Software Product Lines: A succulent Minestrone with lots of Flavours

Giancarlo Succi
Department of Electrical and Software Engineering,
University of Calgary
2500 University Dr. NW., Calgary, AB, Canada T2N 1N4
E-mail: Giancarlo.Succi@enel.ucalgary.ca

Abstract

Software product lines aim at reducing the cost and increasing the quality of software products by producing multiple products synergistically. The underlying assumption is that the benefits from the reuse of domain specific software components will offset the extra cost for the increased organizational complexity.
Product lines benefit from a collection of effects and technologies, such as branding, minimal marginal costs, network externalities, software reuse, and modularity in the development process.

It is essential for a firm embracing a software product line approach to determine the extent to which each of these effects and technologies will be taken into account. However, this is not an easy task because it requires a deep understanding of the target market, its incumbents, and the firm's own position, and technical and financial resources.

1. Introduction

Software product lines are one of the most recent proposals to improve the effectiveness and the efficiency of software firms.

The aim of software product lines is to produce multiple products –often in a given domain, trying to exploit scope economies. Scope economies arise when the cost to develop multiple products or deliver multiple services together is less than the sum of the cost of developing the products or delivering the services individually. Scope economies are common in knowledge intensive sectors (Baumol et al., 1982). The underlying assumption is that the benefits from the reuse of domain specific software components will offset the extra cost for the increased organizational complexity.

Software product lines are appealing; some research has been already performed in the subject, such as (Bass et al., 1999; De Baude and Knauber, 1998). Reuse has been considered the cornerstone of software product lines –see for instance the very clear paper of Poulin (1998). Sometimes additional factors has been taken into account, such as in (Clemens, 1999); in most of the cases, they were mainly impeding factors.

We think that product lines do benefit from a collection of phenomena and technologies, a very important one being reuse. The selection of the target phenomena and technologies to establish a software product line is a strategic decisions based on several factors, such as the structure of the market, the financial wealth of the producers, the stability of the application domain, and so on.

This paper is organized as follows. Section 2 provides a definition of software product lines. Section 3 presents the phenomena and the technologies that can support the establishment of software product lines. Section 4 evidences the strategic decisions that are required to establish a successful software product line. Section 5 draws some conclusions.

2. Software product lines

Software product lines are the establishment of a synergy between the production of multiple products by a software producer or a coordinated group of software producers.

The core idea is to aim at a synergy between the production of multiple products motivated by presence of scope economies: developing multiple products with some interaction between the lifecycles of them is more economically profitable than developing them individually.

Product lines do not require a single producer; also groups of producers can implement them successfully. However, the presence of group of producers require coordination, either explicitly defined as a partnership –look for instance at all the Sun-supported firms producing Java-based software, or implicitly occurring as consequence of standardization or other coordination mechanisms –look at all the history of the Unix operating system. The work by Gaio and Zaninotto (1998) contains an outstanding discussion of this problem; unfortunately, no English translation of it exists yet. In the remaining of this paper we will use the term "producer" generically to refer both to a single producer and to a group of producers.

Products need not be in the same problem or application domain, even if it is more "natural" to exploit product lines in a given domain because of network externalities and software reuse –more on this later.

3. Phenomena and technologies supporting software product lines

Scope economies do not just happen. They require careful investment and smart strategies. In particular, it is important to take advantages of phenomena and technologies that support the establishment of profitable product lines, such as:

- Branding
- Network externalities
- Minimal marginal costs of production
- Sharing of organizational costs
- Software reuse
- Modularity

Branding arises when people prefer a brand they have experienced in the past based on expectations they have on the future behaviour of such brand (Marder, 1997). Branding is a psychological phenomenon. Branding is especially strong in software, where the expectations of quality, reliability, and usability of a product are usually not met. Branding supports the establishment of a product line. Users will feel more comfortable in buying a product of a firm or a consortium if they are already satisfied by another product with a similar brand from the same firm or consortium or from a firm or consortium endorsing the new product.

