

Organizational Learning Through the Process of Enhancing Information Systems

Dana Edberg
University of Nevada, Reno
dte@unr.edu

Lorne Olfman
Claremont Graduate University
lorne.olfman@cgu.edu

Abstract

Business publications have embraced the idea of learning as crucial to empowering individuals and helping them manage the surprises inherent in a dynamic environment. While many agree about the importance of learning, there is little research about the practical mechanisms used to transfer knowledge from the individual to the organization. Computerized information systems codify the established procedures for an organization. Changes to those systems could represent new knowledge now shared with an entire organization. The process of software enhancement is an opportunity to view a method of knowledge transfer. This research examined the software enhancement process in fifteen work groups at five organizations. Enhancement requests in the work groups were reviewed to determine their underlying causes. It was found that software enhancement is a manifestation of organizational learning since the learning from individuals is disseminated to other members of the organization when software is enhanced.

1. Introduction

A way for information systems (IS) groups to encourage organizational learning may be through the unglamorous, often ignored and frequently avoided task of software maintenance. An organization is considered to "learn" when an individual's knowledge becomes part of the standards and routines of the organization [28],[31]. Computerized information systems (IS) take the business rules of an organization and codify those rules into a set of procedures that can be followed routinely. The embodiment of work activities into a computerized information system could be considered a form of organizational learning because the individuals who create the system specifications have their knowledge embedded into the information system.

An information system, however, embodies the rules and procedures of an organization at a fixed period of time - when it is originally installed. Changes to those rules and procedures are incorporated only through changes to the software. For an existing information

system to incorporate new individual knowledge and relay that knowledge throughout the organization, it must be enhanced during the process of software maintenance.

Software maintenance is work performed to change an existing software system after that system has been transferred to its intended recipient. While software maintenance is often perceived as just another name for software correction, prior research has shown that a majority of the work performed during maintenance is to enhance, rather than repair, existing systems [44],[23].

Existing research and practice views software maintenance as an expense incurred after development that should be contained through better development methodologies, better evaluation of system characteristics, better enforcement of programming standards; and more participation of users during initial systems development [20],[6],[13]. The implication of this research is that if we did it right the first time, there would be no need to continue enhancing software. It is an interesting paradox that as businesses are being exhorted to adapt to changing circumstances [37] and encouraged to learn to customize [22], the computerized business information systems that support those organizations are supposed to stay much the same from the day they are installed to the day they die.

This study addresses that paradox and seeks to understand the relationship between learning and software enhancements. To better understand user learning and the software enhancement process the question addressed by this study is: How does learning affect the need for software enhancements?

This question was addressed by investigating the software enhancement process from the IS practitioner and user perspectives. We concentrated on computerized information systems that had been used for at least six months. We interviewed personnel in the IS groups as well as users of each system to gain insight into what motivated the need for software enhancement and how those needs were satisfied.

2. Background

Current literature has embraced the idea of learning as crucial to empowering individuals and dealing with a volatile business environment [37],[16],[27],[39]. There is an implicit belief in this work that learning will lead to success because it will provide the flexibility and innovation needed to deal with the unexpected. While there is a paucity of quantitative research linking organizational success to learning, there is a rich vein of case studies examining the potential link between the two [37],[38],[14],[30].

2.1 Types of Learning

Organizational learning has been defined primarily through its results, such as a quantifiable improvement in activities, increased available knowledge for decision making or sustainable competitive advantage, and its processes, such as the ways companies build routines or socialize employees into a corporate culture [34]. Not all organizational learning, however, is the same. Theorists have posited two general categories of organizational learning: Exploration and exploitation [32],[1]¹). Exploration involves questioning existing boundaries and experimenting with new ideas to discover innovative solutions, while exploitation refers to learning performed within currently accepted norms and boundaries. Exploitation refines current operations while exploration results in completely new methods of doing business. Both types of learning are believed to enhance the competitive ability of an organization [32], but explorative learning is rare for both individuals and organizations while organizations frequently use exploitative learning to refine business processes [3],[32].

2.2 Process for Organizational Learning

The actual process of organizational learning, or “how” a given work unit or complete organization acquires and shares knowledge is of interest in the literature [19]. It has been suggested that it takes the exploration of an individual, or an external event, to trigger the organizational learning cycle [37],[30],[35],[31]. Other theorists [11],[9] believe that new knowledge can be generated in the interaction between members of a group. Transferring knowledge from individual to organization,

and guaranteeing the ongoing learning of the organization from the individual is a key concept in the literature [4],[34],[21].

