Post-Implementation Management in Large-Scale Management Information Systems: Changes, Incident Reports, Help Desk Calls, and Business Process

H. Keith Edwards
Department of Computer Science
University of Hawaii-Hilo
200 W. Kawili Street, Hilo, HI 96720, U.S.A.
hedwards@hawaii.edu

ABSTRACT

Proper management of the post-implementation phase of the software development life cycle is necessary in order to achieve proper strategic alignment between the technical operation of the system and organizational strategy. For this reason, it is important to understand the relationship between technical changes to the system, business process changes, process support, and the actual use of the system. This study examines the relationships among these aspects of system maintenance and process support for a key information system at a large, industrial Fortune 500 company. In this case study, we see strong relationships between system use, business process, and the technical maintenance of the system, with the business process acting as the driving force for many of these system-support activities. This case study provides empirical validation for strategic alignment philosophies and models, such as Leavitt’s Model, during the maintenance phase of the software development life cycle.

Keywords: Management information systems, configuration management, implementation issues, maintenance

1. INTRODUCTION

The post-implementation phase of the software development life cycle is one of the most important aspects of software engineering [31, 40]. The proper maintenance and change management of implemented systems are critical for organizational success. In fact, continual change and system evolution are necessary for the system to maintain continual usefulness. In spite of the importance of the maintenance phase, this is one of the least studied areas of the software development life cycle. As a result, there are very few case studies that examine the nature of the post-implementation phase for bespoke systems in industrial systems.

Volume 3, Number 2, December 2008
This research investigates system metrics, process support activities, and business process changes for the post-implementation phase of an inventory requisitioning system at a Fortune 500 company. In particular, this research explores the relationships among change requests, help desk reports, system incident fixes, system utilization, and the business process in order to begin to understand the interaction between these factors in the operation and maintenance of large scale information systems.

The rest of this paper is organized as follows. Section 2 examines research related to the areas of implemented systems, configuration management strategies and support. Section 2 also discusses information related to existing studies, many of which examine the role of problem reports and incident fixes for open-source, off-the-shelf software systems such as Mozilla and Eclipse. Section 3 provides background on the system itself, the change control process used by the company, and the various data sets comprising the artifacts from the maintenance phase of the software development life cycle. Section 4 provides details on the analysis of the data sets and discussion on the analysis methods, and Section 5 provides conclusions and presents possibilities for future work.

2. RELATED WORK: SOFTWARE MAINTENANCE, CONFIGURATION MANAGEMENT AND CHANGE

Implementation and program maintenance are the final stage of the software development life cycle [21, 39]. Sommerville [39] defines the implementation stage of the software development life cycle as “the process of converting a system specification into an executable system.” The word process in this definition shows that this is not merely a destination, but an on-going process that involves the continual modification of the system to meet the requirements of its operating environment.

The need for applying changes to implemented software has been evident since the development of the first systems. Early research in software engineering shows the need for a disciplined and rigorous process to control changes that are incorporated into the final product [1, 2]. Later work, such as Joeris [16], looks at how to provide basic management function for both change management and configuration management within the implementation phase of the software development life cycle. Other work, such as Davis [7], takes a broader view by examining the role that the overall development life cycle plays in the software configuration management process in the post-waterfall model era.

An underlying consideration in any view of the change management process is that large-scale information systems constitute socio-technical systems that interleave people, process, and technology [40]. Leavitt’s model (Figure 1) is perhaps the seminal piece of related work in the arena of management information systems for understanding the role of the implemented system within the overall organization [22]. Leavitt’s model examines the relationship between business process, change, management processes, and information systems. The
model hypothesizes that changes to any one aspect of the model will cause changes in the other areas of the model so that the organization can maintain its overall strategic alignment. Since this research examines changes to the system resulting from changes to both the business process and the underlying technology, we can use Leavitt’s model as a tool for understanding the relationships among several of the aspects of the implemented system.

![Figure 1. Leavitt’s Model](image-url)

2.1. Change Control Process and Configuration Management Strategies

Several articles in the related literature point to the need for, and examine, the nature of change control and configuration management processes that form a critical part of the implementation phase of the software development life cycle [3, 43]. For example, Joeris [16] provides a summary of desirable characteristics for a change management process, such as managing the process of change, as well as managing the artifacts of the change process. Chillakanti [6] motivates the need for incorporating security into the change management process. Sato, et al. [33] discuss Hewlett Packard’s change management process and its impact on the customer experience, but focuses on off-the-shelf products as opposed to bespoke systems. Nguyen [27, 28, 29] examines the software configuration environment in object-oriented and Web-based systems, but treats the more general subject matter than a specific system. Other work such as [35] looks at configuration control for evolutionary systems, whereas Simmonds [36] examines the configuration management process on the PACT software engineering system.