The *marginal cost* to produce an extra unit of a software product is *minimal*. Therefore, producers can segment the market freely, with the only constraint of opportunity costs. This can be profitably used dealing with network externalities.

Network externalities occur when users value a product more when there are other users of such product or of other "compatible" products (Economides, 1996). Software is an industry with strong network externalities coming from the need to exchange information between users and products and the advantage of exchanging experience among users of a product. Developers of product lines can organize the cross-compatibility of the different products to maximize network externalities; users of a product in a product line will achieve much higher benefit buying another product in the product line rather than a product from competitors, ceteribus paribus. In addition, the minimal marginal cost of software products can be profitable used to enlarge network externalities. Product lines developers can provide a product in the product line free or at a substantial discount creating value for the other, fully priced products in the product line.

Sharing organizational costs refer possible positive synergies in marketing and distributing the software products in the line. In marketing, while advertising the feature of one product of the line it is possible to evidence the need for another product in the line. In distribution, it is possible to take advantage of the distribution channels of one product of the line to distribute also the other product. Web-based marketing and distribution is an example of such possible synergies; software products of the same line may share the same web server and may have in their pages links to each-others showing not only the features of one product but the network externalities arising from using multiple products.

Software reuse is the reuse of existing software components in new products. Software reuse is believed to increase productivity and quality. A product line offers the opportunity of identifying common components in the line, thus fully exploiting the advantages of software reuse. Often product line reuse is based on domain-centered software architectures (De Baude et al., 1998; Bass et al., 1998).

Modularity is the attempt of reducing the complexity of the production by organizing it into separate, cohesive units that relate one-another according to a well-defined protocol (Simos, 1982). Modularity can be applied to the product, that is the software system being developed –this is the essence not only of "Modular Programming" but also of most of the proposed programming paradigms, including object orientation, logic programming, and functional programming. Modularity can be applied to the development process itself. This is particularly useful in software product lines where the complexity of production tend to be very large.

4. Strategies and decisions

From the analysis of the phenomena and technologies it is evident that there are two order of scope economies that can come from product lines:

- Savings coming from the added value that products in the product lines have for users with respect to individual products
- Savings coming from distributing some of the fixed costs of components across multiple products; this include synergetic marketing (Churchill and Peter., 1998), sharing distribution channels, software reuse

These scope economies are present throughout the development cycle. Figure 1 presents a summary of this view in a very simplified software product line lifecycle.

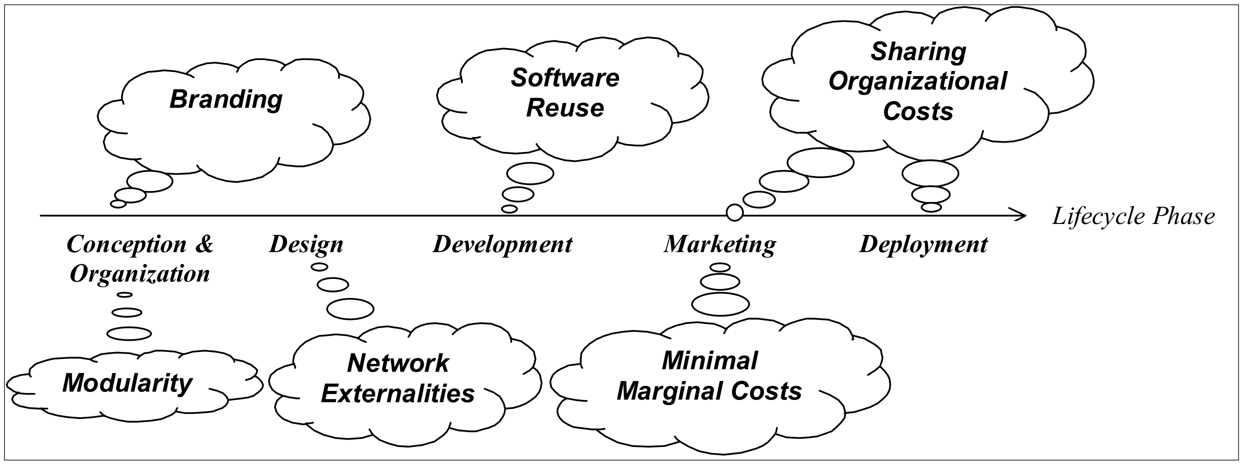

Figure 1: Opportunities for scope economies in software product lines

Taking advantage of these phenomena and technologies may require expensive up-front and operational costs. This is why, deciding on the production of product lines it is important to determine where the savings are going to be most relevant for the target product line and then studying whether such savings are likely to require feasible investments.

