# Software product architectural integrity during organizational distress

By

Scott Miller

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering in Telecommunications Technology Management

Department of Systems and Computer Engineering

Carleton University

Ottawa, ON, Canada, K1S 5B6

January 28, 2006

*Your file Votre référence*
*ISBN: 978-0-494-16484-6*
*Our file Notre référence*
*ISBN: 978-0-494-16484-6*

**ABSTRACT**

This exploratory research examines how firms integrate the software from the open source software projects that they establish into their market offers and how they use these projects to gain competitive advantage. Data on twelve companies is used to identify: (i) types of market offers that are based on open source software, (ii) ways open source software is integrated with proprietary assets into market offers, (iii) insights gained from using Teece's commercialization framework to examine an OSS as a complementary asset, (iv) ways companies use open source software projects to change their competitive environments, (v) benefits and risks of setting up open source projects and (vi) what can be done with open source software relative to alliances, standards and freeware. This research provides concrete examples of how companies use open source software projects as complementary assets for competitive purposes.

**ACKNOWLEDGEMENTS**

I am forever grateful to my supervisor Professor Tony Bailetti for his guidance, dedication, and suggestions throughout this study. Professor Bailetti was a very active supervisor and he invested as much time in this research as I did. Our discussions about this research have been very stimulating and educational. I would also like to thank Professor John Callahan for all his inputs, suggestions, support, and ideas for improvement to this study from beginning until the end. Professors Bailetti and Callahan both dedicated themselves to my success and to the success of all the other students in the TTM program.

I would like to thank my wife, Lea, and my family for all the love, support, encouragement, and understanding while I was completing this study.

This thesis was developed as part of the Competing in Open Environments research program led by Tony Bailetti, Carleton University, and Peter Carbone, Nortel. The support received from Nortel for this research and the M.Eng. in Telecommunications Technology Management program is gratefully acknowledged.

**Acknowledgements**

Many people have helped me with this research, and I want to thank them all for their help, time and encouragement. In particular I thank my supervisor Dr. John Callahan, who has been a source of patient help and encouragement throughout.

I would like to express my appreciation to my employer for permitting me to utilize data from one of their products. Identifying names and places have been modified to protect their identity; and specific organizational size numbers are presented as percentages in order to preserve corporate confidentiality.

Support begins at home; I would like to acknowledge the patience, support and occasional nudge to "get working" from my wife Diana and daughter Laura. Without them this work would not have been completed.

**Table of contents**

## List of tables

## List of figures

## List of appendices

# 1 Introduction

The focus of this grounded study was a large telephony network infrastructure product. The objective of the study was to extend the existing literature on the relationship between product architecture and the structure of the organization that designed it. As a grounded study, the focus changed as the data emerged. Initially I thought that I would find that the product architecture had eroded and through analysis would be in a position to make recommendations on how to approach maintenance of the architecture to position the product for growth post-downturn. What I uncovered is that the product architecture in this case was resilient, and did not significantly erode even though the product development organization went through significant staff reductions.

'Conway's Law' as attributed to Melvin Conway [Conway, 1968] states: "the structure of a system mirrors the structure of the organization that designed it". In this framework lies the significance of product structure within the context of the organization; and opens up opportunities to study the linkages between what is designed, how it is designed, and the impacts that the organization and its structure have on the end product.

Tying product architecture together with Conway's Law creates the framework for this study: Changes to the organization have an impact on the product architecture.

Since the architecture did not degrade in the case of this product, there are key insights into how they enabled this stability, the impact of staff reductions in key organizational functions on post release error report arrivals, increasing saleable feature creation in smaller teams. In addition, insight into a situation where software code churn; which is often used as a predictor of design activity and post-release error report arrival rates may not correlate.

The maintenance and intelligent extension of the product's architecture by the development organization provided the stable footing needed to enable increased sales to a larger available market during and after the downturn.

There are 3 key areas of contribution from this research. The first is addition to the growing body of literature inspired by the relationship between product architecture and the structure of the design organization (Conway's Law).

A set of findings is developed from this longitudinal study. These findings can be tested against a broader range of products, markets and organizations to test the general applicability of the findings.

As an outcome of the findings development discussion, emergent recommendations from this product's successful development and deployment become evident. These may be applicable to other products and markets undergoing a downturn as strategy inputs towards maintaining their own architectural stability.

This Thesis is organized into seven sections. The first introduces this research at a high level. Section 2 provides brief review of the product and its positioning over the study period. The third section looks at the current state of the art with respect to this topic as published in key journal articles. Section 4 outlines the research method utilized for the study. The fifth section presents the data that has been collected within the individual study contexts, as well as between them where linkages exist. The insights from key staff members are also captured in Section 5. Section 6 draws out the insights from the raw data, and develops the logic that uncovers the findings uncovered through this study. The conclusions are captured in Section 7, and encapsulate what was uncovered, as well as the limitations and opportunities for further study.

## 1.1 Research method

A longitudinal research method was employed to study how to maintain product architecture during market downturns. The longitudinal method has been successfully applied to several studies; including those for innovation processes [Poole, Van De Ven, Dooley and Holmes, 2000], creativity and technological learning [Kazanjian, Drazin and Glynn, 2000], change theory and practice [Pettigrew, 1990] and theory building [Christensen, Carlile and Sundahl, 2002].

This research was based on a single case study of the development of a large software based product developed for the infrastructure of telephone network operators. The method as outlined by Pettigrew [1990] to study change theory and practice was applied. The temporal longitudinal links between the contexts of market, organization and product were examined.

## 1.2 Researcher involvement

The researcher was employed in the development organization that developed the product over much of the study period. Every effort has been made to provide a view of what occurred as an unbiased observer. Where possible, facts and raw data were utilized to obtain results so that researcher bias could be minimized. Where this was not possible, interviews with key

contributors to the program were held in order to provide a diversity of views and details of tacit data not retrievable from ex-post analysis of the raw data.

## 2 A brief review of the product

The focus of the research is a single large telecommunications network infrastructure product (over 5 million non-commented lines of software source code), deployed to service providers through several releases globally over a 5-year study period. This scalable platform product utilizes multiple distributed processors interconnected via a multi-protocol messaging system; and is designed to be highly available and have high reliability. During the study period, the product went from small deployments in small telephony networks, through the telecom boom, bust and bottom, all the while continuing to service the market and experience increased revenues. It grew into larger deployments in larger markets over each release as a result of increasing product capabilities and capacity.

The development organization was part of a large, full line telecom network infrastructure company. The development organization expanded, contracted and changed structurally, as did the telecom market during the study period. The product provided the framework for the introduction of capacity increases and new capabilities in new markets, while continuing to service the existing product market over the study period.

The product was originally targeted for a single specific market segment; as a replacement for a legacy product. The replacement of the legacy product was necessary to enable the development company to offer increasingly larger capacities to network operators, and to reduce their operational expenses. The motivation for the company was to improve revenue, and have a next-generation platform for serving new emerging market opportunities.

There were five major product releases during the study period, labeled TPR1 through TPR5. These were deployed to the market in 2Q1998, 4Q1999, 4Q2000, 4Q2001 and 3Q2002 respectively.

This initial product offering provided router capabilities within the telephony network. Over the first three software releases (TPR1 through TPR3), the product link capacity increased and a variety of physical interface types were introduced to bring the new product offering towards parity with the legacy product.

Over the course of the study period, the product was re-positioned to capture a new opportunity. The original target of functional equivalence with the legacy product providing router functionality in the fourth release was deferred in favor of this new opportunity that positioned the product for

continued growth in a new area not well defined when the product development was originally planned. The first release with this new capability was TPR4, where a stand-alone gateway capability was introduced onto the platform product.

The TPR4 release was deployed to the new market segment as an initial installation that provided the gateway capability only. Subsequent releases starting with TPR5 included functionality to support the original market opportunity, as well as providing software configurable co-residency of the new capability. This co-residency was enabled by software provisioning. The ability to offer both functions on the same physical hardware provided additional operational and capital expense savings opportunities for customers, as the gateway capability became necessary within their network.

The survival of the product and organization during the market turbulence, coupled with the detailed data available provides a rare opportunity to study the development organization, its product, how it is structured, how it was successful, and what this or other organizations could do to maintain their product through future downturns.

Additional details of the product during the study are detailed in Section 5.3.

# 3 Literature review

Two main areas of research literature were reviewed as the basis for this research: the foundation for the research method, and existing, documented linkages between the three horizontal components of market, product development organization and product architecture. The literature review in this section documents the works reviewed to provide the foundation for undertaking the research.

Over the course of the study, additional literature was uncovered to explain or understand key findings. When these additional works provide context or background to findings, they are interspersed inline in subsequent Thesis sections.

## 3.1 Research method

This study broadens the scope of available research on product development by taking advantage of the quantity and quality of data available for this case. The longitudinal study method uncovers the contextual changes over time, and provides a method for mapping between the three contexts of market, development organization and product.

Lewis [2001] identified that there is considerable scope for more longitudinal investigations of new product development. Although the analysis of a single product may not be easily generalized, it can deepen our understanding of the linkages between product architecture and organizational structure. The success of the development organization and its product during the telecom meltdown of 2001 provides an extreme bound for uncovering factors important in maintaining product quality during market downturns. The insights gathered in this study as testable findings can be further tested for applicability in other endeavors.

As noted by Eisenhardt: "....given the limited number of cases which can usually be studied, it makes sense to choose cases such as extreme situations and polar types in which the process of interest is 'transparently observable'....." [Eisenhardt, 1989:537].

### 3.2 Product and market considerations

The market itself does not directly perform design activities on a product. Organizations are structured so that market requirements are captured and used as the planning input to a development cycle [Royce, 1970]. In essence, the market channels its requirements through the development organization, which in turn interprets those requirements and implements them on the product for delivery to the market.

## 3.3 Development organization and product considerations

### 3.3.1 Development organization – Conway's Law

Conway [1968] wrote that as a consequence of the communication between people during the development of a product, product architecture mirrors the organization that builds it. This concept has become known as "Conway's Law" The application of Conway's Law to software product architecture in the literature is becoming more common ([Herbsleb and Grinter, 1999], [Bowman and Holt, 1993], [Dekel and Kane, 2002]).

The communication and coordination aspects of integrating a single large software product release that linked directly to Conway's Law [Herbsleb and Grinter, 1999]. This paper documented the critical need for informal communication channels that could spontaneously be invoked as a mechanism to deal with project ambiguity. The implication from a product release perspective is that if the informal communications network across the organization was not functioning to its full potential; the coordination problems across the project became exaggerated, even if formal documentation and processes were in place.

Bowman and Holt [1993] employed Conway's Law [1998] as a tool for documenting and recovery of software architecture, extending the toolset for

reverse engineering a software system. Studying three large software systems (Linux, Mozilla and a commercial software development system) ranging in size from 800,000 to 3.8 million lines of code, they provided empirical validation that the structure of a software system is a direct reflection of the structure of the development team, validating Conway's Law.

Conway had two ideas in his original paper [Conway, 1968] that are relevant to this study. The first is the linkage between the product and its development organization, originally attributed to Conway as a linear relationship; and validated through the study by Bowman and Holt (1998). The second key idea is a structure-preserving relationship between the product and the development team. Conway called this a homomorphism.

Homomorphism is defined as : "A transformation of one set into another that preserves in the second set the operations between the members of the first set.". Applying the idea of homomorphism is the essence of this study in two key dimensions. First, within the longitudinal study itself, event data are linked and contextual linkages are drawn out between events to generate opportunities for further research, and to identify opportunities to maintain product architecture.

Second, homomorphism as used by Conway introduces the idea that Foote and Yoder [1997] expanded on – namely the most frequently deployed software architecture is one of a casually structured system. In the case of Conway, he described this as follows, "...the homomorphism insures that the structure of the system will reflect the disintegration which has occurred in the design organization".

### 3.3.2 Development organization – tacit considerations

Adaptability and resiliency have been identified as attributes of successful organizations (Conway [1968], Brookes [1995] and Collins [2001]). Intrapersonal functional diversity of the individual members of the organization has a positive impact on team process and performance [Bunderson and Sutcliffe, 2002].

Booz, Allen, Hamilton [2003] have developed a questionnaire for measuring the adaptability and resiliency of organizations. They call the measure Organizational DNA™. They use Organizational DNA™ to categorize organizations into six types: Resilient, Just-in-time, Military, Overmanaged, Outgrown, Fits-and-starts, Passive-Aggressive or Inconclusive. The concept is that if your organization is adaptable, it is one that is poised to succeed. Booz, Allen, Hamilton have developed an online questionnaire that measures

structure, decision rights, motivators and information flow within an organization, and provides an overall organizational profile.

### 3.3.3 Product considerations and architecture

Product architecture is a platform product's key enabler for adding features and capabilities quickly [Foote and Yoder, 1997]. Solid product architecture also eases maintainability [Land, 2002]; improves product quality [Lundberg, Bosch, Haggander and Bengtsson, 1999], and enables software re-use [Foote and Opdyke, 1995].

The definition of software architecture has proven both difficult and controversial. The Software Engineering Institute lists over 25 published definitions and many more contributor definitions. Bass, Clements, Kazman [2003] define software architecture as "the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them".

Diversity of opinion on what the definition of software architecture extends into techniques for capturing, categorizing, measuring and mapping. The importance of measuring and tracking the development and maintenance of software has been recognized in many studies, for example the work of Briand, Morasca, and Basili [1993]. In the case of that study, the change in

the architecture and a measure of its "improvement" or "degradation" is required.

In this study, two key software development process tools are employed: software churn analysis release over release, and a commercially available CASE tool, KlocWork. Software churn has been shown to be positively correlated to the number of problems found during product test [Elbaum and Munson, 1998].

KlocWork Insight is used to examine the release over release changes of key software attributes, including defect density, low-level interface problems, clusters, and system complexity. These measures track product line improvement or degradation.

Architectural stability [Jazayeri, 2002] measures of product quality are possible through predictive analysis and retrospective analysis. KlocWork and software churn measures are providing the predictive data; while field trouble reports of each release provide data for retrospective analysis. Retrospective analysis is possible because field trouble reports are generated after the development activity has taken place, and act as an overall measure of development through detailing what errors escaped the product development cycle.

KlocWork also provides insight into product architecture through its "cluster" metric. The KlocWork definition of a cluster is "a group of files that have a cyclical dependency on each other". The existence of clusters can be a sign of architectural erosion.

### 3.4 Market and development organization considerations

The linkage between the market and the development organization can be conceptualized in this study as that of a proxy relationship. The market works through the development organization as a proxy, to create the product that will meet the market need. This is a symbiotic relationship, with two principle communication paths:

1. At the front end of the development cycle, when new requirements are captured and negotiated for content in an upcoming product release
2. At the completion of the development cycle, where the product releases to the customer, and is deployed

At the completion of the development cycle, deployment is often the first exposure that the market has to a product that encapsulates the market's original requirements. It is during deployment that the market reports numbers of product error reports. Error reports can be lack of functionality

originally specified in market requirements, or errors introduced as a result of flawed development activity.