All of the work here points to the need to modify the change management in relation to the current system under consideration. Although general work such as [31, 40] can provide a starting point for the discussion, it is generally necessary to have a change control process that closely aligns with the needs of the sponsoring organization.
Although the need for a disciplined change management process cannot be understated, the purpose of this research is not to supplant any of these existing processes nor to examine the particular change control strategy in isolation from the rest of the system maintenance artifacts. Rather, this research focuses on the change control process as it relates to the business process and to the implementation of change.

2.2. Support for Configuration Management

Calabough [5] points to the need for change and configuration management support and develops a tool along with a process to these ends. There is a novel approach here in terms of developing a configuration management system for naval tactical systems and with its use of expert systems technology. Here, the focus is on the control of the configuration management process from managerial and technical standpoints, rather than on examining the results of the maintenance phase of the project.

Several articles [13, 23, 24, 34, 41] examine the configuration management process in order to manage component versions in the build. For example, Thomas [41] looks at a tool for control of the object management process, whereas Mei, et al. [24] look at a model for improving component-based software development through software configuration management. Schuster, et al. [34] look at support for configuration management in large-scale work-flow systems.

Again, these treat the post-implementation phase of the software development life cycle from a developer-oriented point of view. Although the correct version of individual components is an important consideration in deploying an effective version of the system, it is also exceedingly important to view the process from end-user and organizational standpoints.

Feiler [13] examines how process support can assist in the implementation of configuration management tools, particularly through separation of mechanisms and policy. A general treatment of support for configuration management can be found in Estublier [10, 11, 12] and in German, et al. [15], who provide a framework for describing tools for the mining of software repositories [16].

Tools such as HATARI [38] can be effectively used to implement and manage changes to the configuration of software products. In particular, HATARI looks to minimize the risk of fix-inducing changes such as those discussed in [44] to the software through the detection of locations where changes have been historically risky within the software. HATARI has been integrated into the Eclipse Development Environment.
2.3. Configuration Management Case Studies

A final area of research is that of configuration management case studies. These are sparse, but will most likely differ from our research in the scale of the effort, the amount of data examined, the granularity of the examination, and the time over which the system was examined. There is also a focus on open-source systems development and meta-studies that examine numerous projects at a high level view. Our study extends for five years and examines approximately 600 separate change requests, more than 6,000 incident reports, and 8,513 help desk calls. It also looks at a single bespoke system in great detail.

Dietel [8] provides a look at the impact of instituting a change management process into the organization and how this impacts the employees in the organization. The paper evolves the process from “ignorant anarchy” to a more rigorous and disciplined process. This case study examines the evolution of the change control process from a personnel standpoint and provides many insights into the work necessary to achieve a rigorous process that suits the organizational needs.

Sliwerski, et al. [37] provide a study of changes in open-source software such as Mozilla and Eclipse. Their research examined the impact of fix-inducing changes on the system and provided recommendations as to how to minimize later fixes due to these changes. Further work by this group [38] looks at how to relate the software version history to the bug database.

Likewise, Mockus, et al. [26] use data on Mozilla and Apache to understand source code change history, problem reports, and productivity in order to understand the open-source development process. Koponen [19] looks at the open-source maintenance process for these same two products. Later work by Koponen [18, 20] examines an evaluation framework for open-source software projects. This work examines five open-source projects in order to examine problem tracking. The paper looks at project scale, efficiency, and defect management. One of the important findings here is that changes to the software are not necessarily connected to defects, which fact the researcher attributes to a possible lack of management in such projects.

Kim and Whitehead [17] examine the time to fix for bugs in ArgoUML and PostgresSQL, two open-source, off-the-shelf systems. In this research, the researchers look at the number of days required to fix bugs in each of these products and at the average time and distribution of time required for all the bug fixes. The majority of bugs in these programs require between 100 and 300 days to fix, with the median bug fix time being approximately 200 days.

The preceding case studies examine the post-implementation phase of large software systems. However, they tend to deal with open-source, off-the-shelf products designed for the mass market as opposed to bespoke systems that are designed to provide a competitive advantage to a business in a proprietary manner. Understanding the nature of configuration management in open-source systems and the patterns of bugs introduced into the system through fix-inducing
changes are laudable goals. However, an understanding of this portion of the life cycle for bespoke systems helps to paint a more complete picture of the process, since proprietary bespoke systems play a large role in the global economy.

