Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 80 (2007) 74–91
The Journal of Systems and Software
www.elsevier.com/locate/jss

# Experimental use of code delta, code churn, and rate of change to understand software product line evolution

Samuel A. Ajila [a,*], Razvan T. Dumitrescu [b]

[a] Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, ON, Canada K1S 5B6
[b] AFORE Solutions Inc. 2081 Merivale Road, Suite 1300, Ottawa, ON, Canada K2G 1G9

Received 18 October 2005; received in revised form 11 May 2006; accepted 26 May 2006
Available online 11 July 2006

**Abstract**

This research is a longitudinal study of change processes. It links changes in the product line architecture of a large telecommunications equipment supplier with the company's customers, inner context, and eight line card products over six-year period. There are three important time related constructs in this study: the time it takes to develop a new product line release; the frequency in which a metric is collected; and the frequency at which financial results and metrics related to the customer layer are collected and made available. Data collection has been organized by product release. The original goal of this research is to study the economic impact of market reposition on the product line and identify metrics that can be used to records changes in product line. We later look at the product line evolution vis-à-vis the changes in the products that form the product line. Our results show that there is no relationship between the size of the code added to the product line and the number of designers required to develop and test it; and there is a positive relationship between designer turnover and impact of change.


*Keywords:* Software product line; Software evolution; Code delta; Code churn; Rate of change; Impact analysis; Software metric

## 1. Introduction

A software product line is a dynamic, long-lived system of artifacts that needs to evolve for a variety of reasons. Possible sources of change in product line include changes driven by an individual product, changes targeted to the entire product line, and repositioning of an architectural components from individual product to the product line (Bailetti et al., 2004; Ajila and Kaba, 2004; Bosh and Svahnberg, 1999). Modeling product lines provides substantial reuse advantages (Clements and Northrop, 2001) such as – *reduce time to market, reduce product development costs, improved process predictability, increased product quality, achieve large-scale productivity gains*, and *increase customer satisfaction*. The quality of any product line depends largely on the importance attached to the development process and its evolution (Ajila, 1994; Lehman, 1980). A software product is the result of a set of activities, which appeal to various competence and knowledge. Each activity manipulates different kinds of artifacts (Ajila, 1994, 1995) (requirements, design, coding, testing, etc.).

The individual evolution of a product in a software product line is affected by the evolution of the reused components as well as the evolution of the rest of the assets involved in the product line architecture (Ajila and Kaba, 2004). In traditional software engineering (single product), a sizable amount of money and time is spent doing software product evolution. This is so because of the complicity involved in the maintenance of software. For software product lines, the evolution of assets is even more complex compared to the traditional approach. The reason is that most assets depend on multiple software systems (Cook et al., 2001).

* Corresponding author. Tel.: +1 613 520 2600x2673; fax: +1 613 520 5727.
*E-mail addresses:* ajila@sce.carleton.ca (S.A. Ajila), razvan.dumitrescu@aforesolutions.com (R.T. Dumitrescu).

0164-1212/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2006.05.034

This research is a longitudinal study of change processes (Pettigrew, 1990). It links changes in the product line architecture of a large telecommunications equipment supplier with the company's customers, inner context, and eight line card products over six-year period. In this paper, customers refer to the companies that buy telecommunications switches from the supplier and the economic, social, and political circumstances in which they operate. Inner context refers to the characteristics of the product development unit's structure, designers, and inter-group relationships. Products refer to the four line card products that were repositioned so they could also be sold to the new target market and the four line card products that were developed for the new target market. The period of interest is marked by when top management decided to change the target market for its telecommunications switches to when the required product line repositioning was achieved. This period includes years of growth and downturn.

The original goals of this research are to study the economic impact of market reposition on the product line and to identify metrics that can be used to records changes in product line, customers, inner context, and product layers, and to identify potential links between the layers. Initial result of this study was reported in a paper (Bailetti et al., 2004). No attempt was made to study the complexity of the product line due to the impact of change. As time goes on it became apparent that we need to look at the product line evolution vis-à-vis the changes in the products that form the product line. In this later study the inner context and customers layers are limited to the cases where it is necessary to refer to the size of the designers involved in the products change. This is done in order to study the complexity of the code. We must understand that the evolution we are looking at here is more complex than that of a single product. We have four products that are developed before the IT market down turn and later changed to suit the new market. In addition four new products within the same product line were started to complement the initial adapted products making eight products in all. So, we are dealing with two issues – product line commonality and variability (Cook et al., 2001; O'Brian and Stoermer, 2001). In this paper, we seek to answer the question: "Has the product line become more complex over time due to the changes?" As software products change over time, it is sometimes difficult to understand and measure the effect of the changes. We hope to be able to show, numerically, how each product in the product line has evolved and how this evolution has affected the product line complexity and quality.

## 2. Background

### 2.1. Assumptions

Although this paper looks at the product line architecture and products layers, in the original goals we established three important assumptions (Bailetti et al., 2004) that are still valid. These assumptions are all consistent with the "contextualize" method (Pettigrew, 1990):

- That change in product line architecture needs to be studied in the context of other levels of analysis. For the purpose of this research, we assume that the source of change in product line architecture is the asymmetries between three interconnected levels: customers, inner context, and products. We expect that activities at some level may be more visible and rapid than at other levels.
- That identifying the temporal interconnectedness of changes in the product line architecture, customers, inner context, and products layers is important. For example, change in product line architecture is both constrained by and helps shape the other three levels of analysis.
- That history needs to be understood as events, chronology, structures, and underlying logics not just a series of events.

### 2.2. The company, the technology, and the study period

The company selected for this research provides hardware and software for communication and information to companies all over the world. It has a proven record of innovation and technological breakthroughs. Currently, the company has approximately 25,000 employees and has offices in the US, Canada, the UK, Continental Europe, Africa, Asia, and Australia.

From 1994 to April 1999, the business division that was responsible for the product line examined in this research was part of a US based company with 2800 employees. A large telecommunication equipment manufacturer with approximately 45,000 employees acquired the medium size company in April 1999. In September 1999, as a result of the ownership change, a new top management team was appointed to lead the business division responsible for the product line that is the focal point of this study. This was the only significant change in the top management team during the study period.

As of June 1997, the target for the product line was the enterprise market only. An institutionalized effort to reach the service provider market started in November 1997, when a service provider group was created within the marketing division of the business unit. The first results of this effort occurred in the second quarter of 1998. The company started using established service providers and competitive local exchange carriers as sales channels to reach the enterprise market.

In January 1998, a key strategic decision was made to incorporate web based user interfaces to the product line. Before this date, command line and SNMP (simple network management protocol) were the only available user interfaces for switch management. This was done because web interfaces were increasingly common on competitors' equipment. At the technical level, this decision was

implemented by adding a user interface convergence layer, which represents the point of separation between the internal management database abstractions and various user interfaces.

In September 1998, the business unit's top management team announced a strategy for service provider market penetration. The strategy was comprised of the following major points:

- create a new product line dedicated exclusively for the service provider market;
- reposition the enterprise based product line as access devices for new service provider product line.

As a result of this decision, a new engineering PLU (product line unit) was created in October 1998. The first two products of the service provider product line were released in November 1999. A total of four products were released exclusively for the service provider product line until the first quarter of 2002. The development of two additional products for the service provider product line was cancelled early 2002.

In March 2001, a decision was made to "merge" the two product lines. De facto this decision repositioned the products belonging to the enterprise product line to the service provider market.

In January 2002, the service provider PLU was formally merged with the Enterprise PLU creating a common enterprise and service provider PLU.

When conducting longitudinal studies of change, the choice of the start and end of the study period is critical. Time influences the perspectives of the researcher and set the frame of reference for what product line changes are seen and how these changes are explained. The study period selected for this research is six-year period starting from June 1, 1997 to November 30, 2003. During 1996, the company in question undertook a massive restructuring of the hardware support and base operating system for the product line. A new electronic repository system for design documents was inaugurated in June 1997. Development of the first product covered by this research started in May 1997. The first line of code was submitted to the revision control server (RCS) in June 1997. The first formal effort to address the needs of the service provider market occurred in November 1997 when a marketing group was established with a mandate to target the service provider market. The first release of the last product examined in this study was introduced in November 2003. During the study period, the strategic target market of the company changed from the enterprise market to the service provider market. This study allows enough time after the repositioning decision was made to observe the impact of such decision on the product line. In addition, the four line cards repositioned to the new market were written in C programming and language. The management and the development team decided to continue to use C to program the four line cards developed for the new market. We did not consider the user interface that was developed using a different programming language. In any case, this is not part of the product line core.

