Understanding the Requirements for Information System Documentation: An Empirical Investigation

Abdulaziz Jazzar
International Systems Engineering, Riyadh 11514 Saudi Arabia
and
Walt Scacchi
University of Southern California, Los Angeles, CA 90089-1421
scacchi@gilligan.usc.edu

Abstract

Software and Information Systems (IS) documents are a common product of large IS development efforts. These documents are produced and consumed through a variety of documentation processes. These processes involve developers and users working within complex organizational settings, as well as with the focal system under development. These organizational settings facilitate and constrain IS documentation and development efforts in complicated ways. Accordingly, we present, analyze, and compare cases from field studies of three different IS development efforts in a large industrial corporation. Based on these studies, we identify a new set of variables and hypotheses that we believe represent a more plausible set of requirements for IS documentation products and processes in different organizational settings. In this regard, we utilize the concept of viewing IS documentation requirements as hypotheses to be tested, refined, or refuted.

1 Introduction

A characteristic of most sofware and IS is the large amount of documents that are produced to support system development and evolution. These documents are employed to help manage the development process as well as to facilitate IS use and maintenance. After all, who among us has not taken a multitude of opportunities to either review, criticize, or author some form of IS documentation? Similarly, many of us may prefer to try to find someone to ask, who may have already read the documentation, to explain how some system function was developed, or how to use some system function, rather than facing the chore of looking into the available documentation to resolve the query. Nonetheless, the production and consumption of IS documents is often problematic, costly, and frequently regretted.

Typically, the problems of software or IS documentation are described in terms of the difficulty in keeping software or IS documentation *consistent, up-to-date, correct, cost effective,* and *understandable* [5, 10, 2, 9]. Other formulations of the problems are expressed in terms of the inadequacy of the methods of representation needed to describe the system in a *consistent, complete* and *precise* way [14, 1]. Finally, others view the problems in terms of poor training, communication errors, document incomprehensibility, or insufficient IS standards [6, 7, 12], where solutions represent tools that promote *traceability, standardization, consistency, conversion into hypertext* and *ease of document development and online browsing* [8, 15, 16, 3]. The problem, therefore, has been generally formulated in terms of the difficulties in developing and maintaining documents as *products* with a set of desirable characteristics, but without regard to the *processes* and *organizational settings* in which they are produced and consumed.

These solution approaches have led to mixed results in easing the problems of IS documentation. For example, document standards emphasize the importance of documentation throughout a system's life cycle and give a certain amount of structure to the documentation process. Unfortunately, this is often done at the expense of equating *good* documentation with either *more* or *less* documentation depending on which sources of expertise one consults. Accordingly, good documentation effectively requires documenting either *everything* or *almost everything* to some degree. The principle behind this approach is "you never know what might be needed," and in any case, people often complain from under-documented systems.

In spite of standards and tools, organizational computing systems still end up with unused and/or unusable documents. IS documents that do exist generally continue to be (a) unavailable when needed, (b) inconsistent, (c) untraceable across a system's life cycle transformations, (d) difficult to update, and (e) infrequently produced on a timely and cost effective basis. Documentation can in some sense be considered the least liked task of IS development.

In this paper, we argue that none of the proposed solutions cited above has produced the desired comprehensive solution. Missing is a more fully developed and empirically grounded understanding of what shapes the effectiveness and quality of IS documentation products and processes. Our research is directed toward this goal of developing such an understanding in a way that can be further tested, refined, or refuted.

We describe an empirical investigation that identifies a set of variables and hypotheses which provide a more articulate model of IS documentation requirements than previously available. These variables and hypotheses are derived from a comparative study of IS documentation efforts in three field study cases. We then examine relationships between a subset of three variables and various documentation products and processes as the basis for constructing the hypotheses. In our view, these hypotheses collectively begin to model the requirements for achieving effective and quality IS documentation products and processes. Finally, we conclude with some observations on how this new understanding of the interrelationships between IS documentation products, processes, and organizational settings should lead to revised set of requirements for IS documentation tools, techniques, and standards. But first, we describe the research methods we employed for identifying and constructing these documentation requirements hypotheses..