Within the product development cycle, product verification (PV) as a function within a development organization acts as the error locators for the design activity; while the design function within an organization corrects the errors after they are reported.

Capturing post release error reports against the RQMS metrics [Telcordia GR-929, 2002] provides the detailed data for retrospective product and organizational analysis.

## 4 Research method details

A grounded research methodology [Glaser and Strauss, 1967] was utilized on a single case [Eisenhardt, 1991]. Following Pettigrew [1990], quantitative data on three key contexts of the case were gathered and analyzed:

- Overall market size
- Organizational size and structure
- Software architecture and quality

In addition, personal interviews with key members of the product development organization were carried out. The data from these interviews were used to clarify and augment the quantitative data.

The following research process was followed, with additional details as to the method and outcomes in the balance of Section 4 through Section 6:

**Initial data gathering activity:**

- Software static predictive analysis was performed on each TPR software release
- Organizational headcount data was gathered and plotted

- Market data was captured and plotted
- Software error reports were captured from the defect reporting database, and plotted across each of the TPR releases

**Second literature review:**

Based on the researchers' review of the data from the initial data gathering activity, works were searched for and reviewed to categorize the organization in the face of change in a quantitative way. This led to the discovery of the OrgDNA tool as a categorization tool to understand the character of this development organization.

**Initial interviews**

Two separate interviews, each with software development team members of one hour duration were done. During the interviews, the OrgDNA questions were posed and responses captured for input into the tool. The interviewees were shown the plots of the software predictive code analysis and software error arrivals over time and releases.

## Third literature review

A third literature review was undertaken to look for foundations in software predictive measurement and post arrival error reports.

## Subsequent interviews

The next interviews were also one hour in duration, and were held with a first level manager, a senior manager and one of the formal architects from the development organization during the study period. They were also shown and asked to comment on the same data plots as the initial interviewees. They were asked the questions from the OrgDNA questionnaire for later input, just as the designers in the initial interviews. As their position within the organization was above the designers, they were also asked questions from Appendix F to provide the researcher access to more tacit understanding of the strategic decisions of the organization during the downturn that were within their purview.

## Final data analysis

A final round of data gathering and analysis was performed based on areas identified from the interviews. This led to gathering data to ascertain:

- The size of the product verification team and the software architects during the study period.
- The organizational efficiency in delivering saleable features
- The mix of product functions deployed in the later software releases during and after the product re-positioning.

**Final literature review**

A final literature review was completed to identify if any of the research findings had been previously identified or studied.

### 4.1 Study period

The study deals with the five-year period from January 1, 1997 to December 31, 2002. During this period, five product releases were developed and deployed to the market. The study period start date was chosen to coincide with the full development cycle of the initial product release that was deployed in the third quarter of 1998. The end date was the bottom of the trough of the telecom bubble burst.

In order to capture a consistent and settled view of each product release, a consistent period of post-release time was chosen. This period of time was 9 quarters; as during this time the number of field reported issues from the data

had settled in most cases. This time selection also coincided with the original product strategy to support each software release for two years (8 quarters) post full commercial availability. Data for field reported problems was only examined in the aforementioned 9 quarter time period. Overall, this required that field trouble report data extended through to December 31, 2004.

### 4.2 Time

There are key time windows and periods throughout this study. Key ones being:

- Release development time
- Deployment time to maturity (field issue settling)
- Periodicity of data availability/measurement

### 4.3 Study Contexts

The three contexts for this research are: Market, Organization, and Product. Changes to individual contexts have impacts to the others through relationships based on roles and processes; where processes can be formally defined or more informal.

A pictorial representation is helpful for visualizing the contexts and their interconnections (Figure 1 Research Model):

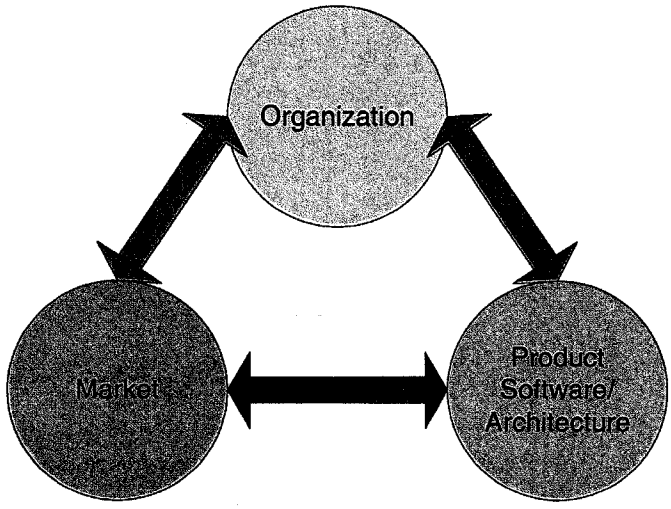


**Figure 1 Research Model**

### 4.3.1 The market

The NASDAQ Telecommunications Index, Symbol IXTC was chosen as a broadly accepted measure of the health, size and liquidity of the entire telecommunications sector including customers and vendors. The market for the product studied is measured directly as the number of units deployed and in-service globally. Both market measures were captured quarterly.

Since the product was re-positioned in the TPR4 release, I measured the number of sites that deployed the new product capability during the TPR4 and TPR5 release. This was done in order to understand the extent of mix of sites utilizing the new product capability versus the sites that took the new release but were using the product's original capabilities.

| Metric | Measurement Interval | What was measured? |
|---|---|---|
| NASDAQ: ITXC | End of each quarter | %/% of the peak value of the index over the study period. The maximum value is the 100% reference point, with all other measures being in relation to that 100% value |
| Product deployments | End of each quarter | Running count of in-service systems |
| Product deployments | Per release | The peak number of in-service systems running a software release. |
| Product capabilities | Per release | The mixture of sites as a percentage running the original product capability and the re-positioned product capability. Totals to 100% of deployed sites. |

**Table 1 Metrics for changes to the market**

Product deployments per release were measured as a base count of sites on particular releases. For example, software churn per site on a particular release utilizes this data.

Product capabilities (saleable features) per release were measured also, to understand what contribution the product repositioning played in new product deployments. This provides insight into the growth rate of the new capability within the study period, and speaks to the organizations adaptability to develop and deploy this new capability.

### 4.3.2 Organization

The development organization has been principally examined by the number of individuals (headcount) associated with the development and design, verification and field support of the product. Refinements to this dataset around specific job functions are possible due to organization chart availability through the study.

As the development organization being studied was part of a full line telecommunications infrastructure company, changes to the health of the overall telecommunications market were directly reflected to the entire company. As a constituent part of the larger company, the overall market effects were relayed to this development organization.

Interviews with key members of the organization were done to further uncover informal as opposed to formal organizational structure, decision rights, information flow and organizational motivators.

Additional event data was captured through organizational notices and archival emails.

The headcount data for the development organization was captured quarterly from historic organizational charts. Other data seen as important from other

sources, such as organizational notices was captured and added to the organization timeline so that it could be included in the analysis.

| Metric | Measurement Interval | What was measured? |
|---|---|---|
| Development organization headcount | End of each quarter | Number of direct employees in the development organization, measured as a percentage of maximum value during the study period. |
| Development organization formal architects | End of each quarter | Number of formal product architects that were part of the development organization, measured as a percentage of the development organization. |
| Development organization product verification strength | End of each quarter | Number of product verification members that were part of the development organization, measured as a percentage of the development organization |

**Table 2 Metrics for changes to the organization**

Within the development organization itself, capturing its overall size allows an understanding of the connective-ness and causality in relation to the corporation and the market. Based on the interviews held with the development organization members, the number of people performing architecture and product verification functions was extracted from the overall development organization numbers. This data was extracted to validate their interview insights.

### 4.3.3 Product architecture/quality

The product architecture was examined for each of the five software releases for this product; TPR1 through TPR5. At each release the following predictive analysis was performed on the software source code base:

- Software code churn
- Software code base size
- Software defect density
- Number of features deployed per release
- Software code churn per feature
- System complexity
- Percentage of software files that are cyclically dependent

| Metric | Measurement Interval | What was measured? |
|---|---|---|
| Code churn | Per release | Sum of non-comment lines of source software added, deleted or modified. An overall measure of development effort. |
| Code base size | Per release | Number of lines of code (LOC) in the product. Growth release over release is measured after TPR1. |
| Defect density | Per release | Predictive number of errors per thousand LOC (KLOC) |
| Feature count | Per release | Number of features the development organization introduced in each release. |
| Churn per feature | Per release | A measure of how much design activity was performed to delivery a field deployable feature |
| System complexity | Per release | A measure of the Cyclomatic Complexity, which correlates to maintainability and testability of the product |
| Clusters of cyclically dependant files | Per release | A relative measure of product architecture erosion across releases |

**Table 3 Metrics for changes to the product**

The predictive measures in Table 3 represent standard and well-known measuring tools for software products. They were measured for each TPR release to document the underlying changes to the product over the study period. These metrics were not being measured at the time of the study.

The development organization utilized a stage-gate process for managing and tracking their development cycle. A series of milestones were scheduled, with a set of criteria pre-defined and agreed to by the development organization that would be tested prior to passing each of the gates. The gate

process follows a waterfall model for development that is widely practiced. In the case of this development organization, they tracked higher level metrics to determine readiness to move through each gate. No direct software metrics were tracked or measured.

The higher level metrics that were product related that were used to determine readiness to move through the gate prior to the beginning of field deployment were:

- All design activities completed.
- Product verification (PV) cycle complete, with 100% of planned test cases executed and greater than 96% pass rate.
- Zero (0) service impacting problem reports that did not have software resolution or plans for resolution in place from the development cycle that had just completed PV.
- System capacity measurements captured on the final load, with all measures meeting or exceeding the associated standards defined criteria.

Because the software predictive measures from Table 3 were not measured during the study to provide input into design process changes, they represent an unregulated measure of the product change for each metric over time for each metric. They also provided a basis for discussion of the organization

and market linkages to the product with the development organization interviewees.

Retrospective analysis of the number of post release error reports was performed. These post release errors are those which are reported by customers after the product has been released to the market. They capture feedback on the product quality exiting the development organization. This post release data was captured for all critical and major errors. These errors report problems which may be service impacting. They were captured from the release date for 9 quarters following the initial release deployment. The nine quarter period was selected after reviewing the error report arrival data for the product and determining that this was a consistent settling period where issues reduced to a low amount across all product releases. This provided a consistent time window for measurement after first deployment.

From these errors, a random sample of the reports during each release was taken and categorized them as either software faults or non-software faults. Non-software faults were outside the scope of this development organization to remedy and were not analyzed.

From the random sample of software faults, individual issues were reviewed. Each individual issue was analyzed and inspected to determine if it had

originated as a result of development in the release in which it was reported, or a prior release. For example, if an error was reported in TPR2, and originated as a result of development in TPR2; it would fall into the former category. An error reported in TPR2 that originated in TPR1 would fall into the latter category.

This review was done by examining each of the individual error reports from the sample on an individual basis. The software source code fix was identified in the software library, and the introduction of the software error was traced back to its initial introduction into the software base. This retrospective data analysis provided insight into the efficiency of the organization in identifying and correcting software errors. All software errors may not be fixed. For instance, some could be non-reproducible faults for which no software fix is created or delivered. By looking at how many faults were in this "no fault found" category, an understanding of the level of knowledge of the product by the organization, and its effectiveness at fixing the source of the errors emerges.

### 4.4 Interviews

Interviews were held with staff from the development organization to draw out insights from the data analysis. During interviews with an architect, first level manager, senior manager, and two designers, product predictive and

historical data was presented and comments were invited from the interviewees. These interviews were held after an initial review of the data analysis, and were limited to one hour in length.

In order to understand the organization's capacity for performance, all interviewers were asked questions from the Organizational DNA web-based query tool. This was the only common set of questions asked during all interviews. The questions asked are reproduced from the website (http://www.orgdna.com) are in Appendix E – OrgDNA Questions.

The initial interviews were with designers within the development organization. They were unstructured, and were presentations of the product predictive data. Questions to probe the comments were utilized to capture insights not apparent from the raw data and analysis. Their insight triggered the subsequent analysis of the products' post release error report arrivals.

For the remaining interviewees, the interviews were semi-structured. A subset of the predictive data presented to the designers was presented, with comments captured from their review.

Specifically, the data in Figure 1, the organization event timeline from Appendix A – Market Data, and the graphic presentations of Figure 2, Figure 14, and Figure 19.

The subsequent interviews were held with a first level people manager, a senior manager and a product architect within the development organization. Interviews were semi-structured; with the predictive software analysis data presented to them. A set of prepared questions (Appendix F – Interview Questions) was used as needed during the interviews to draw out their individual perspectives on insights or questions that arose during the interviews with the developers. These questions also clarified the researcher's understanding of the responsibilities of the person being interviewed. These interviews brought out additional insights over and above those from the developers, and also identified other areas that required further research. These interviews were limited to one hour in duration, and were all held within a period of 5 days.

As a result of these interviews, additional data were gathered and analyzed based on interview responses in the following areas: Architects within the development organization, Product Verification (PV) staff levels within the development organization, Number of deployments of new product capabilities within each release, and source and type of post-release errors.

## 5 Changes in each context

Examining the three dimensions outlined in the research model (Figure 1) individually lays the framework for identifying the temporal connections between the three contexts.

Raw data for all graphs are located in the Appendices.

### 5.1 Market changes

The size of the telecommunications market as measured by the NASDAQ index depicted in Figure 2 is quite striking. The ramp up and ramp down of the market contrasts quite graphically with the number of in-service systems under study.

This data was collected to understand the product footprint changes with respect to the market. It is evident from Figure 2 that the product deployments continued to grow even though the entire market did not. The growth in the number of deployed systems is a key tenet of this study; the product was a success in a falling market, and provides a basis for understanding what was done to achieve this when other companies and products were unable to weather the downturn.

The product itself was re-positioned for the TPR4 release. To capture and visualize how these new capabilities were being deployed to the market, Figure 3 was plotted. Figure 3 shows the deployment of each release to the field, and the percentage of customers who took each release that utilized the capabilities of the product as it was originally conceived; and the percentage of customers that deployed the product using it's new capability initially deployed in TPR4. TPR5 had an 80/20 mix of sites running the original versus the new platform capability.