### 2.3. Data collection

There are three important time related constructs in this study: the time it takes to develop a new product line release; the frequency in which a metric is collected; and the frequency at which financial results and metrics related to the customer layer are collected and made available. Data collection has been organized by product release. The time it takes to develop a product line release ranges between 6 and 10 months. Metrics for these were collected quarterly. Financial results are reported every 6 months, using a financial year as a time reference. There is a 3-quarter offset between the financial year and the calendar year.

Data were collected to:

- emphasize actions taken by the company and changes in the four layers over time;
- examine the reciprocal relationships between changes in product line architecture and customers, inner context, and products;
- describes and analyses the often competing versions of reality seen by actors in change processes;
- take into account the historical evolution of ideas and actions for change and the constraints within which decision makers operate.

Data collection involved documentary and archive data that includes:

- product line requirements documents;
- product line architecture and design documents;
- core assets architecture and design documents;
- individual product requirements documents;
- individual product architecture and design documents;
- code stream for core assets;
- code stream for products;
- hardware support for each product;
- observational material.

In general, collected data are aggregated and reported quarterly. Such granularity will attenuate the impact of the expected irregularities in the initial development phases.

### 2.4. Software and software development process metrics

The metrics used in this study are based on extensive literature review and exploratory study of the data and team project management (Stark and Kern, 1994).

Lehman (1980) studied the evolution of software products over long periods of time. He concluded that, in order to avoid decay, a software product must change at a certain rate according to changing market requirements.

According to Lehman (1980), the rate of change in the product requirements is not constant over time.

Fenton and Pfleeger (1997) argued for the use of software size as a good predictor of development "costs" and the use of lines of code (LOC) and number of files to measure size. They also define the number of modules in a design as a metric for size. The number of modules is obtained from the definition of modules in the design documents. Fenton and Pfleeger (1997) define development time (measured in man-month) as an absolute measurement for the duration of a software development process.

Hall and Munson (2000) define the notion of "code churn" as the absolute number of changes made on a module during a development activity. Code churn refers to the intensity of the development activity, rather than the final result of it. Code churn can be used independently of its context to compare different development activities across an organization. Elbaum and Munson (1998) have found out a high correlation between the code churn and the number of problems discovered during the testing phase of a software product.

Clark and Wheelwright (1992) have identified four types of team structures: functional, lightweight, heavyweight and autonomous, and examined their appropriateness for different project contexts. Cain and McCrindle (2002) examine the link between organizational structure and the architecture of a software product. They found that having a tightly coupled development team leads to coupled software modules. Herbsleb and Grinter (1999) examined the impact of multi-site development teams on software architecture. They discovered that dispersed development teams tend to produce less stable software architecture. Brooks (1995) studied the effects of development team instability. He concluded that adding more programmers to a late project usually increases the development effort. Cook et al. (2001) built on Lehman's work (Lehman, 1980) and examined multi-release software products. They noted that an organization establishes, over time, processes which can accommodate only a certain expected rate of change.

From the literature review and our exploratory study of the data, we learned the following lessons:

(i) Architecting a product line involves a trade off between commonality and distinctiveness. Product line architecture is an iterative process that involves many stakeholders. The goal of the process is to separate variabilities (assets that provide distinctiveness to individual products) from commonalities (assets which belong to the product line). The intent is for individual products to achieve the distinctiveness required by their own functional specifications, while the overall level of commonality remains high.

(ii) LOC (lines of code) is a good metric for software artifacts. The size of a software artifact is a good predictor of development cost and the number of modules is indicative of the structure of software components. Development activity is characterized by development time and by code churn, which is a better indicator of the intensity of the product development effort.

(iii) Organizational structure influences the outcome of the software development activity. There is a relationship between the organizational structure of a software development group and the results of the development activity. In time, an organization develops processes able to handle a certain rate of change for a software product; however, product requirements change at a rate that is not constant over time. This is a big problem.

We then use the lessons learned to develop all the metrics in Section 3 (Tables 2–5).

### 2.5. Code delta, code churn, and rate of change

Various measurement techniques have been studied and used in software engineering. A great deal of work has been done especially in the area of using measurements of software systems to identify fault-prone components and to predict code complexity and quality after changes have been applied. Examples of these types of works include "Measuring Software Evolution", "Detection of Fault-Prone Programs", and "Lines of Code Metric as a Predictor of Program Faults" (Khoshgoftaar and Munson, 1990; Munson and Khoshgoftaar, 1992; Munson and Werries, 1996). Another example is the work of Lehman et al. on the "Laws of Evolution" (Lehman, 1980; Lehman and Ramil, 2001). In this paper, we borrow code delta, code churn, and rate of change ideas and tailor them to product line evolution (Khoshgoftaar and Munson, 1990; Munson and Khoshgoftaar, 1992; Munson and Werries, 1996; Elbaum and Munson, 1998; Hall and Munson, 2000; Nikora and Munson, 2003; Lehman, 1980; Lehman and Ramil, 2001). We summarize below three concepts – code delta, code churn, and rate of change. Readers who are interested in the theoretical framework for these concepts should please refer to the papers cited above.

A software product consists of one or more software modules. Each new release of the software is built from a set of software modules. The new release may contain some of the same modules as the previous version, some entirely new modules, and it may even omit some modules that were present in the old release. For example, given two releases of a software product (SP) R1 and R2 if the value of the chosen metric for R2 is greater than that for R1 on a particular module, this may indicate an increase in module complexity as measured by the chosen metric. Conversely, if the value of the chosen metric for R2 on a particular module is less than that of R1, this indicates a decrease in module complexity with respect to the chosen metric.

As software evolve through a series of changes (or builds), its complexity will change and this may affect the quality of the system. Software complexity is measured

by a set of metrics. An example is the lines of code (LOC) metric. It is not only enough just to look at the changes in the LOC but it is very important to see how these changes came about. Code delta by definition is the difference between two builds as computed for a particular metric (Khoshgoftaar and Munson, 1990; Munson and Khoshgoftaar, 1992; Munson and Werries, 1996). Code delta, therefore, can be calculated for lines of code. These deltas represent how the code has change (increased or decreased) with respect to the chosen metric. One of the limitations of code delta is that it is not an indication of how much change has the system undergone. If between releases, several modules are removed and replaced with modules of similar complexity, the code delta calculated will be near zero. Therefore, we need to measure the code churn to accompany code delta in order to indicate how much change has occurred. Code churn is calculated in similar manner to code delta but code churn is the absolute value of the code delta.

It has been proven that code delta and code churn is a good measure for estimating the impact of code change (Khoshgoftaar and Munson, 1990; Munson and Khoshgoftaar, 1992; Munson and Werries, 1996; Elbaum and Munson, 1998). Complexity metrics can provide good information on the differences between the products of a software product line and among the different releases of a particular product and a product line. There is enough evidence to conclude that relative complexity together with code deltas and code churn, are closely associated with measures of program quality. Code churn and code deltas can be used for aspects of software quality such as software faults that are very difficult to measure.

In this study, the evolution of volume of change across multiple releases is measured using the rate of change developed by Lehman (1980) and Lehman and Ramil (2001). The rate of change approximates the experience a company has with changes to a product or product line. The rate of change is calculated as follows:

Let $m_i$ be the value of an attribute for release $i$ (i.e., number of modules). We define incremental change for release $i$ as:

$$dm_i = abs(m_i - m_{i-1});$$

The rate of change ($rc_i$) for attribute $a$ at release $i$ is defined as the mean of incremental changes for all previous releases, not including release $i$.

Rate of change is useful to position a new release into a "historical" context (Fig. 1). Using this concept, Lehman provides a theory for evaluating the risks of a new release. Let $s_i$ be the standard deviation of the incremental growth over the last $N$ releases, not including release $i$. $N$ is a well defined number.

A release is:

- safe if $dm_i < rc_i$;
- risky if $rc_i < dm_i < rc_i + 2s_i$;
- very risky if $dm_i > rc_i + 2s_i$.

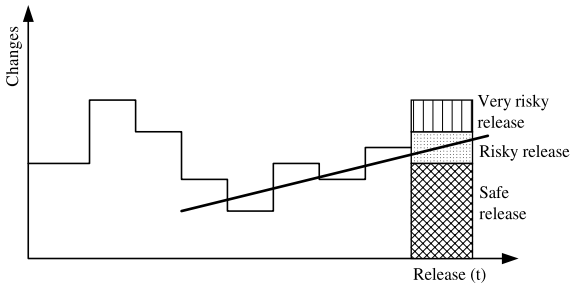


Fig. 1. Evolution of the safe rate of change.