2 Research design

We follow a comparative case study research method. The mode of our analysis is to conduct field studies of IS development efforts in complex, real-world settings. This allows us to investigate the contextual circumstances for changes in the documentation products and processes in organizations. Our focus is not on individual participants or small groups, but rather on the flow of work and problems experienced by people participating in a large IS development effort within an organizational setting. As such, the unit of analysis in our studies spans the production and consumption of IS documents within specific organizational settings. Thus, we examine the products (documents), processes (document production and consumption) and organizational settings and situations of IS documentation, as well as their interrelationships.

The criteria for selecting a study case was that the IS development effort being studied exhibit some changes in its documentation products and/or processes. As a research strategy, case studies are good for revealing the depth and complexity of situated processes or phenomena, but often lack generalization. In turn, to increase generalization, multiple case studies must be employed, and their analyses conducted at different levels. In this study, our analyses are conducted at both the case-specific and case-comparative levels.

Three case studies are presented, analyzed, and compared. The cases represent IS development efforts that vary in increasing complexity and orgnizational diversity. The cases were investigated over a period of 18 months. Data was collected from structured interviews with more than 50 people including corporate executives, department managers, project leaders, and systems programmers, as well as contractor staff. In addition, we also collected and reviewed documents and artifacts from these development efforts where appropriate. Details on the collected data including the interview schedule and structure are available from the authors.

The first case study investigated changes in the working arrangements of a data processing center (DPC) due to the adoption of a new software system. The second case study examined two different DPCs and the changes in working conditions that had to be made as one software system moved from one DPC to another. The third case study looked at circumstances that necessitated the adoption of two sets of documentation processes and products within one DPC.

Our objective in conducting this comparative case study is to identify a set of empirically grounded variables and hypotheses that characterize the conditions and relationships that shape the production and consumption of IS documentation. In this regard, our view is that these hypotheses will represent a new set of requirements for managing and supporting IS documentation processes. Clearly, we cannot claim that this small sample of IS development cases is necessarily representative of the entire population of IS development efforts. Thus, our hypotheses do not represent a definitive set of documentation requirements, but instead represent a suggestive set of requirements that are grounded in empirical observation and comparative analysis. Nonetheless, as hypotheses, they are testable in other IS development efforts, which can lead to their further validation, refinement, or refutation. We believe this is a more substantial and testable basis for understanding the requirements for IS documentation than has been used in most of the approaches cited in the Introduction.

3 Case study data

The following three study cases are all related to the same overall organization. The organization will be referred to as the STBG corporation (a pseudonym). However, each study case exhibits different circumstances that are in accord with the selection criteria.

The STBG corporation is a large organization concerned with flying and maintaining aircraft, and with training aircraft maintenance technicians. The company has seven geographically remote sites where flying, maintenance, and training activities take place. The STBG main corporate offices maintain control over all the company's general policy affairs, while each site is entrusted with the day-to-day operation of its local flying, maintenance, and training operations.

3.1 The T&T case

This case concerns the development of a system called T&T that tracked STBG students throughout their training phases. The T&T system was based on another IS called NRP. NRP was a batch-oriented system developed by a subcontractor to the STBG corporation that was responsible for training a particular subgroup of the STBG students. The NRP system was used internally by the subcontractor to produce monthly reports as a means of maintaining student records and tracking their progress. It was also used to produce student performance reports required under its contract with STBG. After interacting with the NRP system for 3-4 years, the training department staff within STBG was so fond of the system that they decided to use it and expand its capabilities in order to track all STBG students.

