A QUALITATIVE AND QUANTITATIVE ANALYSIS OF A LARGE SCALE BANKING SYSTEMS MIGRATION PROJECT

by

S. DUTTA*
M. LEE**
and
L. VAN WASSENHOVE†

98/18/TM (RISE)

This working paper was written within the context of the Research Initiative in Software Excellence (RISE) at INSEAD. Financial support was given by the European Software Institute.

* Associate Professor of Information Systems at INSEAD, Boulevard de Constance, 77305 Fontainebleau Cedex, France.

** Program Manager, RISE at INSEAD, Boulevard de Constance, 77305 Fontainebleau Cedex, France.

† The John H. Loudon Professor of International Management, Professor of Operations Management at INSEAD, Boulevard de Constance, 77305 Fontainebleau Cedex, France.

Printed at INSEAD, Fontainebleau, France.

A Qualitative and Quantitative Analysis of a

Large Scale Banking Systems Migration Project

Soumitra Dutta, Michael Lee and Luk Van Wassenhove[1]
Research Initiative in Software Excellence (RISE)
INSEAD
Blvd De Constance
Fontainebleau, France 77305

---

[1] Author names are listed in alphabetical order.

Soumitra Dutta is Associate Professor at INSEAD. His email address is: soumitra.dutta@insead.fr.

Michael Lee is Program Manager of Research Initiative in Software Excellence (RISE). His email address is: michael.lee@insead.fr

Luk Van Wassenhove is Professor at INSEAD. His e-mail address is: luk.van.wassenhove@insead.fr

A Qualitative and Quantitative Analysis of a

Large Scale Banking Systems Migration Project

Executive Summary

In this paper, we present a qualitative and quantitative analysis of a large scale multi-year banking systems migration project at Kansallis Bank, a large Finnish bank, with the goal of linking changes in management and organization to changes in software productivity. A case history is presented along with an analysis of function-point based software metrics data, collected from over 60 projects spanning 12 years and consuming more than 900 man-years of effort. From this data we are able to show 1) that management and organizational issues contribute significantly to software project productivity, 2) the importance of a metric program for effective resource allocation and schedule planning, and 3) that senior management involvement is paramount to project success.

A Qualitative and Quantitative Analysis of a Large Scale Banking Systems Migration Project

Abstract

In this paper, we present a qualitative and quantitative analysis of a large scale multi-year banking systems migration project at Kansallis Bank, a large Finnish bank, with the goal of linking changes in management and organization to changes in software productivity. A case history is presented along with an analysis of function-point based software metrics data, collected from over 60 projects spanning 12 years and consuming more than 900 man-years of effort. Qualitatively, the project was marked by three major phases in which management became increasingly involved through project management and reorganization of the IT responsibilities within Kansallis. Although many aspects of these changes were not be captured explicitly in the project database, they manifested themselves through changes in software productivity. From this data we are able to show 1) that management and organizational issues contribute significantly to software project productivity, 2) the importance of a metric program for effective resource allocation and schedule planning, and 3) that senior management involvement is paramount to project success. The size and duration of the Kansallis Migration project combined with the increasing occurrence of similar scale projects make these results unique, significant, and timely.

Introduction

Kansallis Bank was established in 1889 with strong roots in Finnish nationalism and language. The success and growth of Kansallis Bank was continuous and stable through the decades. By the early 1990s, Kansallis Bank was the biggest commercial bank in Finland with almost 10,000 employees and more than 450 branches in Finland and 10 subsidiaries abroad.

During the period 1960 to 1995, Kansallis bank witnessed the evolution of three generations of banking systems. The first generation batch-computing systems[a] running on Siemens, GE and IBM 360 computers were in use from 1963 to 1981. The second generation on-line banking system[b] was in operation from 1973 to 1995. The

third generation multi-functional banking system[c] became fully operational across the bank in February 1995. All three (generation) systems were developed in-house by the technology staff of Kansallis bank.

The migration from the first generation batch processing environment to the on-line transaction processing system was started in 1972 and finished in 1981. The last transactions on the batch processing system continued till the end of the migration in 1981 as branches were progressively phased onto the on-line system. A larger and more complex project was the migration from the second generation on-line system to the multi-functional banking system environment. This second migration was started in 1982 and completed in early 1995. Analogous to the first migration, the use of the on-line system continued till Feb. 1995 while the third generation system was being rolled out.

This research presents both a qualitative and quantitative analysis of the second migration project in Kansallis Bank. The entire migration project took a little over 12 years to complete and required over 900 man-years of effort! System migration/integration projects of such a scale are common in business today for a number of reasons as described below.

Large Scale Software Projects

Software forms the foundations of today's information-based economy. Virtually all manufacturing and service operations are dependent upon the successful deployment of effective software systems. Software glitches have been known to shut off significant sections of the telecommunications infrastructure of large telecom operators such as AT&T for unacceptably long periods. Software forms the life-blood of several information-intensive service sectors such as airlines, hotels, banks and car rental agencies. Airline companies such as American and United have had to bear millions of dollars in revenue losses when their reservation systems have crashed, even for a few hours, due to software problems. With the ubiquitous presence of software in all corners of a company's operations, any significant upgrade of systems requires a massive undertaking of effort within organizations.

Despite all the visible progress in computing, software development remains a source of problems. Software projects are often reported to breach budget and time constraints in a relatively dramatic fashion. Gibbs (Gibbs 1994) notes that 'for every six new large-scale software systems that are put into operation, two others are canceled. The average software project overshoots its schedule by half; larger projects generally do worse'. Boehm (Boehm 1981) has written that 'the frequency of these software project disasters is a serious concern' and has quoted the results of a study which found that 35% of 600 surveyed firms had at least one runaway software project each. Martin (Martin 1994) notes that while problems such as requirements specification, architectural integrity, cost estimation and project management were discussed in the literature starting in the 1960s, 'these same problems are still too common'. Van Genuchten (Van Genuchten 1991) provides a good summary of research into some of the schedule related problems faced by software development projects.

Thus it is not uncommon to find system migration projects which require a significant amount of time and effort. The issues are similar for systems integration projects, which are also becoming more common as companies either try to integrate their different company-wide systems or merge with other firms. The issue of mergers is particularly acute in the banking sector where size and the resulting economies of scale are driving banks and financial institutions world-wide to merge with each other to create a few global giants. A good example of such a trend is the recently created United Bank of Switzerland, formed by the merger of two large Swiss banks, Union Bank of Switzerland and Swiss Bank Corporation.

Focus and Structure of Paper

While large scale systems migration/integration projects are becoming more frequent, the pressures on organizations to successfully manage them within short periods are increasing. Gone are the days when a firm could afford to spend a decade on a large scale system migration/integration project. About two and a half years ago, when one of the most profitable banks in Portugal, Banco Comercial Portugues (BCP), took over the largest (private) Portuguese bank, Banco Portugal Atlantico (BPA), the Board of the combined entity made it a top *commercial* priority to merge the IT platforms of

the two banks within a span of two and a half years. While BCP has been largely successful in managing this integration project[d], stories abound of other organizations where such system migration projects tend to go on for a lifetime. A good example of this is the upgrade being undertaken by the FAA of the US air-traffic control systems. Started in 1981, the project is eight years behind schedule, $5 billion over budget and still with no hope for a quick solution[e]!

While large scale system migration projects are both common and important, there has been little systematic research on them. This is due to a number of reasons. Often such integration/migration projects are strategic for a firm's success and thus few firms are ready to open themselves up to external observers. Also, typically such projects experience significant personnel turnover, tool/technology changes and schedule/effort overruns and all these factors reduce the chances of the firm welcoming external researchers. Finally, such projects typically extend over several years and thus demand dedicated research attention and consistency (in use of research instruments) over an extended time period.

Our involvement with the system migration experience of Kansallis Bank started late in the process when Pekka Forselius, one of the key project managers for the system migration project, joined the RISE team at INSEAD in 1995 as a Research Associate. He helped to rebuild a qualitative description of the entire migration project based on his own memory, interviews with other Kansallis managers and on a variety of internal Kansallis documents. Fortunately, Pekka and his fellow software project managers had utilized a commercially available formal tool, the Laturi System (Nevalainen 1994), to collect a variety of project related data. The Laturi System is commonly used in Scandinavian countries for measurement of software metrics and software project cost estimation (see description in a later section). The Laturi System provided us with a database of numerical data on a number of projects undertaken as part of the system migration project.

Thus, we are able to present in this paper both a qualitative and quantitative analysis of the large scale system migration project undertaken by Kansallis Bank. Due to the lack of prior research in this important domain, we believe that our analysis of

Kansallis Bank's experiences is unique and that it provides a useful learning opportunity for both academic researchers and practitioners.

A primary objective of this paper is to identify the factors, both qualitative and quantitative, in the Kansallis migration project which contributed to the variations in productivity over the duration of the system migration project. The relative importance of the engineering/technical issues and the associated organizational/managerial factors are analyzed with the help of a process improvement framework. These observations are then used to formulate general guidelines for effective management of large scale migration projects.

The remainder of this paper is structured as follows. We first present a qualitative summary of the Kansallis system migration project. This helps to provide an overview of the primary system-migration related events as they unfolded in the recent history of Kansallis Bank. Next, we embark on a detailed quantitative analysis of the system migration project, based on software metrics data collected with the help of the Laturi system. This quantitative analysis is conducted at different levels:

- Individual variables: an analysis of which variables individually explain the greatest variations in productivity.

- Combinations of variables: an analysis of which combinations of variables provide the best explanation for variations in productivity over the course of the Kansallis system migration project.

- Project phases: an analysis of how different factors influenced productivity during the different phases of the system migration project.

- SPICE categories: an analysis of productivity models built on database variables grouped according to the SPICE framework (Dorling 1993).

After having presented the qualitative overview and quantitative analysis of the Kansallis system migration project, we provide a detailed discussion of the results, with a particular emphasis on their relationships to prior research. The emphasis in this section, is on attempting to draw relevant managerial and research conclusions

from the experiences of Kansallis Bank. The final section provides some concluding comments and outlines avenues for future research.

A Qualitative Analysis of the Kansallis Systems Migration Project

Situation in the Early 1980s

By the early 1980s, Kansallis Bank's internally developed on-line banking system (running on a Bull mainframe environment) was handling almost one million on-line transactions each banking day. Juhani Peltola, a Development Manager, commented on the motivation for starting on the migration project:

> "We knew that the technology had evolved significantly since the early 70s when we had made most of our choices. We couldn't see an easy evolutionary way to adapt to the new technology. We were also in a peculiar way, conscious of the truth that the code wasn't very well structured and that the application documents had not been maintained since the development projects had ended."