2.3 Organizational Learning through Software Enhancements

Viewed from an organizational learning perspective, the development, implementation and maintenance of an information system could be considered a practical example of the learning process. User and IS professionals make tacit knowledge explicit during the development of initial system specifications. After a system is in production, the knowledge to use it becomes tacit once again while users exploit its potential. In time, a user might realize that there are enhancements that could be made that would help the system better support his or her work activities. The tacit knowledge of the user must become explicit for the IS professional to make enhancements to the system, thus translating the knowledge of the individual into organizational knowledge that can benefit others outside of the individual.

2.4 Research into Software Enhancements

Much of the previous research in software maintenance has concentrated on examining maintenance efforts for classification purposes and to estimate the scope of maintenance activities in organizations [23],[48],[12],[29],[43]. Since this research early noted that software maintenance takes a large percentage of the IS budget [18],[29] many studies have tried to find ways to control and constrain maintenance. IS research in software maintenance has looked at how to control software maintenance from a software engineering perspective [17],[20],[5],[13], a process definition and management perspective [46],[7],[8], and a maintenance activity management perspective [45],[24],[42]. Findings from this research show the need for greater understanding of the user's role in maintenance [17],[45] since the existing research has not done a good job delving into the reasons why software is enhanced [42].

In 1980, M. M. Lehman [25] published an article deriving a set of “laws” concerning software evolution. The first law explains that software programs are subject to continuing change; they are never completed because programs continue to evolve to meet user needs [25],[26]. By reviewing the reasons that software enhancements were requested, this research attempts to better understand the need for continual software evolution. This study looks at the following questions:

¹ Argyris and Schon referred to the two categories as single-loop and double-loop learning. The terms “exploitation” and “exploration” will be used in this paper as synonymous with single-loop and double-loop learning respectively.

1. Is software enhancement a result of individual learning?
2. How does software enhancement reflect organizational learning?
3. What type of learning (exploitative or explorative) triggers software enhancement?

3. Research Method

The objective of this study is to develop theory grounded in empirical evidence. The research design for this study is what Yin [47] has labeled an "embedded multiple case" design. The primary unit of analysis for this study was the organization, but embedded within each organization was a set of "work groups". A "work group" consisted of a group of people using the same information system(s) to support their work activities. For example, the three work groups embedded within one of the case studies (a set of hospitals) included the Accounts Payable, Hospital Admitting, and Pharmacy Control groups. Embedded work groups were studied in order to find out about the dynamics of learning within a given contextual situation. It was important in addressing the research questions to understand how individual learning was translated into shared routines within the work group. As a result, each work group within a case study site was analyzed separately to answer the question concerning individual learning and software enhancement. The organizational learning research question, however, was directed at the organizational level which is why the primary unit of analysis was the organization rather than the work group [47].

Five cases were researched for this project. Each case included three embedded work groups. A multiple-case design is considered to be similar to multiple experiments; it allows for replication logic [15]. Each case was treated as an individual experiment. The cases were analyzed separately and inferences from one case were applied to the next case. A multiple-case research design is often considered more robust than a single case because it is believed to produce a richer theoretical framework than a single case [47],[15].

3.1 Data Collection

Data were collected through interviews, questionnaires, observation, and study of documentation. In-depth, face-to-face, semi-structured interviews formed the primary method of data gathering. Interviews were conducted with representatives of the IS group and with members of the user work groups. Between 18 to 33 individuals were interviewed at each of the five case study sites.

Informal questionnaires were used to gather information about the demographics and structure of the IS group, as well details about the application portfolio maintained at each site. A questionnaire was also used to gather budgetary information at each site.

Documentation concerning the maintenance process was available at all sites. This documentation included formal process descriptions, request forms, request logs, prioritization formulas, and communication concerning enhancement requests. Documentation about the maintenance process was gathered from both IS and the user work groups.

3.2 Data Analysis

Each case was analyzed separately, one work group at a time. To ensure that the analysis and resulting theory was grounded in the data obtained from the study, a "constant comparison" approach was used [15],[41]. The forms of analysis used have been defined as "pattern-matching" and "explanation-building" [47].

3.2.1. Within-case analysis. Each work group within a case was analyzed separately using first a descriptive and then an interpretive coding scheme [33]. Descriptive codes were used to organize and coordinate data by work group. Pattern codes were developed from the interpretive codes by a system of theoretical memoing [33]. As the interpretive codes were reviewed, memos on specific theoretical topics were written to summarize research issues and possible explanations for those issues. Cross-embedded group matrices were developed to analyze the pattern codes in order to analyze all data for a given case study site. A theoretical model emerged from the first case analysis and this model was enhanced, modified and refined as additional cases were analyzed.