Table 1
Product line release dates

| Release | Release date | Products introduced | Products repositioned |
|---|---|---|---|
| 1 | March 1998 | P(C1) | |
| 2 | August 1998 | | |
| 3 | October 1998 | P(F1) | |
| 4 | February 1999 | | |
| 5 | November 1999 | P(I1) | |
| 6 | February 2000 | | |
| 7 | May 2000 | | |
| 8 | July 2000 | | |
| 9 | December 2000 | | |
| 10 | March 2001 | | |
| 11 | June 2001 | P(E10) | |
| 12 | September 2001 | | |
| 13 | March 2002 | | |
| 14 | July 2002 | P(F3), P(C2) | P(I1) |
| 15 | January 2003 | P(I3) | |
| 16 | April 2003 | P(E1) | P(F1) |
| 17 | November 2003 | | P(C1), P(E1) |

According to this theory, each release increases or decreases the experience the organization has with changes on a particular product. After several releases of small changes (e.g., if the product was only in maintenance mode), the experience with a particular product decreases, and a new change intensive release is risky.

Releases are auxiliary constructs, which apply to both the product and product line layers. From customer perspective, a release is the result of a development process. A release occurs after the completion of a full testing regression cycle. During the study period, the product line introduced 17 releases (cf. Table 1). Each new product line release represents a new release for all individual products that belong to the product line.

## 3. Description of the product line (PL)

For the purpose of this research, a product line refers to the code comprising the set of core assets and the development effort required to change it. The common set of core assets refers to the lines of code that all products in the product line shared to satisfy the needs of enterprise and service provider customers (Bosh and Svahnberg, 1999;

Clements and Northrop, 2001). The set of core assets has two attributes: size and structure. Size is measured in terms of the number of LOC while the structure is measured in terms of the number of independent modules. Product line growth measures the incremental change in the size of the product line over a reporting period. For each quarter, product line growth was calculated as the number of LOC added to the product line during the quarter.

Code churn is used to measure development effort (Khoshgoftaar and Munson, 1990; Munson and Khoshgoftaar, 1992; Munson and Werries, 1996). We think that code churn is a better measure of development effort in this study because three projects included in the study were started and stopped at various times due to internal organization issues. Therefore, using development time in this case will be misleading. For each quarter, the impact of change is measured as the ratio between the code churn and code growth. A value close to one reflects changes with little modifications to the existing code base, while larger values reflect important modifications.

Some products (in the product line) may require changes only to the core assets, while others may target the core assets and a particular product. Within each project, development effort shall be measured separately for product line (core assets) and for each product.

There are three possible sources of change in the product line architecture: (i) Changes driven by individual product, i.e., changes made at the product line level to accommodate a new product, or support changes in an existing product, but without any intention to generalize the changes to other products. (ii) Changes targeted to the entire product line, i.e., when there is an explicit request to implement a certain behavior on several products within the product line. (iii) Repositioning of an architectural component (asset) from an individual product to the product line.

For each quarter in the study period, we shall report a normalized weighted score for each or the three possible sources of change described above. The score for each source of change shall be reported as:

$$\text{Score}(\text{class-of-change}) = \sum \frac{\text{Size-of-change-for-the-class}}{\text{Total-change-per-quarter}}$$

An adjusted product line growth shall be calculated each quarter by excluding changes due to asset repositioning from the quarterly volume of change. While asset repositioning represents a valid source of change for the product line, there is relatively less effort involved in such change (the code exists already) than for the code generated through the other two sources of change. Table 2 shows the metrics used to measure change in the product line layer. The data interval is quarterly.

During the study period, the declared objective of the company's top management team was to introduce a new product line release every 6 months. Table 1 provides the dates of the 17 product line releases that fall within the study period. The first release was introduced in March 1998 and the last in November 2003. Some of the releases are special. A special release refers to a release developed for a particular customer for the purpose of meeting a demonstration or lab testing deadline. Special releases contain a specific subset of the features and products that would have been included in the next regular release, and are replaced with next regular release as soon as such release becomes available. We will not consider special releases when dealing with rate of change and safeness of releases (Lehman, 1980; Lehman and Ramil, 2001) because this will constitute double evaluation of change. The life cycle of a product has two phases – a pre-release phase and a post-release phase. The pre-release phase is comprised of the bulk of development activities. The post-release phase is comprised of maintenance and incremental development activities. The repositioning projects of the four products form part of the post-release development activities (cf. Table A10).

### 3.1. Metrics

Data were collected to emphasize the changes in the product and product line layers over the study period. We also examine the reciprocal relationships between changes in the product line and product layers. Data collection involves documentary and archive data that includes: product line requirements, product line architecture and design, core assets architecture and design, individual product requirements, individual product architecture and design, code stream for core assets, and code stream for products. Typically, metrics were reported on a quarterly basis (cf. Tables 2 and 3).

Table 2
Metrics used to record changes in product line layer

| Metric | What was measured |
|---|---|
| Size of code in the product line | Number of LOC of the product line at the end of each quarter |
| Number of modules | Number of modules in the product line, as defined by the compiling/building process |
| Code churn | Sum of lines of product line code added, deleted, and modified during a given quarter |
| Source of change | Relative size of change for each source of change: product line, individual product, and asset reposition |
| Product line growth | Increase or decrease in the size of product line code from the previous quarter, measured in LOC |
| Impact of change | Ratio between the code churn and product line growth |
| Adjusted product line growth | Difference between product line growth and size of changes related to asset-repositioning activities |

Table 3
Metrics for product layer

| Metric | Data interval | What was measured |
|---|---|---|
| Size of product code | Quarterly | The number of LOC of the product code |
| Code churn | Quarterly | The sum of added, deleted and modified lines of code for individual product |
| Changes on product line | Collected once on first release | The size of the change (in LOC) made to the product line code required to accommodate the new product |
| Code churn on product line | Collected once on first release | The sum of added, deleted and modified lines of code belonging to the product line code that have been changed in order to accommodate the new product |

*3.1.1. Product line*

Table 2 shows the metrics used to measure change in the product line layer. The interval for data collection in each metric is quarterly.

The efficiency of the product line with regard to a product is calculated as the ratio between the size of the product and the size of changes required in the product line in order to support the newly introduced product. This metric measures an attribute of the platform rather than an attribute of the product.

*3.1.2. Product*

The product layer is comprised of eight products – P(C1), P(F1), P(I1), P(F3), P(C2), P(I3), P(E1), and P(E10). Each product is a line card. Four line cards were repositioned from old target market (enterprises) to new target market (service providers). Four line cards were developed explicitly for the new target market. From the architectural perspective, a product uses a subset of the product line core assets together with a set of architectural components that belong only to that product. We used the same approach as for the product line to measure changes in products. Table 3 gives the different metrics used to record changes in product layer.

*3.1.3. Customer*

The customer refers to the companies that buy telecommunications products from the supplier and the economic, social, and political circumstances in which they operate. The customers were classified based on the type of network they operate as: enterprise and service providers. Enterprise customers operate networks which carry traffic originating or terminating within the organization that owns the network. An enterprise network is a private network. Service provider customers multiplex traffic from various sources across its network. It is neither the originator, nor the terminator of the traffic (except some control traffic). A service provider sells "services" and contracts to its customers. A service provider network is a shared network. Customers influence products directly through explicit requirements and indirectly through increases in the buying power of their customers or potential customers. The features of the product line may influence the pool of potential customers. A change in the product line architecture may extend the pool of potential customers for the product line or it may reduce it. For example, introducing European interfaces for intercontinental networks as a result of explicit requests from North American service providers may create an opportunity to sell the products in the product line in Asia or Europe. The addition of a feature that does not satisfy customer requirements may reduce ability to sell the products in the product line. Table 4 shows the metrics used to measure change in the customer layer.

*3.1.4. Inner context*

The inner context layer includes constructs that are internal to the company but are not part of the product or customer. Three constructs of the inner context are examined in this research: structure, designers, and functional group. Three organizational entities are relevant to this study: product development unit, product line unit, and functional groups. The product development unit (PDU) refers to the R&D group. A product line unit (PLU) is a subdivision within the product development unit. A PLU is established as an organizational tool to clearly identify the development resources allocated to each product line. The company conceptualized a PLU as an informal group comprised of several functional groups working together on the same product line. The PDU is responsible for the product line and the individual products

Table 4
Metrics for customer layer

| Metric | Data interval | What was measured |
|---|---|---|
| Sales | Collected semiannually | Sales in millions of British Pounds (GBP) |
| Size of projects undertaken to satisfy customers' change requests | Collected quarterly and reported semiannually | Number of LOC |
| Source of change for the product line | Collected quarterly and reported semiannually | Numbers of LOC changed |

Table 5
Metrics for inner context

| Metric | Data interval | What was measured |
|---|---|---|
| Number of designers | Collected quarterly, reported per product and product line | Number of designers from each functional group assigned to individual product and product line |
| Number of designers in the business unit | Quarterly | Number of designers in each functional group that is part of the business |
| Number of code review | Quarterly | A score of 1 is given if the code review belongs to a different group |

during the entire study period. A functional group represents the group of designers with a common set of skills. A functional manager led each functional group. There were four distinct sites where the functional groups were located, three in North America and one in Europe. These sites will be referred to as HQ, S1, S2 and S3. HQ denotes that the designers were located at the company's headquarters, where the company's top management team was located. S1, S2 and S3 refer to the remote locations where the designers worked.