A decision to replace the current on-line banking system with a new multi-function system (MFS) was taken by the board of Kansallis Bank initially in 1982. The driver for the migration plan was the technology division and the justifications for starting the migration were primarily technical. Targets were set for the new system: "openness", "complete integration", "ease of use" and "client server architecture"[f]. Overall, the goal was to design and build the "next generation banking system".

The new IT architecture at Kansallis Bank was seen in six layers as illustrated in Figure 1. The lower three levels, Hardware Service, Operating System Service and System Software Service formed the hardware and software foundations for the bank's IT architecture. The IT systems for these layers were to be largely supplied by external vendors. The upper three layers, Technical Service, Application Service and End-user Service contained critical value-adding banking applications and these systems were to be developed in-house.

Figure 1 about here

It was not difficult to get people in the business divisions to support the idea. A MFS-project team was established comprising several middle managers from both the technology and business divisions. Manu Ahola, a bank director and one of the managers involved in the MFS project commented on the level of initial planning:

> "It wasn't a real project, but we didn't know it then. There was a faint idea of the vision but there were no schedules, no cost estimates and no plans! Like many other development programs, this big migration was just a migration, a long chain of sequential and parallel projects, a program without any conscious phases."

The First Phase (1983-1986)

Pekka Forselius, one of the software project managers described one of the early challenges faced by the MFS project:

> "Starting a migration project is never easy - specially when you do not really know how big it is. By just relying on their imagination, the small group that was responsible of planning and managing the software development in the new environment produced the first schedules. It should all be done in three or four years, in approximately 15 projects."

The choice of IBM as the mainframe environment for the future was made in 1983, but the selection process of the client (PC) platform took an additional year. Many architecture projects were started in parallel in 1983 as well as several different working groups to think about implementation principles. However, as the decision about the client environment was not known, all reports produced were full of "Ifs" and question marks.

A select group of pioneer software developers underwent a four months training program at the IBM Training Center in Helsinki to learn about operating systems and database management systems in the IBM mainframe environment[g]. A major problem was the lack of opportunities to practice the new knowledge. While limited IBM hardware was given for pilot tests to the hardware specialists, this was not accessible

to the software developers. Also, no software development environment and/or tools were available on the IBMs in Kansallis Bank.

Finally, in 1984 the decision to choose Nokia as the supplier of client hardware was made. Kansallis Bank was the first customer for Nokia's new MPS10 minicomputer (which were to be used as the local servers in each branch) and Nokia's MPS4 microcomputers (desktop 286 PCs). The novelty of the machines had important consequences for the MFS project, as explained by Maritta Tynninen, a project manager:

> "Very soon the MPS10 proved to be more of a concept than a real computer. We were the only customer using it - it was not in use anywhere else but for Nokia's research laboratories. There was no development environment; no testing tools...nothing!"

Maritta Tynninen reflected on the early years of the MFS migration:

> "The main goals for the new system were set, but there were no quantitative targets. Thousands of hours were spent in talking about standards and general solutions but there was not a word about productivity! All methods and practices used in the (second generation) Bull environment were banned. It was important to find something better and more modern for the MFS project, no matter if the current practice happened to be a best practice! During the first phase of the migration, more working hours were used to develop methods and standards than to develop applications in the new environment. Looking back, this was probably good because 75 % of the code produced in those days was replaced totally within the next 3 years!"

Due to missing measurable targets for projects or systems, regular measurement practices were not used. This was despite a strong pre-existing tradition of weekly reporting within the IT division. Overruns in both cost and time was the rule, and not the exception. Instead of being late by a few weeks, projects were typically delayed by a year or more. A retrospective analysis of some of the more important migration projects conducted during this period showed that the productivity level was on average 7 function points per man-month (FP/Mmonth)[h]. The real productivity was

much worse because several cancelled projects with estimated productivity rates of around 3 FP/Mmonth were omitted from the analysis.

It was dawning on everyone that it made no sense to continue on this track. A couple of external consultants were called in to help IT Management in late 1985. This resulted in some basic improvements like producing an Information Systems Architecture and establishing a Project Department. The aim of the Information Systems Architecture was to define guidelines for an ideal banking system for Kansallis Bank and the Project Department was set up to develop an applicable project management system. The initial confusion was over when the first version of a long term software plan, including about 100 planned migration projects, was created and published in 1987. The plan was produced by the Project Department under the control of the management group of the IT division.

The Second Phase (1987-1990)

During this period the business environment was conducive to growth and growth of the IT sector was accelerating. As a result, staff volatility was high with software professionals moving with ease from one firm to the other. Programming tools were becoming more advanced and the level of sophistication of software development was on the rise and support staff were busy creating quality requirements. Organizational changes were taking place within Kansallis Bank, like decentralizing IT functions and adopting performance based compensation systems.

This was the phase of enthusiastic and idealistic software development within Kansallis Bank. As IT functions were getting decentralized, parallel projects were started in different departments. Maritta Tynninen described it as follows:

> "During these years, all flowers were allowed to bloom. Less than 50 % of all projects undertaken by the IT department were related to migration. There were a number of different outsourced projects developing separate systems for "independent" departments in the business sectors. Even if the project was classified to be a migration project[i], the owner was very innovative to change requirements when they invented new user needs."

Several projects at the application layer (see Figure 1) were completed in 1987 and 1988 (see Figure 2). These projects included the Customer and Product databases and other systems critical for enabling business applications in the End-user layer (see Figure 1). Around this time, end-users could also start accessing business information through the Investment Information System and the Customer Inquiry System. However, these systems did not exploit the power of the PCs, which were simply emulating text terminals without using their own intelligence[j].

Figure 2 about here

Soon after the completion of the first version of the application layer systems, planning for the next version of these systems started. It was envisioned that the new version would exploit the power of PCs and the new platform, and lead to a true client server architecture within the bank. Development projects to build the second version of the application layer systems were started immediately without having fully tested the first version. This caused several problems, and progress was at times slow and painful.

Taken as a whole, the migration project seemed to progress well and there was a flurry of activity within the IT department. While project metrics were being collected from the very beginning, it was not until the second phase that a formal method for analyzing productivity and effort was initiated. For this purpose, Kansallis adopted the Laturi System; a family of methods for advanced software quality, productivity, and risk estimation (more details on Laturi will be presented in a later section). Along with the adoption of Laturi, a new project management tool, Hessu, was also developed and adopted and resulted in improvements in long term planning of projects. In April 1989, the Project Department completed a holistic long term plan which included all planned software development work in the bank from the start of 1989 to the end of 1996. There were about 650 projects or maintenance jobs divided into 6 main categories: 80 Bull maintenance jobs, 20 Bull development projects, 150 migration projects, 100 IBM development projects, 250 IBM maintenance jobs and 50 development projects to other platforms.

Management training began to be imparted at all levels in the organization, with special modules for project leaders, primarily from the line functions. The operational responsibility for the migration clearly moved from the IT department to the business organization. The phase of enthusiastic and idealistic software development in Kansallis Bank ended in late 1990 when members of the Board started to take notice of the burgeoning IT costs. A Board member, Heikki Koivisto, was made responsible for the IT functions of the bank. He started by asking for the extra cost of running two different mainframe environments to be tabulated, something which had not really been done before. As a result of Koivisto's investigation, the IT decentralization process was stopped and reversed with major organizational consequences. An internal IT group, Kansallis Data was formed (in June 1990) as an independent profit center with the objectives of improving productivity, reducing costs and concentrating on finishing the migration.

Later, Board member of the Banking Group and Chairman of the Board of Kansallis Data, Veikko Ylitalo, took over responsibility for the migration. Veikko Ylitalo felt that the deadline of 1996 was too late and asked for it to be moved 2 years ahead to 1994. All Bull applications were inventoried once again in an attempt to try and find other means to replace them and speed up the pace of the migration. Hannu Sivonen, a business manager with a technology background, reported to Veikko Ylitalo and produced periodic reviews on the status of the migration for the Board. This interest and involvement of the Board was a major change as the migration had received negligible attention at the Board level previously.

The Third Phase (1991-1994)

The early 90's saw the start of a major recession in the Finnish IT sector. All major Finnish banks started writing off credit losses and suffering from the excesses of previous years. There was also a significant shift in attitudes towards IT within the line, as summarized by a manager:

> "For the first time in the history of Kansallis Bank, the line organization started taking responsibility for IT projects. Managers understood that IT was no longer a hobby for strange, bearded individuals in their pullovers and

sandals; rather it was something that had to be managed actively by the business."

The IT costs of Kansallis Bank decreased by 10-15% each year. IT staff, both administrative and operational were reduced. No new IT staff were added due to the imminent end of the migration. Natural reduction in staff was minimal due to the recession. No new managerial posts were created in the shrinking organization. It was difficult to change jobs due to the recession. Training was minimized and IT staff were concerned about their professional value. Due to financial constraints, the practice of giving project bonuses was stopped.

Due to the reduction in the overall schedule of the migration project by two years, there was a concentrated focus on projects related to the migration. Only a few new business-critical projects (such as tele-banking) were approved in addition to the migration projects. A software developer noted:

> "The pace of projects was relentless. You finished a project and there was another similar project waiting for you! We worked very hard!"

More than 60% of the new IBM applications were prepared and 50% of the transactional workload was moved from the Bull to the IBM environment during this four year period. This was an impressive achievement as 90% of the applications in the Bull environment were still running at the start of this period.

Several new applications were rolled out successfully in this period. All indicators used to measure the quality of application services pointed to the same positive direction. The number of operating errors per month was lower as compared to a few years ago when the practice of measuring and collecting data was started. The productivity level of projects increased to 15 function points per man month. No new methodologies or tools were implemented, but productivity improved by involving the methodology staff more intimately within projects. Management of the monthly reporting was decentralized to the business divisions. Risk management was improved at project level and different specialists from line functions worked together with the project manager from the early stages of the project.

The results were exceptional relative to both the history of IT within Kansallis Bank and the Finnish banking sector in general. An internal audit in February 1995 showed that the twenty five last migration projects, which lasted for about 8 months each, were late on average by only 8 days each. The productivity of IT within Kansallis Bank in the early 90s was known to be 50% higher than the Finnish banking sector's average; it was found that only 5 of the 25 projects had any cost overruns. On average the project costs were 10% less than those estimated.

The migration project ended when the last Bull Mainframe applications were terminated at the end of February 1995, just 8 weeks late of the revised schedule.

A Quantitative Analysis of the Kansallis Systems Migration Project

The qualitative analysis presented in the previous part of the paper was designed to provide a look at the rich story behind the migration project, focusing on management and organizational changes that occurred during the project's lifetime. In this section, we present a quantitative analysis of the migration project with two classes of intended goals: to identify those factors and combination of factors that have the most influence on software productivity across all phases and to examine how software productivity evolved throughout the phases of the project in association with organizational and management factor changes.