3.2.2. Cross-case analysis. Cross-case analysis was performed by case, and also by work group. Matrices were used to compare cases based on the pattern codes developed within each case. Each pattern code was examined for coherence and strength across the cases. Some pattern codes were not replicated across cases, and those were not included in the final models and propositions. This process of data reduction [33] eliminated much of the richness of description for the cases, but allowed a more generalized theory to emerge.

4. Findings - Description and Analysis

The five case sites included a mixture of non-profit and for-profit organizations as summarized in Table 1. As shown in Table 1, the sites existed in disparate external

and internal environments and were from very different industries. The sites were chosen specifically to evaluate learning in both volatile and non-volatile external and internal environments.

4.1 Employee Education

The types of employees needed by the organizations were very different. At one end of the range GMT required many highly educated engineering and manufacturing employees so management placed great value on education and sought personnel with college degrees. At the other end of the spectrum was BEA. The organization hired many entry-level workers to staff hotel and casino operations. College degrees were not required of the majority of the employees, and this emphasis was visible throughout the organization.

4.2 IS Group Characteristics

The relative emphasis on information systems differed among the organizations, but all of the IS groups could have been considered "factory" or "support" rather than "strategic" or "turnaround" using the information systems strategic grid [10]. The structure of the information systems group was remarkably similar among these organizations in differing industries. The manager of IS was primarily a technical representative dealing strictly with the infrastructure for business systems and rarely

participating outside that area of expertise. The IS organization was divided by functional specialty area, such as operations, systems and programming, and networking and telecommunications. The systems and programming area was divided by application in all cases. The specific application area was dependent upon the industry. For example, pharmacy was an application area for HHS, while manufacturing was an application area for GMT. Two of the organizations had some of their development and maintenance personnel in line departments, while three did not. There was no separate area or personnel for system maintenance from systems development at any of the sites - it was combined with systems and programming in all organizations.

4.3 Findings about the Relationship between Software Enhancement and Organizational Learning

Only one case (GMT) performed more new development than software maintenance. As has been found in existing research [45],[23],[48] the rest of the cases spent more IS resources on maintenance than new development.

4.3.1 What were the reasons for software maintenance? As had been found in prior studies [20],[45], functional enhancements requested by users were the primary reasons for software maintenance at the

Table 1. Summary of case study organizations.

Size is provided differently depending on the level of confidentiality requested by the organization and the kind of measure most appropriate to the type of organization

	GMT	SCG	HHS	APP	BEA
<i>Industry</i>	equipment manufacturing. profit.	county government. non-profit.	hospital. non-profit.	public utility. non-profit.	hotels and casinos. profit.
<i>Size*</i>	Total Revenue: \$744 million. Net Income: \$137 million.	All gov't funds: \$274 million. General Fund: \$165 million. 318,900 population. 6,000 sq. mile area.	Total Revenue: \$312 million. 3 hospitals. 449 beds total.	Total Revenue: \$663 million. Net Income: \$74 million. 286,000 electricity customers.	Total Revenue: \$280 million. 5 properties. total rooms: 1,462.
<i>External environment</i>	<ul style="list-style-type: none"> • very volatile. • increased competition. • new markets. 	<ul style="list-style-type: none"> • relatively stable. • high community growth (4%/year over last 20 years). 	<ul style="list-style-type: none"> • changes in regulation and insurance. • high growth community. 	<ul style="list-style-type: none"> • has been regulated, but in process of deregulation. 	<ul style="list-style-type: none"> • highly competitive. • established industry. • new markets.
<i>Internal environment</i>	<ul style="list-style-type: none"> • high mgmt. turnover. • reengineering processes. • new acquisition activity. 	<ul style="list-style-type: none"> • very stable mgmt. • stable processes. 	<ul style="list-style-type: none"> • stable mgmt. • some change in processes. • decreasing control by physicians. 	<ul style="list-style-type: none"> • very high mgmt. change. • stable processes. • active merger negotiations. 	<ul style="list-style-type: none"> • very stable corporate mgmt. • high change in property mgmt. • stable processes. • high growth.
<i>Total # of employees</i>	2,600.	2,385.	425 physicians. 1,800 staff.	1,500.	3,500.

case study organizations Over 60% of the software maintenance requests at all organizations were for functional enhancements. The enhancements included new reports, additional data items, enhanced data input validation, reformatted screens, integration with other systems, enhanced help systems, and simplified interfaces. Not all of those requests were met with software modifications. Many of the enhancement requests were for capabilities already contained within the software, so one of the major functions of IS people responsible for software development and maintenance was to identify those capabilities and teach the users to make best use of the software. This form of teaching was considered to be part of the "maintenance" umbrella, since the same support people were responsible for all phases of customer application support.