A careful analysis of the strategic positioning of the firm is required, especially with reference to the supporting phenomena and technologies. Factors requiring specific attention follow.

- The structure of the market
- The position of the producer in the market
- The financial situation of the producer
- The structure of the users' base
- The expected variability in the application domain
- The technical competence of the producers
- The quality of the process of the producer

The *structure of the market* is the first factor to take into consideration. Producers should try to understand (1) whether the market they plan to enter is a natural monopoly, is an oligopoly, or is an open market(Baumol et al., 1992); (2) the number, the mutual positions, and the wealth of the incumbents; (3) the compatibility and interoperability of the products of the incumbents; (4) the relations between the target market and other lateral markets in terms of compatibility or complementarity of products. This understanding is important for introducing any product, but is critical for product lines where to goal is to achieve a synergy from the presence of multiple products. If the market is a natural monopoly with a very strong incumbent and very limited relations with other markets, there is little hope to succeed at all. Open markets with various relations with other markets are the best for product lines and offer opportunities for network externalities. In addition, the structure of the market influences the possibility of sharing organizational costs between products in the product line.

The *position of the producer in the market* refers to whether the producer is a new entrant, an already established incumbent or the market leader. This is essential to understand the possibility to take advantage of network externalities and branding effects. A new entrant in a market close to lateral markets is unlikely to take advantage of branding, but can still try to take advantage of externalities, if some of the incumbents have products that are open to or interoperable with compatible products. An incumbent in a market open to later markets can take full advantage of both branding and network externalities to establish a software product line.

The *financial situation of the producer*, the *reliability of the predictions of the future users' base* and the *expected variability in the application domain* are the key factors in determining the scope of the strategy and consequently the investment to make. Producers are not advised to perform large up-front investments, such as establishing a domain-oriented library of reusable components, when

1. they are short of funds, or
2. the target markets do not offer reliable predictions of the future users' base, or

3. the products are likely to change a lot without a predictable variation pattern –for instance, for the high rate of innovation.

On the contrary, when producers can invest money in uncertain operations, when the market is fairly stable, or when the variability of the application domain is predictable it could be wise to establish a fully-fledged reuse program.

In addition, the financial situation of the producers and the structure of the users' base influence strongly how to take advantage of minimal marginal costs. The structure of the users' base is also a key factors in the determination of the feasibility to share organizational costs.

The *technical competence of the producer* must be considered before starting a reuse program, especially if couple with the introduction of new technologies, such as object orientation. It would be risky to introduce object orientation in a business environment with experienced Cobol programmers, without an up-front commitment to devote time and effort to training. In addition it is important to analyze it before deciding to adopt any business process reengineering or improvement strategy, such as the adoption of a modular development process.

The *quality of the process of the producer* is a reference point when deciding to adopt a modular production process.

Table 1 contains a summary of the discussion. The rows contain the major factors to consider before establishing a given product line. Columns 3 to 8 contain the major phenomena and technologies that help establishing a product line. Column 2 refers to the strategic decision of whether or not to start a software product line. An X in a cell indicates that the corresponding factor should be considered when deciding on taking advantage of the corresponding phenomena / technology.

Notice that table 1 is clearly implying that it is not wise to try to use all the phenomena and technologies to establish one single software product line. Producers' strategists should careful select only the few essential ones. In particular there can be successful product lines not taking advantage at all on software reuse!

5. Conclusions

The establishment of a software product line is a complex operation that requires considerable more attention than just introducing a product, since there are complex interactions among products in the product lines and between products in the product lines and other incumbent.

