Quantifying Product Line Benefits

Peter Knauber, Fraunhofer IESE, Germany, peter.knauber@iese.fhg.de
Jesus Bermejo, Telvent, Spain, jesus.bermejo@telvent.abengoa.com
Günter Böckle, Siemens AG, Corporate R&D, Germany,
guenter.w.boeckle@mchp.siemens.de
Julio Cesar Sampaio do Prado Leite, PUC-Rio, Brasilia,
julio@inf.puc-rio.br
Frank van der Linden, Philips Medical Systems, The Netherlands,
frank.van.der.linden@philips.com
Linda Northrop, SEI, USA, lmn@sei.cmu.edu
Michael Stark, University of Maryland, USA, mstark@cs.umd.edu
David M. Weiss, Avaya, USA, weiss@avaya.com

Abstract. Software product lines promise benefits like development and maintenance effort reduction, time to market decrease, and quality improvement, all resulting from planned and systematic reuse of common core assets. However, very little quantitative data has been measured so far to prove these promises.
This paper formulates and discusses 7 hypotheses on how the promised advantages would look like in a quantitative way. It is meant to be a starting point for discussion on how to quantify which product line benefits and how they can be measured.

Introduction

We understand the technology for creating and using product lines reasonably well. When motivating why some organization should adopt product line development, it is relatively easy to find qualitative arguments for why product lines help to satisfy goals such as time to market, but it is still an open issue how to create convincing business cases based on quantitative analyses. At the moment we can only hypothesize the form of the quantitative relationship (cf. [1, 3, 4, 5]); we urgently need to perform experiments and gather data to confirm or reject our hypotheses.

As result of that lack of quantitative data, we have few examples of well-constructed business cases for product-line engineering. From a business point of view, we should be able to identify at least five convincing arguments, based on economic analyses, for the use of product line engineering. A convincing business case should have the property that it shows how the use of product line engineering leads to fulfilment of the goals of those we ask to invest in the technology.

Business cases will be different according to the viewpoint of the client. For instance, a product manager has different goals for a product line than does a software development manager, a software engineer, or a software quality assurance manager. The product manager, for example, may be interested in achieving a short time to market

with the products from his/her product line, whereas the software development manager may be interested in reducing the cost of producing the next version of his/her product. When talking to the product manager, we should be able to show a business case that demonstrates that applying software product line engineering to the set of products that he/she is responsible for will directly help to achieve a low time to market. This business case should contain a quantitative demonstration of the relationship between time to market and product line engineering, e.g., a graph such as shown in Figure 2, which has been constructed solely as an example, without the benefit of real data.

During a Dagustuhl workshop, the authors spent four days discussing how to have successful business cases for the product line idea. Starting from a suggestion of one of us (Weiss), who had before thought of showing the economic benefit of product line software production by means of economic curves, we started to think how we could draw curves representing benefits of product lines against the traditional way of software production. Based on each one's experience and knowledge of the subject, we started discussing the shapes of the curves. After several hours of debate we have reached an agreement over the overall shapes. After doing this, one of us (Knauber), using the Excel´s features, came up with the shapes as presented in this paper.

The set of diagrams given in this position paper would strongly support any product line business case if only they were based on real-world data, that is, not just hypothetical. The paper is meant to be a starting point for discussion on which promised product line benefits should be quantified and how these can be measured. To those being able to access respective data, the paper is meant as a strong motivation to collect, consolidate, and publish them.

Quantified Hypotheses of Product Line Benefits

The following 7 hypotheses try to quantify benefits that can be achieved through product line development over single system (also called stove-pipe) development. The corresponding graphs do not show any scale units because they are only meant to illustrate the respective hypothesis, that is, not to give precise numbers.