There has been a substantial amount of research in the literature focused on determining causes for variations in project productivity. In an analysis of data from the NASA/Goddard Space Flight Center, Bailey and Basili (Baily and Basili 1981) identified 21 productivity parameters and Boehm's COCOMO Model (Boehm 1981) identified 15 software factors which had a major impact on productivity. Productivity has been found to vary with hardware constraints (Boehm 1981; Vosburgh 1984) and tool usage, modern programming practices (Boehm 1981; Brooks 1981; Vosburgh 1984; Jones 1991), programmer experience (Watson 1977; Boehm 1981; Thadhani 1984; Vosburgh 1984; Jones 1991), team size(Brooks 1975; Conte 1986), duration (Aaron 1976; Belady 1979) and project size (Albrecht 1979; Behrens 1983; Vosburgh 1984; Conte 1986; Putnam 1992). Although similar in nature, the following quantitative analysis is unique because it is based on a dataset of projects related to the multi-year large scale systems migration within one organization (Kansallis Bank) and

can be linked to the organizational and management changes which occurred within the organization during the migration period.

Design of Quantitative Analysis

This quantitative analysis is made possible by examining data on the migration collected through the use of the Laturi System. The system, developed by the Information Technology Development Center of Finland and employed by many Scandinavian companies, includes procedures and methods for collecting project data and computing software productivity and size estimates. Parameters included in the Laturi System cover engineering, management, and organization issues and are given in Appendix Table A1. Information on individual projects are obtained through manual analysis of software projects. Although the Laturi System was not adopted until the second phase, projects completed before the adoption time were analyzed retroactively (software project information was recorded in the first phase, however not using the Laturi format).

The Kansallis project database is comprised of data extracted from 63 individual projects using the Laturi System (Nevalainen 1994) and includes information on projects whose lifetimes occurred during one of three different phases of the migration. Of the 63 projects, 38 are associated with the migration and 25 are other projects that occurred during the lifetime of the migration. These projects are included in the database to illustrate the differences between the migration and non-migration projects.

The productivity metric used throughout the quantitative analysis is *function points per man hour*, which is commonly used in business software development. Function points were first proposed by Albrecht in (Albrecht 1979). The basic idea of function point analysis is to treat software modules as black boxes and to measure the complexity of a module according to the behavior observed externally. In the initial framework proposed by Albrecht, the logical size of a software module was based on the number of inputs, number of outputs, number of inquiries, and the number of interfaces a module had. These values were aggregated using a weighted sum and adjusted using calibration factors. Several revisions have been made concerning the counting rules and overall structure of the metric.

The Laturi System is based on function point analysis, but differs slightly from Albrecht models. Like the model proposed in (Albrecht 1983), the Laturi System aggregates measurements related to inputs, outputs, logical files, inquiries, interfaces, and entities, however, the weighting coefficients are different: they are calibrated for Scandinavian software projects and employ a different structure to express variable complexity. The system also includes provisions for incorporating other productivity factors rating team characteristics such as staff tool skill, staff analysis skill, and project skill levels. For example, as seen in Table A1, the input count variable has been broken down into several sub-variables according to complexity. The modifications to the function point counting procedure has in part eliminated the calibration/adjustment required in the Albrecht model. Man-hours expended by the supplier is a recorded variable in the database and hence productivity is computed using an equation of the form:

$$Productivity = \frac{\Sigma w_i f_{inputs} + \Sigma w_i f_{outputs} + \Sigma w_i f_{logical\ files} + \Sigma w_i f_{inquiries} + \Sigma w_i f_{interfaces}}{man\text{-}hours}$$

where the w_i's represent the weights for each level of complexity of the factors. We must caution that debates still revolve around the validity of function point analysis as a measurement system and that techniques for computing software size and are beyond the scope of this paper.

There are four parts to our quantitative analysis:

1. **productivity variations due to individual variables** - The first part of the quantitative analysis was concerned with determining which individual variables, both class and non-class variables, explained the greatest amount of variation in productivity. In an analysis in which a class variable has a limited number of observations in particular classes, those classes were eliminated and the analysis was performed on the subset of well represented classes. The results of the analysis of all individual class variables are presented in this section.

2. **productivity variations due to combinations of variables** - In the second part of the quantitative analysis, combinations of two class variables were analyzed to find

the model which could explain the highest amount of productivity variance. A summary table of results and a matrix of productivity values for the best 2-class variable model are presented. The results of models based on combinations of all variables are presented in this section.

3. **productivity variations across phase** - In the third part of the quantitative analysis, the influence of the different variables on productivity for each of the three phases outlined in the earlier qualitative description of the Kansallis migration is described. This is interesting because there were distinct shifts in the level of management involvement and adoption of project management practices across the three phases.

4. **productivity variations in the SPICE framework** - In the final part of the quantitative analysis, we use the well accepted SPICE framework to categorize the variables in the database and then examine each category's effect on productivity. Both analyses over the lifetime of the migration project and through each phase are given.

Part I of Quantitative Analysis: Individual variables

The goal of this part of the quantitative analysis is to identify which of the class and non-class variables had the most influence on productivity. In our analysis, the influence of a variable is quantified by measuring the amount of variation in productivity accounted for by using the variable in question as the independent variable and productivity as the dependent variable (Maxwell, Van Wassenhove et al. 1996). Models that related the each of the variables listed in Table A1 to productivity were constructed using the SAS General Linear Models procedure (1989). The option to analyze the variance of unbalanced data was used for this analysis. Both additive and multiplicative (log) models were fit to the data.

Additive: $Productivity = a + b \times x_1 + c \times x_2 + \ldots . (1)$

Multiplicative: $Productivity = a \times x_1^b \times x_2^c \times \ldots . (2)$

where a is a constant which varies with the significant class variable(s)[k].

Analysis of Variance by Individual Class Variables

The variance explained by individual class variables are summarized in Table 1. Among the information listed in the table for each variable is, the number of observations used to calculate the variance captured, the type of model constructed, and the sign of the correlation (either positive or negative). The table also includes results from analysis of class variable subsets where appropriate.

The single variable which explained the greatest amount of variance in productivity in the dataset was User Interface (21%). This was followed by the Project Length (16.3%), Case Tool Type (11.8%), Phase (10.2%), Centralization of Software (7.4%), Migration project (7.2%), and Intended Market (5.6%) variables (See Table 1). A few other variables - Language, Hardware, and Centralization of Database - provided strong numerical results when analysis was performed on the entire database. However, these variables produced statistically not significant results when outlier projects were eliminated.

Table 1 about here

The influence of language on software productivity is a well studied topic and variations in productivity due to language choice have been shown to be significant (Albrecht 1979; Behrens 1983; Kitchenham 1992; Maxwell, Van Wassenhove et al. 1996). For example, in Albrecht's seminal paper on function point analysis, he compares PL/1 and COBOL and finds PL/1 to be more productive(Albrecht 1979). It is also quite easy to imagine that higher level languages lead to higher productivity rates naturally as illustrated by assembler vs. ADA productivity comparisons made in Maxwell (Maxwell, Van Wassenhove et al. 1996). In the Kansallis database, examining language alone on the entire database explained 57% of the variation in productivity. However, when the subset of Languages with 3 or more observations was considered the results were no longer significant.

The influence of Case Tool Type on software productivity also has also been studied in the literature (Boehm 1981; Banker R.D. 1991; Jones 1991; Kitchenham 1992; Nevalainen 1994; Maxwell, Van Wassenhove et al. 1996). Many Case Tools now integrate syntax checking and debugging facilities leading to significant reductions in

the coding-compiling-test cycle. In the Kansallis database, Case Tool Type explained 11.8% of the variance in productivity.

User Interface alone explained 21% of the variation in productivity in the entire database. Projects involving graphical interfaces were nearly three times more productive than projects with a character interface (0.27 vs. 0.09 FP/Hour). Similar studies on the effect of customer interface and productivity and effort have appeared in the literature (Watson 1977; Baily and Basili 1981; Brooks 1981; Vosburgh 1984; Kitchenham 1992). The fact that the graphical interface projects appeared more in the later phases account for the overall increased productivity level (an increase in productivity in graphical interface projects with phase was observed and will be discussed in part 3 of the quantitative analysis). In addition, most of the graphical interface modules were reported as having used a code generating tool. It is possible that the productivity on these projects is artificially high due to the fact that the Laturi measurements are made on the generated code (although function point sizing may treat this issue better than Lines of Code sizing (LOC), it is still not a panacea for software sizing).

The centralization of the software alone explains 7.4% of the variation in productivity. The highest productivity was obtained when the software was completely in the client. This also can be attributed to the fact that the client based software projects were relatively small. The lowest productivity exhibited are those projects developed for the server only. The server-only projects are also the earlier projects as they formed more of the service level functions and existed during the less mature phases of the migration.

Project length is a class variable derived from the project duration: Short projects were defined as being less than 12 months duration, Medium projects were between 12 and 18 months duration, and Long projects were greater than 18 months duration[1]. Alone, Project Length explained 16.3% of the variance in productivity. Short projects had the highest productivity levels. Medium and Long projects had similar productivity levels, but lower than the Short projects (0.10, 0.09, and 0.13 FP/Hour respectively). Similar results have appeared in the literature (Aaron 1976; Belady 1979; Putnam 1992; Nevalainen 1994; Maxwell, Van Wassenhove et al. 1996). One common

hypothesis that considers this effect is that long projects are exposed to more volatility in requirements. In this migration project, the productivity dependence on project length could also be attributed to the fact that longer projects are also the older projects, which did not benefit as much from the development process improvements.

As outlined in the qualitative analysis, the migration project was marked by distinct phases attributed to the changes in organization and management of the project. The effect of these changes are reflected in the increase in Software Development productivity in later phases. Phase alone explains 10.2% of the variation in productivity and on average productivity increased 11% per year between 1983 and 1993. A more detailed analysis of the factors leading to this increase in productivity is presented in a later section.

Whether or not a project belongs to the migration effort explains 7.2% of the productivity variance. Thirty eight of the projects included in the Kansallis database are migration projects (the other twenty five are non-migration related projects). Migration projects have a lower productivity level than non-migration projects with 0.09 and 0.14 FP/Hour respectively. In fact, the top five productive projects were non-migration projects with starting dates in the third phase (more on this in the discussion).

Analysis of Variance of Individual Non-Class Variables

The results of the analysis of individual non-class variables are presented in Table 2. The hours worked, duration, requirements volatility, use of standards, number of calculations, number of users, quality requirements and efficiency requirements show a negative impact on productivity. Unfortunately, the quality and efficiency dimensions of software are not integrated into the productivity measure. Discussions surrounding these issues have appeared in the literature and are beyond the scope of this paper.