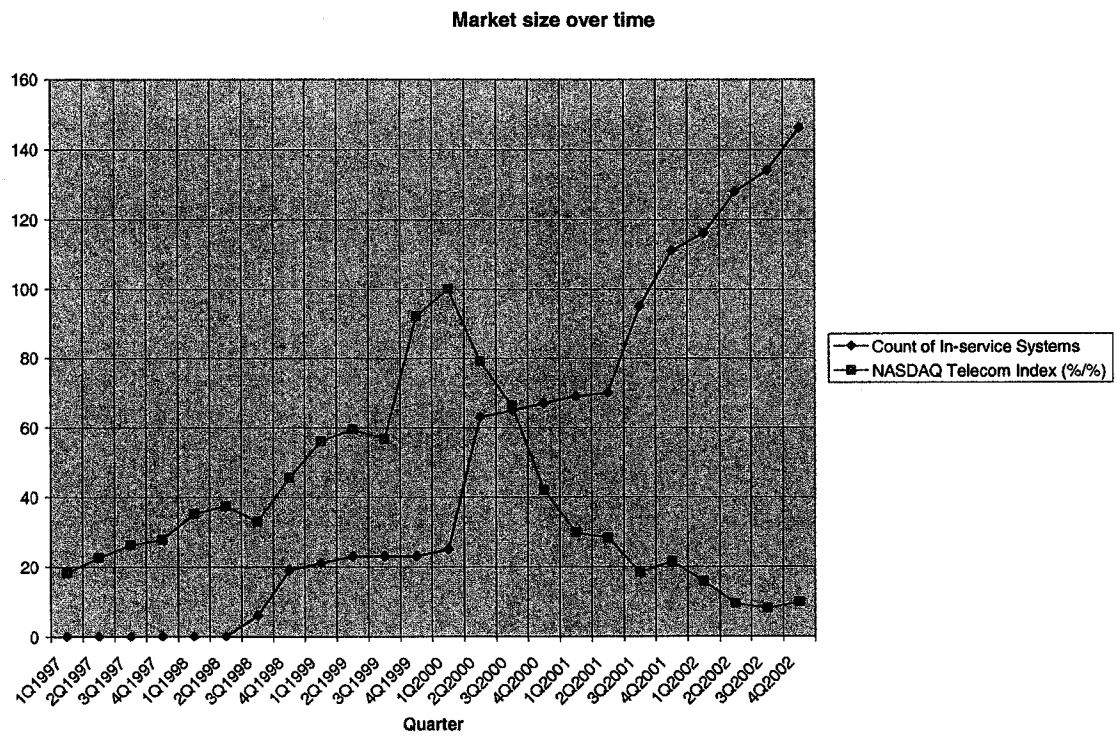


**Figure 2 Market size**

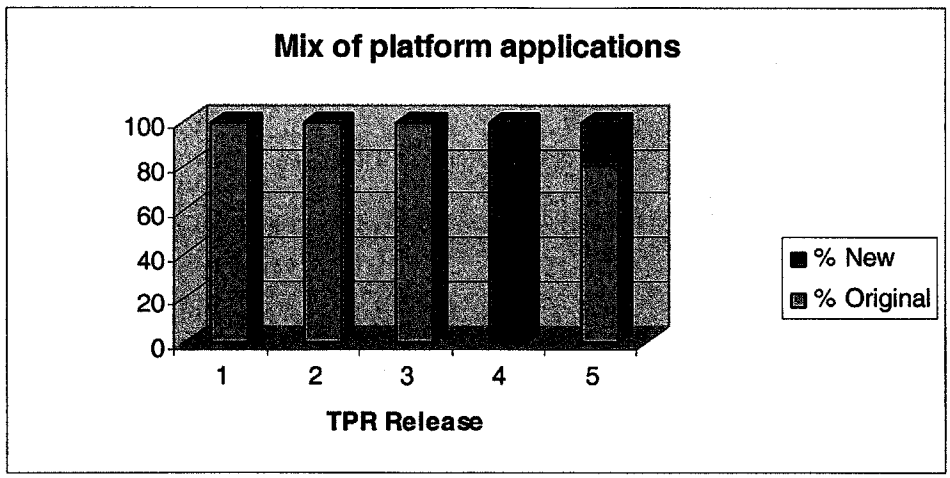


**Figure 3 Deployed product capabilities per release**

## 5.2 Organization changes

There are several measures of the organization's empirical size and makeup presented. One of the interviewees indicated that "we had creative architects that were able to figure out how to create features with a small amount of software change". The size of the organization overlaid with the number of product architects and the release milestones is presented in Figure 4.

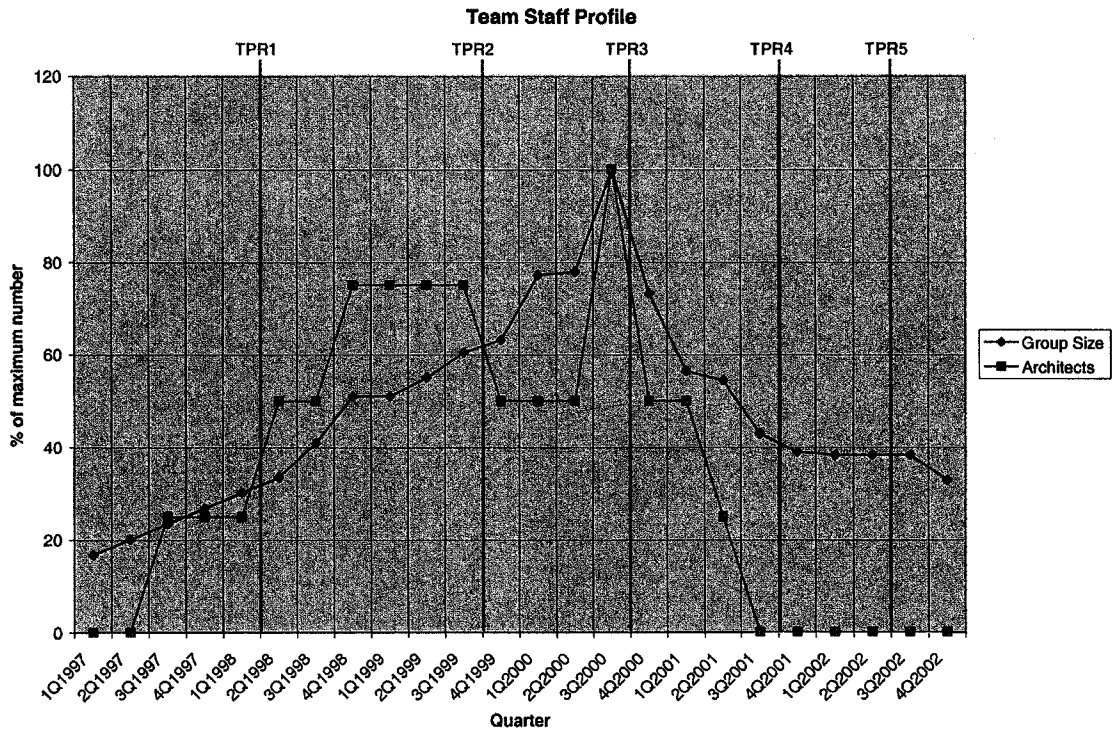


**Figure 4 Development organization staff profile**

The development organization underwent several changes over the course of the study period. There were 3 main structural phases to the reporting structure:

1. The original product owners from product conception through 4Q1999
2. A totally new to the product organization in a new location from 1Q2000 to the end of the study
3. Organizational re-alignment from the original functional alignment with the product architecture to a feature delivery model with virtual teams, from 3Q2001 to the end of the study

From the interviews, insights for these three structural phases were gathered. The initial phase, during which the original product creators were the development organization, the managers who were interviewed said that the development organization was very architecturally focused. They wanted to get it right. Their market knowledge and experience was not as great as the organization that became product owners in stage 2 (above).

When ownership of the product was transferred at the start of 2000, the receiving organization was responsible for creating the transition plan to infuse the technology into their development organization. The interviewees did not feel that this transition was particularly well executed and cited the following reasons:

- Some of the original owners were not very forthcoming with information on their areas; and were more focused on their new opportunity rather than providing a solid transfer of technology knowledge.
- The transition plan was accelerated such that it became more of a hand-off rather than a transition. The new organization members were forced to leverage their past experience on the legacy product in order to perform development.

Stage 3 as noted above changed the dynamic of the organization substantially. Not only was the architecturally aligned development organization model lost, but the number of development organization members was also reduced, and the formal structure of product architects guiding the high-level system view was lost. The architects became software developers within the development organization with feature deliverables – something that had previously not been done. As the architects blended into the development organization or left for other opportunities, designers within the organization stepped in to fulfill those roles in areas in which they had become knowledgeable from their feature development activities.

There were several other compelling organizational events that were uncovered through data analysis and through confirming interviews with key staff members:

1. A key executive who nurtured the product from 3Q1997 when he was the product general manager through to 3Q2002 when he left the company.
2. Three changes to the director of the development organization.
3. The initial product development director was promoted from the legacy team who eventually became product owners. His tenure was from 1Q1998 through 2Q1999.

4. The second product development director was from the initial product development team; and one of the peer designers that seeded the group initially, having had some market experience. His tenure was from 2Q1999 through 4Q1999.
5. The third product director was once again from the legacy team, and was director from 4Q1999 through the end of the study.
6. The move from an integral imbedded marketing team to a larger marketing organization within the company during 2Q2001.
7. The relocation of the development team in 3Q2001 from their own building onto a large corporate campus. The development teams were dispersed in pockets through the campus based on their dominant functional alignment. For example, the product verification (PV) team was located in adjacent offices.

### 5.3 Product changes

The product was an immature product in a mainstream market. This can be substantiated in three ways. First, the functionality addressed by the product was formally defined by standards in 1981 [ITU-T Q7x Series, 1981]. Product development activity began in 1996 – within the 15 to 20 year technology maturation timeframe outlined by Redwine and Riddle [1985].

Second, existing products in the market provided the same nodal function that the initial release(s) of the product. In fact, existing products were in the "Main Street" stage of their lifecycle [Moore, 1991]. Incremental feature development was being carried out on existing products to address market near term needs. The opportunity for the product was that existing products did not have the ability to scale further to meet future market growth needs, and were physically large and expensive to operate – requiring high power consumption, for example.

The third point of support for the product position relative to the market was uncovered through interviews. The new product opportunity was targeted to quickly reach and then surpass existing product capabilities. It took advantage of newer hardware technologies, and was smaller physically and had lower operating costs than existing products. It was designed from the start to be scaleable to meet foreseeable market growth demands, and included a user-friendly Graphical User Interface (GUI), making day-to-day operations easier and speeding operations staff training cycles.

The product was originally intended to perform a single function within the telephony network as a router, and achieve parity in function with greater capacity than the legacy product it was to replace by the fourth release. This original plan was modified to reposition the product into a new target market.

This repositioning took place in the fourth release, which was deployed to a small number of customers prior to merging the original product with the new in the fifth release.

The merged product did not have parity with the original product in the fifth release, either in function or in capacity.

Many detailed metrics were captured to measure and abstract the product, development activity, architectural change and quality. The predictive analysis from the software analysis is presented graphically initially, followed by the feedback or post-deployment data from the trouble report analysis.

As a software product, literature relating to development and measurement over time is a development necessity. A set of "laws" has been put forward by Lehman and Ramil [2001] relating to software development activity outlines these phenomena.

### 5.3.1 Predictive product measures

Predictive product measures are the result of static software analysis on the code base. These provide the basis for analysis of the product architecture and quality.

The first measure is that of software churn. This has been shown to be a statistically significant predictor of field failures for a software system [Nagappan et al, 2004].

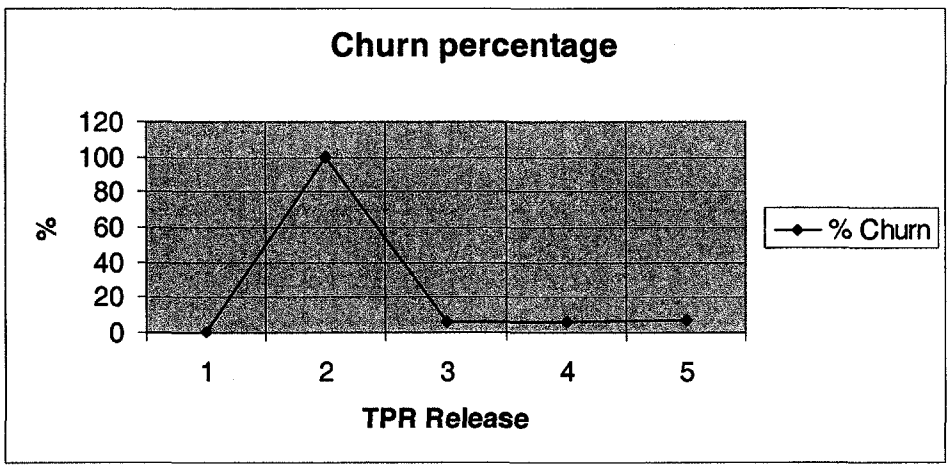


**Figure 5 Software churn as a percentage of the highest churned release**

Relative software churn was highest in the second (TPR2) release (Figure 5). This release was a significant one from a content perspective, and opened the product up to the possibility of global deployment. The TPR1 release was taken as the starting point for the churn studies, so a churn percentage of 0 is expected.

The software churn is a measure of software development activity, and includes new software plus existing software that has been modified. Figure 6 graphs the size of the software code base for each release. From the graph, it is evident that there is modest growth in the amount of new

source code between releases, as the growth in the number of lines of code between releases is relatively modest.

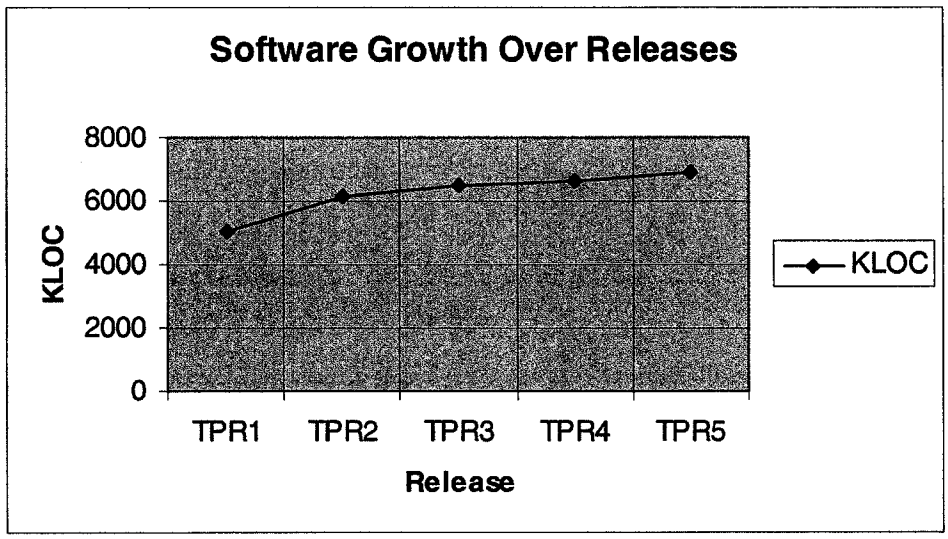


**Figure 6 Software code base size**

Also from Figure 6, the introduction of the TRP4 release does not show a significant increase in the amount of new software. This validates the insight from one of the interviewees, that new capabilities could be added without a large amount of new software being introduced.

