decision, traditional development practice can be considered essentially as a degenerate product line approach that accommodates no variability at all. DsE process adoption treats reversion to conventional development as an option that is evaluated against various levels of product line investment.

Given our argument that building a product line for the production of a single changing product is technically feasible, what are the considerations that motivate actually doing this? To explore this question, we will consider various types of situations that might lead an organization to adopt a product line approach and see if this leads us to any single-product cases.

The utility of the product line concept is the framework it prescribes for anticipating and facilitating diversity and change. Different products may be created to provide a more customized fit to specific needs, such as for a particular market niche, customer, operational context, or even individual user. The motivation for changing a product (or equivalently for deriving a new product line instance) is to create a better fit to the customers' current needs.

The case for an organization to adopt a product line approach is most easily made when there is a recognized need to provide a set of similar products to multiple customers constituting a coherent market. The alternative of building and maintaining those products individually tends to lead in practice to arbitrary divergence of the products over time and increased maintenance costs. Similarly, trying to collapse differing needs of customers into a single product is easier for the provider to manage but forces customers to conform to the resulting capabilities of that product.

The option of creating a single product that can be customized during installation or at runtime is really just a form of product line and its developer would benefit from applying product line practices in building it. In particular, benefit comes from being able to distinguish between customizations that provide a tailored product with the specific capabilities needed (and omits others that would make the product more complicated to use or less efficient) versus included capabilities of the product that allow it to dynamically adapt to a changing operational context.

Another reasonably straightforward argument for a product line approach exists for an organization whose market consists of a single customer who needs a product that can be customized for differing operational contexts. An example might be a company that provides software for producing tax returns, with a customized version derived corresponding to each user's circumstances. While the conventional approach is to build a single product that accommodates the totality of all capabilities needed, deriving a product line instance for each use is also feasible. Customized instances could provide a simpler, more streamlined user experience. Another example might be a product (consisting of equipment, software, and supplies) that supports the operations of a military unit, that is differently equipped and supported depending on its operational mission and environment.

A less obvious case is the organization that has a single customer who needs a single long-lived product. Even in such a single-product case, the capabilities that a customer needs are rarely static. New needs emerge and change over time as circumstances change (e.g., laws and regulations being modified require changes in the rules and processes by which the enterprise operates). A product line approach can improve the ability of the developer to adapt the product more quickly if such changing circumstances have been anticipated as variabilities in the product family.

To be fair, the perspective of conventional development approaches is that building a solution with specific capabilities is sufficiently difficult without trying to anticipate and prepare for changes that may or may not be needed at some future time. Instead, as changes are needed, the then existing product is directly modified to reflect those changes. The counter to this argument is that software developers must already routinely consider alternative solutions and adjust their solutions as their understanding of needs change. The product line treatment of this would be to rationalize these changes with a decision model that exposes the motivations and implications of potential alternatives as they arise. The value of this is that it provides a degree of understanding that enhances the ability of developers to anticipate and respond more rapidly and reliably to changes that do occur.

A related common objection to creating a product line capability is the notion that we cannot predict what a customer is going to need in the future and any attempt to do so is wasted effort. In some sense, this is true – our ability to understand what a customer needs today is hardly better and, by the time we do understand those needs, they will probably have changed to some degree. However, the conceit of a product line approach is that leverage comes not from knowing precisely what changes the future will bring but only what types of changes the future are most likely to bring. With this insight and based on the reality that we can never build software that will accommodate all conceivable changes with equal ease, we can create a capability that is more amenable to more likely changes because we have predicted that other changes are not as likely to occur. If unexpected changes are needed (and they will be), our goal is not that those be as easy as expected changes but that they be no harder than if we had predicted that the software would never change at all (which is in effect what we do when we build software in a conventional way).

V. Product Line Practices for a Single Product

Narrowing our focus further, what practices does a product line approach include that could apply and provide benefit in the development of a single product even if we expect to never change it once it has been delivered?

When building a single product, a decision model would provide benefit, as Dijkstra advised, by exposing how the product (both the problem and its solution) differs from alternative products that might have been built. In the context of a development effort that recognizes the identification and analysis of alternatives as an aspect of normal engineering practice, this imposes very little if any additional development effort by instituting a standard form for organizing and retaining information about alternatives considered, including underlying assumptions and rationale for choices made. The value of this comes both in the short run by exposing these decisions and tradeoffs for user validation and in the long run by retaining an institutional memory of the rationale for the product's final state as an aid in understanding its expected behavior.

Another product line technique that can benefit single product development is the potential for use of adaptable components. Libraries of pre-built "reusable" software components that are broadly useful over different types of products are already in widespread use within and across product development enterprises. Adaptable components are a concrete representation for a family of similar software components from which customized components can be derived. An adaptable component is characterized by an associated set of instantiation parameters that control derivation of specific instances. It is nothing more or less than a simple, flexible, and highly predictable form of component generator, eliminating the need for over-generalized component implementations or the proliferation of multiple versions of components required to accommodate differing needs.

A third product line technique is to apply product line practices narrowly in support of testing efforts. The intent would be to systematically represent differences in potential operational contexts as variabilities that would drive derivation of alternative test scenarios and associated expected results. This would reduce the effort required to create and evaluate a sufficient set of different scenarios to properly evaluate the product before delivery into operational use.

VI. Summary

Diversity across a homogeneous population is conceptually indistinguishable from a progression of prospective changes in an individual. Product lines are a formalism for analyzing and describing both diversity and change. As such, product line practices are applicable to the production of a set of similar products, including versions of a single product, both concurrently and serially.

References

[1] G. H. Campbell, "Domain-specific Engineering", Proceedings. San Jose, CA: Embedded Systems Conference, 1997. <http://www.domain-specific.com/PDFfiles/DsE-RSP.pdf>

[2] G. H. Campbell, S. R. Faulk, and D. M. Weiss, Introduction to Synthesis. Herndon, VA: Software Productivity Consortium, June 1990.

[3] E. W. Dijkstra, "On Program Families", Structured Programming. London: Academic Press, 1972, pp. 39–41.