The Project Duration is used to derive the Project Length class variable and the 15% explanation of variance is similar to the result presented on Project Length. The hours worked, duration and requirements volatility have been linked before in prior research. Quality and standards use are closely interrelated by definition and have

been shown to be highly correlated. By definition, standards are part of overall quality systems (Pressman 1997). Although the use of these quality related methods can decrease productivity locally, the overall picture that takes into account later reuse must be considered. The negative correlation with the number of calculations can be easily justified through increased complexity.

Other notable factors that have positive impacts on productivity are the number of very easy functions, the start year of the project, the tool skills of the staff, and the number of output functions. All other variables taken individually were found to be statistically not significant.

Table 2 about here

Part II of Quantitative Analysis: Combinations of Variables

Software development is a complex process, with many interrelated factors affecting productivity. Although individual factors are interesting from the point of view of identifying simple project control mechanisms, identifying influential combinations usually leads to higher levels of performance. In this part of the quantitative analysis, simple combinations of relevant subsets of class variables were analyzed to discover which combinations explain the highest amount of productivity variance. Table 3 summarizes the four best models found that use only two class variables. The Length variable appears in most of these models reflecting its significant influence even when taken as an individual variable. This result supports findings in earlier studies on length and productivity. One of the models that combines phase and length represents a entirely management influenced model. Both the variables, Length in particular, in this model are management controllable (Boehm 1987). The Length of a project is very much related to the requirements volatility. The management often has the ability to dictate which requirements to include (and throw away) and therefore has some degree of control over the length of the project.

Table 4 summarizes the best model for combinations of two class and two non-class variables. These models were built by first taking the best two class model and then adding in two non class variables. In the majority of these models, variables related to engineering issues, such as number of easy functions or efficiency requirements, were

added. The amount of variance explained is over 80% higher than with using only two class variables. In two of these models, efficiency and quality requirements appear, both of which are also management controllable.

Table 5 presents an improved combination model. Many of the variables in the model of Table 5 have appeared in the models described above.

Tables 3, 4 and 5 about here

Part III of Quantitative Analysis: Variations Across Phase

Many of the non-quantifiable, mainly organizational and managerial factors, changed along the three phases of the project as described in the qualitative analysis. It is important to note that the phases came into being not by design, but rather emerged through the retrospective analysis (by Pekka Forselius and his colleagues within Kansallis Bank) of the changing role and level of involvement of management. Hence, in this section of the analysis we studied how each of the key variables, identified in the previous two quantitative analyses, changed with phase and how each of the variables reflected the changes in managerial and organizational factors.

Figure 3 about here

The variation in observed productivity along the phases is as shown in Figure 3. Early in the project, the migration group was driven primarily by technology. Although general concepts existed, they did not have clear system goals due to the lack of direction from the business divisions. As a result, there were no quantitative targets and time and cost overruns were rampant. It was not until the end of the first phase that an overall Information Systems Architecture was designed and a Project Department set up.

The formal collection and usage of metrics, via the Laturi System, was initiated in the second phase and formal project management techniques were adopted indicating a shift in responsibility to the management. The management involvement in the migration project was further manifested by the appointment of a board member responsible for IT functions and through the setting up of Kansallis Data as an independent profit center with the goals of improving productivity.

Figure 4 about here

The productivity factors (measured on a scale of 1 to 5 with 1 being the lowest and 5 the highest) exhibited significant change with phase as shown in Figure 4. Quite naturally, due to technology maturation, the use of tools and tool skills of staff increased. In addition, the top management initiative in training led to increases in team skills of staff.

The transfer of the migration effort to the business units led to more refined goals and thus reduced the requirements volatility, software logical complexity, and average project duration. Each of these factors contributed to improvements in productivity.

The average duration of the projects was also found to decrease with phase. The mean duration of 26.4 months in phase 1 decreased to 18.1 in phase 2 and 14.5 in phase 3. As described earlier, the mean productivity in FP/hr was highest for projects of short lengths (0.136) and decreased for projects of medium (0.104) and long (0.87) lengths. This explained a further 12.8% of the productivity variation across phases.

Some of the technical/engineering factors which also changed with phase significantly are user interface, centralization of software and case tool usage. The user interface was classified into two categories - graphical and character-based. Projects for the graphical interface had a productivity of 0.267 FP/hr whereas those for the character interface had that of 0.094 FP/hr. The use of graphical interface went up from zero in Phase 1 to 3.3% in Phase 2 to 14.3% in Phase 3 leading to significant improvements in productivity.

With the adoption of the client-server architecture for the banking systems, there was also a gradual shift of the development platform towards the same (see Table 6). The productivity levels associated with the centralization of software in server (mainframe) and in client (client-server) were markedly different. On an average the centralization of software in the server led to productivity levels of 0.0886 FP/hr while those for software in client were 0.2 FP/hr and a combination of the two had productivity of 0.121 FP/hr.

Table 6 about here

Case tool usage led to significant increases in productivity. The average productivity of projects without case tools was found to be 0.0968 FP/hr while that using case tools was 0.1217 FP/hr. Case tools usage increased from 0% in phase 1 to 10% in phase 2 to 32% in phase 3. The associated increase in productivity accounted for 11.6% of the variation. Cusumano and Kemerer and Boehm have also reported increase in productivity with tool usage (Boehm 1987; Kemerer 1987; Cusumano 1990). Boehm points out that usage of tools leads to larger savings in the later stages of projects (coding and testing) (Boehm 1987).

Figure 5 provides a slightly different perspective on the variation in productivity over phase. The figure shows migration and non-migration projects plotted in the function point size (log of the inverse of function point size) and supplier work hours space. In this diagram, projects on the left edge of the datacloud, or frontier, closer to the origin represent the most efficient projects - for a given project size, a project on the frontier consumes the least amount of supplier work hours. The diagram shows that migration projects started in phase 3 did not improve overall productivity as compared with those started in phase 2. However, improvements are seen in that for an identical amount of supplier work hours, projects with increased function point size were completed.

Figure 5 about here

Part IV of Quantitative Analysis: *SPICE Based Analysis*

The SPICE (Software Process Improvement and Capability dEtermination) model framework was developed specifically for software engineering (Dorling 1993). SPICE emphasizes issues related to software project management and quality and improvement of software processes. The SPICE framework has the twin objectives of facilitating both process improvement and capability determination. Towards these ends, it distinguishes between two types of practices: base practices and generic practices. Base practices cover the core processes related to software development and are grouped into five process categories. Generic practices refer to the implementation and institutionalization of processes within an organization and help in the determination of the appropriate capability level of an organization.

The SPICE framework is constructed from elements in technical/engineering, project management, organizational, customer-supplier, and support areas. According to the SPICE categories, we classified the Kansallis database variables as shown in Appendix Table A2.

By classifying the variables in the Kansallis database into various categories such as technical and managerial, we can examine how each of the categories impact productivity and identify those that influence productivity most. In addition, looking at the results from the quantitative analysis using such a framework can help us interpret the results within the context of the qualitative analysis. It is important to note that different classifications are possible as some of the variables could arguably be placed in more than one category.

The number of variables within each category varied widely with most being classified into engineering issues and none in support issues. The percentage of variables classified in each category are mentioned within brackets.

- Customer-Supplier Relations (4%): These include the variables dealing with managing customer requirements, providing support and service and assessing customer satisfaction.

- Engineering/Technical Issues (74%): These include the majority of the technical variables relating to system requirements and design and development and maintenance of the software.

- Project Management Issues (12%): These include project planning, coordination and managing resources, risks and quality.

- Support Issues (0%): These include documentation development, quality assurance and problem resolution.

- Organizational/Personnel Issues (10%): These include process definition and improvement, training, enabling reuse and provision of amiable development environment and facilities.

The analysis in the section follows a similar track as the analyses appearing in part II and III. The difference in this analysis is that models are built directly grouping variables in a given SPICE category and examining their combined contribution towards the variation in productivity. Non significant variables are eliminated and a refined model is computed. Only the variables which remained statistically significant were retained in the final models.

Each of the categories delineated above explain a significant amount of the variation in productivity, highlighting the need to focus on all the factors for achieving process improvement. The relative importance of these factors, however, was observed to be quite different. Treating the entire Kansallis database, across all phases the Engineering Issues accounted for the maximum variance explained in productivity of 36.5% while Project Management explained 29.7% and Organizational Issues 5.9%.

An analysis of the migration projects only reveals slightly different results. The Engineering issues remain dominant at 32.5%, but the Project Management Category drops to 9.6% and the Organizational Issues increases to 13.9%.

Although the analysis indicates that the Engineering / Technical Issues offer stronger explanation power than the Project Management and Organizational Issues, this could be due to the fact that Engineering Issues lend themselves more easily to quantification. In addition, many of the management, organizational, and changes in strategy manifest themselves through these technical variables. For example, although the following variables are technical in nature, management and strategy influence them: volatility of requirements, use of software and modern programming practices, and platform requirements.

There were many other non-quantifiable factors which changed with phase and contribute to the observed variation more than anything else. This is also evident from the fact that only 6 variables in Project Management and 5 in Organizational / Personnel Issues explained 29.7% and 5.9% of the variation in productivity, whereas the 37 variables in Engineering explained only 36.5%. Organizational Issues appear to have a stronger impact on the migration projects as compared to the non-migration projects. This is interesting in light of the increased management attention and participation in the latter two phases of the migration project.

The results obtained are in accordance with previous research. Van Genuchten (Van Genuchten 1991), attributes 45% of schedule differences to Organization related reasons. Even in Boehm's COCOMO model organizational factors like personnel/team capability and applications experience and project scheduling factors like timing and schedule constraints are just as important as other technical factors (Boehm 1987).

Hence, both engineering and managerial issues need to be addressed effectively to achieve success in software projects. However, the latter tends to be neglected because it is more difficult to quantify. One step towards overcoming this pitfall is to adopt a company-wide metrics program along with a process improvement framework (Daskalantonakis 1992). As in the Kansallis Bank Migration case, the metrics were instrumental in monitoring project behavior and the process improvement/maturity framework provided guidelines towards improving both the software process and the organizational and management structure of the company.

Discussion of Results

In summary, our analysis of the Kansallis Bank Migration project has identified both quantitative and qualitative changes through the project. The qualitative analysis documented managerial and organizational changes and the quantitative analysis focused on software productivity through the life of the migration project. In this section, we identify the significant links between the qualitative and quantitative changes and provide some discussion on these points.

As seen in the first section, the majority of the managerial and organizational changes occurred during the second and third phases of the migration project. One of the more significant changes was that the project shifted from being technology driven to business goal driven. In the early part of the project, the IT people were responsible for defining goals and application specifications. Left to their own devices, the technology group came up with features exploiting the new technologies. However, the goals were vague in that while conceptual specifications were made, concrete projects were not defined. This was not the case until the management stepped in and shifted the responsibility of the project towards individual business units. This change is reflected in an 33% increase in average amount of customer effort supplied in phase

two for migration projects. In addition, with the added customer effort, both the volatility of requirements and the project length decreased. Along with these changes came an increase of over 50% in productivity for migration projects started in phase 2. This finding supports prior research linking both shorter project lengths and decreased requirement volatility to increased productivity. The linkage between personnel, organizational-related reasons, and project success has been reported in the past (Brooks 1975; Conte 1986). Repeatedly in the literature, a key message has been that people are the key resource in the process of software development and it is necessary to keep them motivated (DeMarco and Lister 1987; Parnas 1997). For example, Van Genuchten (Van Genuchten 1991) attributes about 60% of productivity variations to personnel and organizational-related reasons.