The KlocWork tool provides several predictive metrics on the software base; defect density (Figure 7), and the aggregate system complexity as depicted in Figure 10.

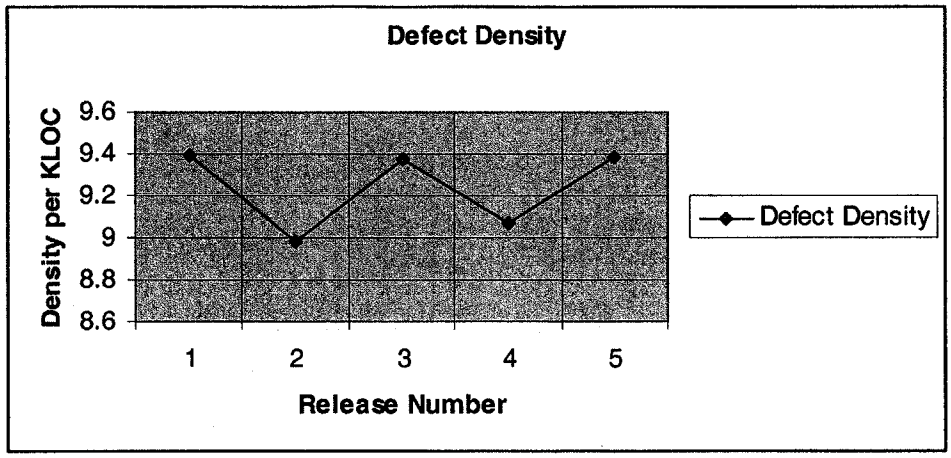


**Figure 7 Software quality attribute - defect density**

Defect density is an average of the number of software coding errors in the system per thousand lines (KLOC) of non-commented source code. A lower number indicates software with fewer errors. This data provides a measure of the quality of the development organizations software release over release. Again it is interesting to note that from a predictive static software analysis perspective there are no significant jumps in the defect density. In fact the releases with the greatest software change (TPR2 for its introduction of in-service upgrade, and TPR4 for the new product capability) have lower software defect densities.

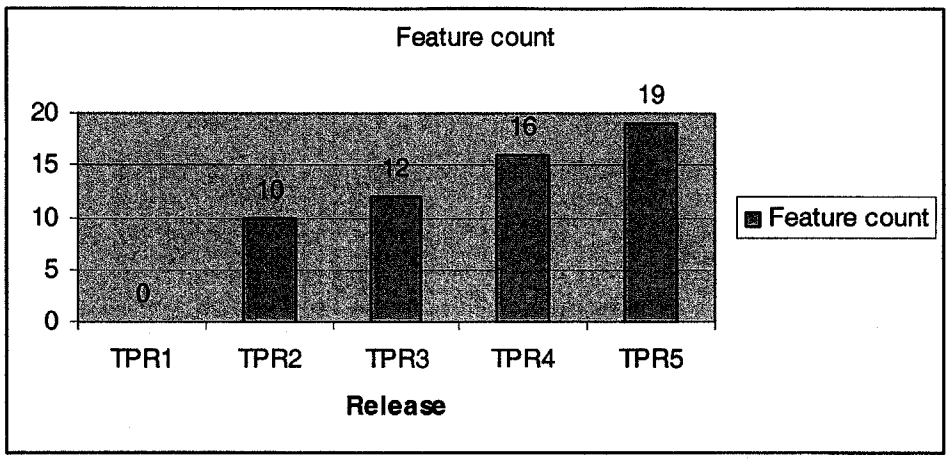


**Figure 8 Number of features per release**

From a product release perspective, the number of saleable features deployed is tracked in Figure 8. As the initial release, TPR1 content was not included; this view is of incremental additions of product capabilities release over release. For example, TPR5 had 19 new features available that were over and above the capabilities provided in TPR4. This data was collected to understand how much design activity was going on relative to the software churn.

The increase in the number of saleable features release over release signals significant event data in relation to the temporal mappings. The shrinking development organization created more saleable features being a key finding, even though the amount of software per feature gets smaller. This is not normal development, and is an indicator that the shrinking development

organization was becoming more efficient and creative in their development of new capabilities for the market.

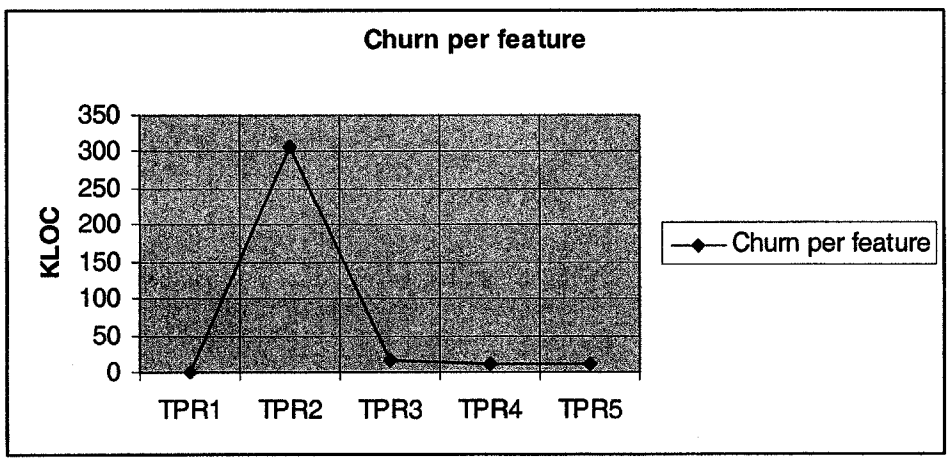


**Figure 9 Overall software churn per delivered feature**

From a software development activity point of view, a further analysis of the data is depicted in Figure 9. This data is obtained by dividing the total release churn by the number of features from Figure 8. Within the product context itself, the amount of software development activity on a per feature basis is quite high in TPR2. Subsequent releases show a declining amount of development activity, even though the number of features in those releases increases.

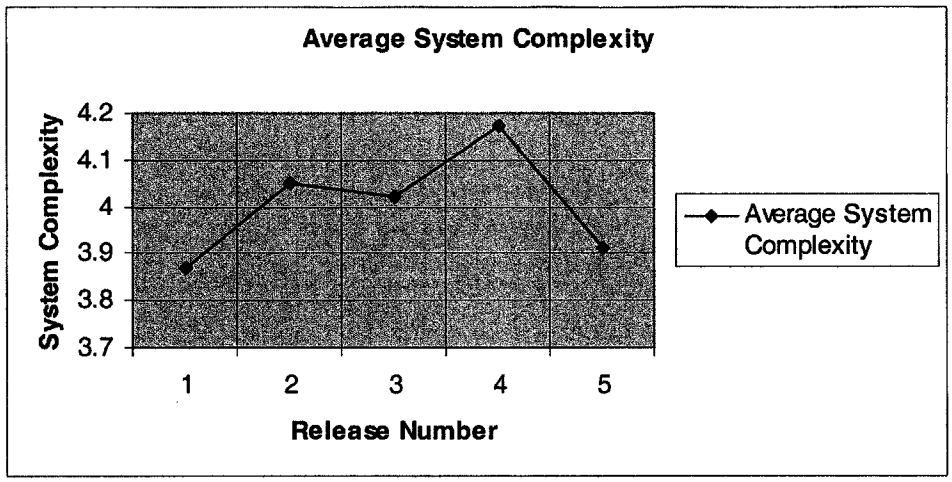


**Figure 10 Average system complexity**

The average system complexity metric is calculated from the predictive static analysis of the software. This is a metric that is widely considered to be a broad measure of the soundness of a software program, and a better number indicates to the development organization that they can have confidence in their program. In this case, the metric depicted in Figure 10 is the measure for the entire software base of the system. From a raw numbers point of view, a score of less than 10 denotes a simple program with little risk. The actual calculation is the result of the McCabe Cyclomatic Complexity.

This metric was captured to understand the aggregate risk of all of the design changes from a software architecture perspective. This includes an understanding of whether there was a test planning exposure, or if there was a release over release complexity build-up. The build-up of complexity may

have been a predictor of the need to re-engineer parts of the system to improve their maintainability, testability, and stability.

With the introduction of new product capabilities; it was initially expected that the product complexity would have increased significantly, such that the system would fall into the domain of a risky program (indicated by a score between 11 and 20).

The score stayed within the realm of a program without much risk, with only minor fluctuations at the key TPR2 and TPR4 releases.

From an architectural perspective, the stability or erosion of the architecture is captured by the clusters of cyclically dependent files metric and plotted release over release in Figure 11.

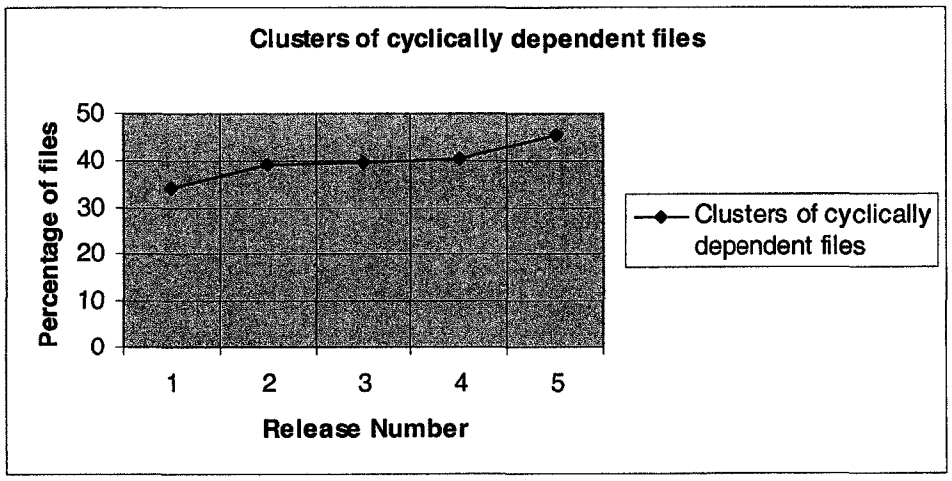


**Figure 11 Clusters of cyclically dependent files**

This is the key architectural metric captured from the software analysis; it indicates software architecture erosion points. The architecture eroded slightly between TPR1 and TRP2, and between TPR4 and TRP5. Between TRP2 and TRP4, the architecture as measured by this metric was relatively stable.

The TPR2 release introduced key system capabilities: targeting the system to a global market, expanding the number of physical interfaces (in number and type), and introducing a full in-service software upgrade that converted data between the previous release and the new release without manual intervention. The product at this time was focused on providing the standards defined network function described in Section 3.2, which was as a replacement for the legacy product within the network.

The TPR5 release was the first widely deployed release that incorporated a new and emerging product capability that had been introduced to a limited market in TPR4. This new capability was an extension of the product from its original market. From a product perspective, the original nodal function was extended to enable the new capability.

In addition to the new platform capability, the original nodal function continued to be offered through software partitioning on the same software and hardware platform. In effect both nodal functions were offered in a co-resident fashion, with software provisioning permitting inter-working between the product capabilities on the same hardware.

A view of the percentage of deployments using the original and new capability on each software release was depicted in Figure 3.

### 5.3.2 Product performance measures

Within the context of actual performance measures, data from the product error reporting database relating specifically to significant field reported defects was utilized. These are the Critical and Major classes of issues as defined in GR-929, the standard that defines quality and reliability for telecommunications networks [Telcordia, 2002].

In the product context itself, issue arrival on a per release basis across time was captured and is presented in Figure 12.

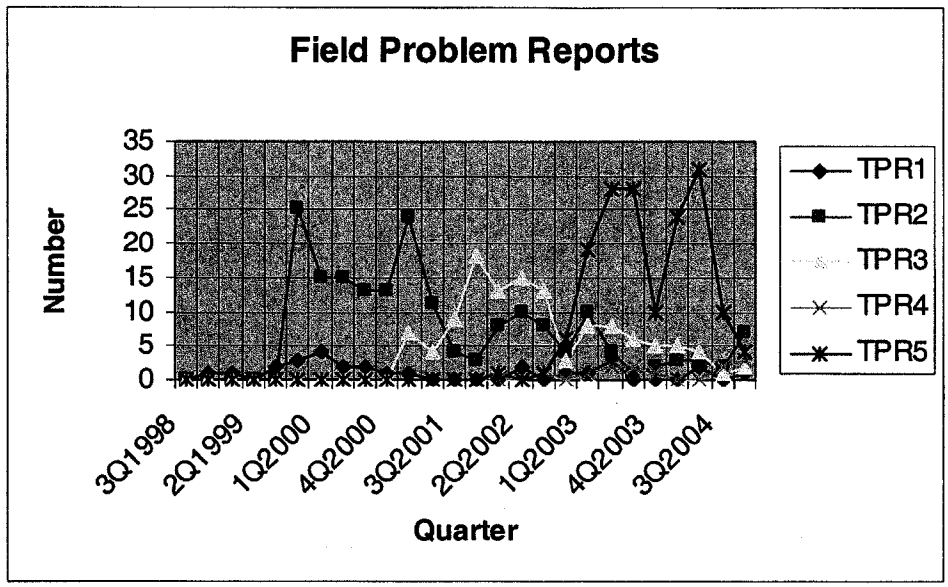


**Figure 12 Field reported problem reports**

Examining the three dimensions outlined in the picture of the research model individually in Sections 5.1 through 5.3 lays the framework for identifying the temporal connections between them.

In Sections 5.4 through 5.6, the temporal mapping data is presented. The first level of analysis captures the directly coupled temporal effects between individual contexts uncovered from the study. First level effects are those which were identified directly across the contexts. For example, if the overall market shrank along with the organization with minimal time lag between changes.

Where secondary effects were observed, they are presented separately within their relevant sections. For instance, if the market turned down and the organization did not, or there was a lag, there may be other factors that could explain this phenomenon. These factors provide additional insight into plans and decisions for the development organization and the product.

### 5.4 Contextual changes between market and organization

In this study, the organization's ability to address the market need is being measured by the number of deployed units, number of features delivered per release and post deployment product defects.

Direct links between the market and the development organization were few. The one key link relates market expansion and contraction to the effect on the development organization. This relationship is evident when plotting the development organization size with the NASDAQ Telecommunications Index (Figure 13).

The release dates for individual product releases have been overlaid on the plot to allow examination of secondary effects.