The second construct of the inner context is designers. Designers are individuals who architect, design, implement and test the products and the product line. Each designer formally belongs to one of the engineering functional groups. Designers are a particular class of stakeholders in the product definition process. Their input to requirements definition is sought and taken into account. However, individuals belonging to the marketing function did make the final decisions about product or product line requirements. For each line card product, changes in the number of designers allocated to it at each stage of the product's life were recorded. At each stage of a development project, new designers may be assigned to the project, while existing designers were reassigned to other projects. While there is a "natural" or expected designer turnover rate at each stage of the development of an individual product, there is no expected designer turnover rate for the product line. For example, it is expected that the bulk of hardware designers will leave a product development project early in the design phase, while the bulk of testing engineers will be assigned to the project at a late stage. For the development of the product line, the need for a particular set of skills depends mainly on the ongoing requirements. The numbers of engineers from each of the functional groups that were assigned to a product or a product line were recorded (using electronic timesheets). While an engineer may be assigned to different products, the major portion of the engineer's task related to a single product. In this study, we assume that – in terms of impact to the product or product line code – adding a new designer to a project will have the same effect as removing one from the project. This situation is conceptualized as an instance of designer turnover.

Two metrics were used to measure the relationships among functional groups: (i) proportion of individuals assigned to an individual product and the product line that belong to a functional group; and (ii) code submissions reviewed by members of other functional groups. The first metric measured the proportion of people in an individual product or product line that came from each functional group. The group that accounted for the largest proportion of total designers assigned to an individual product or the product line was deemed to have more power than the other groups. The second metric is used to examine the power a functional group had over other groups focused on code submissions. By power we mean the extent to which a functional group influences another functional group. Table 5 shows the metrics used to measure change in the inner context.

## 4. Results – product and product line evolution

Tables A1–A8 provide the changes in individual products in terms of LOC, code churn, and code delta. Table A9 shows the product line size – LOC, product line code delta, number of modules in the product line, and the product line code churn. The size of the product line code increased four times over the six-year study period. Table A9 shows that the size of the product line grew from approximately 887,162 LOC in the second quarter of 1997 to 2,384,772 LOC in the second quarter of 2003 and that the number of modules increased from 49 to 171 during the same period. Fig. 2 graphically illustrates the size of the product line during the study period. Three stages in the evolution of the product line can be identified:

- A fast growth phase (Q2-97 to Q2-99), during which the product line code size grew by 66%.

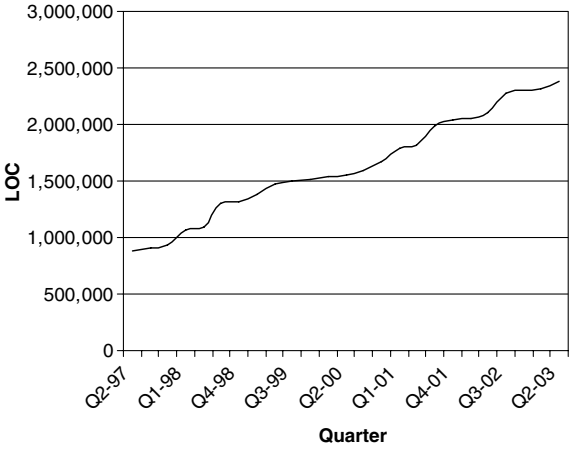


Fig. 2. Product line growth in LOC.

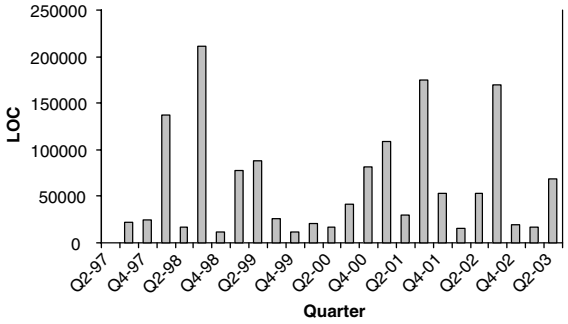


Fig. 3. Product line growth in LOC.

- A stale phase (Q3-99 to Q3-00), during which the product line code size grew by only 7%.
- A revival phase (Q4-00 to Q2-03), during which the product line code size grew by 49%.

The impact of change calculated at the product line level varies between 1.09 (Q4-00) and 4.30 (Q4-99). The impact of change is the ratio between code churn and product line code delta.

Fig. 3 shows the product line growth for each quarter in the study period.

### 4.1. Rate of change and safeness of releases

Table 6 describes the risk and safety thresholds for regular releases between 1999 and 2003. There are nine regular releases for the period. For each release, the safe rate of change as well as the risk threshold has been calculated using three previous releases for statistical mean and the product line growth between releases as base metric. The table shows that there were four risky releases and that none of the releases during the study period can be deemed "very risky".

### 4.2. Designers' turnover and code churn

Table A10 provides the number of designers assigned to the product, the designer turnover, the code churn and the development time. The total assignments of designers per product, as well as the code churn and development activities follow a rather expected pattern over the life of the product. During the pre-release phase, there seems to be a positive relationship between designer turnover and the impact of change (i.e., the ratio between code churn and product growth for a particular product). A similar relationship does not seem to hold for post-release changes.

### 4.3. Testing and correlations

Table 7 shows that the Pearson correlation coefficient between code size and test effort. Testing effort does not seem to depend on the product's target market. For example, P(I3) which targeted both service provider and enterprise markets, required eight person quarters of testing, while P(C1), one fifth of P(I3)'s size and targeted for one market only, requires 10 person quarters of testing.

Table 8 provides the correlation coefficient between product growth and number of designers assigned. Correlation has been calculated using product data from Tables A1–A8. Number of quarters for which data have been available varies per product and it excludes the asset-repositioning activity.

### 4.4. Product line efficiency

The efficiency of the product line measures the easiness to which a product line supports a particular new product (Table 9). This is a cross layer metric; it characterizes the

Table 6
Safe rate of change and risk thresholds

| Release | Risky threshold | Unsafe threshold | Actual change | Result |
|---|---|---|---|---|
| R1 | 140,903.0 | 370,019.4 | 77,741 | Safe |
| R2 | 125,112.5 | 347,275.5 | 125,524 | Risky |
| R3 | 125,194.8 | 240,106.2 | 37,720 | Safe |
| R4 | 110,615.7 | 198,534 | 41,566 | Safe |
| R5 | 100,751.4 | 199,992.8 | 190,508 | Risky |
| R6 | 111,971.0 | 286,217.3 | 203,874 | Risky |
| R7 | 122,182.4 | 302,379.1 | 68,686 | Safe |
| R8 | 116,832.8 | 265,818.4 | 223,615 | Risky |
| R9 | 126,540.3 | 295,198.7 | 105,617 | Safe |

Table 7
Product size and testing effort

| Product | Size | Testing effort (person quarters) |
|---|---|---|
| P(C1) | 8787 | 10 |
| P(F1) | 154,985 | 13 |
| P(I1) | 87,779 | 32 |
| P(E1) | 181,434 | 20 |
| P(C2) | 121,138 | 21 |
| P(F3) | 237,621 | 26 |
| P(I3) | 43,859 | 8 |
| P(E10) | 161,053 | 28 |

Pearson correlation $r = 0.53$, $r_2 = 0.28$, $p = 0.08$.