In addition to the open-source studies, there are studies that examine the artifacts from proprietary software projects. For example, Parnin, Gorg, and Rugaber [30] explore the concept of revision history for source code by looking at both the source code revision history and the interaction history for source code created by developers at defense contractors. The revision history explored in this article is similar to Sliwerski, et al. [37], with regard to when changes induce fixes. The authors use an additional data set in terms of the interaction history and look at how the programmers access various parts of the program within the integrated development environment.

A second paper examining bespoke systems [42] looks at the productivity of developers in industrial projects in Japan. In this article, the focus is on calculating developer productivity based on various factors such as outsourcing and the upstream process ratio. The researchers found that limiting the amount of outsourcing and higher upstream process ratios greatly contributed to programmer productivity.

Although these case studies examine multiple data sources and look at proprietary projects as a case study, they examine the software maintenance from a developer standpoint as opposed to the end-user and organizational perspectives.

Bhatt, et al. [4] examine the maintenance process for 127 projects in 27 different organizations. Here, the focus is on the relationship between the software characteristics, maintenance team capability, customer attributes, and organizational climate as they relate to the software maintenance effort. The data in this paper was collected through questionnaires. Though the paper examines numerous qualitative attributes such as leadership, the authors also note that the business criticality of the system has a direct impact on the time spent by the maintenance team. This work takes a more detailed look at this type of factor, but explores the relationships using data from the maintenance phase as opposed to questionnaires.

Earlier research on this project [9] began to establish relationships among system utilization, system changes, and the business process. This work extends [9] by examining the data in greater detail and incorporating additional data sets such as the help desk report as well as the bug fix reports. These aspects of the maintenance phase were not available for the earlier research.
3. DESCRIPTION OF SYSTEM ENVIRONMENT, CHANGE CONTROL PROCESS, DATA SET, AND RESEARCH QUESTIONS

This section describes the various aspects of the case study. In particular, it describes the computer system itself, the change control process used to effect changes within the system and business environment, and the various artifacts from the post-implementation phase of the software development life cycle. This section also discusses the research questions that are to be analyzed in the analysis section.

3.1. System Environment

In this study, we examine the post-implementation history of an inventory and requisitioning system used by a large global manufacturer. The overall system itself is divided into four main modules that represent the core business processes of the company.

The first module is responsible for specification and scheduling of the individual products; i.e., the type of product that will be built, along with when and where it will be built.

The second module allows users to specify a bill of material for individual products; i.e., to indicate which individual parts are used in a particular product. However, many of the parts are based on the specification and are automatically populated, based on the current components given in the engineering database.

The third module supports the ordering of the parts from the bill of material. Here, it is possible for analysts to place orders for components from the bill of material, to apply controls over orders that are already in process, and to track the status of orders and suppliers. There are also facilities to request the order of individual items that support the product, but may not appear on the bill of material.

The fourth module supports warehousing and logistic operations such as receiving, storing, allocating, and shipping the individual parts. This module also includes controls for the site that ultimately builds the product, such as a function to indicate the completion of the product within the scheduling module. Finally, there is a module for administrative functions such as creating users and allowing access to the system.

The system was originally released in 1997 in the United States, United Kingdom, and Germany. It is now deployed in numerous locations within seven countries, and has thousands of individual users. It currently has 325 different screens that support queries, updates, and administrative functions.

The system is supported by full-time staff who are dedicated to representing the users and who are also responsible for explaining and cascading the business process throughout the organization. There is also a systems staff that is responsible for ensuring that the system functions correctly from a technical standpoint. These two groups work together to facilitate a holistic approach to system configuration and maintenance.
3.2. Change Control Process

Since this is a bespoke system that was designed to support the business process, it is not surprising that the system would need to change as the business process continued to evolve. In much the same way, the organization has created the change control and configuration management processes to support the business processes. Of course, there is a separate process designated to deal with fixes to the system that are the result of bugs in the software; this process focuses much more on the technology and system than on the business process.

The change control process begins with an individual user requesting a change to the system. Typically, such a change is proposed in order to augment the system’s ability to support the user’s job or to rectify a situation in which it is unable to do so. Hence, all changes are analyzed by a business process analyst, who determines whether the change is needed to support the core business process of the organization.

If the change is required to support the business process, then the analyst assigns a priority to the change. The business process analyst then meets with representatives from the systems staff. This group analyzes the request to see whether it is actually required (i.e., whether there are workarounds) to support the business process and whether it is technically feasible. As part of the request, the person proposing the change needs to document the savings associated with the change. This is compared with a work estimate from the systems group in order to determine a cost-benefit ratio [21].