Most of the requests for software enhancements could be considered as a way to exploit current processes. The requests were made to refine a task, or gain a little more information to make a decision. Few of the requests resulted in large changes to the existing software or made big modifications to work procedures. Software enhancements that did not make fundamental changes in work processes were accepted in the work groups and easily tolerated by the IS organizations. The IS organizations considered these requests "superfluous" or "frosting on the cake," but they were willing to make the changes. There appeared to be pressure among members of a work group to suppress the need for large changes in work activities. For example, members of the Pharmacy Control group at HHS attempted to transfer a group member who was trying to use technology to fundamentally reengineer the group's processes to another department. There was consistent peer pressure among groups to avoid the changes that might result from exploratory learning. This pressure was echoed by the IS organization, whose members frequently avoided the implementation of any complex or fundamental changes to existing systems.

Far down the scale from functional enhancements was software correction and repair. These requests accounted for about 10-15% of the software maintenance activity. The remaining software maintenance activities could be attributed to adaptive maintenance. None of the organizations were moving to a completely new hardware platform, but some were off-loading specific applications to a different architecture or to a different graphical interface.

4.3.2 What were the reasons for software enhancements? Reasons for software enhancement requests can be classified into four categories:

- **External Changes:** This factor includes all changes in the environment external to the organization. External changes include such modifications as: Regulatory changes (example: a hospital with new Medicare reporting needs); new requirements from integrated systems (example: a Sheriff's information system that must connect to the Department of Motor Vehicles – when the DMV changes its data format, the Sheriff's system must change as well); and new competition.
- **Internal Changes:** This factor includes all changes inside an organization. Internal changes include management changes, new policies, new products or services, and acquisitions of other organizations. Anything that changes inside a company, except the technological environment of computer-based information systems, is part of this factor.
- **Technical Environment Changes:** This factor is listed separately from internal changes because it produces its own set of software enhancements. Changes in the technical environment can result from the purchase of new software. For example, a new information system may require enhancements to existing packages for data integration and user interface. Another example of a technical environment change is the move to a web browser-based interface.
- **Learning:** This factor includes all changes that are a result of individual or group learning. When an individual gains knowledge about his/her work activities and wants to change the software used to support those activities, then the category of "learning" was considered to be influencing the need for software enhancement. For example, a member of the budget office at SCG had a set of specialized calculations he performed that provided improved decision making for his office. He devised these calculations from ideas gained while using his existing software. He requested a software change so that his new calculations would be incorporated in the standard budgeting package so that the calculations would be used across the county, sharing his knowledge and saving time for others in the organization. Individual learning may not have occurred only from using the existing software. Individual learning also includes knowledge gained from such factors as: external exposure, networking and internal reflection.

Table 2. Reasons for software enhancement requests by embedded work group.

	Learning	Internal Changes	External Changes	Tech. Env. Changes	Examples of Software Enhancements
GMT					
Order Management	10%	20%	20%	50%	Interface with new software package. Many functional enhancements made to make reports and user interface of the new package look like the old.
Accounts Payable	50%	30%	5%	15%	Management changed existing processes based on their exposure to software in use at other organizations. They consolidated processes, eliminated the need for matching invoices with receiving documents, and reduced the need for data entry with a new imaging system.
Machine Placement	70%	5%	25%	0%	Requests included new reports, data items, calculations, and modified screens. New accounting procedures were being developed based on the flexibility of the technology.
SCG					
Law Enforcement	10%	60%	25%	5%	Management changes were responsible for most requests for enhancements. Management did not work directly with the system – management mainly received reports and information from the users of the system.
District Attorney	55%	10%	25%	15%	Package was acquired and modified to fit the users. Requests were to add data to screens, provide additional help screens, and make fewer keystroke operations.
Budget Office	50%	20%	20%	10%	Learning resulted in new reports, new calculations, additional data items, modified screens, and more integration/consolidation with other systems.
HHS					
Accounts Payable	5%	35%	60%	0%	Changes were a result of installation of new systems for suppliers. Suppliers incorporated their own inventory control in systems used by the hospitals.
Hospital Admitting	25%	55%	25%	0%	Changes were a result of new management. New procedures such as online insurance confirmation, reservations, and one-step exits.
Pharmacy Control	50%	5%	20%	25%	Changes included new pharmacy delivery and distribution mechanisms, new methods of control, more comprehensive reporting, and new prescription review procedures.
APP					
Human Resources	0%	0%	100%	0%	Requests for governmental and regulatory changes.
Purchasing	50%	10%	20%	20%	Requests outstanding for automated requisitioning, intranet-based work flow, and internet connection to suppliers.
Customer Billing	20%	10%	70%	0%	Regulatory changes caused most change requests. Past regulatory influences included new billing information to be provided for customers and new calculations of bills. Current deregulation triggering most changes.
BEA					
Hotel Administration	20%	50%	30%	0%	Ideas to incorporate yield management techniques to hotel room pricing. Ideas came from new management and competition..
Credit and Collections	15%	15%	0%	70%	Most requests made to make new package have same input interface and informational structure as old package.
Accounting	25%	35%	40%	0%	New regulatory reporting requirements for gaming accounting. New decision making reports required by new general manager.