Table 8
Correlation coefficient between product growth and number of designers assigned

| Product | Pre-release correlation | Pre- and post-release correlation |
|---|---|---|
| P(C1) | −0.12 | 0.52 |
| P(F1) | 0.03 | 0.58 |
| P(I1) | 0.35 | 0.66 |
| P(E1) | 0.11 | 0.11 |
| C3 | 0.31 | 0.69 |
| P(F3) | −0.01 | 0.08 |
| P(I3) | 0.99 | 0.99 |
| P(E10) | 0.31 | 0.6 |

Table 9
Product line efficiency at product release

| Product | Product size | Product line change | Product line efficiency |
|---|---|---|---|
| P(C1) | 8787 | 1221 | 7.20 |
| P(F1) | 154,985 | 11,872 | 13.05 |
| P(I1) | 87,779 | 18,762 | 4.68 |
| P(E10) | 161,053 | 6102 | 26.39 |
| P(C2) | 121,138 | 7881 | 15.37 |
| P(F3) | 237,621 | 9971 | 23.83 |
| P(I3) | 43,859 | 1542 | 28.44 |
| P(E1) | 181,434 | 1109 | 163.60 |

Table 10
Efficiency of product line with regards to repositioning projects

| Product | Product repositioning change | Product line repositioning change | Efficiency |
|---|---|---|---|
| P(C1) | 465 | 32 | 14.5 |
| P(F1) | 508 | 302 | 1.6 |
| P(I1) | 1130 | 2812 | 0.4 |
| P(E10) | 942 | 287 | 3.2 |

product line, at a certain point in time, with respect to a particular product. A higher efficiency value indicates an easier integration of the new product in the product line. For the products which have been repositioned, we have measured the product line efficiency twice: once with regards to the product itself and with regards to the repositioning activity (Table 10).

Product line efficiency does not decrease significantly over time, (in fact, the four products introduced in 2002–2003 do have higher efficiency rates than products introduced in 1998–1999). Such evolution signifies a product line which did not enter a period of decay.

Prior to first product reposition (P(I1)), product line itself did undergo changes required by the repositioning process. Such product line changes are not reflected in metrics for a particular product. Within this context, the first repositioned product (P(I1)) did bear a higher percentage of product line changes, while the last product which has been repositioned (C3) had a smaller amount of requirements to the product line.

## 5. Observations

This section first identifies the potential links between changes in four layers of this study: customer, inner context, product line and product, and then positions the most important findings of this study within three major subsections.

### 5.1. Customer and product line layers

Fig. 4 illustrates the plots of sales, the LOC added to the product line, and the LOC added to the product line minus the lines of code directly related to the repositioning of

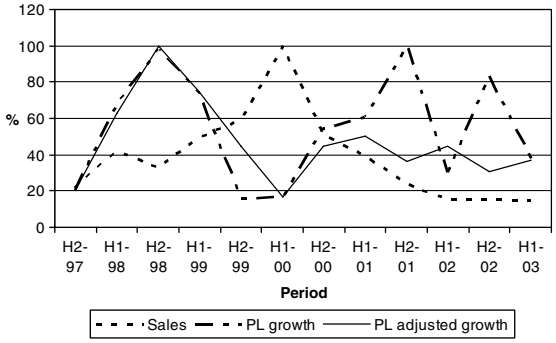


Fig. 4. Sales, Product line growth, and product line adjusted growth.

assets between the individual products and product line decision (cf. Table A11). For example, maximum sales occurred in the first half of 2000 (denoted H1-00). Accordingly, the sales for the second half of 1997 were 21.40% of H1-00 sales.

Three observations of interest can be made. First, Fig. 4 suggests that sales may be negatively related to the LOC added to the product line. During the market growth phase, LOC added to the product line decreases and sales increase. During the market decline phase, LOC added to the product line increases as sales decrease. The second interesting observation is that the LOC added to the product line were at the highest levels when the two strategic decisions were made: the January 1998 decision to incorporate a Web based user interface to the entire product line and the March 2001 decision to merge the product lines for the service providers and the enterprise markets. The third interesting observation is that after the market reposition decision (March 2001), the curve that reflects the LOC added to the product line adjusted for the size of repositioned assets (adjusted product line growth) closely follows sales. Prior to March 2001, this curve followed the curve for the LOC added to the product line.

Fig. 5 illustrates sales and the number of LOC of independent product line development over time (cf. Table

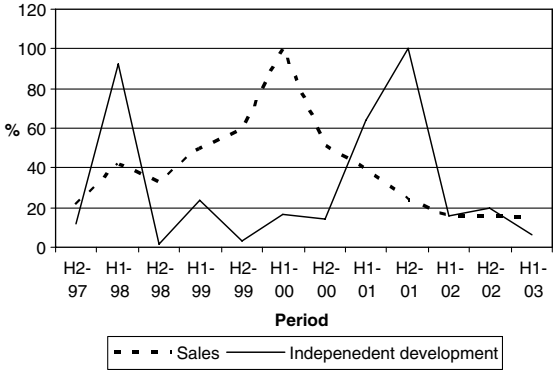


Fig. 5. Relationships between sales and independent development.

A11). The data suggest that the number of LOC of independent product line development increased at the time that two strategic decisions were made: the January 1998 decision to incorporate a Web based user interface to the entire product line and the March 2001 decision to merge the product lines for the service providers and the enterprise markets. Figs. 4 and 5 are both derived from Table A11.

### 5.2. Inner context and product line layers

The Pearson correlation coefficient between the 24 quarterly observations of the number of LOC added to the product line and the number of software designers assigned to develop the product line is 0.009 and is not significant at $p < .10$ (two-tailed). Further examination of the data shows that the quarterly output per designer varies from 102 LOC/designer (fourth quarter 1999) to 2069 LOC/designer (third quarter 1998).

The Pearson correlation coefficient between the 24 quarterly observations of the number of LOC added to the product line and the number of test engineers assigned to the product line is 0.075 and is not significant at $p < .10$ (two-tailed).

The low correlation results suggests that, for this particular product line, the number of LOC added to the product line is not a good predictor of the number of designers required to undertake development and testing in the product line.

### 5.3. Product and product line layers

Fig. 6 shows the breakdown of changes in the number of LOC added to the product line due to individual product lines, product line development, asset repositioning, and maintenance (including small projects). It also shows that at the beginning of the study period, most of the changes to the product line were driven by product line related projects. Around the time the market repositioning decision was taken, significant asset-repositioning activities started to take place. These asset-repositioning activities include moving assets from individual products to the product line.

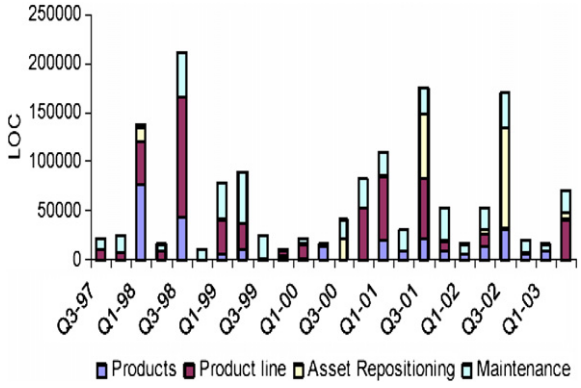


Fig. 6. Profile of changes to the product line code by source of change.

Table 11
Impact of change to the product line for each product

| Product | PL churn | LOC added to PL | Impact of change | Location |
|---|---|---|---|---|
| P(E10) | 23,221 | 6102 | 3.81 | HQ |
| P(F1) | 16,542 | 11,872 | 1.39 | Remote |
| P(C1) | 1401 | 1221 | 1.15 | Remote |
| P(E1) | 11,982 | 1109 | 10.80 | HQ |
| P(C2) | 20,886 | 7881 | 2.65 | Remote |
| P(F3) | 14,311 | 9971 | 1.44 | Remote |
| P(I3) | 2871 | 1542 | 1.86 | Remote |
| P(I1) | 33,212 | 18,762 | 1.77 | HQ/remote |

Changes to the product line can be classified into two types: product line restructuring and product line extension. Product line restructuring includes changes to the existing code undertaken to support a new feature. Such operations entail high code churn, low number of LOC added, and very few new files added to the product line. Product line extensions occur when new a module, class or other dedicated entity is added to the product line for the purpose of handling a specific scenario, even at the expense of duplicating code. The new entity interacts with the old 'core' via a few well defined Application Program Interfaces (APIs).

Table 11 shows the impact of change generated in the product line, the team that generates the change and the location of the team. The impact of change is calculated as the ratio between the number of LOC added to the product line and the associated code churn. Table 11 also shows that the impact of change scores for products developed by remote sites are usually lower than the scores for products developed by headquarters. The maximum score obtained by a product developed by a remote site is 2.65 while the maximum score obtained by a product developed at headquarters is 10.80. In addition, the table shows that groups located at headquarters made many changes to the product line code, but added few lines of code while remote groups added a larger amount of code but made fewer changes. This suggests that the remote groups tend to duplicate existing code more, modify it and then added to the product line. The rationale for these results resides in the fact that the product line was traditionally owned by functional groups located in headquarters. Therefore, designers belonging to groups located in headquarters felt more at ease changing their own or their collocated peer code.

## 6. Summary and conclusions

We start this section by summarizing the key decisions the company's top management team made that are relevant to this research.