The subcontractor had its own data processing center that had originally developed NRP. The DP center had a staff of 15 and ran its software on a large IBM mainframe computer. NRP had some 400 programs and was developed and enhanced by a group of 7-10 programmers over a period of 3-4 years. The subcontractor was given detailed documentation standards to follow. The subcontractor's project leader conceded however that they were not followed to the letter when developing the NRP software.

The T&T system was to be developed by the IS department at the STBG corporate offices. An arrangement was made with the subcontractor to hand over all related sources of the NRP software and databases, together with all available documentation. A subset of the programmers that were involved in developing and maintaining NRP were available by appointment to answer any queries regarding the software.

The new T&T system was envisioned to be an on-line system that included most of NRP's features as well as new reports and inquiry capabilities peculiar to the STBG corporation. While the mainframe-based NRP included some 400 programs that only produced monthly reports, the new T&T system was to run on a different IBM machine that ran a different but compatible operating system.

The IS department had a set of software documentation standards that were reasonably enforced but that were different from the ones used for developing NRP. The department was divided into three groups: operations and technical support group, development and maintenance group, and quality assurance group. The T&T development team was part of the development and maintenance group. An extensive analysis was conducted with the users before starting the design phase. This process was made easier by the fact that the original NRP was available and thus could be treated as an "operational prototype" for the T&T system.

The training department staff at the STBG main office were the main users of this system although remote sites were also going to use it. The users were generally technically oriented and were familiar with the computer reports produced by NRP. Learning how to log-on and interact with the new on-line system was only necessary for a selected number of users. Some user procedures had to be modified and republished in a new version of the standards manual.

The new T&T system was to include extensive interactions with other software systems such as the student personnel system and the student scholarship system. Both these systems were still under various phases of development at the IS department, but were being developed by different groups of programmers. Often members from the three development teams had to coordinate their efforts, so that each project leader received all the documentation produced by the other two teams.

After receiving the NRP documentation and reviewing it, the T&T development group decided that NRP was not adequately documented. Inconsistencies between the system performance and its documentation were found to be confusing and unacceptable. The IS department developers of T&T decided to contact the original developers of NRP to get their assistance in upgrading the NRP documentation before attempting to extend the system's use and capabilities. The documentation upgrade effort was the first phase of developing the T&T system. This phase ran parallel with the effort of moving NRP and its database to run on the local IBM machine. The manager of the IS department at STBG stated more than once that he was not totally convinced of the necessity of the redocumentation effort. He stated that he gave in reluctantly after the head of the T&T development team personally pleaded with him.

3.1.1 Discussion of the T&T case

NRP was running quite satisfactorily in its original location even though the documentation standards were not strictly enforced. It seemed very likely that the new T&T system could have been developed without the documentation upgrade effort if it was developed by the same group that developed/maintained the original NRP. This suggests that an *undocumented knowledge base* existed in addition to the documented one. This informal knowledge base was captured in the minds of the people involved with the NRP development and maintenance process, and it appears to have played a supplementary yet critical role in the overall knowledge required to develop, maintain, and use the new software system.

Nonetheless, two significant conditions can be identified:

(1) When the maintainers of NRP were no longer a subset of the development team, the documentation that was successfully used for its maintenance was no longer sufficient. This suggests that the *the relationship between the developers and maintainers* could affect the documentation requirements of a project.

(2) NRP had almost no interaction with other software systems. All the data and the processing that were needed to produce its reports were not shared by any other system. By contrast, the T&T software had to export and import data to other software systems such as the personnel system and the scholarship system. The extent of *interconnectivity of the system to other systems* thus may account for the change in documentation requirements that occurred.

3.2 The R2S2 case

Spare part purchases for all STBG aircraft maintenance activities were made through a Central Supply Facility (CSF). Requests from remote sites were funneled to the CSF and when spare parts were received, CSF distributed them back to the remote sites. Due to a recent surge in business for STBG, the CSF manual system for ordering, receiving and distributing spare parts became a bottleneck that was hampering the company's efforts to meet the new demand.