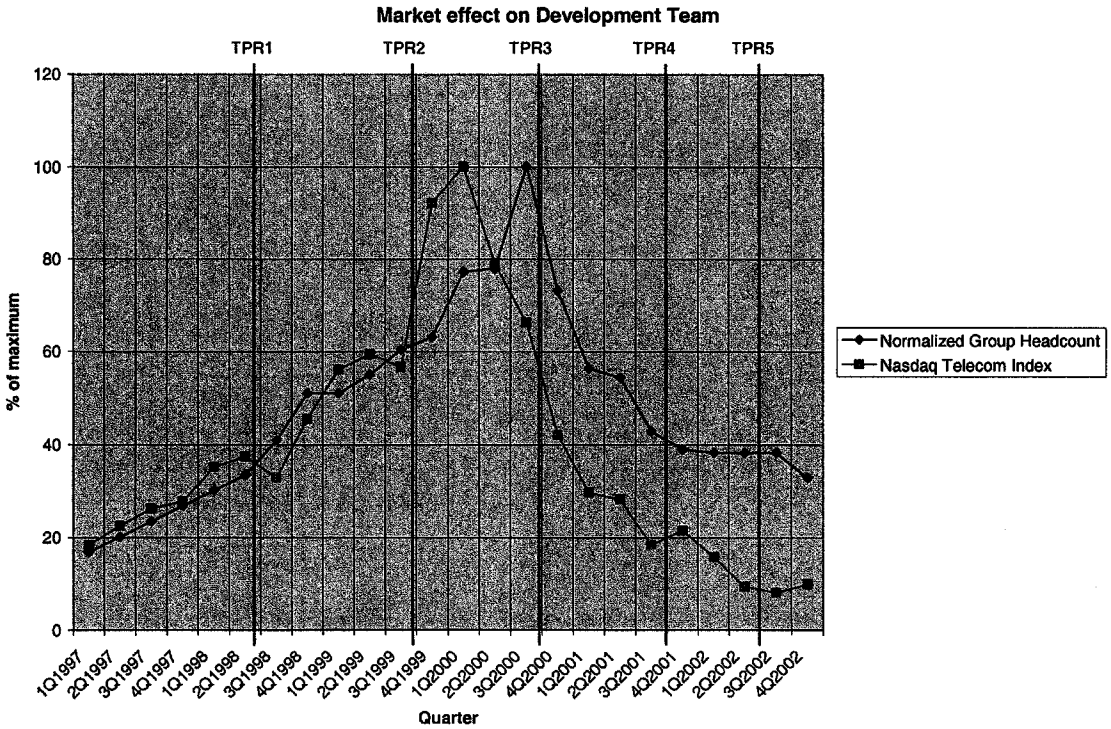


**Figure 13 Market effect on development organization**

Examining Figure 13 uncovers two key findings. The first is a direct effect, and that is the growth in the organization tracked fairly closely without lag along with the market until TPR2. It was post TPR2 deployment in 1Q2000 that the product was transferred to the new development organization. There is a lag of 2 quarters while the new owners ramped up their organizations size to meet the market. This was accomplished through a large career fair in 3Q2000 that invited candidates to be interviewed onsite and receive a job offer during the fair itself.

The second is that relative to the market, the development organization did not reduce in size comparable to the market after the market peak. Based on the very close tracking during the market rise, it would be expected that the development organization size during the denouement would track similarly in proportion.

Interviews with senior managers of the product development organization uncovered that there was a need to reposition the product into a new market space, and the necessary investment was made to enable this. The managers who were interviewed conceded that had the case for re-positioning the product not occurred, the development organization and/or its product would have been divested.

### 5.4.1 Market and organization secondary effects – Market size

The TPR3 release was the last release that offered the original platform capability to the market as a stand-alone application.

With the reduction in organizational size, the decision to rationalize to a common development stream was made. By rationalizing on a common development stream, both platform applications would share a common software release name, dates and enable co-residency of the applications.

Without rationalizing on the same stream, separate product releases would be needed to deploy the individual products to the field.

This repositioning was seen as the way to continue to serve the market on a common development cycle as a cost saving measure. The new capabilities were prototyped in their own release and kept separate from the main release stream, being introduced as TPR4. During this release, only the new capabilities were productized for the market.

### 5.4.2 Market and organization secondary effects – Organizational effort

The other secondary effect relates to the market need for product capabilities. Examining Figure 8 from Section 5.3.1 through the lens of the development organization size clearly shows that, although the organization reduced in size overall, the amount of features developed for the product and deployed for the market increased.

## 5.5 Organization and product

The development organization underwent several changes over the course of the study period. There were three main structural phases to the reporting structure, as discussed in Section 5.2. These phases are elaborated below with additional details relating to product linkages.

The first phase of the organization's structure was that of design peers – each designer was part of the single architectural team. The designers had full responsibility for the function, structure, scalability and performance of their software areas. Discussions relating to overall function of the system were team decisions. Integration between functions was carried out in an ad hoc manner between affected parties, with the overall deliver validation performed by product verification (PV). This phase was from product development conception through to 2Q1997.

When the first release had entered trials and deployment, the second phase of the organization's structure is apparent starting in 3Q1997. This period was characterized by functionally aligned teams, who had development responsibilities for their areas of software expertise. They owned and worked on the software related to those functions no matter the function of the overall product release. This period was characterized by the introduction of a strong product technical architecture team. The architects defined up-front the structure of the system and its capabilities prior to their introduction into the system. The architects during this time created the model from which the design team planned and executed their work. Integration was more formally done between designers working on complementary functionality, than when individual designers simply coded their functionality and then declared

completion of the design. A project manager was brought into to provide measurable development milestones and date tracking to manage the development project.

This structure continued while the product transitioned from one country to another in 1Q2000. The individual architects changed, as did their areas of expertise compared to the original architecture team. Since the product had delivered its second release to the market, the focus was more on new capabilities for the platform product, as well as extending the existing fan-out and capacity capabilities. The new architecture team continued to positively map out the new direction for the product as input to the functionally aligned development teams. The TPR2 and TPR3 releases were developed and deployed during this timeframe

Just prior to deployment of the TPR4 release, and after the peak of the market, there was a change in architect involvement. The individuals who were previously architects stayed with the organization, but became less involved in future directions of the product and more involved in day to day development activities. This change occurred in 3Q2001.

Graphically, the percentage of formal product architects is depicted in Figure 14.

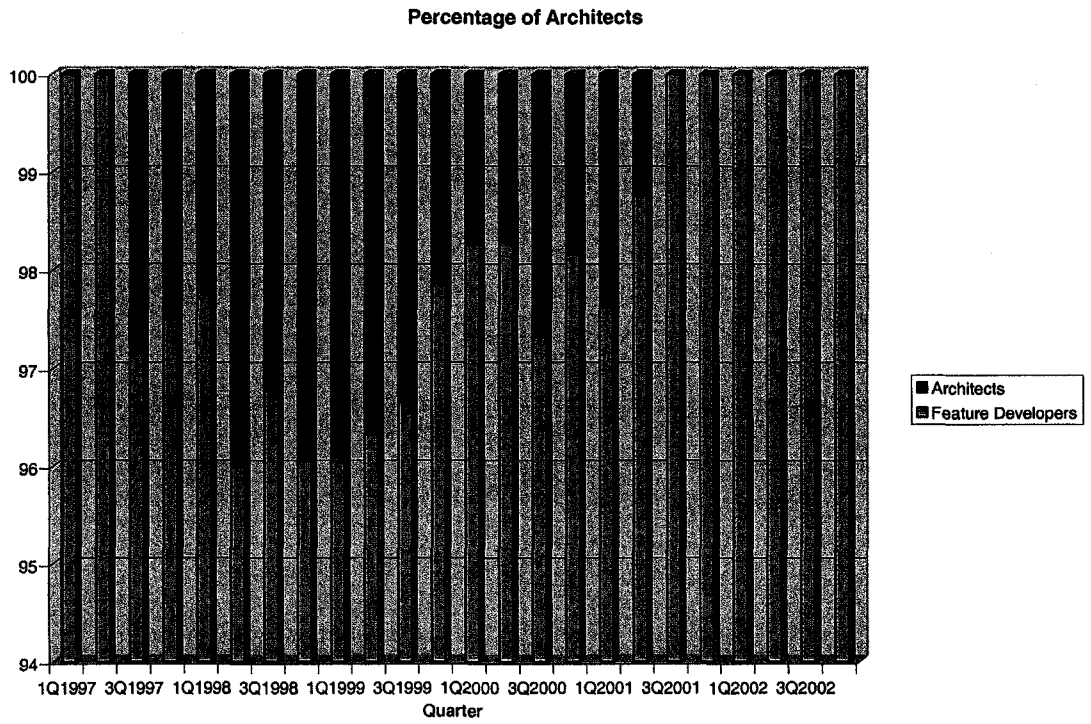


**Figure 14 Percentage of architects in the development organization**

Since the formal product architects were now developers, they took on normal software development activities as well as continuing to act as informal architects in the areas that they were previously the formal architects.

From the interviews with the software developers, they identified with individual developers who had become skilled within the areas they were developing through their day to day development activities. These developers developed a sense of ownership in these areas; and became subject matter experts. The informal networks within the development organization

recognized them as the experts, and they were called upon to work through challenging architectural and functional problems during the development cycle. These developers became the de-facto architects for these areas of the software that were not able to be covered by a formal architect, since there were no longer any formal architects.

## 5.6 Product and market

As examined in Section 5.1, the market size and makeup changed substantially over the study period. Coming directly from Figure 2 and Figure 3, the re-positioning of the product enabled the product to continue an upward trend of deployments.

### 5.6.1 Product and market secondary effects – field reported error reports

The development cycle brings requirements from the market to the development organization which then implements functionality on the product to meet the market's testable requirements. The product verification cycle occurs between the development activity and deployment of the newly developed functionality.

Feedback on how well the development organization understood the requirements, how well they implemented the requirements, from the market

perspective is via post-release error reporting. Between the development cycle and the deployment stands the product verification (PV) cycle.

The PV team has an important role to play, as they are the group that verifies whether or not the testable requirements originally stipulated by the market are met, as well as whether validating an error-free implementation by the software developers. In essence, PV represents the last quality checkpoint prior to product release. If there are missed requirements or errors that escape PV, they will be seen by customers once the product is deployed. To quote Brooks [1995], "The product-testing group then is the surrogate customer, specialized for finding flaws".

The incidence of field reported error arrivals on a per release basis is depicted as an aggregate in Figure 12. By further refining the analysis, additional effects become apparent.

Plotting out the reported errors over time for TPR1 through TPR5 across the study period and nine quarters beyond the forward analysis in this study reveals the total field issue arrival profile in Figure 15.

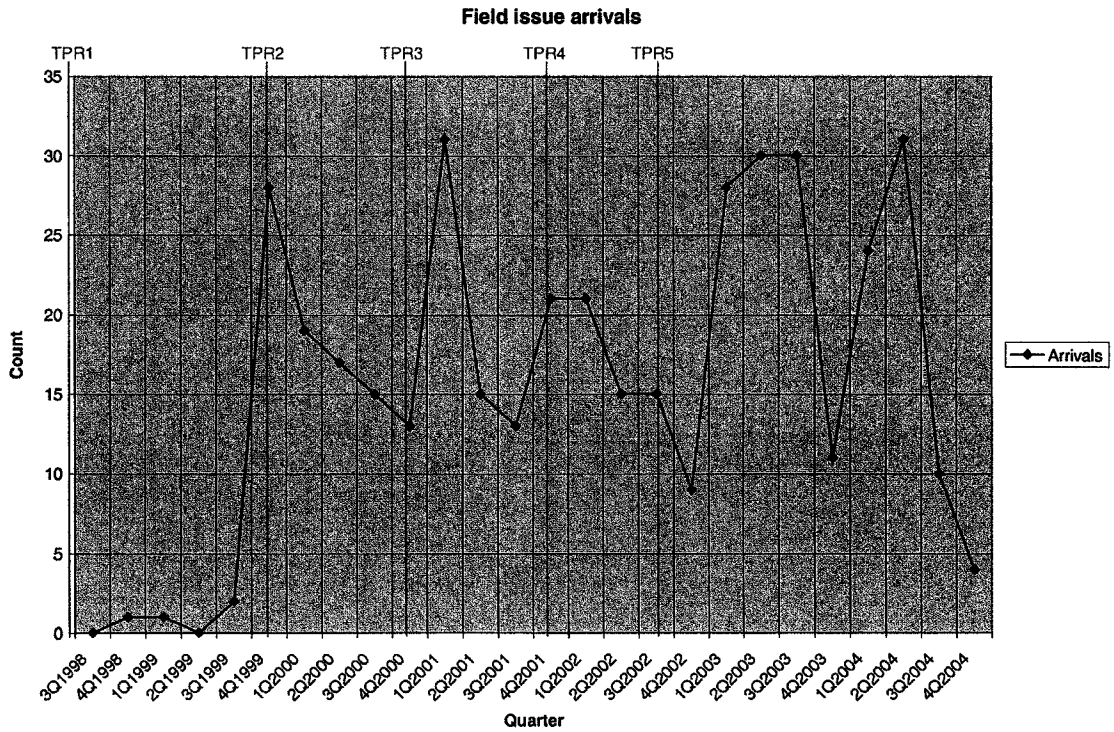


**Figure 15 Field issue arrival profile**

Figure 15 in general identifies an increase in arrivals coincident with initial deployment of individual releases to the market. The release strategy for this product includes a select set of friendly customers who take the product into their network and perform focused testing on new features and general system testing. After this period, which was typically three months in duration, the load is made generally available to the entire market. For the purposes of this study, all major or critical issues after initial deployment are included in the field issue arrivals.

From the data of all reported problems, the average numbers of issues reported per release on per-site bases were computed. These data were graphically compared to the churn data originally presented in Figure 5, and added to Figure 16. This graphic contradicts the findings in Nagappan et al. [2004], which is that the amount of software churn correlates to the number of software defects that will be introduced into a software release. All things being equal, since the software churn for this product was stable during the market downturn, the number of issues per site should have been relatively constant in the absence of other factors.

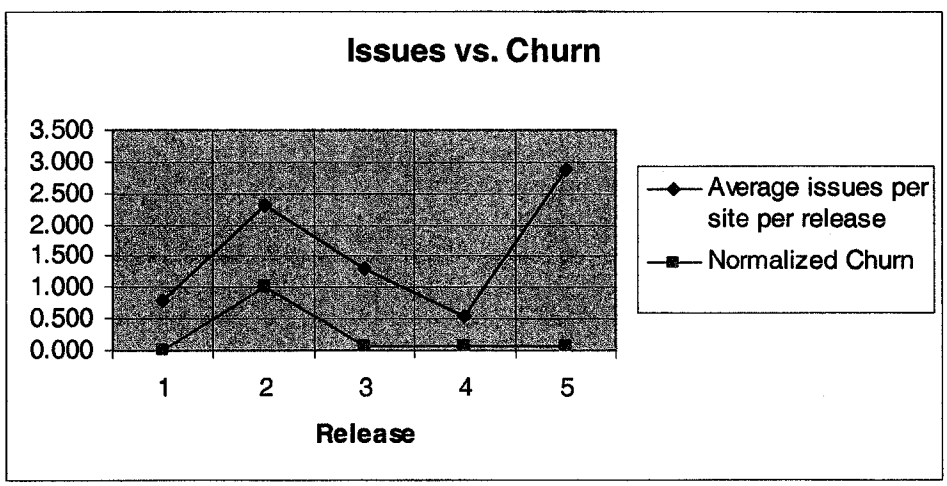


**Figure 16 Average problem reports per site versus software churn**

Several individual factors are put forward in the next sections as possible explanations for the software churn predictive quality measure not aligning with the market deployment of this product.