The reasons for software enhancements could not be generalized to a given case study site, but were more consistent among requests in a work group, as a result, the reasons for software enhancements are presented by work group rather than by case. Table 2 divides the motivation behind software enhancement requests for the work groups into the four categories described previously. Learning was the main motivator behind the requests in about 40% of the work groups. This number can be a bit deceiving because even in the cases where learning was the primary motivator, the other reasons were important, too. Most of the changes attributed to learning were for slight revisions to software-controlled work procedures. These changes were small revisions to the existing software, such as changes in screen formatting or additional reports.

An interesting finding was the reaction of users to the idea of learning possibly contributing to the need for enhancements. Most users said that their requests for software enhancements came from external or internal changes when asked point blank during an interview about the motivation of the request. Users said they had to have the changes because of some force other than themselves. Few users initially identified their requests as derived from learning. During further conversation and examination of specific change requests, though, many users admitted that the changes they wanted were to “make my life easier,” “save me three hours a day,” “make me feel better about this decision” or “cut down on

paperwork.” It became apparent during interviews that many requests attributed to outside forces were actually at the instigation of the user. There appeared to be a desire to make the change requests “not my fault” (on the part of the user) so that they would appear more acceptable to the IS organization. Almost uniformly among users in work groups, there was a strong belief that the IS organization did not want to enhance software and that changes had to be justified in some way other than it would help work activities. The interviewees in IS organizations agreed, frequently commenting that the enhancements required by users were “superfluous” and, in the opinion of IS, not necessary for users to do a good job. There was a consistent conflict between work groups and IS organizations at each case about what constituted a necessary enhancement to software.

5. Study Questions and Hypotheses

The first question pursued by this study was: *Is software enhancement a result of learning?* Learning was one factor contributing to the need for ongoing software enhancement, as depicted in Figure 1. Software enhancement was more a reflection of individual learning than learning within a work group since individuals, not groups, devised new ways to use computerized information systems to support their work activities at the case sites. After coming up with a new idea, an individual usually had to convince the work group that it

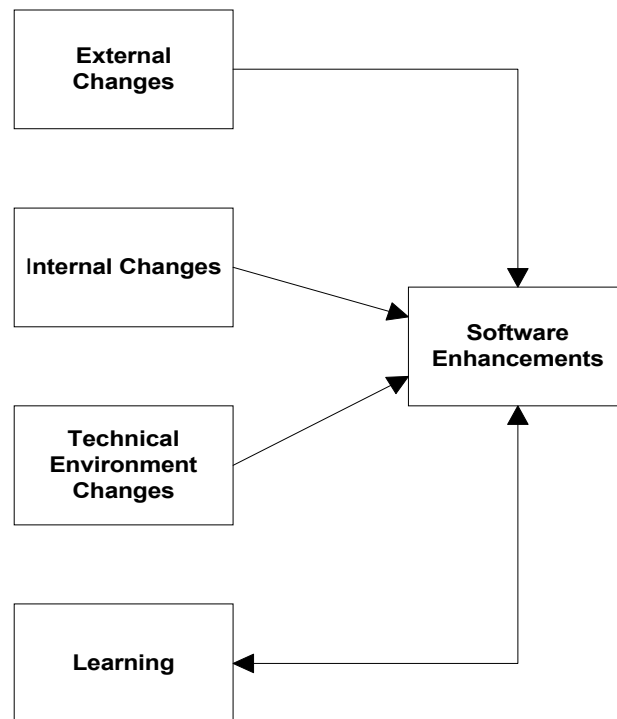


Figure 1. Factors instigating software enhancement requests.

was a good idea and should be applied to the computerized information system. The individual then had to convince the responsible IS professional that the request was necessary and important. This finding with respect to software enhancement agrees with literature that asserts learning comes from individuals instead of organizations [4],[35],[40] rather than the idea that learning evolves both from individuals and groups as they interact [11]. However, in the case of system requirements determination, the latter perspective probably holds.