If the proposed change meets all of the preceding conditions, the systems group negotiates a target date with the business process group and provides an estimated cost for the change.

The systems group then implements the change in the development system and reports completion of the change to the business process analysts. Typically, the systems group will cascade the change into the testing and training environments to allow for evaluation without impacting the production operations of the organization.

Once the change is available, the business process analysts and possibly the person who proposed the change will test the change. If the change works according to its specification and results in no errors, it is released into the production environment of the system as part of a batch of changes. The systems group releases batches of changes into the production system several times throughout the year to maintain a more stable production environment. At the same time as the release occurs, support functions such as training and the help desk will provide users with new information anpiy the change. Typically, this takes the form of a bulletin within the system. However, it is possible that new training courses may be developed if the extent of the change is large enough.

In many ways, the change control process mirrors generic change control processes such as those discussed in textbooks such as [31, 40]. However, a key difference with this process is the analysis based on business need, cost and
impact on the core business processes and the system itself. A second difference is the support for changes through cascading information through training and the help desk.

3.3. **Description of the Data Sets**

This section examines the four major artifacts from the post-implementation phase of the system. The particular documents under consideration are a summary of change control requests, screen utilization data, an archive of help desk calls, and a summary of bug fixes for the system.

This information provides data on not just the technical aspects of the system, but also on the business process that the system supports. All of these documents relate solely to the post-implementation phase of the project, which makes it markedly different from a traditional Software Configuration Management (SCM) archive such as those described in Pressman [31], which are more likely to include artifacts from the technical development of the system (e.g., design specifications, source code, data models, etc.).

3.3.1. **Change Control Requests**

The first document is a synopsis of the change control requests for the overall system. This synopsis provides detailed information about the 668 change controls that have been applied to the current version of the system, as well as a summary of the implementation status and the prioritization of the change requests.

For the individual change controls, the archive provides a detailed summary of each request. In particular, the information begins with a change request number and a module to which the change request belongs. As previously stated, the system comprises four major areas, and it is possible for the request to belong to more than one area. In such cases, the change is categorized according to the area of the system where it will have the greatest impact. Some changes are categorized as all. Furthermore, the archive provides a description of the change request, along with the date that the change was requested.

There is also a place for the status of the change request. Requests can have a status of *new, evaluation, accepted, or implemented*. Requests can also be placed on hold, rejected, or canceled, based on the business need(s) and the availability of system resources to support the change. Along with the status of the request, each request has a priority. There are four categories of priority that range from 1-ASAP (as soon as possible) to 4-Low, based on business need. It is exceedingly rare for a change request to receive a priority of 4, as it would indicate that it is not needed to support the core business.

Although business needs drive the change control process, the systems group also provides input into the process. In particular, the systems group provides an estimate of the system hours required to complete the request and a target completion date for the request. These are compared with the expected
benefit that the change will bring to the company as part of a cost-benefit ratio analysis [21] and against the actual completion date in order to monitor systems productivity.

Finally, the archive provides check boxes for updating the business process documentation, training materials, and help screens in order to ensure that users of the system receive timely information regarding operational changes to the system.

3.3.2. Screen Utilization Data

The second document in the data set is the access information for the 325 individual screens in the system. This document breaks down the number of hits for each screen by year. It begins in 2000 and runs until 2005, since access information for individual information was not available prior to the year 2000.

This document indicates the screen number, screen name, module, and the number of “hits” for that screen for each of the years that the screen was active. For most of the screens, this is the entire range of six years. However, some new screens were introduced during this period and several low-use screens were phased out during this time and contained data for only 1-2 years.

3.3.3. Help Desk Data

In addition to the change control data and the information on screen utilization, the data gathered from the help desk serves as another artifact from the post-implementation phase of the system. Although not as rich in detail as the change control request archive, the help desk data provides the following information:

- **Incident ID** – A unique ID assigned to the help desk call
- **Problem Detail** – A category indicating the area of the particular problem, generally corresponding to the business area where the problem was encountered, although there are also categories for access and log-in issues
- **Specific Issue** – More detailed information concerning the nature of the call and often a screen number or details on the nature of the problem
- **Date/Time** – The date and time of the call to the help desk

3.3.4. System Incidents Reports

The system incidents report archive is akin to a traditional bug report archive. For the system in this research, there is more than four years’ worth of data ranging from January 18, 2002, to September 29, 2006, comprising 6,329 separate reports.