#### 5.6.1.1 A larger number of users find a larger number of errors

The idea that the number of errors found is related to the number of users is outlined in Brooks [1995]. In the case of this study, this concept is does not track for the releases in the study.

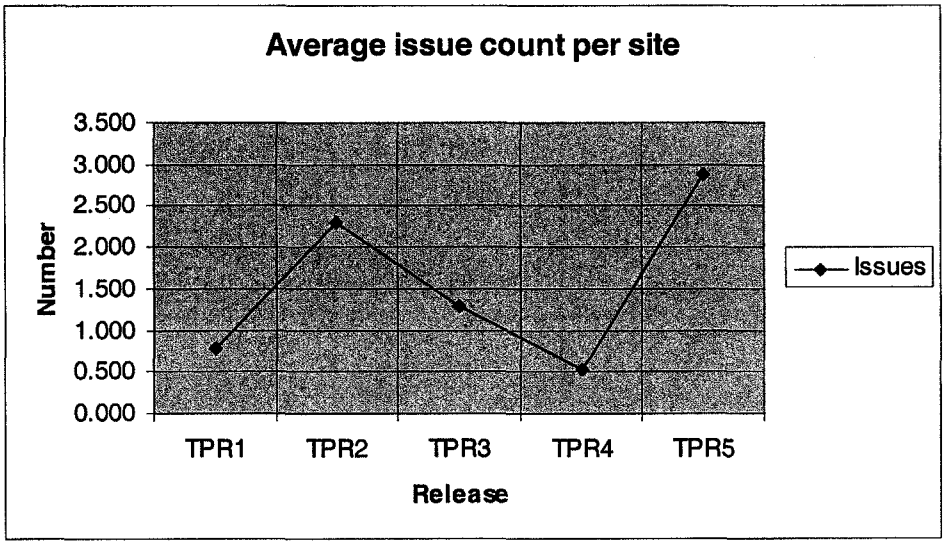


**Figure 17 Issues per in-service site for each software release**

Examining Figure 17, and by dividing the number of issues on a release by the number of deployed sites, the number of issues on a per site basis varies considerably. Normalizing the data in Figure 17 and overlaying a normalization of the number of sites per release (TPR3 had the largest deployment of the studied releases), the plot of Figure 18 results.

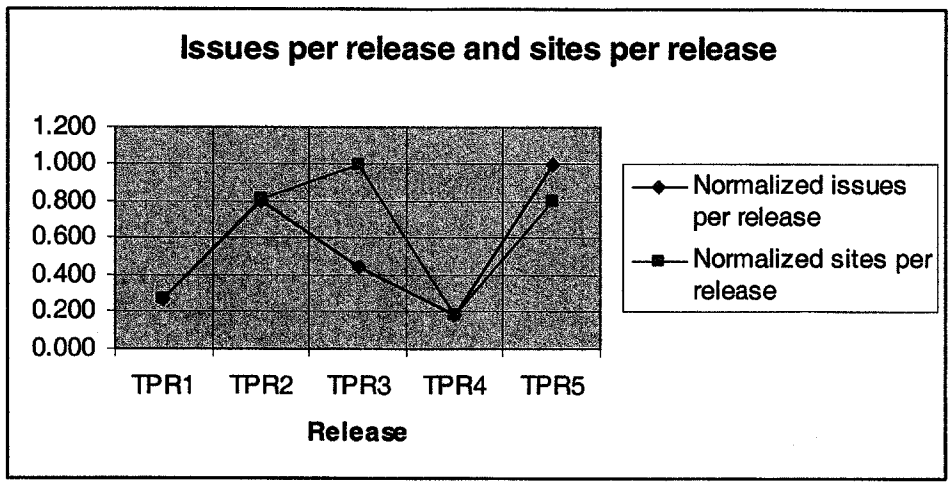


**Figure 18 Issues per release and sites per release**

#### 5.6.1.2 Organizational efficiency increases

The entire list of field reported problems were sorted by product release. Then, within each release, where possible a statistically significant random sample of the problem reports was catalogued. For TPR4, there were insufficient numbers of field reports to perform statistical analysis; so all of the problem reports were included in the analysis. From this data, it was possible to examine individual issues with the benefit of hindsight to ascertain a picture of organizational efficiency in correcting post-release field error reports.

Efficiency measures are captured two ways: first, by looking at the percentage of issues reported from the field that upon ex-post inspection were reported back to the field as "No Fault Found" and, second, by looking at the percentage of software errors for which a software solution was

delivered. These data are presented in Table 4. Note that the totals do not equal 100%, as the data relating to hardware failure or questions regarding operation of the system were not included in this analysis.

| Measure | TPR2 | TPR3 | TPR4 | TPR5 |
|---|---|---|---|---|
| % No Fault Found | 46% | 42% | 66% | 39% |
| % Software Fixes Delivered | 29% | 19% | 0% | 47% |

**Table 4 Organizational Error Correction Efficiency**

A few observations can be made from the data in Table 4. First, the aggregate efficiency of the organization (sum of no fault found and software fixes delivered) when perturbed by change is less than when it is not. The aggregate efficiency refers to problems that are within the scope of the organization to remedy, so include software problems only, some fixed, and others that no fault could be identified for correction. The TPR3 and TPR4 releases were deployed after significant organizational changes and product positioning changes respectively.

Second, efficiency overall is best measured by reviewing the "% Software Fixes Delivered" data. The TPR3 release was the first delivered by the new development team. The release itself was developed by a full team. There was a reduction to the size of the organization, however, coincident with its

early deployment. A deeper understanding of what the optimum mix between organization and support is an area for further study: Additional data points would be required to empirically determine this.

Third, the TPR4 release represented the point of product re-positioning. The TPR4 post-release error resolution efficiency is quite low. This could be related to the architectural re-alignment or the organizational downsizing size pressure. It is evident that all field reported issues were replied to with no fault found.

Fourth, TPR5 shows a remarkable spike in the post-release error resolution efficiency. This indicates either a singular anomaly, where the errors were contained in areas that the organization had high knowledge of and were able to isolate the root cause quickly. It could also indicate that the development organization itself had absorbed the impact of becoming product owners and the organizational size reduction and were able to better cope with errors in their software. In this case the organization has found a way to effectively resolve post release error reports.

#### 5.6.1.3 Organizational resiliency

In order to ascertain the adaptability and resiliency of this development organization, the questions from the OrgDNA Profiler as detailed in Section

4.4 were posed to the interviewees. In all cases, the organization was typed as "Resilient".

From the tools' descriptor, this categorized the development organization as:

> "Adaptive to market changes" yet "steadfast" in business strategy. This forward-looking organization anticipates changes routinely and addresses them proactively.

Indeed, this organization succeeded in the face of the market downturn with increasing product deployments.

#### 5.6.1.4 A smaller organization generates more errors that they cannot find before deployment

Logically speaking, a development organization that is smaller would normally be expected to develop fewer features than a larger development organization. However, in the case of this development organization, the number of saleable features per release increased (Figure 8), while the size of the team decreased (Figure 4). This is in some conflict with the post-release error resolution efficiency increase noted in Section 5.6.1.2.

It appears that there are other factors that influence the organization outputs into the product. In this case, it is possible that the reducing organization either generates more errors or cannot find those that they generate during the development cycle or both.

To explore this, the size of the product verification component of the development organization as a percentage of the overall development organization was examined. Figure 19 plots the Product Verification (PV) team as a percentage of the entire development organization over the study period.

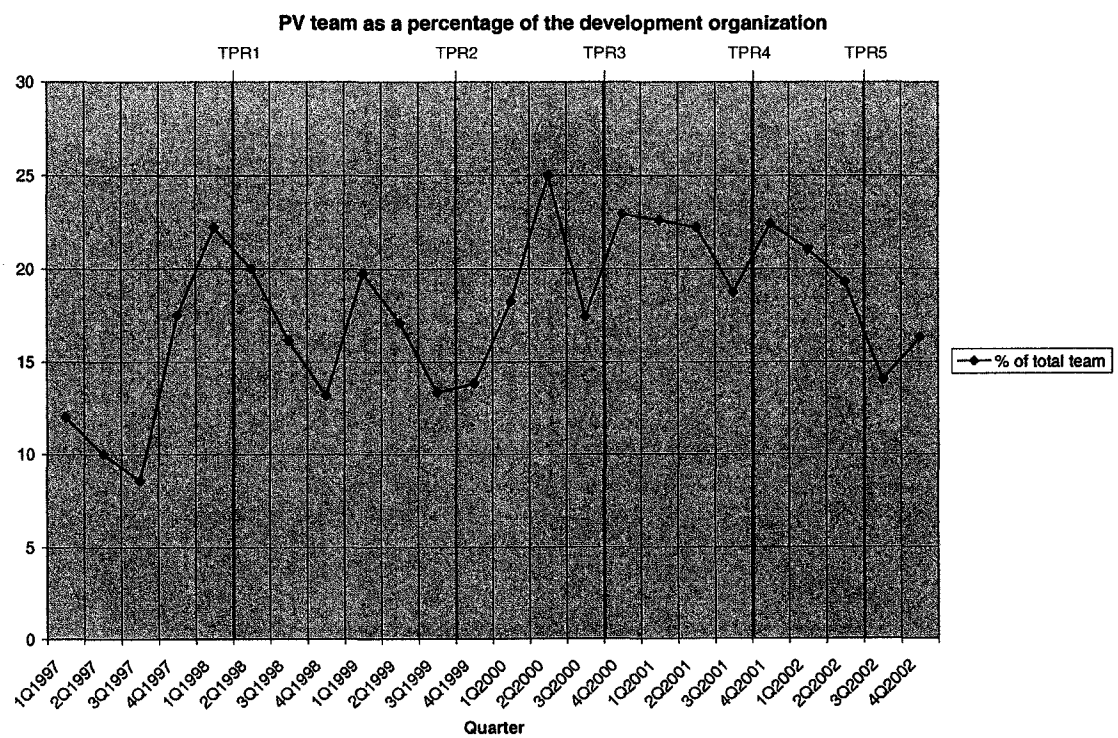


**Figure 19 PV team size as a percentage of the overall development organization**

From Figure 19, it is evident that the product PV team size fluctuated across the study period. When looking at this plot in relation to Figure 16, TPR2 and TPR5 had the highest number of post release problem reports from the field. These releases were also the ones that had the lowest number of PV members leading up to the release dates of their respective releases.

## 5.7 Interviews

Some of the data and analysis in Sections 5.1 through 5.6 was undertaken as a result of interviews with key development organization staff. Two software developers, a product architect, a first level manager, and a senior manager all of whom were active development organization members through the study period were interviewed.

The following tabulates the key insights uncovered by the interviews that resulted in additional data gathering and analysis post-interview:

- The need to look at post release defects as well as the predictive analysis
    - This led to a literature review of literature on retrospective analysis and how churn which is a predictive analysis tool could predict future error reports.
- Architectural creativity in delivering features with little churn.

- This led to analyzing the amount of code churn per feature and the number of features delivered per release as measures of organizational efficiency.

- Need to split growth against original application and newly positioned one.
  - Triggered a review of the number of deployed sites per product capability, which uncovered the growth components of the product towards the total deployment.
- Product test was not as solid as it could have been.
  - This comment led to a further review of the number of PV team members as a percentage of the entire development organization.

## 6 Summary analysis

In this section, the four principle findings of this study are presented, as well as two secondary findings. Table 5 summarizes these findings, with further details for each of the findings expanded in Sections 6.1 through 6.4.

| Finding | Relevance |
|---|---|
| Product architecture is reslient and does not degrade quickly in small, well integrated teams. | A product with a well thought-out architecture is extensible for a time in the face of staff downward pressure. Product architecture is not a manpower issue. |
| As the overall market shrinks, the power within the organization moves to those who are closer to customers | The people who hold the budget authority shifts from individuals close to staff to a smaller number of people who are close to customers. |
| As the organization shrinks, and the amount of work stays the same or increases, the ability of the development team to find errors decreases. | In the absence of organizational stress, error rates correlate to software churn, however during times of stress, error rates increase. |
| As the organization shrinks, the reduction of the product verification team within the development organization below a certain point leads to a significant increase in post deployment trouble reports. | There is a tipping point that exists organizationally for the product verification team within a software development organization. If this point is crossed, the number of defects that will escape the product development organization will increase. This provides insight into the minimum staff level to deliver a product free of errors. |
| During a market downturn, having a resilient development organization whose members can become intra-personally diverse is important. | A development organization with a broad range of experience with a "can-do" attitude during a market downturn can lead to product success during a market downturn and enable product stability. |

**Table 5 Thesis finding summary**

The product under study went through 2 phases of product architecture and 3 organizational phases.

The first product phase was developed by the original product design organization from the start of the study until the product was moved to a new development team in 4Q1999. The original product was targeted as a routing node within the network to modernize existing products in the market that provided this capability.

The second product phase began in 2000, just as the telecom market was peaking (Figure 13) when the new development organization were becoming product owners. This phase introduced a new network function onto the existing platform product - that of a gateway. The original router and the new gateway were complimentary functions that were controlled by software to be co-resident on the same physical hardware.

The two phases of the product capabilities are discussed in Section 5.3, with a graphical representation in Figure 3, where the product capabilities per release are tabulated. The TPR1 through TPR3 release had router functionality only; whereas TPR4 and TPR5 had gateway functionality.

The first organizational phase was staffed by the original development organization. The development organization was aligned functionally within broad areas that mirrored the platform's software structure (platform, applications, GUI). This phase continued through to the initial deployment of the TPR2 release.

The second organizational phase started with the product transition to an established development team at the end 4Q1999. This team had many years of experience with the legacy product that was to be displaced by this new product offering.

In 3Q2001, the third organizational phase began. The development organization from phase two re-aligned to a small number of large departments that were not aligned based on the product's software architecture.

Organizational changes are discussed in Section 5.2.

## 6.1 Connections between organization and product

The original development team was the product owner for phase 1 of the product development organization. They functioned as a team of peers for the TPR1 release. There was no strong project management for this release, and

very light formal process. The work was broken down in size so that one designer could deliver that functionality to the release. The individual designers had accountability for the design from requirements through to coding and unit test. The interviews held with the development manager and senior manager highlighted that there were only a small amount of peer reviews for their designs in the TPR1 release. The interviews also brought out that the individual designers were very strong minded about the product architecture, particularly in delineating the boundaries between functional areas within the product - because individuals created and understood their software to their interfaces with other software elements they interacted with.