In September 1998, the Business Unit's top management team announced a strategy for service provider market penetration. The strategy was comprised of the following major points: (i) Creates a new product line dedicated exclusively for the service provider market. (ii) Reposition the enterprise based product line as access devices for the new service provider product line.

As a result of this decision, a new engineering PLU was created in October 1998. The first two products of the service provider product line were released in November 1999. A total of four products were released exclusively for the service provider product line until the first quarter of 2002. The development of two additional products for the service provider product line was cancelled early 2002.

In March 2001, a decision was made to "merge" the two product lines. De facto this decision repositioned the products belonging to the enterprise product line to the service provider market.

In January 2002, the service provider product line unit (PLU) was formally merged with the Enterprise PLU creating a common enterprise and service provider PLU.

### 6.1. Conclusions

In the first place, the decision to reposition the product line resulted in:

- *Increase in the number of designers allocated to the product line*. In December 2000, prior to the product line reposition decision, there were 126 designers assigned to the product line. This number increased to 155 in the next quarter, when the repositioning decision was taken, followed by increases to 170 and 165 in the subsequent quarters.
- *Increase in the product line growth and code churn*. Between Q4-99 and Q4-00, prior to the repositioning decision, the product line growth was, on average, approximately 34 KLOC per quarter. In the subsequent five quarters (from the moment the decision was taken until the first reposition product has been released) the average product line growth was approximately 65 KLOC per quarter. In terms of code churn, it increases from an average of 58,418 in the five quarters (prior to the repositioning decision) to 96,628 in the five quarters after the repositioning decision.
- *Increase in independent development asset reposition from products to the product line*. The size of independent development increased after the product line repositioning decision has been taken. There are two major reasons for such increase:
  - The reposition work itself accounts for speculative development.
  - The relationship with new customer base comprised of service providers was weak; therefore, most subsequent development was speculative rather than based on explicit requests.

In absolute terms, the average independent development in the five quarters prior to the repositioning decision was on average 9753 LOC per quarter, while in the subsequent five quarters, the independent development averaged 35,774 LOC per quarter. In relative terms, the independent development represented 28.20% of the total product line growth in the five quarters before the repositioning decision, and increased to 54.84% of the total product line growth in the five quarters subsequent to the repositioning decision.

Product line growth due to asset repositioning (from individual products to the product line) in the five quarters prior to the reposition decision, accounted for 21,060 LOC, representing 12.18% of the total product line growth. In the five quarters following the market repositioning decision, asset repositioning accounted for 73,192 LOC, representing 22.44% of the product line growth.

Secondly, our study concludes that code size is not associated with designer assignment, both at the product and product line levels. A rationale for such result at the product level may be the fact that each designer assigned to a product during the pre-release development phase, developed skills in a very narrow area. While this rationale applies, to a certain extent, to the product line as well, the output per designer varies, over the study period. Such variation may be explained, in part, by over and under allocation of resources, through managerial decision, during various stages of the product line development. Such result is consistent with a study done by Elbaum and Munson (1998) which has found consistently low correlations between the number of designers and the code churn for some medium size (150 KLOC) software products.

Thirdly, we also conclude that code size is not a good predictor of testing effort (measured by allocated test engineers) at either product or product line levels. Other metrics, like code coupling and cohesion are more appropriate for such estimation (Fenton and Pfleeger, 1997). At the same time, contextual factors (like the target market) influence the amount of testing activities required for a particular release of the product line. In addition, testing effort does not seem to depend on the product's target market (cf. Table 11). For example, P(I3) which targeted both service provider and enterprise markets, required eight person quarters of testing, while P(C1), one fifth of P(I3)'s size and targeted for one market only, did require 10 person quarters of testing.

Fourthly, during the pre-release phase, there seems to be a positive relationship between designer turnover and the impact of change. A similar relationship does not seem to hold for post-release changes.

Finally, we outline below four important findings from this research:

- *Code size and designer assignment.* Our study concludes that code size is not associated with designer assignment, both at the product and product line level. In addition, code size is not a good predictor of testing effort. There are three particularities of a telecommunication product, which may restrict the ability of predicting the required testing effort based on the software metrics only. (i) A telecom product is a hardware–software system and in many cases an important part of the decisional complexity is moved to the hardware components; (ii) the behavior of a telecom product is usually driven by asynchronous events; (iii) different applications have different levels of tolerance to defects. For example, voice products are simpler than data products – however they require a much higher level of resiliency. In addition, contextual factors (like the target market) influence the amount of testing activities required for a particular release of the product line.
- *Effects of team collocation.* From the formal process perspective, all teams included in this study have equal privileges regarding to changes to the product line code. There is no explicit approval process required in order to change a code originated by a different team; each designer is free to change any part of the code according to his/her needs. However, this study found that project teams are more likely to change existing code originated by collocated designers, rather than changing code, which has been originated by designers from remote locations. This result confirms the findings of Herbsleb and Grinter (1999) and Brooks (1995), related to the role of informal contact between developers. A large amount of information related to the behavior of the code is exchanged through informal communication. Such communication is minimal between non-collocated teams. At the same time, this result qualifies the finding of Cain and McCrindle (2002), related to coordination between teams. In the case of a product line development, an organization may want to use different locations for the product line development versus individual product development as an informal tool for enhancing product line stability. Less informal interaction may lead to a more formal process in the product line changes; which may be, in particular contexts, a desired outcome (Ramanujan et al., 2000).
- *Market effect on product and product line development.* This research finds that market downturn has an explicit impact on development activities. The first reaction to the market downturn by the studied organization was to increase the development of its core product line. Such result qualifies Lehman's fifth law of software evolution, which predicts a long term decrease of the rate of growth for a given software product (Lehman, 1980) by restricting its applicability to stable or growing market conditions. (Lehman's law applies to the studied product line during the market growth phase.) The consistency of such impact across multiple organizations should be subject to further study.

Table 12
Product line moment correlations

| | Product line LOC | PL code delta | No. of modules |
|---|---|---|---|
| PL delta | 0.07656 | | |
| No. of modules | 0.98570 | 0.03251 | |
| PL code churn | −0.8532 | 0.87703 | −0.12164 |

- *Product line code complexity.* The Pearson moment correlations among the experimental variables are shown in Table 12. This is calculated from the data in Table A9. One interesting observation here is the relationship between code churn and code delta. Had all changes resulted in new code, the correlation reported would have been 1.0. The reported correlation of 0.87703 is very close to 1.0 which means that the majority of the changes made over the study period resulted in increased code complexity (Schneidewind, 2001). Also of interest is the relationship between code churn and number of modules. This correlation is very low. We can conclude that the number of modules involved in the change is not related to the magnitude of the change.

### 6.2. Limitations and further work

We see a number of limitations in this study. First, this study focused on one product line of one company. Results may not be generalized to other product lines in the same company, other companies in the industry, or other industries. The second limitation is that changes in the product line and individual products were measured using the number of LOC. The use of this metric is limited because it does not account for code complexity. Important descriptors of code complexity, such as code coupling and code cohesion, and testing activity, such as number of bugs found, were not measured in this study. Thirdly, the financial performance of the individual products was not included in this study. In the fourth place is that contextual factors (like the target market) influence the amount of testing activities required for a particular release of the product line.

The scope of this research may be extended by trying to answer two questions – Is it common behavior across organizations to respond to a decline in sales by increasing development of the code in individual product or product line targeted towards a stable or growing market? Does the power of functional groups (closest to the customers) increases during sales decline? A second research direction is to determine if testing the relationships observed in this study can be used to predict development and testing requirements for a product line? Thirdly, one can try to use case studies to examine how senior and junior software designers split responsibilities and the diffusion of the knowledge about the product line across functional groups. In addition, it will be very interesting to see the impact of the split responsibilities and knowledge diffusion on the economics of software product line development. Fourthly,

a new layer in this study will be to will be an issue related to the efficiency of the development team, productivity, localization, soundness of product line development plan, and quality plan. In this study we have already addressed the issue of team collocation which is somewhat related to localization. This was also addressed by Herbsleb and Grinter (1999) and Brooks (1995). Lastly, it would also be interesting to mathematically model the market behavior in order to predict when changes are necessary to the product line or when repositioning a product is required.

## Appendix. Data for individual products

See Tables A1–A11.