A successful introduction of a product line can rely on phenomena and technologies, such as branding, network externalities, minimal marginal costs of production, sharing of organizational costs, software reuse, and modularity.

	Go/Stop	**Branding**	**Network Externalities**	**Minimal Marginal Cost**	**Sharing Org. Costs**	**Software Reuse**	**Modularity**
Structure of the market	**X**		**X**		**X**		
Position of the producer in the market		**X**	**X**				
Financial situation of the producer				**X**		**X**	
Structure of the users' base				**X**	**X**	**X**	
Expected variability in the application domain						**X**	
Technical competence of the producers						**X**	
Quality of the process of the producer						**X**	**X**

Table 1: Factors to consider when deciding on phenomena / technologies to support software product lines

However, such devices do not come for free and their successful applications depend on a careful study of the operating environment. In particular, producers should always have a clear picture of what they are aiming to do, where the competition is and what it is planning to do, and what are the financial and technical resources available.

This does not solve at all the problem of software product lines, it just rationalize some of the problems involved with their successful establishment.

Intentionally, some aspects have been completely left out since they are more commonly found in software engineering papers, such as the extent to which reuse should be applied, the decision of generality vs. specificity in the domain components, the prediction of the variability in the domain, the model of externalities, the role of a software architecture, and so on.

Also, as briefly mentioned, product lines require careful coordination between products within the product line and between subjects producing product lines.

Acknowledgements

This research has been partly funded by the University of Calgary, the Canadian National Science and Engineering Council, the Government of Alberta. The author thanks Enrico Zaninotto for the stimulating discussions that are the basis of this work, Francesca Pizzirani and Eric Liu for diligently reviewing this manuscript.

References

Bass, L., G. Campbell, P. Clements, L. Northrop, D. Smith. *Third Product Line Practice Workshop Report*, CMU/SEI-99-TR-003, March 1999 (This report can be found at the URL: http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr003.pdf)

Bass, L., P. Clements, and R. Katzman. *Software Architecture in Practice*, Addison Wesley, 1998.

Baumol, W.J., J.C. Panzar, and R.D. Willig. *Contestable Markets and The Theory of Industry Structure*, Harcourt Brace Jovanovich, Inc., 1982.

Clements, P. "Essential Product Line Practices" *Proceedings of the Ninth Annual Workshop on Institutionalizing Software Reuse (WISR 9)*, Austin, TX, January 1999 (This paper can be found at URL: http://www.umcs.maine.edu/~ftp/wisr/wisr9/final-papers/Clements.html)

DeBaud, J.M., O. Flege, and P. Knauber. "PuLSE-DSSA - A Method for the Development of Software Reference Architectures", *Proceedings of the 3rd International Workshop on Software Architecture (ISAW-3)*, Orlando, FL, November 1998

DeBaud, J.M., and P. Knauber. "Applying PuLSE for Software Product Line Development." *Proceedings of the European Reuse Workshop '98*, Madrid, E, November 1998

Economides, N., *The Economics of Networks*, International Journal of Industrial Organization, **16** (4) 1996

Gaio, L. and E. Zaninotto. *Standardizzazione e modelli di produzione post-fordisti* CEDAM, Padova, I, 1998

Churchill, G.A. and J.P. Peter. *Marketing – Creating Value for Customers*, 2nd Edition, Irwin McGraw-Hill, 1998

Marder, E., *The Law of Choice*, Simon and Schuster Inc., 1997

Poulin, J. "Software Architectures, Product Lines, and DSSAs: Choosing the Appropriate Level of Abstraction" *Proceedings of the Eighth Annual Workshop on Institutionalizing Software Reuse (WISR 8)*, Ohio State University, Columbus, OH, March 1997 (This paper can be found at URL: http://www.umcs.maine.edu/~ftp/wisr/wisr8/papers/poulin/poulin.html).

Simon, H., *The Science of the Artificial*, The MIT Press, Cambridge, MA, 1982