The two-way arrow between individual learning and software enhancements in Figure 1 depicts the observation that software enhancements, once completed, appeared to encourage additional learning, which then frequently led to further software enhancements. This finding aligns with Orlikowski's [36] concept of "interpretive flexibility," or the degree to which users of a software-based technology understand its constructive nature and changeability. Orlikowski [36] speculated that people would have a higher degree of interpretive flexibility during initial development since during that time people would be more aware that technology could change. She also suggested that as the technology becomes entrenched it takes on the characteristics of an organizational structure and then people lose their ability to perceive it as changeable. The findings in this study show that a user's interpretive flexibility appears to increase during software enhancement, which then contributes to further learning. Data from the cases showed that efforts by IS to reduce the number of software enhancements by such methods as delaying completion of requests, requiring extensive cost justifications, and verbally discouraging users from submitting requests appeared to be correlated with a reduction in individual learning. This relationship of software enhancement and learning leads to the following:

Proposition 1: The level of requests completed successfully for software enhancements is positively related to the level of individual learning in a work group that is dependent upon computerized information systems to complete work activities.

The second question motivating this study was: **How does software enhancement reflect organizational learning?** Software enhancement could be considered a method of disseminating the knowledge of an individual to a group because modifications to a system translate the learning of an individual into shared routines for the whole group. For some of the members of the work groups, the application of a software enhancement allowed an individual to make changes to work group procedures that would have been more difficult to do

without the direct enforcement of a computerized system. It would have been easier for the work group to slip back into old procedures without the constant reinforcement and structure of a computerized system. One interviewee at GMT summarized the profound implications of software enhancements on organizational procedures by stating "once the program is changed, then everyone has to do it like I do (referring to his method of calculating costs)". While initial software development incorporates the tacit and explicit knowledge of a work group into an information system, ongoing learning of the group is reflected in the enhancements made to the information system. Thus, software enhancements are a form of organizational learning as stated in the following:

Proposition 2: Software enhancements are a way for individual members of a work group using computerized information systems to transfer their knowledge to the rest of the group.

The third question of this study asked: **What type of learning (exploitative or explorative) triggers software enhancement?** Exploitation and exploration-type learning motivates software enhancements. Based on the data from the study, exploitative learning lent itself to small refinements in systems, while explorative learning always demanded structural changes to procedures and data. As has been noted in prior research [2], exploitative learning was far more prevalent in the work groups observed for this study. Both the members of the work groups and the IS organizations appeared more comfortable with software enhancements that refined rather than reengineered existing work processes and the systems that supported those processes. People who suggested small improvements to processes were more accepted in their groups while those suggesting radical changes were either loved or hated by the other members. It was a surprise that anyone was able to explore new methods of applying technology to business problems at all. Those people who did so steered or bulldozed their way through roadblocks established by members of their own work groups and members of the IS organization. While members of the IS organization dismissed most requests for software enhancements as "unnecessary," they were most willing to fulfill those that required the fewest number of structural changes. As a result, exploitative learning was encouraged by the IS organization while explorative learning was suppressed. These observations are formalized in the following:

Proposition 3: The majority of software enhancement requests resulting from learning are a result of exploitative learning.

Proposition 4: IS organizations emphasize the fulfillment of software enhancement requests that reflect exploitation learning over explorative learning.

6. Conclusion

This study is unique because it merges the technical literature on software maintenance with the managerially-oriented research into organizational learning and thus contributes to both areas. Existing literature in software maintenance speculated that learning does contribute to the need for software enhancement [26],[7] but the structure of that contribution had not been defined previously. This study expands existing research by formalizing the relationship between learning and software enhancement, categorizing the factors that contribute to software enhancements from the perspective of those actually making the enhancement requests, and delineating the relationship between learning and enhancements. These relationships, as captured in the four propositions stated above, constitute a testable theory of the role of software maintenance in organizations.