When the product was slow to achieve readiness to deliver to the market, a project office was put in place, and the more senior developers were formed into a formal architecture team. The product began to achieve field ready stability and capacity after the architecture team took the lead to resolve the integration problems that were created by the team of peers' development approach.

This first organizational phase continued through to the initial deployment of the TPR2 release, with the original product development organization creating the features for the release from requirements through to field delivery. As a lesson learned from the first release, a more formal

development process was put in place for TPR2. This enabled the development cycle to proceed with greater schedule compliance, and more process required design artifacts were generated.

The second organizational phase marked the transition of the product from the original owners to the legacy development team. Structurally, the organization was set up to mirror the organization that had owned the product in phase 1, functional departments aligned with the product architecture (platform, applications, GUI). This phase continued through to 3Q2001.

The third phase of the organization marked a change in development team structure. The departments were managed by a single people manager. Within the departments there were technical project managers given accountability for delivery of features to the product releases. Individual designers were assigned to the product releases for feature development from requirements through to deployment to the field.

During this third phase, the development organization was under significant duress, as there had been significant layoffs leading up to this re-organization. The formal architecture team was dissolved, and the architects who remained became designers or technical project managers.

It was during this third phase that the gateway functionality was added to the product.

From the interviews with designers, managers and the architect; these insights emerged:

- The decision on which members of the design team would remain with the organization during the downturn was motivated by their system knowledge, work volume and quality, ability to learn, previous market knowledge and adaptability.
- The architects were creative in coming up with ways to augment the existing architecture to create the new gateway functionality without compromising the system's architecture and with minimal software churn (Section 5.3).
- Individuals who had been architects and had become developers or technical project managers as a result of the organizational change continued to fulfill the architecture role on top of their new role (Section 5.5).
- Some individuals, who were designers in phase 2, became architects for their areas of expertise in phase 3. These informal architects were called upon by the general design community for their expertise more and more as time passed after the re-organization.

Tacitly, the organization through its own actions became intra-personally diverse and well integrated with a strong informal communications network.

From the product predictive measures, the architecture did not significantly erode. The system complexity (Figure 10) and the Clusters of cyclically dependent files (Figure 11) indicate this. The organization did deliver an increasing number of features (Figure 8), with low churn per feature (Figure 9), low predicted defects (Figure 7).

**Finding #1**

These insights bring out the first finding: Product architecture is resilient and does not erode quickly in small, well integrated teams

## 6.2 Connections between organization and market

The telecom market through the study period went through a boom and bust cycle that had been without precedent. As a development organization within the market, there were impacts to the size of the organization that mirrored the market as outlined in Section 5.4 and Figure 13.

However, this development organization continued to see increased deployments of their product throughout the market turbulence, as documented in Figure 2. The market for this product increased, but the size of the development organization mirrored the overall telecom market, rather than its particular market segment.

From the interviews with the managers and product architect, it came to light that there were many factors at play that drove these decisions. The need to reduce the development organization headcount was mandated corporately, so although the market for this product continued to grow and generate revenue, the staff level was controlled through corporate allocation.

This reality led to the shift of the product to new market opportunities that were revenue driven as recognized corporately and enabled increased funding to flow to the organization to develop the gateway functionality. The managers and product architect were quick to point out that a product line manager and the development organization director worked closely together in order to create a market driven vision for the future that would enable the organization to flourish.

This increased funding is visible in Figure 13 after the peak headcount in the development organization. Prior to the market peak, the size of the

development organization tracks the market. After the market peaked, the reduction to this development organization was less than would be expected based on the overall market size.

The product line manager and development director enabled the implementation of the feature development plans that resulted in increased features being deployed release over release (Figure 8).

From a post-release perspective, the need to provide superior service to the existing customer base became imperative. The interviews held with the software developers as part of the study quickly isolated the power of the product field support team within the organization. Critical and major issues that were reported by customers took precedence over design activities during the third organizational phase. In other words, when a challenging problem arrived from an important customer, the support team had the power to pull people from across the organization to isolate and remedy the cause of the software fault. This is in stark contrast to the first two organizational phases of the study.

During the first two organizational phases of the study, field problems were the sole responsibility of the field support team to resolve. Interrupting the

development cycle was discouraged, and only permitted when key strategic customers escalated their issues.

The organizational efficiency tabulated in Table 4 from Section 5.6.1.2 shows one of the effects of this change: The overall organization became more efficient at providing software solutions to customers when the designers became involved in correcting defects.

**Finding #2**

This development leads to the second finding: As the market shrinks, the power within the organization moves to those who are closer to customers

### 6.3 Connections between product and market

When examining the predictive and retrospective product measures, there is an important insight that emerges. The key facts that bring this out are:

- The organization itself was reduced in numbers until there were about 1/3 of the peak numbers of staff working in the product development organization.
- The amount of development activity increased gradually starting in TPR3 through TPR5 as measured through software churn (Figure 5

and Figure 9) and through the increasing number of saleable features per release that were developed (Figure 8).

- The number of errors that were reported from the field increased (Figure 12, Figure 15 and Figure 18).
- The introduction of the gateway functionality in TPR4 enabled 20% of the total deployments of TPR5 utilizing this product capability.

The number of software errors correlates with software churn in the TPR2 release, when there was a large development team. However, in TPR5 when there was a smaller development team that created features with lower total software churn and lower software churn per feature than in TPR2, there were more errors reported by customers.

Logically speaking, a greater volume of users should generate a corresponding volume of errors, as discussed in Section 5.6.1.1. In the case of this study, the effect of the number of users was damped out by generating the number of issues per deployed site. The resulting graphs (Figure 17 and Figure 18 shows that there were varying numbers of issues per site one each release. In particular the number of reported issues only correlates with the software churn (Figure 16) for the first three software releases (TPR1 through TPR3). In particular, the issues reported for the TPR5 release was very high in relation to any release previously released to the market.

**Finding #3**

As an organization shrinks and the amount of work stays the same or increases, the ability of the development organization to find errors decreases; the third finding of this study.

In trying to understand how lower measured development activity as measured by churn could result in higher error arrivals after the release had been deployed to the market, a comment from the senior manager during his interview provided a lead. This lead was followed, with the data captured in Figure 19.

The product verification (PV) team represents the last development function prior to product deployment. They are responsible for validation of new features against the original testable requirements, as well as overall verification of the system. Verification includes regressive and stability testing to confirm successful integration of the new features into the product release.

PV is the last quality control point in the development cycle. If errors escape the development organization through its design processes, then these errors will be released to the market, and will be reported as post-release errors.

The size of the PV team as a constituent of the entire development organization was captured and plotted in Figure 19 from Section 5.6.1.4.

By examining the size of the PV team leading up to each release, the PV team size at TPR2 and TPR5 is lower than the other releases. The TPR2 and TPR5 releases are also the releases with the highest number of field reported problem reports from the retrospective analysis. In the case of this product, it seems that there is a point below which there is insufficient numbers of PV team members to perform the validation and verification of the product. In this case, that number is between 15 and 20%.

There is literature that says that there is "no optimal team size" [Boodoo et al, 2000] in relation to software inspection. The validation and verification of software has standards entrenched requirements, just like software design [IEEE 1012-1986]. Direct links in the literature relating to the size of the verification team could not be found.

In the absence of direct available literature on PV team size, and logically applying the team size result from software inspection [Boodoo et al, 2000], the fourth finding of this study emerges.

**Finding #4**

The fourth finding emerges from this data: As the organization shrinks, the reduction of the product verification team within the development organization below a certain point leads to a significant increase in post deployment trouble reports.

## 6.4 Team resilience, process and performance

By querying the interviewees with the questionnaire and entering their responses in the OrgDNA web tool, a measured description of the organization was obtained. Questions from the web tool are captured in Appendix E – OrgDNA Questions. Although a small sample of five team members were interviewed, the results were unanimous: This team was a "Resilient Organization" as measured by the tool.

As a resilient organization, they were able to anticipate changes routinely and address them proactively; responding immediately, thoroughly and constructively.

An example of this organizations resilience was uncovered through interviews with the manager and architect. They indicated that the team members who were to remain with the development organization in the face of the

mandated headcount reductions were chosen deliberately. They were the people who were motivated, experienced, adaptable team players, who understood the products' customers from multiple perspectives. Team members that could communicate effectively with customers to understand their needs and find creative solutions to address those needs were selected to be part of the organization going into the downturn remained in the development organization.

This led to a search for whether this decision process could have been a contributing factor to the success of the organization and the product in the face of the overall market downturn. The study on the importance of intrapersonal functional diversity provided a possible insight into this organizations success.

The study by [Bunderson and Sutcliffe, 2002] as introduced in Section 3.3.2 outlines the positive correlation between the effectiveness of the development organization when team members have broad functional experience in a variety of different areas.

**Finding #5**

During a market downturn, having a resilient development organization whose members can become intra-personally diverse is important.

# 7 Conclusions, limitations and opportunities for further research

## 7.1 Conclusions

The objective of this study was to:

1. Extend the existing literature on the links between a product and the organization that developed it.
2. To identify findings from this research that may be tested in other product ecosystems.
3. To identify management recommendations on strategies for maintaining architectural stability during a market downturn in order to position a product for continued growth.

### 7.1.1 Product, market and development organization linkage

The shift of the organization from functionally aligned to project aligned in phase three of the organization, marked a reversal of Conway's law. In this case, the organization's informal communications network was structured by the product. The development organization closed ranks around the product architecture in order to continue to develop features for the market.

Being able to furnish in-demand features to the market drove increased deployment during the downturn. This generated revenue during a time when

survival was paramount. A resilient organization that had intrapersonal diversity was able to work closely with the market to succeed.

### 7.1.2 Key findings for test

The following findings from this research are candidates for further study and statistical analysis:

Product architecture is resilient, and does not erode quickly in small, well integrated teams.

As the market shrinks, the power within the organization moves to those who are closer to customers.

As an organization shrinks and the amount of work stays the same or increases, the ability of the development organization to find errors decreases

Reduction of the product verification (PV) team within a development organization below a certain point, leads to a significant increase in post-deployment error reports.

### 7.1.3 Management recommendations

The following recommendations emerge that may be of assistance to managers on similar telecom market infrastructure products during a market downturn.

When reducing development organization headcount, creating a resilient team with intrapersonal functional diversity may improve your product's market position.

Measure of product market size, software growth, software churn, software quality and the product architecture should be employed to guide decisions relating to the organization size and structure. The development organization studied here did not generate the metrics captured in this Thesis during the course of the development of the TPR releases. Had they done so, the decisions that were made may have created an environment for even greater success.

Based on the data that emerged from this study, providing visibility to the organization to post release problem reports may provide a decision input towards organizational changes necessary to reduce not only the problem reports, but the expense of correcting them after they are in the hands of customers in the market. During stable times or times of market and

organizational growth, such as occurred during the TPR1 and TPR2 releases for this product, software churn measurement may be an effective tool for measuring defects. With increasing software churn, a corresponding increase in software inspection to increase software quality [Boodoo, 2000] should be undertaken.

As an overall measure, the availability of post-release error reports is a solid overall measure of the quality of a product. Continued measurement and trend analysis of this data should be employed to understand end to end development cycle effectiveness in the delivery of high-quality products. Increases in error reports should then be used to regulate the design activity to balance the need to delivery products that are new and novel to the market, but while balancing the quality needs of the product in the market.

Adding more people to a program need not increase the amount of saleable features created by the development organization. This organization increased the number of saleable features release over release even though the size of the development team was dramatically reduced in size after the market peak.

Well conceived platform product architecture may lead to success, and should be created prior to engaging in specific product development. The

ability to readily adapt the platform to multiple product functions within a broad market creates flexibility. This flexibility permits creative architects in a customer focused development organization to adapt the platform product, creating new product functions. These product functions can be delivered to the market to meet the needs of more customers.

## 7.2 Limitations

There are several limitations that were identified as a result of performing this study. The most evident is that this study includes a single product. This fact may limit the general applicability of the findings to other products or markets; regardless of the development organization that is involved. To overcome this limitation, testing of the hypotheses developed here across a statistically significant number of samples would remove this current limitation.

There are 3 additional limitations related to data availability and analysis; software analysis tools, the number of available releases, and financial data. For software analysis, there are few broadly available, generally applicable tools for tracking changes to software product architecture. This may be a consequence of the number of available definitions for software architecture, which could be a limiting factor in development of metrics that could be used to track changes to a software product along this dimension.

As a single product, the five year study period was a long time in market terms, but from a product point of view only included 5 main releases. The number of available releases for study limited the hypotheses development to an empirical one rather than a statistical one.

As a development organization within a large corporation, profit and loss financial data relating to the development organization was not available. This may have provided further insight into the number of development team members during the downturn: High revenues on this specific product may have enabled the development organization to carry more staff even though on the whole the market couldn't justify more than a proportional staffing level.

The other limitation of this study related to available interview subjects. Only development team members could be traced down and interviewed. This limited the scope of managerial insights that could be captured to direct product team members and managers. High level corporate strategies and decisions beyond the product development team were not uncovered. For the purposes of this study this is viewed as relatively minor in contribution, as the development team could only control things within their product scope.

## 7.3 Opportunities for further research

There are four opportunities for further research that have been uncovered through this study.

The first set of research opportunities is within the scope of the hypotheses developed here: Namely to test them on a statistically significant number of products within the telecom market, as well as to extend the testing of the hypotheses to other markets. These activities would validate the general applicability of the findings of this study.

An additional area for further research is within software engineering itself. Being able to measure and categorize the architecture and track the changes of a products' architecture over time with an accepted standard method would be of broad application. This ability would provide a more holistic measurable view of a system; and provide data for management decisions around investments to improve the system.

The main finding in this study was that the product architecture was resilient, even though it was repositioned to a new market opportunity with a development organization that was under stress. Studying how resilient product architecture is, what measurements would abstract the architecture to

a representation of its stability, and how and when to look at architectural reinforcement would be of high benefit and value to practitioners.

Finally, further study within the area of the role, size and scope of product verification and validation within a development organization is understudied. Practical recommendations relating to how to structure an organization to achieve specific business results permit managers of software products to set up and structure their organizations for success.

## References


Bass, Len, Clements, Paul, and Kazman, Rick (2003) *Software Architecture in Practice*, second edition, Addison Wesley Professional, Boston, MA.

Boodoo, S., El-Emam, K., Laitenberger, O., Madhavji, N. (2000) *The Optimal Team Size for UML Design Inspections, National Research Council of Canada,* NRC/ERB-1081, NRC44149, 29 pages.