Table A1
LOC, code churn, and code delta for product P(E1)

| Quarter | LOC | Code churn | Code delta |
|---|---|---|---|
| Q2-99 | 2663 | 2770 | 2770 |
| Q3-99 | 28,916 | 44,769 | 26,253 |
| Q4-99 | 36,300 | 90,615 | 7384 |
| Q1-00 | 74,135 | 144,138 | 37,835 |
| Q2-00 | 112,405 | 157,606 | 38,270 |
| Q3-00 | 141,705 | 109,709 | 29,300 |
| Q4-00 | 149,584 | 17,169 | 7879 |
| Q1-01 | 162,126 | 59,982 | 12,542 |
| Q2-01 | 161,053 | 6871 | 1073 |
| Q3-01 | 169,703 | 22,078 | 8650 |
| Q4-01 | 169,712 | 215 | 9 |
| Q1-02 | 169,849 | 337 | 137 |
| Q2-02 | 169,980 | 170 | 131 |
| Q3-02 | 171,188 | 1307 | 1208 |
| Q4-02 | 148,972 | 28,716 | 22,216 |
| Q1-03 | 151,403 | 4188 | 2431 |
| Q2-03 | 151,552 | 265 | 149 |

Table A2
LOC, code churn, and code delta for product P(F1)

| Quarter | LOC | Code churn | Code delta |
|---|---|---|---|
| Q3-97 | 2875 | 2983 | 2875 |
| Q4-97 | 37,669 | 55,413 | 34,794 |
| Q1-98 | 79,982 | 80,973 | 42,313 |
| Q2-98 | 139,116 | 238,715 | 59,134 |
| Q3-98 | 152,431 | 34,934 | 13,315 |
| Q4-98 | 154,985 | 9946 | 2554 |
| Q1-99 | 165,423 | 12,430 | 10,438 |
| Q2-99 | 186,887 | 24,745 | 21,464 |
| Q3-99 | 189,922 | 9410 | 3035 |
| Q4-99 | 196,043 | 11,839 | 6121 |
| Q1-00 | 196,498 | 734 | 455 |
| Q2-00 | 196,504 | 25 | 6 |
| Q3-00 | 199,997 | 8487 | 3493 |
| Q4-00 | 200,347 | 1845 | 350 |
| Q1-01 | 200,347 | 4 | 0 |
| Q2-01 | 200,566 | 412 | 219 |
| Q3-01 | 200,569 | 8 | 3 |
| Q4-01 | 200,569 | 4 | 0 |
| Q1-02 | 200,670 | 105 | 101 |
| Q2-02 | 200,695 | 20 | 25 |
| Q3-02 | 200,694 | 6 | 1 |
| Q4-02 | 200,699 | 16 | 5 |
| Q1-03 | 201,324 | 1629 | 625 |
| Q2-03 | 202,178 | 8263 | 854 |

Table A3
LOC, code churn, and code delta for product P(C1)

| Quarter | LOC | Code churn | Code delta |
|---|---|---|---|
| Q2-97 | 3836 | 4766 | 3836 |
| Q3-97 | 6590 | 6703 | 2754 |
| Q4-97 | 8303 | 7509 | 1713 |
| Q1-98 | 8332 | 4344 | 29 |
| Q2-98 | 8787 | 508 | 455 |
| Q3-98 | 8810 | 212 | 23 |
| Q4-98 | 8829 | 25 | 19 |
| Q1-99 | 8847 | 28 | 18 |
| Q2-99 | 8847 | 4 | 0 |
| Q3-99 | 8848 | 3 | 1 |
| Q4-99 | 8852 | 5 | 4 |
| Q1-00 | 8860 | 17 | 8 |
| Q2-00 | 8860 | 1 | 0 |
| Q3-00 | 8864 | 6 | 4 |
| Q4-00 | 8864 | 1 | 0 |
| Q1-01 | 8930 | 142 | 66 |
| Q2-01 | 8922 | 67 | −8 |
| Q3-01 | 8922 | 1 | 0 |
| Q4-01 | 8937 | 26 | 15 |
| Q1-02 | 8937 | 2 | 0 |
| Q2-02 | 8937 | 4 | 0 |
| Q3-02 | 8944 | 7 | 7 |
| Q4-02 | 8944 | 3 | 0 |
| Q1-03 | 8942 | 9 | −2 |
| Q2-03 | 9407 | 588 | 465 |

Table A4
LOC, code churn, and code delta for product P(E10)

| Quarter | LOC | Code churn | Code delta |
|---|---|---|---|
| Q3-01 | 122,778 | 128,716 | 122,778 |
| Q4-01 | 140,122 | 22,651 | 17,344 |
| Q1-02 | 153,520 | 17,777 | 13,398 |
| Q2-02 | 177,453 | 30,871 | 23,933 |
| Q3-02 | 177,063 | 2236 | −390 |
| Q4-02 | 181,355 | 4989 | 4292 |
| Q1-03 | 181,464 | 172 | 109 |
| Q2-03 | 181,436 | 44 | −28 |

Table A5
LOC, code churn, and code delta for product P(C2)

| Quarter | LOC | Code churn | Code delta |
|---|---|---|---|
| Q2-01 | 1302 | 1654 | 1302 |
| Q3-01 | 93,248 | 126,543 | 91,946 |
| Q4-01 | 112,061 | 48,765 | 18,813 |
| Q1-02 | 118,647 | 12,543 | 6586 |
| Q2-02 | 121,138 | 5432 | 2491 |
| Q3-02 | 13,065 | 1432 | −108,073[a] |
| Q4-02 | 13,176 | 287 | 111 |
| Q1-03 | 13,964 | 991 | 788 |
| Q2-03 | 14,048 | 154 | 84 |

[a] This is due to asset repositioning to the product line.

Table A6
LOC, code churn, and code delta for product P(F3)

| Quarter | LOC | Code churn | Code delta |
|---|---|---|---|
| Q2-01 | 5421 | 6898 | 5421 |
| Q3-01 | 138,162 | 163,625 | 132,741 |
| Q4-01 | 200,187 | 155,241 | 62,025 |
| Q1-02 | 209,827 | 43,251 | 9640 |
| Q2-02 | 237,621 | 56,524 | 27,794 |
| Q3-02 | 295,184 | 125,413 | 57,563 |
| Q4-02 | 285,317 | 66,524 | −9867 |
| Q1-03 | 291,005 | 10,872 | 5688 |
| Q2-03 | 293,212 | 2652 | 2207 |

Table A7
LOC, code churn, and code delta for product P(I1)

| Code | LOC | Code churn | Code delta |
|---|---|---|---|
| Q2-02 | 11,064 | 16,543 | 11,064 |
| Q3-02 | 31,071 | 38,432 | 20,007 |
| Q4-02 | 43,426 | 47,002 | 12,355 |
| Q1-03 | 43,859 | 10,265 | 433 |
| Q2-03 | 45,541 | 4872 | 1682 |

Table A8
LOC, code churn, and code delta for product P(I3)

| Quarter | LOC | Code churn | Code delta |
|---|---|---|---|
| Q3-98 | 33,737 | 52,881 | 33,737 |
| Q4-98 | 47,565 | 39,712 | 13,828 |
| Q1-99 | 70,852 | 64,577 | 23,287 |
| Q2-99 | 83,195 | 36,645 | 12,343 |
| Q3-99 | 84,353 | 18,700 | 1158 |
| Q4-99 | 87,779 | 10,430 | 3426 |
| Q1-00 | 88,937 | 4779 | 1158 |
| Q2-00 | 90,304 | 2008 | 1367 |
| Q3-00 | 91,606 | 1877 | 1302 |
| Q4-00 | 91,616 | 32 | 10 |
| Q1-01 | 91,616 | 2 | 0 |
| Q2-01 | 91,616 | 0 | 0 |
| Q3-01 | 91,568 | 388 | −48 |
| Q4-01 | 90,047 | 3969 | −1521 |
| Q1-02 | 90,047 | 203 | 0 |
| Q2-02 | 90,830 | 1775 | 783 |
| Q3-02 | 91,315 | 878 | 485 |
| Q4-02 | 91,330 | 18 | 15 |
| Q1-03 | 91,689 | 382 | 359 |
| Q2-03 | 91,689 | 4 | 0 |