The data included in this archive is quite a bit richer and more structured than the help desk reports and includes 31 different fields to record information about the problems. This research examines information regarding the following fields:
(1) Date / Time Reported  
(2) Defect Type  
(3) Causal Factor Level 1  
(4) Causal Factor Level 2  
(5) Causal Factor Level 3  
(6) Other Cause  
(7) Date / Time Completed  
(8) Solution Hours  

The choice of these fields allows examination of the system problem report as it relates to the business process changes and system utilization.

### 3.4. Research Questions

Based on the available artifacts from the implementation phase, it is possible to examine the system from the standpoint of business drivers, technical drivers, and process support. These correspond to the technology and tasks section, whereas the help desk data provide some insight into the people aspect of the model. The results presented in the following section discuss these individual drivers, as well as the relationships between them.

In particular, this study examines the relationship between screen utilization and change requests in order to understand the business process (i.e., tasks) within the model. In particular, it focuses on how well the system utilization corresponds with the business process changes introduced to the system.

Next, we examine the corrective maintenance aspects of the system as garnered from the system problem reports to explore the technical aspects of the system and relate these to the business process, such as examining the relationship between incident fixes and system utilization.

Finally, we examine the process support (i.e., people) aspect of the system through the help desk reports and their relationship to the technical and business process aspects of the system.

### 4. RESULTS

This section presents the results of our analysis. It is organized according to the business drivers, the technical drivers, and the process support activities. As might be expected, certain aspects of the system have more data than others.

Table 1 displays the total number of data points for each system artifact. As can be seen, there are a large number of screen accesses in relation to the other data points. Of particular interest is that there are only 15,510 help desk calls, incident reports, and change requests in total, meaning that support was required once for every 1,117.52 requests, on average.
Table 1
Number of Data Points for System Artifacts

<table>
<thead>
<tr>
<th>System Aspect</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen Accesses</td>
<td>17,332,792</td>
</tr>
<tr>
<td>Help Desk Calls</td>
<td>8,513</td>
</tr>
<tr>
<td>System Incident Reports</td>
<td>6,329</td>
</tr>
<tr>
<td>Change Control Requests</td>
<td>668</td>
</tr>
</tbody>
</table>

4.1. Business Drivers – Change Controls and Screen Utilization

This section examines the business drivers for the system, particularly the change control data in relation to the screen utilization data. To more easily analyze the data, we calculated the average of screen use patterns for the entire six-year period.

We placed the screens in order of their average number of accesses per year and examined the use of the top 10% and top quartile (25%) of the screens to determine the percentage of the overall traffic that they receive. The results, shown in Table 2, suggest an empirical validation of the “80-20” rule.

Table 2
Hits for the Most-Used Screens

<table>
<thead>
<tr>
<th>Item</th>
<th>Count</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Screen Use</td>
<td>17,332,792</td>
<td></td>
</tr>
<tr>
<td>Top 10% Screen Use</td>
<td>13,770,566</td>
<td>79.45</td>
</tr>
<tr>
<td>Top 25% Screen Use</td>
<td>16,720,453</td>
<td>96.47</td>
</tr>
</tbody>
</table>

Here, we can see that the top quartile of the screens receives about 96% of the traffic, whereas the top 10% of the screens receives about 80% of the traffic for the entire system. Since the screens in the top 10% and the top quartile were designed to support the core business process, this is an anticipated finding. Furthermore, the screens in the lower quartiles were devoted to exception handling and administrative functions.

Next, we analyzed the change request synopsis to determine which screens had undergone change as part of the evolving business process. We then correlated the number of change requests for individual screens with the screen utilization. We also correlated the type of screen (update, admin, or query) with the number of changes. Table 3 presents the results of correlating the change requests with the screen use (average hits), the module from which the screen originated, the screen type, and the number of change requests.
The results in Table 3 show a positive association (0.53 Pearson Product Moment) between the changes to a screen and the number of hits that it receives as part of the overall set of screens. This is somewhat lower than expected as the screens from module 4, which comprise the majority of the system, underwent the fewest screen specific changes overall, whereas screens from modules 2 and 3 underwent more changes for less of the system traffic (see Table 4).

When broken down by module, three of the four exhibit a high correlation between change requests and screen use. Two of the Pearson Product Moment correlations are at 0.69, while the other is at 0.60, which suggests a moderate relationship between the two factors.

Furthermore, the regression shows significant relationships between screen type and average hits. Update and Inquiry screens experience a far greater number of hits than their administrative counterparts. Furthermore, there is a significant relationship between change requests and the average number of hits
for the screen. Hence, the number of hits and change requests can be shown to be predictors of one another. However, there was no relationship between the number of changes requested and the screen type; i.e., update screens were no more likely to receive change requests than query screens.