There are, of course, more benefits that can be expected from product line development than presented in this paper. The ones presented here are restricted to those *addressing the particular interests of product managers*:

1. Decrease the development effort per product, cf. Hypothesis 1.
2. Decrease the time to market per product, cf. Hypothesis 2.
3. Keep time to close customer issues constant (i.e., not proportional to the number of products), cf. Hypothesis 3.
4. Develop more features with a given amount of money, cf. Hypothesis 4.
5. Develop more features within a given amount of time, cf. Hypothesis 5.
6. Decrease the time to integrate COTS components per product, cf. Hypothesis 6.
7. Decrease the time for certification per product, cf. Hypothesis 7.

Other product line benefits (and their quantification) could address the interests of managers, software and system engineers, or quality managers.

The graphs shown for single system development in this section are somewhat simplified or abstracted, that is, they do not explicitly consider “unplanned” savings. These kind of savings could result from learning effects and informal reuse of knowledge and code over several systems. This paper abstracts from these effects because the extend to which they would result in savings of effort or time over a series of products strongly depends on the individuals involved in the development process and can therefore hardly be predicted (or even managed during development). However, they could make a difference especially for very small development teams.

Hypothesis 1:

After some initial investment, the effort that has to be spent per product derived from a product line decreases significantly below the effort that has to be spend for the development of the same product as single system, yielding cumulative effort savings (cf. Figure 1).

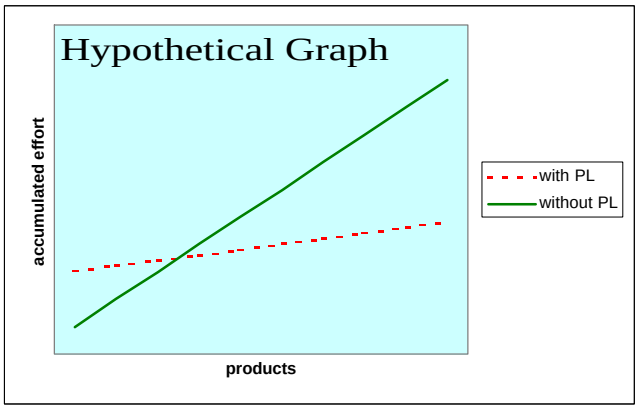

Figure 1 Effort Reduction over Number of Products

Explanation:

When similar products are developed as single systems, the development effort per product can be expected to be roughly constant.

For the setup of a product line, some initial effort has to be invested in planning, architecting, and realization of the product line infrastructure. Thus, the development of the first product line members based on that infrastructure takes longer than in the case of single system development. Once the infrastructure is in place and can simply be reused to derive more products, the effort needed for their development is expected to be constant on a much lower level than the effort needed for single system development. As a result, also the accumulated development effort for product line development will fall below the accumulated effort for development of the respective products in single system fashion.

It has been reported that product line development typically starts to pay with three products (cf. [2, 5]).

Hypothesis 2:

After some initial investment, the time to market per product decreases down to a certain minimum and significantly below the respective time to market in the case of single system development (cf. Figure 2)[1].

Explanation:

With single system development of similar products, a similar time to market per product can be expected.

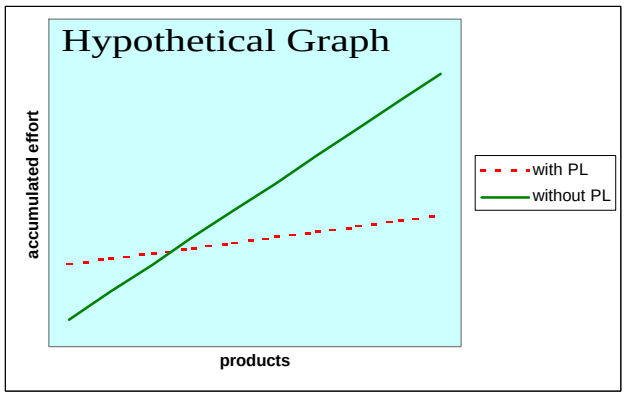

Figure 2 Time to Market Reduction over Number of Products

For product line development, some more planning and development effort is necessary before the first product can be released, causing a higher time to market. It is likely that with the second (and maybe third, fourth, …) product some more product line infrastructure has to be created, causing more effort to develop these product parts than in case of single system development. On the other hand, it is expected that the second (third, forth …) product benefits already from the reuse of existing parts of the product line infrastructure. With more products produced, the infrastructure development and maintenance effort decreases to a certain minimum, the remaining effort per product is then spent on usage of the infrastructure plus the addition of some customer-specifics.