Table A9
Product line (PL) size, growth, number of modules, and code churn

| Quarter | PL LOC | PL code delta | No. of modules | PL code churn |
|---|---|---|---|---|
| Q2-97 | 887,162 | | 49 | |
| Q3-97 | 908,712 | 21,550 | 49 | 29,080 |
| Q4-97 | 932,875 | 24,163 | 57 | 53,776 |
| Q1-98 | 1,070,762 | 137,887 | 65 | 308,881 |
| Q2-98 | 1,087,612 | 16,850 | 65 | 44,144 |
| Q3-98 | 1,298,743 | 211,131 | 87 | 651,321 |
| Q4-98 | 1,309,871 | 11,128 | 91 | 34,090 |
| Q1-99 | 1,387,612 | 77,741 | 97 | 144,970 |
| Q2-99 | 1,476,123 | 88,511 | 102 | 132,599 |
| Q3-99 | 1,501,621 | 25,498 | 107 | 65,933 |
| Q4-99 | 1,513,136 | 11,515 | 112 | 49,603 |
| Q1-00 | 1,534,013 | 20,877 | 114 | 28,981 |
| Q2-00 | 1,550,856 | 16,843 | 117 | 63,680 |
| Q3-00 | 1,592,422 | 41,566 | 124 | 60,083 |
| Q4-00 | 1,674,509 | 82,087 | 136 | 89,744 |
| Q1-01 | 1,782,930 | 108,421 | 139 | 132,296 |
| Q2-01 | 1,812,559 | 29,629 | 139 | 50,767 |
| Q3-01 | 1,986,804 | 174,245 | 139 | 208,572 |
| Q4-01 | 2,040,215 | 53,411 | 155 | 199,050 |
| Q1-02 | 2,055,490 | 15,275 | 156 | 23,394 |
| Q2-02 | 2,108,962 | 53,472 | 161 | 97,991 |
| Q3-02 | 2,279,105 | 170,143 | 163 | 296,774 |
| Q4-02 | 2,298,721 | 19,616 | 165 | 24,413 |
| Q1-03 | 2,315,529 | 16,808 | 171 | 19,718 |
| Q2-03 | 2,384,722 | 69,193 | 171 | 85,755 |

Table A10
Designer assignment, designer turnover, and code churn

| Product | Pre-release | | | | Post-release | | | |
|---|---|---|---|---|---|---|---|---|
| | No. of designers | Designers turnover | Code churn | Development time (months) | No. of designers | Designers turnover | Code churn | Development time (months) |
| P(C1) | 28 | 45 | 23,830 | 24 | 26 | 55 | 1151 | 63 |
| P(F1) | 37 | 50 | 422,964 | 16 | 46 | 69 | 79,982 | 56 |
| P(I1) | 57 | 90 | 222,945 | 30 | 39 | 81 | 16,315 | 43 |
| P(E1) | 36 | 52 | 207,412 | 17 | 0 | 0 | 44 | 3 |
| C3 | 23 | 22 | 194,937 | 16 | 21 | 31 | 2864 | 11 |
| P(F3) | 33 | 29 | 425,593 | 15 | 46 | 55 | 205,461 | 11 |
| P(I3) | 38 | 35 | 112,424 | 8 | 12 | 21 | 4872 | 5 |
| P(E10) | 66 | 93 | 633,629 | 13 | 30 | 57 | 57,276 | 25 |

Table A11
Sales, LOC added to the product line, LOC added to the product line excluding the size of assets repositioned to the product line, and independent development LOC

| | Time period[a] | Sales (%) | LOC added to PL[b] (%) | LOC added to PL excluding assets repositioning[c] (%) | LOC due to independent development (%) |
|---|---|---|---|---|---|
| Market growth phase | H2-97 | 21.40 | 20.08 | 20.57 | 11.96 |
| | H1-98 | 41.53 | 67.97 | 62.68 | 92.02 |
| | H2-98 | 33.05 | 97.63 | 100.00 | 1.81 |
| | H1-99 | 49.58 | 73.03 | 74.80 | 23.76 |
| | H2-99 | 59.32 | 16.26 | 45.00 | 3.45 |
| | H1-00 | 100.00 | 16.57 | 16.97 | 16.49 |
| Market downturn phase | H2-00 | 51.06 | 54.32 | 44.51 | 14.34 |
| | H1-01 | 39.41 | 60.64 | 50.26 | 64.20 |
| | H2-01 | 23.94 | 100.00 | 36.39 | 100.00 |
| | H1-02 | 15.47 | 30.20 | 44.92 | 16.12 |
| | H2-02 | 15.47 | 83.35 | 30.68 | 19.54 |
| | H1-03 | 14.62 | 37.78 | 36.94 | 6.08 |

[a] H2-97 = second half of 1997 and H1-98 = first half of 1998.
[b] Calculated as percentage of the maximum number in the time series of the number of LOC added to the product line.
[c] Calculated as percentage of the maximum number in the time series of the number of LOC added to the product line excluding the size of repositioned assets.

## References


Ajila, S.A., 1994. Software process assistance: management of objects modifications in an integrated software engineering environment. Nigerian Journal of Science 28 (3), 249–264.

Ajila, S.A., 1995. Software maintenance: an approach to impact analysis of objects change. International Journal of Software – Practice and Experience 25 (10), 1155–1181.

Ajila, S.A., Kaba, A.B., 2004. Using traceability mechanisms to support software product line evolution. In: The 2004 IEEE International Conference on Information Reuse and Integration (IEEE IRI-2004), November 8–10, 2004, Las Vegas, NV, USA, pp. 157–162.

Bailetti, A.J., Ajila, S.A., Dumitrescu, R.T., 2004. Experience report on the effect of market reposition on product line evolution. In: The 2004 IEEE International Conference on Information Reuse and Integration (IEEE IRI-2004), November 8–10, 2004, Las Vegas, NV, USA, pp. 151–156.

Bosh, J., Svahnberg, M., 1999. Evolution in software product lines. Journal of Software Maintenance 11 (6), 391–422.

Brooks, F.P., 1995. The mythical man-month, after 20 years. IEEE Software 12 (5), 57–60.

Cain, James, McCrindle, Rachel, 2002. An investigation into the effects of code coupling on team dynamics and productivity. In: Proceedings of the 26th International Computer Software And Applications Conference, Oxford, UK, pp. 907–918.

Clark, K.B., Wheelwright, S.C., 1992. Organizing and leading 'heavyweight' development teams. California Management Review 34 (3), 9–28.

Clements, P., Northrop, L., 2001. Software Product Lines: Practices and Patterns. Addison-Wesley, New York.

Cook, S., He, J., Harrison, R., 2001. Dynamic and static views of software evolution. In: Proceedings of IEEE International Conference on Software Maintenance (ICSM'01). IEEE Computer Society Press, pp. 592–601.

Elbaum, S., Munson, J., 1998. Code churn: a measure for estimating the impact of code change. In: Proceedings of International Conference on Software Maintenance (ICSM '98), Bethesda, MD, November 1998, pp. 24–31.

Fenton, N.E., Pfleeger, S.L., 1997. Software Metrics: A Eigorous and Practical Approach, second ed. International Thompson Computer Press, London.

Hall, G., Munson, J., 2000. Software evolution: code delta and code churn. The Journal of Systems and Software #54, 111–118.

Herbsleb, J.D., Grinter, R.E., 1999. Architectures, coordination and distance: Conway's law and beyond. IEEE Software 15 (5), 63–70.

Khoshgoftaar, T.M., Munson, J.C., 1990. The lines of code metric as a predictor of program faults: a critical analysis. In: Proceedings, Fourteenth Annual International Computer Software and Applications Conference, COMPSAC 90, October 31–November 2, 1990, pp. 408–413.

Lehman, M.M., 1980. On understanding laws, evolution and conservation in the large program life cycle. Journal of Systems and Software 1 (3), 213–221.

Lehman, M.M., Ramil, J., 2001. Rules and tools for software evolution planning and management. Annals of Software Engineering 11, 15–44.

Munson, J.C., Khoshgoftaar, T.M., 1992. The detection of fault-prone programs. IEEE Transaction on Software Engineering 18 (5).

Munson, J.C., Werries, D.S., 1996. Measuring software evolution. In: IEEE Proceedings of METRICS '96.

Nikora, A.P., Munson, J.C., 2003. Understanding the nature of software evolution. In: Proceedings of International Conference on Software Maintenance, ICSM, September 22–26, 2003, pp. 83–93.

O'Brian, L., Stoermer, C., 2001. MAP – mining architectures for product line evaluations. In: Proceedings of the Third Working IEEE/IFIP Conference on Software Architecture, Amsterdam, Netherlands, August 28–31, 2001, pp. 35–45.

Pettigrew, A.M., 1990. Longitudinal field research on change: theory and practice. Organization Science 1 (3), 267–292.

Ramanujan, S., Scamell, R.W., Shah, J.R., 2000. An experimental investigation of the impact of individual, program, and organizational characteristics on software maintenance effort. Journal of Systems and Software 54 (2000), 137–157.

Schneidewind, N.F., 2001. Investigation of logistic regression as a discriminant of software quality. In: Proceedings of the 7th International Software Metrics Symposium, London, April 2001, pp. 328–337.

Stark, G.E., Kern, L.C., 1994. A software metric set for program maintenance management. Journal of Systems and Software 24 (3), 239–245.