Table 5 shows the total screen changes and the number of changes for screens in the top 10%, top quartile, and the other three quartiles, along with their percentage as part of the overall individual screen changes.

Table 5
Screen Changes by Access Amount

<table>
<thead>
<tr>
<th>Description</th>
<th>Number</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Screen Changes</td>
<td>277</td>
<td></td>
</tr>
<tr>
<td>Top 10% Screen Changes</td>
<td>141</td>
<td>50.90</td>
</tr>
<tr>
<td>Top 25% Screen Changes</td>
<td>218</td>
<td>78.70</td>
</tr>
<tr>
<td>Other Three Quartiles</td>
<td>59</td>
<td>21.30</td>
</tr>
</tbody>
</table>

Table 5 shows further evidence supporting the relationship between the average number of hits per screen and the number of changes the screen underwent. In particular, screens that were in the top 10% in terms of access received about 51% of the changes, and the top quartile received about 79% of the changes. The other three quartiles received only 21% of the changes. This result suggests that more widely used screens experience more changes. This phenomenon can be attributed to the fact that the most accessed screens form the core of the business process; this serves as a strong indicator of their continued transformation of the system to match the evolution of the business process.

We also looked at the relationship between screen type, module, and the number of change requests. In particular, we conducted an analysis of variance on the number of change requests based on the module and the screen type as factors. This analysis is shown in Table 6, on the following page.

Table 6 shows that the screen type had a highly statistically significant impact on the number of change requests and presents further evidence that screens assist the core business process are more likely to undergo changes.
4.2. Technical Drivers -- Corrective Maintenance and Incident Reports

In section 4.1, we examined artifacts from the screen utilization data and the change control requests. These data relate to the perfective maintenance and business aspects of the system. In addition to the business process, there are also technical aspects to any organization, as indicated in Leavitt’s model (see Figure 1 earlier). These technical aspects of the maintenance process are most aptly characterized by the systems metrics repository report, a.k.a. the incident report. This section examines the incident reports as well as their relationship to the business process.

The Metrics Repository System Reports contain 6,329 separate reports on technical incidents with the system covering a period from January 2002 until September 2006. Table 7 shows some of the common elements mentioned in incident reports.

Table 7 shows that only 710 of the 6,329 (about 11.2%) incident reports mention specific screens. The most frequent concerns were parts and orders, particularly when these did not populate within the anticipated table. To a lesser extent, shippers for parts created incidents.

Table 7 also shows that reports and printer issues also accounted for more incidents than individual screens. There were a smaller number of incidents for access to the system, individual pieces of data, shippers, installation, how-to questions, and the interface. With the exception of shippers, all of these smaller issues relate to the areas handled by the process support functions of the system.
organization, so it is not surprising that they constitute a smaller number of the issues; there were dedicated business units to support these aspects of the system.

Table 7
Common Elements in Incident Reports

<table>
<thead>
<tr>
<th>Issue</th>
<th>Number of Incidents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parts</td>
<td>1,387</td>
</tr>
<tr>
<td>Order</td>
<td>914</td>
</tr>
<tr>
<td>Printer</td>
<td>778</td>
</tr>
<tr>
<td>Reports</td>
<td>727</td>
</tr>
<tr>
<td>Screens</td>
<td>710</td>
</tr>
<tr>
<td>Access</td>
<td>251</td>
</tr>
<tr>
<td>Data</td>
<td>227</td>
</tr>
<tr>
<td>Shipper</td>
<td>198</td>
</tr>
<tr>
<td>Installation</td>
<td>130</td>
</tr>
<tr>
<td>How To</td>
<td>108</td>
</tr>
<tr>
<td>Interface</td>
<td>30</td>
</tr>
</tbody>
</table>

In examining the relationship between screen use and system incident reports, there is a similar pattern to the screen use and change requests (see Table 5). Table 8 shows that the top 10% of the screens in terms of access were responsible for about 68% of the incident reports involving screen. Likewise, the top 25% of screens in terms of access received 86% of the incidents. This is an even stronger relationship than the one of screen usage and change requests.

Table 8
Percentage of Total Incidents by Screen Access

<table>
<thead>
<tr>
<th>Screen Criteria</th>
<th>Total % Incidents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 10% Screens</td>
<td>67.75</td>
</tr>
<tr>
<td>Top 25% Screens</td>
<td>86.62</td>
</tr>
</tbody>
</table>

This stronger relationship is born out in the Pearson Product Moment correlation between access and changes value of 0.656, which is a more positive association than that in Table 3. Hence, more widely used screens are more likely to generate incident reports as opposed to the lesser used screens. Again this relates to the fact that the most widely used screens correspond most directly to the business process. We also see a correlation of 0.487 between changes and incident reports.
This section also examined the amount of time spent on corrective maintenance (incidents) activities, compared with adaptive maintenance (change control). The results here are not as positive as the previous results.