The case study presented in [2] is a good example of incomplete data, preventing the documentation of a “perfect” business case: the time to market for a series of ten products within a mid-sized company is given, illustrating the relatively long time of development for the first product line member but also the dramatic decrease of time to market after that first product. Unfortunately, the company where the case study is performed has no reference data that would allow the precise calculation of the effort needed if those products were produced as single systems. However, the responsible product

1. The underlying assumption in this argumentation is that the available set of resources is kept constant (as it is most of the time in reality). Otherwise one could set up n full project teams developing n products in parallel within the time of developing one single product.

manager estimates that effort to be roughly constant at a certain value for the products produced so far, implying that the product line started to pay with the third product.

Hypothesis 3:

The time to close customer issues remains almost constant within a product line, whereas it increases proportional to the number of products for single system development (cf. Figure 3)[1].

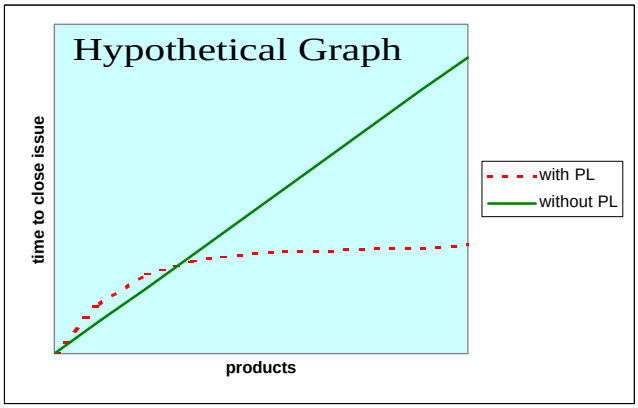

Figure 3 Time to close Customer Issues over Number of Products

Explanation:

As time to close customer issue we understand the time needed to understand customer requests, develop respective solutions and integrate these solutions into the respective products. This is usually perceived as maintenance or evolution of the products.

With single system development of similar products, a similar time per product to close customer issues can be expected: each product is treated separately, implying that the handling of n products takes about n times the time of handling one product.

If the change needed to solve the customer issue affects one product only, then the same time is needed as in the single system case. If it affects more than one product from the line, the respective product parts are likely to be realized in the common infrastructure. Once the change is performed in this infrastructure, it can easily be propagated to all product line members, resulting in less time per product with a growing number of products.

Due to higher complexity of the product line infrastructure vs. the structure of single systems, the time needed to change few products is likely to be higher for product lines than for single systems. But the more products are affected, the more product lines will benefit from performing the change in a central (and consistent!) way.

---

1. As for Hypothesis 2, the assumption is made that the available set of resources is kept constant.

Hypothesis 4:

Beyond a certain minimum investment, more features can be developed with a given amount of money when developed within a product line than without (cf. Figure 4).

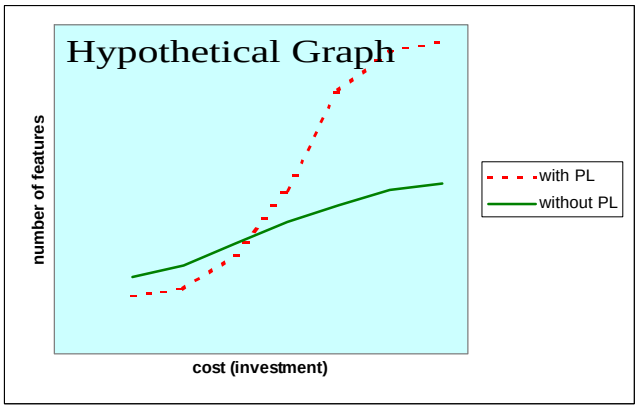

Figure 4 Number of Features developed over Money invested

Explanation:

For single system development, the number of features that can be developed grows roughly proportionally with the money invested. Over time, the curve can be expected to flatten due to feature interaction.