This study also contributes to the research into organizational learning. Prior research called for additional case studies examining ways individuals and groups learn [11]. The case studies in this research demonstrate practical examples of individual learning transferred to the group through software enhancement. A contribution from this demonstration is the concept that the learning of an individual can be disseminated to a group through enhancements to software thus making the process of software enhancement an act of organizational learning. This contribution is significant because it encourages researchers in information systems to expand the research focus of software maintenance beyond the attitudes of constraint and control to incorporate the potentially positive implications of software enhancement. In addition, this study demonstrates another method to transfer knowledge, thus expanding the practical advice that can be given to organizations wishing to establish more ways to encourage organizational learning.

7. References

- [1] Argyris, C. and Schon, D. *Organizational Learning*, Addison-Wesley, Reading, Mass., 1978.
- [2] Argyris, C. and Schon, D.A. *Organizational Learning II: Theory, Method, and Practice*, Addison-Wesley Publishing Company, Reading, Mass., 1996.
- [3] Argyris, C. *On Organizational Learning*, Blackwell Publishers, Cambridge, MA, 1993.
- [4] Bain, A. "Social Defenses Against Organizational Learning," *Human Relations* (51:3), March 1998, pp. 413-429.
- [5] Banker, R.D. and Slaughter, S.A. "A Field Study of Scale Economies in Software Maintenance," *Management Science* (43:12), December 1997, pp. 1709-1725.
- [6] Banker, R.D., Datar, S.M., Kemerer, C.F. and Zweig, D. "Software Complexity and Maintenance Costs," *Communications of the ACM* (36:11), November 1993, pp. 81-0.
- [7] Bennett, K. "Software Evolution: Past, Present and Future," *Information and Software Technology* (38:11), November 1996, pp. 673-680.
- [8] Briand, L.C., Basili, V.R., Kim, Y. and Squier, D.R. "A Change Analysis Process to Characterize Software Maintenance Projects," *Proceedings of the International Conference on Software Maintenance, Victoria, BC, Canada*, September 1994, pp. 38-49.
- [9] Brown, J.S. and Duguid, P. "Organizational Learning and Communities-of-Practice: Toward a Unified View of Working, Learning and Innovation," *Organization Science* (2:1), February 1991, pp. 40-57.
- [10] Cash, J.I., McFarlan, F.W. and McKinney, J.L. *Corporate Information Systems: Text and Cases*, Irwin-McGraw-Hill, Burr Ridge, 1992.
- [11] Cook, S.D.N. and Brown, J.S. "Bridging Epistemologies: The Generative Dance Between Organizational Knowledge and Organizational Knowing," *Organization Science* (10:4), July-August 1999, pp. 381-400.
- [12] Dekleva, S.M. "Software Maintenance: 1990 Status," *Journal of Software Maintenance: Research and Practice* (4:4), December 1992, pp. 233-247 (a).
- [13] Dekleva, S.M. "The Influence of the Information Systems Development Approach on Maintenance," *MIS Quarterly* (16:3), September 1992, pp. 355-372 (b).
- [14] DiBella, A.J., Nevis, E. and Gould, J.M. "Understanding Organizational Learning Capability," *Journal of Management Studies* (33:3), May 1996, pp. 361-379.
- [15] Eisenhardt, K. "Building Theories from Case Study Research," *Academy of Management Review* (14:4), October 1989, pp. 532-550.
- [16] Garvin, D.A. "Building a Learning Organization," *Harvard Business Review*, July-August 1993, pp. 78-0.
- [17] Ghods, M. and Nelson, K.M. "Contributors to Quality During Software Maintenance," *Decision Support Systems* (23:4), October 1998, pp. 361-369.
- [18] Guimaraes, T. "Managing Application Program Maintenance Expenditures," *Communications of the ACM* (28:10), October 1983, pp. 739-746.
- [19] Huber, G.P. "Organizational Learning: The Contributing Processes and the Literatures," *Organization Science* (2:1), 1991, pp. 88-115.