Table 9 shows the average time spent on corrective maintenance as opposed to adaptive maintenance. Here, we can see that there is a statistically significant difference between the time required to fix a technical problem with the system as opposed to the time required to adapt the system to a new business need at the 99% level of confidence.

<table>
<thead>
<tr>
<th>Time on Maintenance</th>
<th>Incidents</th>
<th>Change Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>-0.43</td>
<td>4.00</td>
</tr>
<tr>
<td>Max</td>
<td>8,354.68</td>
<td>2,160.00</td>
</tr>
<tr>
<td>Average</td>
<td>250.06</td>
<td>128.23</td>
</tr>
<tr>
<td>Std. Dev</td>
<td>625.95</td>
<td>254.74</td>
</tr>
<tr>
<td>99% Alpha</td>
<td>20.27</td>
<td>35.33</td>
</tr>
<tr>
<td>Low Conf</td>
<td>229.79</td>
<td>92.90</td>
</tr>
<tr>
<td>High Conf</td>
<td>270.33</td>
<td>163.55</td>
</tr>
</tbody>
</table>

The distribution of time for these two separate types of maintenance is slightly different, as shown in figures 2 and 3.

Figure 2. Distribution of Time Required for Corrective Maintenance
In Figure 2, we can see that the vast majority of corrective maintenance takes place in under two weeks, although there is a heavy tail in this distribution with some tasks (i.e., outliers) taking longer than a year. Another trend in the corrective maintenance is shown in Figure 3.

Figure 3 shows the amount of time required for the rectification of individual incidents as a function of the system time line. This scatter plot shows that there are a larger number of outliers occurring as the system matures. This may be due to the fact that incidents have remained open for longer or may point to structural decay within the system; further investigation is required in order to extrapolate further.

Figure 4 shows the distribution of time required for Adaptive Maintenance. In this figure, we can see that the distribution is skewed slightly to the right, but there is not the same degree of outliers in the distribution. Instead, the systems group was likely to indicate that the amount of time to implement the change was “lots” or “unknown.” Unfortunately, it is impossible to extrapolate as to the meaning of these terms.
There are several possible reasons for the differences between the time spent on corrective maintenance compared with the time spent on adaptive maintenance. First is the fact that there is a controlled process for change requests that allows for negotiation between systems and the business process group. This controlled process also allows time for preliminary analysis and design as to how to implement the change as well as requiring a cost-benefit analysis before proceeding with the change. There is also no formal prioritization process for the changes. Finally, the calculation for these hours may be derived using a different methodology than the one used for change control. As a minimal recommendation, there should be increased discussion on incident reports as they relate to the business process in order to establish a more controlled process that is properly aligned with the core objectives of the organization.

4.3. Process Support – Help Desk Data
The final set of data is that for the process support functions as characterized by the help desk report. These data relate to the process support in terms of accomplishing business related tasks with the system. In terms of its position within the overall implementation environment, the help desk is situated as an entry point for the change control and incident reporting processes. Theoretically, incident reports and change requests should be screened by the help desk and then sent downstream to the appropriate venue. The help desk for this system is fairly busy with 8,513 calls over a period ranging from November 2005 to October 2006. The average inter-arrival time between calls was 15.88 minutes.
This figure excludes data points that had a difference of more than 8 hours to avoid incorporating non-working time into the equation.

As with the incident reports, very few of the calls (1,516/8,513, or about 17.8%) involved individual screens within the system. Table 10 shows that the most frequent concerns were with orders and screens. There are also a large number of calls in the quick-tick category for which the help desk provided a quick answer. These quick ticks are often calls requesting information on access to the system, directions to appropriate screens for business tasks, or interface issues.