Bowman, Ivan T., and Holt, Richard C. (1998) "Software Architecture Recovery Using Conway's Law", *Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative Research (CASCON).*

Briand, L.C., Morasca, S., and Basili, V.R. (1993) "Measuring and assessing maintainability at the end of the high level design", *Proceedings of International Conference on Software Maintenance*, 88-97.

Brooks, Frederick P. (1995) *The Mythical Man-month : Essays on Software Engineering*, Addison Wesley Longman, Boston, MA.

Bunderson, Stuart J., and Sutcliffe, Kathleen M. (2002) "Comparing Alternative Conceptualizations of Functional Diversity in Management Teams: Process and Performance Effects", *Academy of Management Journal*, 45(5), 875-893.

Carlile, Paul, and Christensen, Clayton (2005) "The Cycles of Theory-Building in Management Research", version 6, January 6 (http://innosight.com/documents/Theory Building.pdf).

Collins, James C. (2001) *Good to Great: Why Some Companies Make the Leap....and Others Don't*, HarperCollins Books, New York, NY.

Conway, M.E. (1968) "How Do Committees Invent?", *Datamation*, 14(4), 28-31.

Dekel, David, and Kane, David (May 1, 2002) "Conway's Law Revisited: Successfully Aligning Enterprise Architecture", informIT, (http://www.vraps.com/conway.jsp)

Eisenhardt, K.M. (1991) "Better stories and better constructs: The case for rigor and comparative logic", *Academy of Management Review*, 16(3), 620-627.

Eisenhardt, K.M. (1989) "Building theories from case study research", *Academy of Management Review*, 14(4), 532-550.

Elbaum, S., and Munson, J. (1998), "Code Churn: A measure for estimating the impact of code change", *Proceedings of the International Conference on Software Maintenance*, 24-31.

Foote, Brian, and Yoder, Joseph (1997) "Big Ball of Mud", *4th Conference on Patterns Languages of Programs*, Monticelo, Illinois.

Foote, Brian, and Opdyke, William (1995), "Life-cycle and Refactoring Patterns that Support Evolution and Reuse", *First Conference on Pattern Languages of Programs*, Monticelo, Illinois.

Glaser, B., and Strauss, A. (1967) *The Discovery of Grounded Theory: Strategies of Qualitative Research*, Wiedenfeld and Nicholson, London.

Herbsleb, James D., and Grinter, Rebecca E. (1999) "Splitting the Organization and Integrating the Code: Conway's Law Revisited", *Proceedings, International Conference on Software Engineering*, Los Angeles, CA, 85-95.

Institiute of Electrical and Electronic Engineers, (1986) "IEEE Standard for Software Verification and Validation Plans", *IEEE Standard 1012-1986*, (http://www.ieee.org).

International Telecommunications Union (1996) "Introduction to CCITT Signalling System No. 7", *ITU-T Recommendation Q.700* (http://www.itu.int).

Jazayeri, Mehdi (2002) "On Architectural Stability and Evolution", *Proceedings of the 7th Ada-Europe International Conference on Reliable Software Technologies*, 13-23.

Kazanjian, Robert K., Drazin, Robert, and Glynn, Mary Ann (2000) "Creativity and technological learning: the roles of organization architecture and crisis in large-scale projects", *Journal of Engineering Technology Management*, 17, 273-298.

Land, Rikard (2002) "Software Deterioration and Maintainability – A Model Proposal", *Proceeding of the Second Conference on Software Engineering Research and Practise in Sweden (SERPS).*

Lehman, Meir M., and Ramil, Juan F. (2001) "Rules and Tools for Software Evolution Planning and Management", *Annals of Software Engineering*, 11, 15-44.

Lewis, Michael A. (2001) "Success, Failure and Organisational Competence: A case study of the new product development process", *Journal of Engineering and Technology Management*, 18(2), June, 185-206.

Lundberg, Lars, Bosch, Jan, Haggander, Daniel, and Bengtsson, Per-Olof (1999) "Quality Attributes in Software Architecture Design", Department of Software Engineering and Computer Science, University of Karlskrona/Ronneby, Soft Center, S-372 25 Ronneby, Sweden.

Moore, Geoffrey A. (1991), *Crossing the Chasm*, HarperBusiness, New York, NY.

Nagappan, N., Williams, L., Hudepohl, J., Snipes, W., and Vouk, M. (2004) "Preliminary Results on Using Static Analysis Tools for Software Inspection", *Proceedings of Fifteenth IEEE International Symposium on Software Reliability Engineering*, St. Malo, France, 429-439.

Neilson, Gary, Pasternack, Bruce A., and Mendes, Decio (2003) "The Four Bases of Organizational DNA", *Strategy+Business*, 33 (Reprint No. 03406 – http://www.orgdna.com).

Pettigrew, Andrew M. (1990) "Longitudinal Field Research on Change: Theory and Practice", *Organizational Science*, 1(3), August, 267-292.

Poole, Marshall S., Van de Ven, Andrew H., Dooley, K., and Holmes, Michael E. (2000) *Organizational Change and Innovation Processes, Theory and Methods for Research*, Oxford University Press, New York, NY.

Redwine, Samuel T., and Riddle, William E. (1985) "Software Technology Maturation", *Proceedings of the 8th International Conference on Software Engineering*, London, England: IEEE Computer Society Press, 189-200.

Royce, W. W. (1970) "Managing the development of large software systems: Concepts and techniques," *Proceedings, WESCON*, August.

Telcordia Technologies (2002) "Reliability and Quality Measurements for Telecommunications Systems (RQMS-Wireline) GR-929", (http://telecom-info.telcordia.com/site-cgi/ido/newcust.pl?page=idosearch&docnum=GR-929&).

## Appendices

### Appendix A – Market Data

| Quarter | Release | Number of deployed sites | NASDAQ Telecom Index (IXTC) |
|---|---|---|---|
| 1Q1997 | | | 201.82 |
| 2Q1997 | | | 248.7 |
| 3Q1997 | | | 289.48 |
| 4Q1997 | | | 306.6 |
| 1Q1998 | | | 388.29 |
| 2Q1998 | | | 412.91 |
| 3Q1998 | TPR1 | | 363.53 |
| 4Q1998 | | 19 | 500.91 |
| 1Q1999 | | 21 | 618.3 |
| 2Q1999 | | 23 | 655.1 |
| 3Q1999 | | 23 | 624.79 |
| 4Q1999 | TPR2 | 23 | 1015.4 |
| 1Q2000 | | 25 | 1102.01 |
| 2Q2000 | | 63 | 870.7 |
| 3Q2000 | | 65 | 729.94 |
| 4Q2000 | TPR3 | 67 | 463.44 |
| 1Q2001 | | 69 | 328.08 |
| 2Q2001 | | 70 | 311.62 |
| 3Q2001 | | 95 | 203.1 |
| 4Q2001 | TPR4 | 111 | 236.63 |
| 1Q2002 | | 116 | 173.76 |
| 2Q2002 | | 128 | 103.55 |
| 3Q2002 | TPR5 | 134 | 88.43 |
| 4Q2002 | | 146 | 108.79 |

**Table 6 Market data**

## Appendix B – Organization Data

This is the data on the organization. All numbers are normalized (% of maximum, where the maximum is assigned a value of 100).

| Quarter | Release | Group head-count (%/%) | % architects | % PV | Significant Events |
|---|---|---|---|---|---|
| 1Q1997 | | 16.78 | 0 | 12 | |
| 2Q1997 | | 20.13 | 0 | 10 | |
| 3Q1997 | | 23.49 | 2.9 | 8.6 | Product GM (later executive sponsor) becomes group head |
| 4Q1997 | | 26.85 | 2.5 | 17.5 | |
| 1Q1998 | | 30.2 | 2.2 | 22.2 | |
| 2Q1998 | | 33.56 | 4 | 20 | |
| 3Q1998 | TPR1 | 40.94 | 3.2 | 16.1 | |
| 4Q1998 | | 51 | 4.0 | 13.2 | |
| 1Q1999 | | 51 | 4.0 | 19.7 | |
| 2Q1999 | | 55.03 | 3.7 | 17.1 | |
| 3Q1999 | | 60.4 | 3.3 | 13.3 | |
| 4Q1999 | TPR2 | 63.09 | 2.1 | 13.8 | GM model changed, group autonomy reduces and reporting is into a larger organization |
| 1Q2000 | | 77.18 | 1.7 | 18.3 | Original product owners were re-vectored |
| 2Q2000 | | 77.85 | 1.7 | 25 | |
| 3Q2000 | | 100 | 2.7 | 17.4 | |
| 4Q2000 | TPR3 | 73.15 | 1.8 | 22.9 | |
| 1Q2001 | | 56.38 | 2.4 | 22.6 | |
| 2Q2001 | | 54.36 | 1.2 | 22.2 | Marketing moved from team/ team relocated to main campus and changed to virtual teams |

| Quarter | Release | Group head-count (%/%) | % architects | % PV | Significant Events |
|---|---|---|---|---|---|
| 3Q2001 | | 42.95 | 0 | 18.8 | |
| 4Q2001 | TPR4 | 38.93 | 0 | 22.4 | |
| 1Q2002 | | 38.26 | 0 | 21.1 | |
| 2Q2002 | | 38.26 | 0 | 19.3 | |
| 3Q2002 | TPR5 | 38.26 | 0 | 14 | Product Executive Sponsor departed organization |
| 4Q2002 | | 32.89 | 0 | 16.3 | |

**Table 7 Organizational data**

## Appendix C – Product Data

This is predictive data from the product software base.

| Release | Release Quarter | Codebase Size (KLOC) | Total Code Churn between releases | Clusters of cyclically dependent files (% of files) | Avg. System Comp-lexity | Defect Density per KLOC |
|---|---|---|---|---|---|---|
| TPR1 | 3Q1998 | 5064 | | 34.14 | 3.87 | 9.39 |
| TPR2 | 4Q1999 | 6154 | 3068315 | 39.21 | 4.05 | 8.98 |
| TPR3 | 4Q2000 | 6501 | 189720 | 39.7 | 4.02 | 9.37 |
| TPR4 | 4Q2001 | 6598 | 179565 | 40.27 | 4.17 | 9.07 |
| TPR5 | 3Q2002 | 6889 | 205885 | 45.2 | 3.91 | 9.38 |

**Table 8 Product predictive data**

## Appendix D – Post Release issue arrival data

This is the retrospective data captured from the field for the product.

| Quarter | TPR1 | TPR2 | TPR3 | TPR4 | TPR5 |
|---|---|---|---|---|---|
| 3Q1998 | 0 | 0 | 0 | 0 | 0 |
| 4Q1998 | 1 | 0 | 0 | 0 | 0 |
| 1Q1999 | 1 | 0 | 0 | 0 | 0 |
| 2Q1999 | 0 | 0 | 0 | 0 | 0 |
| 3Q1999 | 2 | 0 | 0 | 0 | 0 |
| 4Q1999 | 3 | 25 | 0 | 0 | 0 |
| 1Q2000 | 4 | 15 | 0 | 0 | 0 |
| 2Q2000 | 2 | 15 | 0 | 0 | 0 |
| 3Q2000 | 2 | 13 | 0 | 0 | 0 |
| 4Q2000 | 0 | 13 | 0 | 0 | 0 |
| 1Q2001 | 0 | 24 | 7 | 0 | 0 |
| 2Q2001 | 0 | 11 | 4 | 0 | 0 |
| 3Q2001 | 0 | 4 | 9 | 0 | 0 |
| 4Q2001 | 0 | 3 | 18 | 0 | 0 |
| 1Q2002 | 0 | 8 | 13 | 0 | 0 |
| 2Q2002 | 0 | 0 | 15 | 0 | 0 |
| 3Q2002 | 0 | 0 | 13 | 1 | 1 |
| 4Q2002 | 0 | 0 | 3 | 0 | 6 |
| 1Q2003 | 0 | 0 | 8 | 1 | 19 |
| 2Q2003 | 0 | 0 | 0 | 2 | 28 |
| 3Q2003 | 0 | 0 | 0 | 2 | 28 |
| 4Q2003 | 0 | 0 | 0 | 1 | 10 |
| 1Q2004 | 0 | 0 | 0 | 0 | 24 |
| 2Q2004 | 0 | 0 | 0 | 0 | 31 |
| 3Q2004 | 0 | 0 | 0 | 0 | 10 |
| 4Q2004 | 0 | 0 | 0 | 0 | 4 |
| Peak deployment | 19 | 57 | 70 | 13 | 56 |

**Table 9 Retrospective product data**

## Appendix E – OrgDNA Questions

To categorize the organizational type, the following questions were asked for input into the OrgDNA profiler online tool:

1. At the middle-management level, the average number of direct reports is: greater than 4 or less than 5.
2. Promotions include lateral moves (from one position to another on the same level in the hierarchy); agree or disagree.
3. "Fast track" employees here can expect promotions; every 3 years or more, or less than every 3 years
4. The culture of this organization can best be described as; persuade and cajole, or command and control
5. Important strategic and operational decisions are quickly translated into action; agree, or disagree.
6. The primary role of corporate staff here is to audit the business units, or support the business units.
7. Managers above me in the hierarchy "get their hands dirty" by getting involved in operating decisions; frequently or rarely.
8. Once made, decisions are often second-guessed; frequently or rarely.
9. Everyone has a good idea of the decisions/actions for which he or she is responsible; agree, or disagree.
10. Overall, this firm deals successfully with discontinuous change in the competitive environment; agree or disagree.
11. Important information about our competitive environment gets to headquarters quickly; agree, or disagree.
12. Field/line employees usually have the information they need to understand the bottom-line impact of their day-to-day choices; agree, or disagree.
13. We rarely send conflicting messages to the marketplace; agree, or disagree.
14. Information flows freely across organizational boundaries; agree, or disagree.
15. Line management has access to the metrics they need to measure the key drivers of their business; agree or disagree.
16. If the firm has a bad year, but a particular division has a good year, the division head would still get a bonus; agree, or disagree.
17. Besides pay, many other things motivate individuals to do a good job; agree, or disagree.
18. The individual performance-appraisal process differentiates among high, adequate, and low performers; agree or disagree.
19. The ability to deliver on performance commitments strongly influences career advancement and compensation; agree, or disagree.

## Appendix F – Interview Questions

The following set of open ended, leading questions were at hand during face to face interviews. These were used to only as required to assist the interviewee in providing candid information pertaining to the predictive data presented.

1. It has been said that the GUI interface was a key selling point and an operational positive for the product. Was this the case, and were there other key success attributes for the product?
2. When the product moved from one location to another at the peak of the telecom bubble, was there a change in product focus, and how did it change?
3. In the three main organizational reporting structures that existed during the study period, what were the key contributions to the corporation in each timeframe?
4. Within these three timeframes, what were the key focuses for the product at each stage?
5. Was there an executive sponsor for the product during the study period? If so, what was it that made that executive a strong sponsor?
6. During the staff reductions post-bubble, what skills or capabilities were the most important in deciding the staff that would remain with the team?