The initiation of a formal metrics program and start of a program management department during the second phase was key to improving and monitoring productivity. Together with the metrics program and project management department, cost overruns and resource allocation were refined during the development process. Andersen (Andersen 1990) provides a good overview and insight into the development process and establishes norms for interpreting metrics values obtained in future projects. Daskalantonakis (Daskalantonakis 1992) recites Motorola's experience with using metrics for process improvement over time across projects and for in-process project control.

In addition, the second phase also exhibited a decrease in the logical complexity of the software for both migration and non migration projects. Several reasons could have given rise to this effect. One possibility is that by refinement of the system architecture, the problem became simplified and more amenable to efficient decomposition. The decision to rework and solidify the IT architecture was purely managerial as finished projects were scarce. The decomposition of the entire project into smaller projects could be both driven by the refined architecture, but also the nature of the projects. The system developed from the bottom up. The lower level services had to be in place before the higher level service could be built. Low level services, by definition, have greater impact on the system and potentially require more complex operations.

Productivity increases in the second phase due to advances in software development methods and tools cannot be ignored. It is well known that software development methodologies improve performance by imposing a structure on the development process (Rook 1986). This was reflected in the increase in tool skills and usage.

In addition, changes in other productivity factors occurred between the first and second phases. Most notably, the efficiency and quality requirements increased. In the individual variable analysis, and intuitively, these factors had a negative impact on productivity. This complicates the analysis as the objectives of high quality and high productivity naturally conflict. Optimizing both simultaneously requires measurements in both dimensions. Since no measures on the error or quality of the software are provided in this database, the effects of these increased requirements levels is unknown.

In the third phase, Kansallis Bank's senior management took more direct control of the migration project as compared with the second phase. At the end of the second phase, Kansallis senior management decided to pare down the migration project scope and schedule to finish two years earlier. This meant that many of the previously desired functions were eliminated from the delivery. In addition, the projects that remained had very firm deadlines. As a result, the duration of projects were shorter on average and high premiums were placed on finishing them on schedule.

Although the productivity of the migration related projects did not improve much, there are a few explanations and observations that explain this result. Firstly, as mentioned previously, both the quality and efficiency requirements increased for projects starting in phase 3 over those starting in phase 2. The increase in this case was fairly significant moving from an average level of 3.89 to 4.38 (the change was 3.75 to 3.89 between phase 1 and 2). Secondly, although the overall logical complexity decreased for migration projects in the third phase, the function point size increased significantly. In addition, the average number of hours worked by the supplier and customer decreased. This signifies that Kansallis became much more efficient at producing high quality applications. (Unfortunately error and quality measurements are not included in the database.) As stated in the qualitative analysis, a large number of applications produced in the third phase worked well upon first use.

What stands out are that the non-migration projects started during the migration had significantly higher productivity values than the migration projects. These non-migration projects had lower quality and efficiency requirements. In addition, the use of case tools in these projects was much higher than in the migration projects. On average, the volatility of requirements were lower and generally customer involvement was higher. Along the phases, these projects exhibited a definite trend towards decrease in project duration, function point size, logical complexity, and supplier and customer work hours. Upon further examination, the highest performing projects in the database belong to a group of non-migration projects whose intended market was marked as 'outsource.' The top performers within this group also were of the graphical user interface type and used code generating tools.

We hasten to point out however, that although the non-migration/migration demarcation suggests that the two classes of projects are independent, the reality is the non-migration projects were equally affected by changes in management directed at the migration project. The organizational changes that took place occurred company wide - a project management department was set up and eventually Kansallis data, an independent entity, was spun off. Hence, by including the non-migration project data, we can have an overall picture of the changing IT situation at Kansallis during the migration period.

The Kansallis senior management participated more in the latter phases of the migration effort than in the earlier phases by implementing both organizational and management changes. The startup of a project management department and metrics collection system in combination with the transfer of responsibility of the migration project from the IT development teams to the business units helped Kansallis to identify concrete goals. The metrics program allowed the management to reschedule and reallocate resources during the project which led to negligible schedule overruns of the projects in the final phase. Although the changes brought upon by management were not directly encoded into the database, their effects manifest themselves in our analysis. To reiterate a key result, while the project management variables represented only 10% of the variables in the database, they contributed up to 40% of the variance in productivity. As shown in the two class models presented in part 2 of the quantitative analysis, the phase variable recurs in several high performing models. The

fact that phase appears so often suggests that many significant non-quantifiable managerial factors manifested themselves only through the phase variable.

Managerial Implications

The implications of the above analysis for organizations undertaking large projects, migration or otherwise, and for all other organizations involved in software development are as follows.

It is important that senior business executives get involved in the management of critical IT projects. The importance of general management participation and involvement in IT has been stressed for a long time. About two decades ago, Adams (Adams 1972) claimed that "the successful implementation of an MIS depends on the active and informed participation of executive management" (p. 54) and Swanson (Swanson 1974) wrote "that management should be 'involved' in MIS development is a popular wisdom" (p. 178). More recently, several other authors have also reiterated such a stance. For example, Nath (Nath 1989) noted in his survey that for both senior general managers and MIS managers "upper management commitment is deemed critical" (p. 71) for aligning MIS with their organization's goals. Jarvenpaa and Ives (Jarvenpaa and Ives 1991) found from a survey of fifty five CEOs that those CEOs who participated in the management of IT were more involved in it and that this in turn led to their firm being more progressive in the use of IT.

Adoption of structured programming techniques and project management tools is key. The software development methodology should include technical methods to assist in analysis, design, coding and testing and also the management procedures to control the development process and deployment of the technical methods. The technical methods provide the basis needed for managerial control while the managerial procedures provide the organization and resources which enable technical development to proceed. Over the last decade, several software maturity models such as the Capability Maturity Model (CMM) (Humphrey 1989; Paulk 1995) and SPICE (Dorling 1993), have been proposed and used within industry for assessing and improving software development processes. The focus is on issues such as whether an organization has appropriate software project management procedures in place and whether the right tools are being utilized for managing software processes.

Adoption of project management techniques would involve activities like planning, scheduling, budgeting and defining milestones. These would, in turn, be derived from the work definition and work breakdown structures established. The project manager needs to establish a project plan which shows the schedule of deliverables and allocation of resources. Project control is a critical part of project management and would involve feedback loops which compare the progress reports with estimates. This can be achieved by having a well-designed metrics program in place.

Software metrics must be goal-oriented and precisely defined. The types of metrics that would generally need to be adopted are process metrics for improving the development and maintenance process, product metrics for improving the software product and project metrics for tracking and controlling projects. For large projects it would also be necessary to include company-wide business unit level metrics to enable comparisons across projects.

The above guidelines are important because organizations cannot today afford to spend a decade or more managing a systems migration project (or a systems integration project). As mentioned in the introductory section, global competitive pressures and customer requirements and forcing organizations to innovate in ever-decreasing cycles. IT system changes have to keep pace with these decreasing business innovation cycles. Firms should strive to avoid mistakes like those made by Kansallis and save time[m].

Conclusions

Our qualitative and quantitative analysis of the multi-year large scale Kansallis bank migration project has identified both key management and technical factors that lead to increased software project productivity. Among the strongest findings in our research are:

1. management and organizational issues contribute significantly to software project productivity. Although many changes in management and organization were not quantified in the project database, their effects on productivity manifested themselves in part through some variables in the database. Most notably the phase variable, whose value is derived from retrospective analysis of the management

changes, appears among the stronger productivity models when used alone and in combination with other variables. Among most significant organizational changes was the shift in responsibility of the migration effort from technical to business units.

2. the adoption of a metric program allowed the Kansallis project management department to more effectively allocate resources and plan schedules. Many projects in the final phase were reported as finishing very near their scheduled completion dates. Before the final phase, the entire migration effort was rescheduled to finish two years earlier. Such effective rescheduling could not have been performed without the project data.

3. senior management involvement is paramount to project success. As the migration project progressed, senior management became increasingly involved leading to increased productivity. Without senior management participation, many of the organizational changes would not have taken place. For example, the formation of the project department, which led to better scheduling, was brought on by top management. In addition, in the end of the second phase, it was top management who made the decision to finish the migration two years before the prior schedule. These changes directly led to other productivity increasing effects: shorter project duration, reduction in requirement volatility, and others.

Grouping the variables into the SPICE categories provided additional support to the hypothesis that Project Management and Organizational, and not just Engineering, issues play significant roles in software project productivity. Although the engineering variables were better represented in the database, their contribution was proportionately lower than the contribution by organizational/personnel issues.

Some of the critical technical issues identified in this database were complexity of software, user interface and case tool usage. Besides these technical factors the other critical project management factors identified were project duration, requirements volatility, and level and usage of standards. While these factors did contribute to the improvements, the driving force for the observed changes was the increasing involvement of management during the project.

This results of this analysis are limited by the assumption that productivity is measured purely as function points per man-hour. However, as the data shows, there is a definite correlation between such factors as quality and efficiency requirements and productivity measured as function points per man-hour. There is a natural conflict in maximizing simultaneously code production, code quality, and code efficiency. Other multi-objective analysis techniques would be useful for investigation in this context. In addition, other variables worthy of including in the database might be those that describe maintenance and reusability dimensions.

In addition, the managerial and organizational dimensions could be augmented with many variables which were not measured. Information on cancelled projects was excluded from the database and hence, the productivity measures are higher than actual. This highlights the need for goal driven design of metrics programs.

Further research using alternative data analysis techniques is warranted. Other nonlinear techniques employing methods such as neural networks or fuzzy logic have the potential of producing higher performance models. A more detailed analysis of the outliers could provide deeper insight into improving software productivity as it is often the case that the search for excellence leads one to the outliers.

Acknowledgement

The authors would like to acknowledge Abhijit Katti, Pekka Forselius and Katrina Maxwell for their contributions and support.