### Table 10
**Common Elements in Help Desk Calls**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Help Desk Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>2,566</td>
</tr>
<tr>
<td>Quick Tick</td>
<td>2,364</td>
</tr>
<tr>
<td>Screens</td>
<td>1,516</td>
</tr>
<tr>
<td>How To</td>
<td>494</td>
</tr>
<tr>
<td>Parts</td>
<td>366</td>
</tr>
<tr>
<td>Access</td>
<td>288</td>
</tr>
<tr>
<td>Printer</td>
<td>80</td>
</tr>
<tr>
<td>Shipper</td>
<td>24</td>
</tr>
<tr>
<td>Data</td>
<td>13</td>
</tr>
<tr>
<td>Reports</td>
<td>8</td>
</tr>
<tr>
<td>Installation</td>
<td>6</td>
</tr>
<tr>
<td>Interface</td>
<td>1</td>
</tr>
</tbody>
</table>

Comparing Table 10 with Table 7 shows an overlap with regard to technical incidents and process support regarding orders and individual screens. This suggests that orders play a central role in accomplishing the business process for the system. Furthermore, the help desk seems to deal with a different range of issues from the systems group, suggesting that the two groups complement one another in their support duties. However, access issues are also important in both areas and their associated artifacts.

Table 11 shows that the top 10% of screens in terms of access received about 68% of the help desk calls relating to specific screens. In addition, the top 25% of screens in terms of access received more than 90% of the help desk calls. This table corresponds almost precisely with Table 5 and Table 8 in terms of the support required for the screens comprising the core business process. Hence, this table suggests the same strategic alignment between the help desk and the core business process that is present in the system incident support and in the change control process.

*International Journal of Business and Information*
We also conducted a Pearson Product Moment correlation between the help desk calls and system use. The results were similar to the correlation between change controls and screen access. In particular, there was a correlation value of 0.539, which suggests a small positive association.

As indicated by the common elements in Table 7 and Table 10, there is not a strong relationship between the incidents handled by the systems group and the help desk. Here, the Pearson Product Moment value was 0.39, which does not indicate a strong relationship between the screens the two groups serviced.

5. CONCLUSIONS AND RECOMMENDATIONS

In this study, we have examined several different artifacts from the post-implementation phase of the software development life cycle in order to understand the interrelationships among the business process for the system, process support, and technical aspects of the system. The analysis of this data shows particularly strong relationships between user access patterns, the change requests, and the business process as characterized by the change requests. These relationships are also present between the technical maintenance of the system and process support activities. In each case, the top quartile of screens (those that make up the core business process) resulted in approximately 90% of the help desk calls, 87% of the system incident reports, and 78% of the change requests. These data suggest a strong alignment between the technical, business, and support aspects of the system with the core business process of the organization as posited by Leavitt’s model.

Although this research deals with a single case study, there are potentially beneficial aspects of this research for software engineering practitioners involved in the post-implementation phase of the software development life cycle. First, it should be possible for organizations to gain insight into the strategic alignment of organization goals with their systems through a diligent evaluation of post-implementation phase artifacts. Second, the data in this case study show that the effort expended on corrective maintenance is significantly different from that expended on adaptive maintenance. One possible explanation for this was the inclusion of discussion and management metrics used in deciding on adaptive maintenance tasks. In terms of more closely aligning the technical maintenance of the system to organization objectives, there should be increased discussion of corrective maintenance with other areas of the organization in order to hash out ideas and evaluate the need for the corrective maintenance.
6. FUTURE WORK

Although this work makes several important contributions to the understanding of the post-implementation phase of the software development life cycle and the relationship of its various components to the strategic alignment of the organization, there are further areas of research that can yield additional insights into this relationship for bespoke systems in industrial environments.

First, additional data sets from large-scale systems that are used in industrial settings would be helpful to understand whether it is possible to generalize the findings in this paper, based on additional data from other systems.

Second, an examination of the data in greater granularity may yield further insight into the nature of the maintenance process. In particular, examination of other fields in the system incident report may yield further insights into the nature of the incidents in the system.

Additionally, a qualitative analysis of the incident descriptions, help desk calls, and change control requests may provide further reinforcement or refinement of the ideas posited in the analysis section.

Finally, an examination of the data from a temporal standpoint may help to yield insights into the stability of the system over a period of time as shown with the distribution of incident fixes. Using a time based analysis would also allow a deeper examination of change controls as they relate to screen accesses.

7. ACKNOWLEDGMENTS

The author gratefully acknowledges the help and support of Dennis O. Holiday, Raymond Lauth, and David Rogers, who were instrumental in explaining the workings of the system and in providing the data for this paper. Further thanks are due to Jim Lawson, who generously provided system use data and to Dr. Errol Yudkos, who provided access to the statistical analysis software.

REFERENCES


International Journal of Business and Information


ABOUT THE AUTHOR

H. Keith Edwards received his Ph.D. from the University of Western Ontario in 2004 and currently serves as an assistant professor of computer science at the University of Hawaii – Hilo. His research interests broadly encompass the areas of software engineering, implementation issues for management information systems, communication in virtual teams, and computer science education.