In product line development, planning and architecting to accommodate future changes is key. Thus, as soon as new features are requested for the first time, re-scoping of the product line and maybe an adjustment of the existing infrastructure in order to support the changed scope has to take place. This can be seen as re-investment in the product line. On one hand, this adjustment implies that it takes initially longer to integrate new features into the product line members than into products built as single systems. On the other hand, it is expected that after the adjustment has been performed, the product line will benefit again from the common infrastructure, meaning that it will be much cheaper to derive existing and new members from the product line than in the case of single systems that have to be handled one after the other.

In order to support more and more new features, the development and maintenance of the product line infrastructure becomes more complex, causing the feature-over-cost curve to flatten (cf. Figure 4).

Hypothesis 5:

With software product line development, more features can be developed within the same amount of time than without (cf. Figure 5).

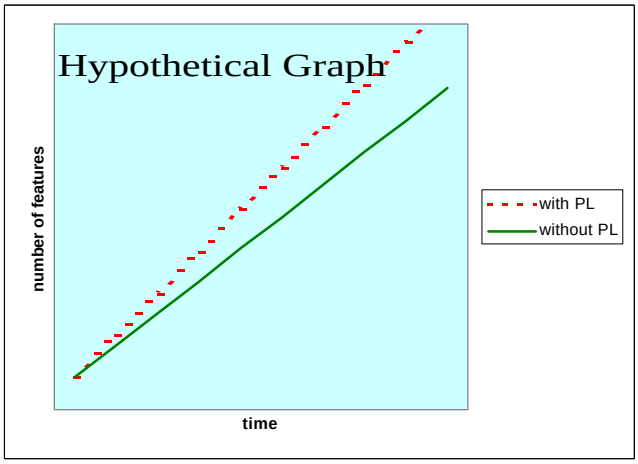

Figure 5 Number of Features developed over Development Time

Explanation:

For single system development, the time needed to develop new features grows roughly proportionally with the number of features.

For product line development, the argument runs similar: the more features have to be developed, the more development time is needed. But, if the respective features affect more than one product line member, it is likely that they have been taken into account during scoping of the product line and that the product line infrastructure is expected to support (or at least allow) the respective extensions and changes. Thus, it takes less effort and less time to integrate the new features than in the case of single system development.

Hypothesis 6:

The time per product to integrate COTS components into a product line decreases with a growing number of products, whereas it remains constant for products produced as single systems (cf. Figure 6)[1].

Explanation:

For single system development, each product is treated separately, taking about the same amount of time.

In the case of product line development, if the COTS integration affects only a single product from the line, the time needed for integration corresponds to the time needed in the single system situation. If the integration affects more than one product from the

1. As for Hypothesis 2, the assumption is made that the available set of resources is kept constant.

line, the respective product parts are realized in the common infrastructure. Once a COTS component has been integrated into this common infrastructure, all product line members can benefit from it at once.

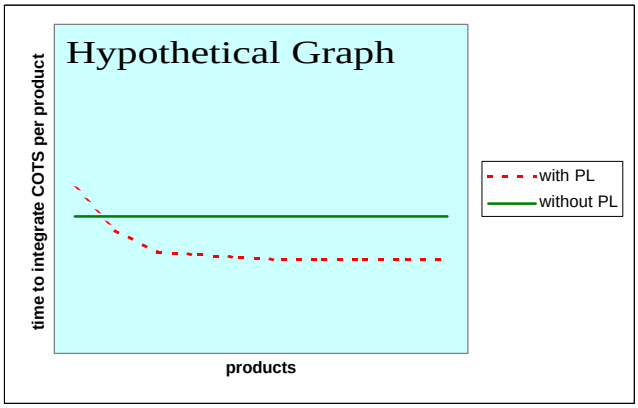

Figure 6 Time to integrate COTS per product over Number of Products

On the other hand, the initial effort for integration may be higher in the product line case due to some higher complexity of the product line infrastructure.