- [20] Kemerer, C. and Slaughter, S.A. "Determinants of Software Maintenance Profiles: An Empirical Investigation," *Journal of Software Maintenance: Research and Practice* (9:4), 1997, pp. 235-251.
- [21] Kim, D.H. "The Link Between Individual and Organizational Learning," *Sloan Management Review* (35:1), Fall 1993, pp. 37-50.
- [22] Kotha, S. "Mass-customization: A Strategy for Knowledge Creation and Organizational Learning," *International Journal of Technology Management* (11:7), July 1996, pp. 846-859.
- [23] Krogstie, J. "On the Distinction between Functional Development and Functional Maintenance," *Journal of Software Maintenance: Research and Practice* (7:6), November-December 1995, pp. 383-403.
- [24] Layzell, P.J. and Macaulay, L.A. "An Investigation into Software Maintenance - Perception and Practices," *Journal of Software Maintenance: Research and Practice* (6:3), May-June 1994, pp. 102-120.
- [25] Lehman, M.M. "Programs, Life Cycles, and Laws of Software Evolution," *Proceedings of the IEEE* (68:9), September 1980, pp. 1060-1076.
- [26] Lehman, M.M. "Softwares Future: Managing Evolution," *IEEE Software*, January-February 1998, pp. 40-44.
- [27] Leonard-Barton, D. "The Factory as a Learning Laboratory," *Sloan Management Review* (34:1), Fall 1992, pp. 23-38.
- [28] Levitt, B. and March, J.G. "Organizational Learning," *Annual Review of Sociology* (14), 1988, pp. 319-340.
- [29] Lientz, B.P. and Swanson, E.B. "Problems in Application Software Maintenance," *Communications of the ACM* (24:11), November 1981, pp. 763-769.
- [30] Lipshitz, R., Popper, M. and Oz, S. "Building Learning Organizations: The Design and Implementation of Organizational Learning Mechanisms," *Journal of Applied Behavioral Science* (32:3), September 1996, pp. 292-305.
- [31] Mackenzie, K.D. "The Science of an Organization. Part I: A New Model of Organizational Learning," *Human Systems Management* (13:4), 1994, pp. 249-258.
- [32] March, J.G. "Exploration and Exploitation in Organizational Learning," *Organization Science* (2:1), February 1991, pp. 71-87.
- [33] Miles, M.B. and Huberman, A.M. *Qualitative Data Analysis*, Sage Publications, Inc., Thousand Oaks, CA, 1994.
- [34] Miller, D. "A Preliminary Typology of Organizational Learning: Synthesizing the Literature," *Journal of Management* (22:3), 1996, pp. 485-505.
- [35] Nonaka, I. and Takeuchi, H. *The Knowledge Creating Company*, Oxford University Press, New York, 1995.
- [36] Orlikowski, W.J. "The Duality of Technology: Rethinking the Concept of Technology in Organizations," *Organization Science* (3:3), August 1992, pp. 398-427.
- [37] Pascale, R., Millemann, M. and Gioja, L. "Changing the Way we Change," *Harvard Business Review*, November-December 1997, pp. 127-139.
- [38] Prokesch, S.E. "Unleashing the Power of Learning: An Interview with British Petroleum's John Browne," *Harvard Business Review*, September-October 1997, pp. 147-168.
- [39] Senge, P.M. *The Fifth Discipline: The Art and Practice of the Learning Organization*, Bantam Doubleday Dell Publishing Group, New York, NY, 1990.
- [40] Simon, H.A. "Bounded Rationality and Organizational Learning," *Organization Science* (2:1), February 1991, pp. 125-134.
- [41] Strauss, A. and Corbin, J. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*, Sage Publications, Inc., Newbury Park, CA, 1990.
- [42] Swanson, E.B. and Beath, C.M. "Departmentalization in Software Development and Maintenance," *Communications of the ACM* (33:6), June 1990, pp. 658-667.
- [43] Swanson, E.B. "The Dimensions of Maintenance," *Proceedings of the Second International conference on Software Engineering*, San Francisco, 1976, pp. 492-497.
- [44] Tan, W. and Gable, G.G. "Attitudes of Maintenance Personnel Towards Maintenance Work: A Comparative Analysis," *Journal of Software Maintenance: Research and Practice* (10:1), 1998, pp. 59-74.
- [45] Taylor, M. J. and Wood-Harper, A. T. "Methodologies and Software Maintenance," *Journal of Software Maintenance: Research and Practice* (8:5), 1996, pp. 295-308.
- [46] Weiderman, N.H., Bergey, J.K., Smith, D.B. and Tilley, S. R. "Approaches to Legacy System Evolution," *Technical Report, CMS/SEI-97-TR-014*, December 1997, pp. 1-30.
- [47] Yin, R.K. *Case Study Research*, Sage Publications - Applied Social Research Methods Series, , 1994.
- [48] Yip, S.W.L. and Lam, T. "A Software Maintenance Survey," *Proceedings of the 1994 Asia-Pacific Software Engineering Conference*, December 1994, pp. 70-79.