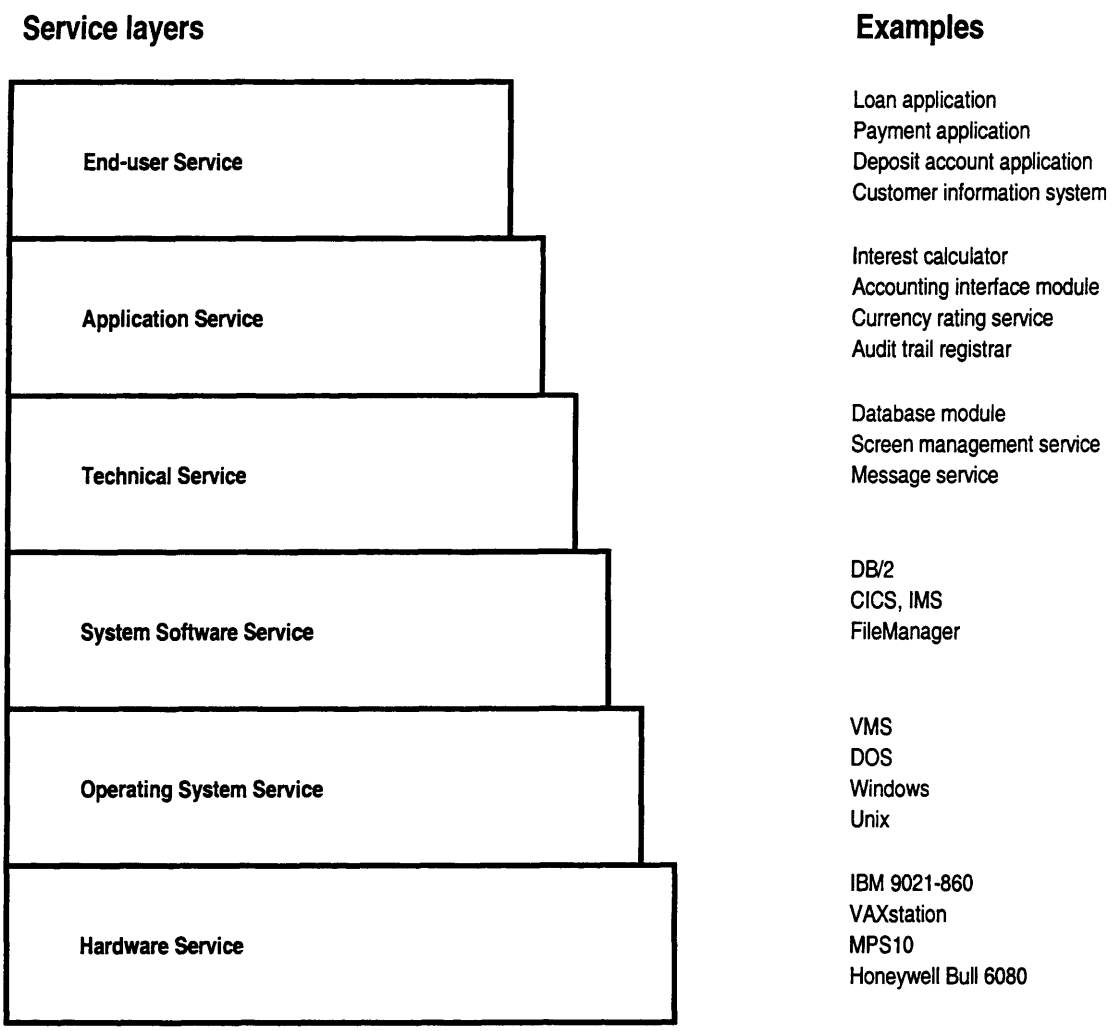

Figure 1: The IT Architecture at Kansallis Bank

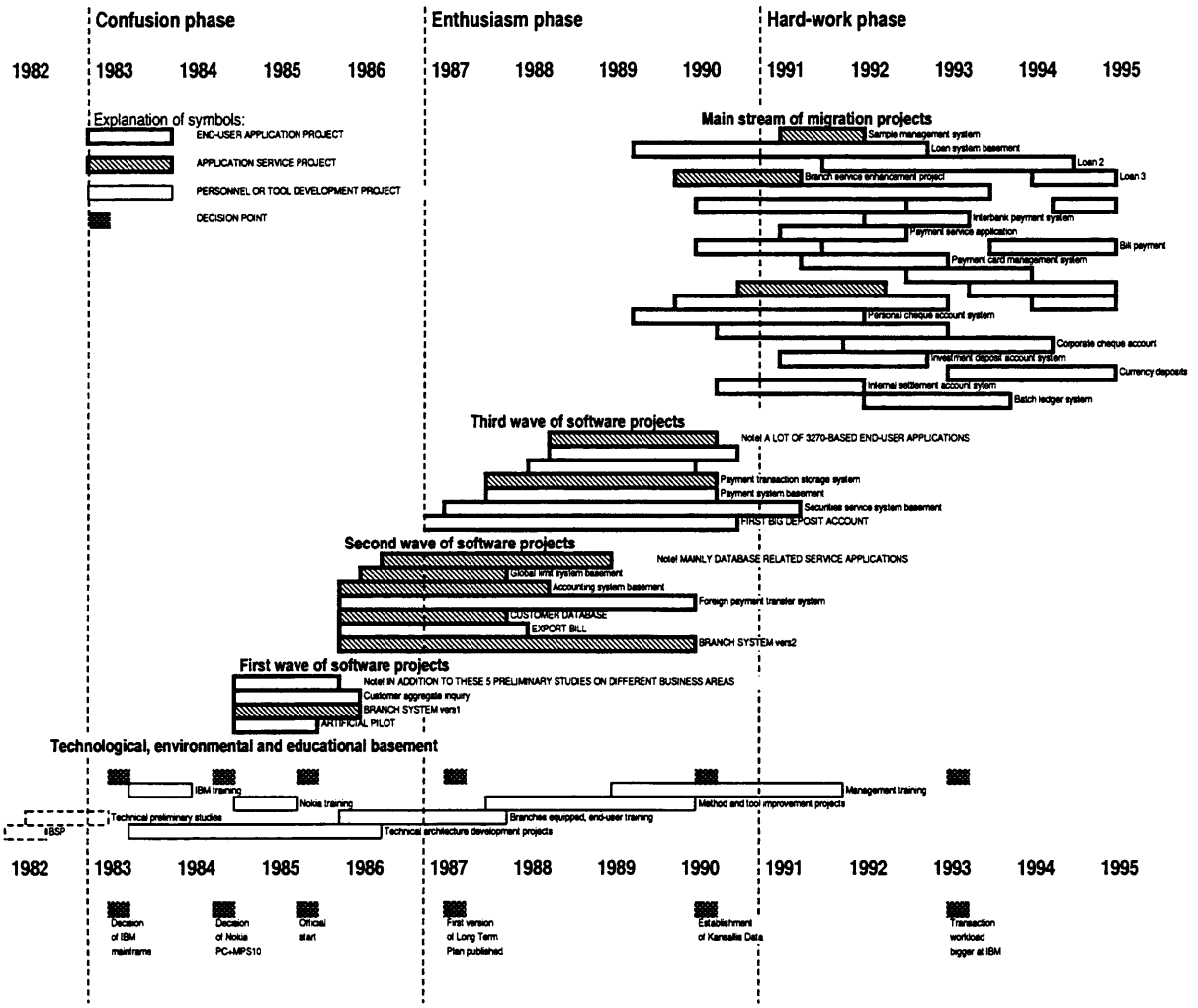

Figure 2: Schedule of the Kansallis Migration Projects

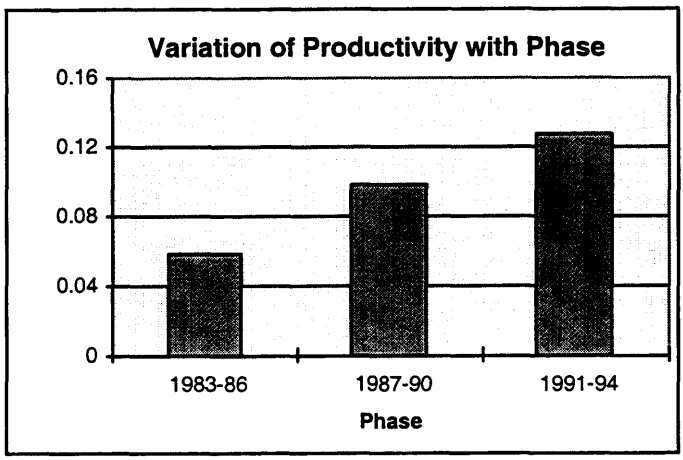

Figure 3: Variation in productivity with phase

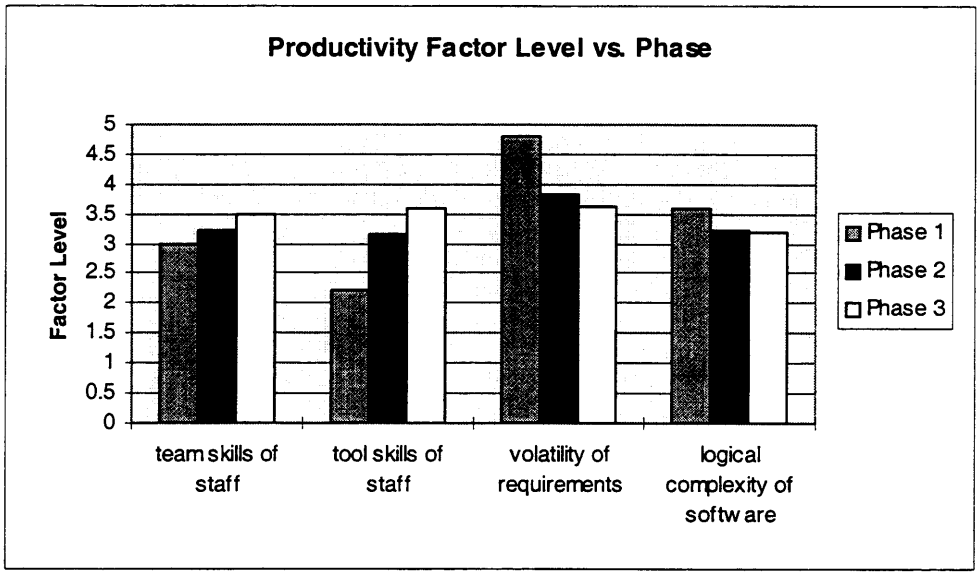

Figure 4: Productivity factor variations with phase

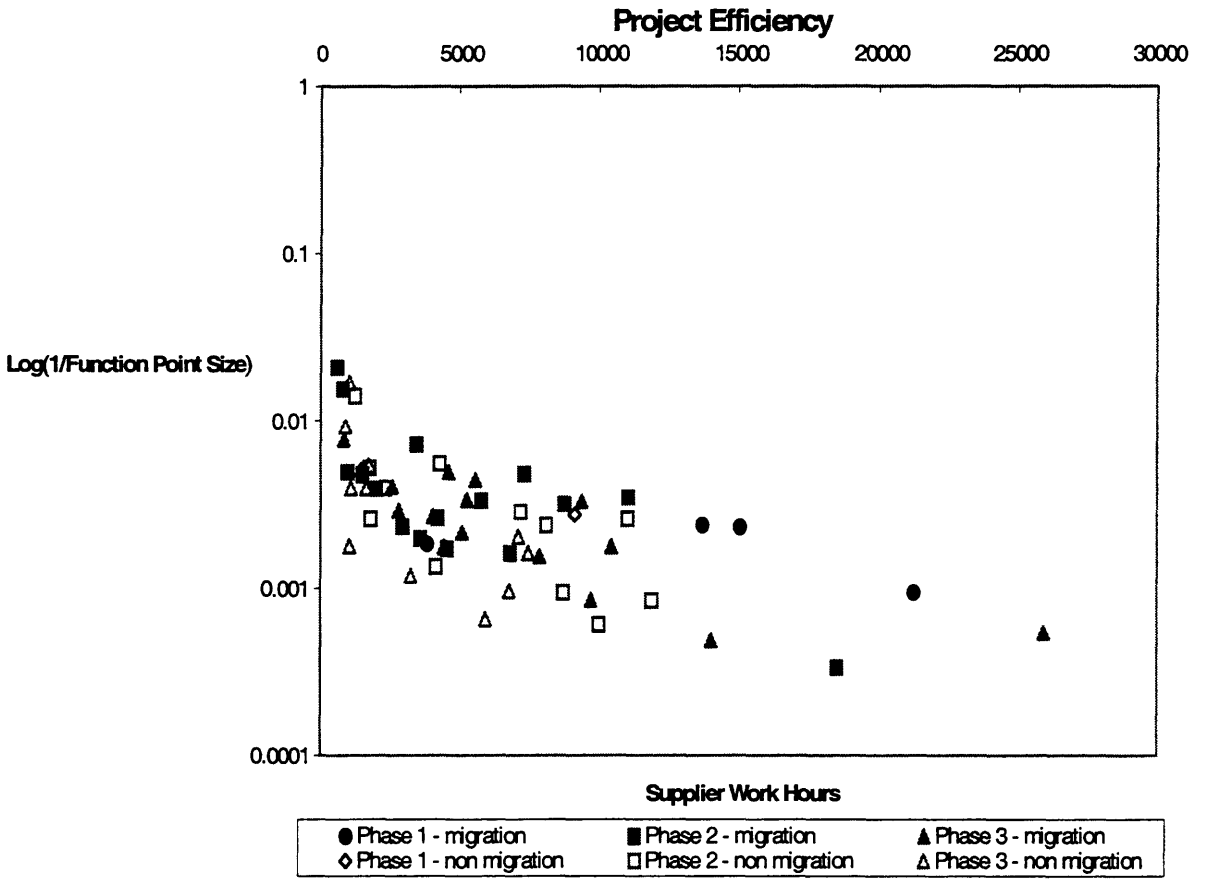

Figure 5: Project Efficiency over Phase

Significant Class Variables

CLASS VARIABLES	NOBS	VARIANCE EXPLAINED	MODEL (significance)
User Interface	62	21%	Add(0.0002)
Length	62	16.3%	Log(0.0051)
Case Tool Type	62	11.8%	Add(0.0623)
Phase	62	10.2%	Log(0.04)
Centralization of Software	62	7.4%	Add(0.1)
Migration Project	62	7.2%	Add(0.003)
Intended Market	62	5.6%	Add(0.0062)

Table 1: Summary of analysis of influence of individual class variables

NON-CLASS VARIABLES	NOBS	VARIANCE EXPLAINED	MODEL (significance)
Hours worked - supplier	62	15%	Log(.0016)
Duration	62	15%	Log(.0011)
Requirements Volatility	62	13.6%	Log(.0080)
Staff - tool skills	62	8.2%	Log(.0099)
Quality Requirements	62	7%	Log(.0428)
Efficiency Requirements	62	6%	Log(.0468)
Use of Standards	62	5.3%	Add(.0133)
No. very easy functions	62	4.2%	Add(.0037)
No. output functions	62	3.4%	Add(.0201)
No. of users	62	2.4%	Add(.0325)

Table 2: Summary of analysis of influence of individual non-class variables

VARIABLES	NOBS	VARIANCE EXPLAINED	MODEL (significance)
User Interface * Length	62	26.7%	Add(.0004)
User Interface * Phase	62	25.6%	Add(.0018)
Centralization of Software * Length	62	24.7%	Add(.0249)
Length * Phase	62	24.0%	Add(.0303)

Table 3: Best two class variables productivity models

VARIABLES	NOBS	VARIANCE EXPLAINED	MODEL (significance)
Phase * Length, No. of entities, Level and usage of standards	62	47.1%	Add(.0001)
User Interface * Length, No. easy functions, Efficiency Requirements	62	46.9%	Add(.0001)
User Interface * Length, No. easy functions, Quality Requirements	62	46.5%	Add(.0001)
Phase * Length, Level and usage of standards, Sum of outputs	62	45.84%	Add(.0001)

Table 4: Best Mixed Productivity Models - 2 class with 2 non class variables

VARIABLES	NOBS	VARIANCE EXPLAINED	MODEL (significance)
Phase * Length, Number of Easy Functions, Quality Requirements, Level and usage of standards, Number of Outputs*	62	50.56%	Add(.0001)

Table 5: Best Combined Phase * Length Productivity Model

Phase	Percent of projects with software		
	in client	in server	combined
1	0%	60%	40%
2	3%	67%	30%
3	11%	43%	46%

Table 6: Variation in centralization of database over phase

(Adams 1972; Swanson 1974; Brooks 1975; Aaron 1976; Watson 1977; Putnam 1978; Albrecht 1979; Belady 1979; Baily and Basili 1981; Basili V.R. 1981; Boehm 1981; Brooks 1981; Lawrence 1981; Albrecht 1983; Behrens 1983; Thadhani 1984; Vosburgh 1984; DeMarco 1985; Conte 1986; Rook 1986; Boehm 1987; DeMarco and Lister 1987; Kemerer 1987; Gilb 1988; Humphrey 1989; Institute 1989; Nath 1989; Andersen 1990; Cusumano 1990; Pfleeger 1990; Banker R.D. 1991; Basili 1991; Jarvenpaa and Ives 1991; Jones 1991; Van Genuchten 1991; Daskalantonakis 1992; Heemstra 1992; Kitchenham 1992; Lederer 1992; Mukhopadhyay T 1992; Mukhopadhyay T. 1992; Putnam 1992; Dorling 1993; Gibbs 1994; Jones 1994; Marciniak 1994; Martin 1994; Matson 1994; Nevalainen 1994; Thomson 1994; Paulk 1995; Maxwell, Van Wassenhove et al. 1996; Parnas 1997; Pressman 1997)

References and Related Publications

Aaron, J. D. (1976). Estimating Resources for Large Programming Systems, Litton Education Publishing, Inc.

Adams, W. (1972). "New Role for Top Management in Computer Applications." Financial Executive(April): 54-56.

Albrecht, A. J. (1979). Measuring Application Development Productivity. *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*, Monterey, CA.