Hypothesis 7:

Beyond a certain minimum of products, the time to certify a product developed within a product line decreases below the time needed to certify products that have been developed as single systems (cf. Figure 7).

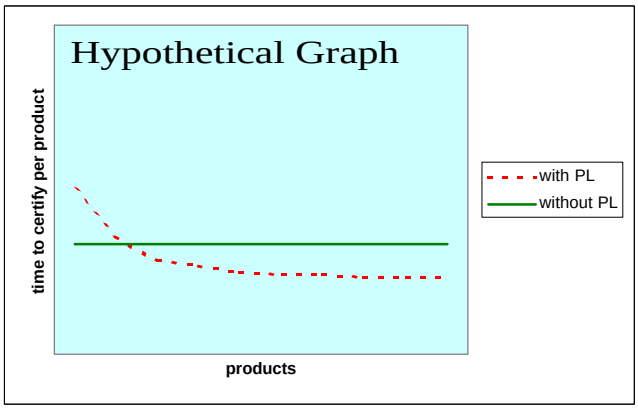

Figure 7 Time to certify Products over Number of Products

Explanation:

For the certification of single systems, each product is treated separately, taking about the same amount of time.

In the beginning, certification of product line members is harder, thus taking more time than for single systems, due to the higher complexity of the (generic) product line infrastructure.

Over time, the time to certify a single product line member is expected to be significantly lower than for stand alone products, due to the following reasons:

- With the certification of more products, parts of the certification procedure of some product line members can be expected to be repeatable directly for others, for example, test plans and test cases are directly reusable.
- These reused product parts can be expected to pass tests directly, thus avoiding costly and time-consuming correction and re-testing activities.
- For product parts where test cases are not directly reusable, still some experience resulting from the similar structure of all product line members speeds up the certification.

Summary

This paper hypothesizes benefits from product line over single system development in a quantitative form. A set of graphs illustrates the hypotheses. The hypotheses given in this paper should support building business cases for product line adoption, they address the particular concerns of product managers like time to market, time to react to customer issues, investment necessary to develop new features, and others.

The graphs are result of a discussion among the authors during a Dagstuhl workshop about Product Family Development in April 2001 [6]. Much more could be said about each graph, taking into account more influencing factors (like the quality of the product line scope and model, the maturity of the domain, or the similarity of the product line members) or estimating the shape curves representing incremental product line introduction would take (which can in most cases be expected somewhere between the product line and the single system curve). These considerations are not presented in this position paper deliberately. Instead, this paper is meant as a starting point of discussion on how to quantify which product line benefits and how they can be measured.

References

[1] Sholom Cohen: *Predicting when Product Line Investment Pays*, Proceedings of 2nd ICSE Workshop on Software Product Lines: Economics, Architectures, and Implications, P. Knauber, G. Succi (Eds.); IESE Technical Report No. 051.00/E

[2] C. Gacek, P. Knauber, K. Schmid, and P.C. Clements: *Successful Software Product Line Development in a Small Organization*; IESE Technical Report No. 013.01/E

[3] J. S. Poulin: *Measuring Software Reus*e; Addison-Wesley, 1997

[4] D.J. Reifer: *Practical Software Reus*e; John Wiley & Sons, New York, 1997

[5] D. Weiss, and R. Lai: *Software Product Line Engineering: A Family Based Software Development Process*; Addison-Wesley, 1999

[6] Product Family Development, Dagstuhl-Seminar, Organizers: G. Böckle, P. C. Clements, H. Obbink, K. Pohl, D. Rombach, Report No. 304, see http://www.dagstuhl.de/DATA/Seminars/01/#01161