Albrecht, A. J. a. G., J.E. Jr. (1983). "Software Function, Source Lines of Code and development Effort Prediction: A Software Science Validation." IEEE Transactions on Software Engineering **SE-9**(November): 639-648.

Andersen, O. (1990). "The use of software engineering data in support of project management." *Software Engineering Journal* **5**(6): 250-56.

Baily, J. W. and V. R. Basili (1981). A Meta-Model for Software Development Resource Expenditures. *Proceedings of the Fifth International Conference on Software Engineering*, San Diego, CA.

Banker R.D., D. S. M., Kemerer C.F. (1991). "A Model to evaluate variables Impacting the Productivity of Software Maintenance Projects." Management Science **37**(1): 1-18.

Basili V.R., F. K. (1981). "Programming Measurement and Estimation in the Software Engineering Laboratory." The Journal of Systems and software **2**: 47-57.

Basili, V. a. M., J (1991). "The Future Engineering of Software: A Management Perspective." Computer **24**(9): 90-96.

Behrens, C. A. (1983). "Measuring the Productivity of Computer Systems Development Activities with Function Points." *IEEE Transactions on Software Engineering* **SE-9**(6): 648-652.

Belady, L. A. a. M. M. L. (1979). The Characteristics of Large Systems. Cambridge, MIT Press.

Boehm, B. (1981). Software Engineering Economics, Prentice-Hall, Inc., Englewood Cliffs.

Boehm, B. W. (1987). "Improving software productivity." *Computer* **September**: 43-57.

Brooks, F. P. (1975). The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Publishing Company.

Brooks, W. D. (1981). "Software Technology Payoff: Some Statistical Evidence." *The Journal of Systems and Software* **2**: 3-19.

Conte, S. D., H.E. Dunsmore and V.Y. Shen (1986). *Software Engineering Metrics and Models*. Menlo Park, California, The Benjamin/Cummings Publishing Company, Inc.

Cusumano, M. A. a. K., C. F (1990). "A Quantitative Analysis of US and Japanese Practice and Performance in Software Development." Management Science **36**(11).

Daskalantonakis, M. K. (1992). "A Practical View of Software Measurement and Implementation Experiences within Motorola." IEEE Transactions on Software Engineering **18**(11).

DeMarco, T. and T. Lister (1987). Peopleware, Dorset House.

DeMarco, T. a. L., T (1985). Programmer Performance and the Effects of the Workplace. *Proceedings of the 8th Intl. Conf. on Software Engineering.*

Dorling, A. (1993). "SPICE: Software Process Improvement and Capability dEtermination." Software Quality Journal **2**: 209-224.

Gibbs, W. W. (1994). "Software's Chronic Crisis." Scientific American(November): 72-81.

Gilb, T. (1988). Principles of Software Engineering Management, Addison-Wesley Publishing Company.

Heemstra, F. J. (1992). "Software cost estimation." Information and software Technology **34**(10): 627-633.

Humphrey, W. S. (1989). Managing the Software Process. Reading, MA, Addison-Wesley.

Institute, S. (1989). SAS/STAT User's Guide, Version 6. Cary, NC, SAS Institute Inc.

Jarvenpaa, S. L. and B. Ives (1991). "Executive Involvement and Participation in the Management of Information Technology." MIS Quarterly **15:2**(June): 205-227.

Jones, C. (1991). Applied Software Measurement: Assuring Productivity and Quality. New York, N.Y, McGraw-Hill, Inc.,.

Jones, C. (1994). "Globalization of Software Supply and Demand." IEEE Software.

Kemerer, C. F. (1987). "An Empirical Validation of Software Cost Estimation Models." Communications of the ACM **30**(5): 416-429.

Kitchenham, B. A. (1992). "Empirical studies of assumptions that underlie software cost estimation models." Information and software technology **34**(4): 211-218.

Lawrence, M. J. (1981). "Programming Methodology, Organizational Environment, and Programming Productivity." *The Journal of Systems and Software* **2**: 257-269.

Lederer, A. L., Prasad J. (1992). "Nine Management Guidelines for better cost Estimating." Communications of the ACM **33**(2): 51-59.

Marciniak, J. J. (1994). Encyclopedia of Software Engineering, John Wiley & Sons Inc. new York.

Martin, R. L. (1994). "How to Solve the Management Crisis." IEEE Software **14**.

Matson, J. E., Barrett B.E., Mellichamp J.M (1994). "Software Development Cost Estimation Using Function Points." IEEE Transactions on Software Engineering **20**(4): 275-287.

Maxwell, K. D., L. Van Wassenhove, et al. (1996). "Software Development Productivity of European Space, Military and Industrial Applications." IEEE Transactions on Software Engineering **22**(10): 706-18.

Mukhopadhyay T, V. S. S., Prietula M.J. (1992). "Examining the feasibility of a case based reasoning Model for Software Effort Estimation." MIS Quarterly(June): 155-171.

Mukhopadhyay T., K. S. (1992). "Software Effort Models for early estimation of Process Control Applications." IEEE Transactions on Software Engineering **18**(10): 915-924.

Nath, R. (1989). "Aligning MIS with the Business Goals." Information and Management **16**: 71-79.

Nevalainen, R. a. M., H (1994). *Laturi System - General Description*, Information Technology Development Center.

Parnas, D. L. (1997). Software Engineering: An Unconsummated Marriage. Communications of the ACM. **40:** 128.

Paulk, M. C. (1995). "The Evolution of the SEI's Capability Maturity Model for Software." Software Process-Improvement and Practice **Pilot Issue**: 3-15.

Pfleeger, S. L. a. M. C. (1990). "Software Metrics in the Process Maturity Framework." Journal of Systems Software **12**(3): 255-262.

Pressman, R. S. (1997). Software Engineering: A Practitioner's Approach. San Francisco, CA, McGraw-Hill.

Putnam, L. H. (1978). "A general Empirical Solution to the Macro Software sizing and estimating problem." IEEE Transactions on Software Engineering **SE-4**(4): 345-361.

Putnam, L. H. a. W. M. (1992). *Measures for Excellence: Reliable Software on Time, within Budget*. Englewood Cliffs, N.J, P.T.R. Prentice Hall, Inc.

Rook, P. (1986). "Controlling software projects." *Software Engineering Journal* **1**(January): 7-16.

Swanson, E. B. (1974). "Management Information Systems: Appreciation and Involvement." Management Science **21:2**(October): 178-188.

Thadhani, A. J. (1984). "Factors Affecting Programmer Productivity during Application Development." *IBM Systems Journal* **23**(1): 19-35.

Thomson, H. E. a. M., P (1994). "The Software Process: A Perspective on Improvement." The Computer Journal **37**(8).

Van Genuchten, M. (1991). "Why is software late? An empirical study of reasons for delay in software development." *IEEE Transactions on Software Engineering* **6**: 582-590.

Vosburgh, B. C., R. Wolverton, B. Albers, H. Malec, S. Hoben and Y. Liu (1984). Productivity Factors and Programming Environments. , *Proceedings of the Seventh International Conference on Software Engineering*.

Watson, C. E. a. C. P. F. (1977). "A method of programming measurement and estimation." IBM Systems Journal **16**(1): 54-73.

APPENDIX

Table A1: Variables in the Kansallis Database

name - project name
worksup - number of hours worked by supplier
workcus - number of hours worked by customer
duration - duration in months
yk - company code **(always 500500)**
bra - basic attributes organization (banking, retail, insurance, etc...) **(always banking)**
app - basic attributes application (customer service, mis, accounting, etc...)
dev - basic attributes development cycle (development/maintenance/other) **(always development)**
har - hardware (networked, mainframe, pc, etc...) target platform
acq - basic attributes development model (waterfall, prototyping, mock-up, etc...)
dba - dbms architecture (hierarchical, relational, network,...) target platform
ifc - user interface (graphical, character) target platform
cdb - centralization of database (fully in client, fully in server, combined)
csw - centralization of software (fully in client, fully in server, combined)
cpr - centralization of user interface (fully in client, fully in server, combined)
dbt1-2 - dbms tools (dbase, focus, oracle,...)
cas1 - case tools (lew/adw, oracle, case...)
toy1-3 - other tools (debugger, wp/word, artemis,...)
lan1,2,3,4 - development languages used
met1-9 - techniques and methods used (conceptual analysis, cost/benefit analysis....)
ope1-3 - operating system (dos, unix, vms,...) target platform, **not in order of importance**
mark - basic attributes intended market (internal/outsourced)
cust - intended number of customers (two ranges 1 or 2-5)
user - intended number of users (4 ranges)
accu - accuracy of effort collection (1 daily collection, 2 weekly, 3 derived afterwards from price)
migrat - (Y migration project, N not migration project)

p1-15 - productivity factors (5 possible values)

p1 - commitment of customer
p2 - capacity of development environment
p3 - availability of staff
p4 - level and usage of standardization
p5 - level and usage of methods
p6 - level and usage of tools
p7 - logical complexity of software
p8 - volatility of requirements
p9 - quality requirements of software
p10 - efficiency requirements of software
p11 - installation/platform requirements
p12 - analysis skills of staff
p13 - application area knowledge of staff
p14 - tool skills of staff
p15 - team skills of staff

function point breakdown variables

inp1 - number of input functions : very easy
inp2 - number of input functions : easy
inp3 - number of input functions : typical
inp4 - number of input functions : complex
inp5 - number of input functions : very complex
inq1-5 - number of inquiry functions (same range as inp1-5)
out1-5 - number of output functions (same range as inp1-5)

int1-5 - number of interfaces (same range as inp1-5)
fil1-5 - number of entities (same range as inp1-5)
las1-5 - number of calculation functions (same range as inp1-5)

breakdown of supplier and customer effort and duration by phases

sup1 - effort by supplier during preliminary study phase
sup2 - effort by supplier during specification phase
sup3 - effort by supplier during design phase
sup4 - effort by supplier during coding/unit testing phase
sup5 - effort by supplier during installation/testing phase
cus1-5 - effort by customer during each phase
dur1-5 - duration of each phase

Table A2: Classification of Kansallis database variables into SPICE categories

Variables classified into Customer-Supplier Relations

p1 - commitment of customer
cust - intended number of customers (two ranges 1 or 2-5)

Variables classified into Engineering Issues

dba - dbms architecture
har - hardware (networked, mainframe, pc, etc...) target platform
dba - dbms architecture (hierarchical, relational, network,...) target platform
ifc - user interface (graphical, character) target platform
cdb - centralization of database (fully in client, fully in server, combined)
csw - centralization of software (fully in client, fully in server, combined)
cpr - centralization of user interface (fully in client, fully in server, combined)
cas1 - case tools (lew/adw, oracle, case...)
p2 - capacity of development environment
p6 - level and usage of tools
p7 - logical complexity of software
p8 - volatility of requirements
p9 - quality requirements of software
p10 - efficiency requirements of software
p11 - installation/platform requirements
mlan - main language
sumsup - sum of hours worked in each phase by supplier
sumcus - sum of hours worked in each phase by customer
sumdur - sum of durations of each phase
nlan - number of different languages used
ndbt - number of different database tools used
ntoy - number of different "other tools" used
nope - number of different operating systems used
fpsize - 5 different sizes of project

very small	<200 fp	fpsize=1
small	200<=fp<400	fpsize=2
medium	400<=fp<1000	fpsize=3
big	1000<=fp<1800	fpsize=4
very big	>=1800	fpsize=5

inp - sum of inputs
inq - sum of inquiries
out - sum of outputs
int - sum of interfaces
fil - sum of entities
las - sum of calculation functions
sumfxn - raw sum of inp, inq, out, int, fil, las
wsumfxn - weighted function points
veasy - number of very easy functions
easy - number of easy functions
typical - number of typical functions
complex - number of complex functions
vcomplex - number of very complex functions

Variables classified into Project Management Issues

nmet - number of different methods used
mark - basic attributes intended market (internal/outsourced)
p4 - level and usage of standardization
p5 - level and usage of methods
phase - 3 different eras

1983 - 1986 confusion era	phase = 1	
1987 - 1990 enthusiasm era	phase = 2	
1991 - 1994 hard work era	phase = 3	

length - 3 different lengths of duration

short project	< 12 months	length=1
medium project	12-18 months	length=2
long project	> 18 months	length=3

Variables classified into Support Issues

none

Variables classified into Organizational Issues

p3 - availability of staff
p12 - analysis skills of staff
p13 - application area knowledge of staff
p14 - tool skills of staff
p15 - team skills of staff

[a] All processing was done in batches with no possibility of real-time response. For example, branches would write out transactions on paper and these paper records would be collected at the end of the day and sent to the central data centre for processing (in batches) during the night.

[b] With the on-line banking systems, it became possible to access data and process transactions in real time. For example, branches could directly check customer account data on their terminals and update the information as needed.

[c] The multi-functional system is an advanced version of the on-line banking systems with a number of new features including: (a) an increase in the use of PCs coupled with a decrease in the dependence on the mainframe; (b) greater integration of banking systems with office automation functions on the PCs; and (c) advanced telecommunications such as network connections to all parts of the organisation including foreign branches.

[d] Based on ongoing research by the authors within BCP.

[e] Air-conrollers Ancient Computers Invite Disaster, Editorial, USA Today, Wednesday, December 24, 1997.

[f] Of course the term "client server" was not in common use at that time, but the core concept of client PCs hooked to server mainframes/minicomputers was contained within the design of the MFS system architecture.

[g] Note that the IBM mainframe environment chosen for the MFS project was different from the BULL mainframe environment used in the second general on-line system.

[h] A function point is a measure of the functionality of a program. Function points are frequently used to measure the complexity and size of business software projects. The median productivity in banking and insurance projects in Finland was 10 FP/Mmonth and 13 FP/Mmonth respectively in 1993.

[i] Major changes in the functionality of systems were typically not welcomed for migration projects. The ideal situation was to port the same applications to the new platform with minimal changes in the functionality of the system.

[j] The goal was to have a true client server architecture in which a central computer (server) acted as a store of programs and the different networked computers (the clients) accessed programs from the server and processed them locally with their own processing powers. In this intermediate stage, the programs were both stored and run on the central mainframe computer. The local PCs simply acted as terminal for transmitting the input commands to the central computer.

[k] For the 2-class models, a is a function of the crossed effects of two class variables.

[l] These cut-off values were defined by project managers within Kansallis Bank.

[m] Pekka Forselius and his colleagues estimate informally that if they had to redo the Kansallis systems migration project, they could have probably done it in less than four years.