The IS department at the head corporate office was charged with finding a suitable solution. The idea of developing a software system that automated the

CSF activities was initially discussed but was regarded as a long-term objective that did not ease the immediate problems. In the opinion of the IS department manger, if STBG was to meet all the new demands and remain competitive, an automated supply system needed to be in place within the next 12 months. Together, the IS and CSF staff investigated available software packages tht support CSF activities. They were able to identify a system called R2S2 that was being used by a sister company that had most of the important features required for automating the CSF functions. Use of this system was then proposed as a temporary solution for three years during which a fully automated supply system that would link CSF to all remote sites would be developed.

R2S2 did not possess all the features required for automating CSF. Consideration was given to modifying the R2S2 system before adopting it, but the department managers feared that this might take a long time. Thus, they decided that the R2S2 system would be installed as is. No major modifications would be made prior to adoption. After negotiating with the sister company, they agreed that a team of approximately 20 IS people, who were currently or recently involved in the development and maintenance of R2S2 system, would be transferred to the STBG IS department. These staff would then install the R2S2 system at CSF as well as maintain and manage it for a period of three years, after which they were to be released back to their old jobs at the sister company. The R2S2 system was to run on the existing computer facility at CSF. This was yet another IBM maingframe machine that for the past two years only ran a batch system called ARX.

ARX was a batch-oriented analysis system that kept track of market trends for aircraft spare parts availability and prices. It was developed by an outside contractor. When ARX was implemented, it was the first automation effort at CSF. It was operated and maintained by a small Data Processing (DP) section at CSF and was run only once a month to generate reports for the various STBG departments.

ARX design and implementation documentation was extensive. Changes done to the system by the DP section of the CSF were numerous but never extensive. They were rarely of an urgent nature and can be generally categorized as routine maintenance. The DP section had no formal documentation standards but rather an informal "standard of practice." Only a few ARX documents were regularly used and of these, the program source code listings were most often in demand. When maintenance tasks were completed, no detailed documentation reviews were made. New program listings were produced and minimal updates and/or editing changes were made to other documentation. Although users were never totally satisfied, the system had been running for over two years with overall positive results in the users' view. The informal documentation standards at the DP section were never challenged as inadequate.

The new R2S2 system was an on-line system, with over 30 geographically remote desktop stations. IS standards prescribed for its original development and maintenance efforts defined detailed checks and reviews of documentation. It was not clear whether these standards were strictly enforced by the sister company. Many of the R2S2 documents that arrived at CSF contained many hand-written remarks written on the margins of documents. The general consensus among the programmers was that the hand-annotated program listings were the most accurate documents.

The DP section at CSF had a staff of 10 who operated and maintained ARX. The section was divided into 2 groups, the system programming and operation group, and the maintenance group. When the new R2S2 system was installed, the staff number mushroomed to over 30 and included an operation group, a system programming group, a maintenance group, and a user interface group. The maintenance/development personnel were grouped by supply functions in the sense that certain supply functions were always maintained by the same group of programmers. Only this group was allowed to modify these functions.

The introduction of the R2S2 system had a noticable effect on the user organization. New supply functions had to be established at the CSF, while many old supply functions were no longer necessary. Learning how to log-on and use the system became a new skill almost everybody at the CSF had to acquire. Old supply procedures had to be modified and new ones had to be written to reflect the automation now in place. Ultimately, a new procedural manual had to be published and disseminated to the users. The DP manager at CSF attained a new and an important status in the organization, since the automated aupply system was his responsibility. Further, supply procedures could no longer be changed without proper consultation with the DP section of CSF.

With the adoption of the R2S2 system, numerous users were involved in completing one supply function. Subsequently, change requests needed to be negotiated and agreed to before being implemented. Strict configuration control were also necessary to control multiple changes to the same function without the risk of conflict. It was increasingly apparent that the informal documentation procedures adopted at the DP section were no longer adequate.

Adopting a rigorous standard of documentation was impractical according to the DP manager. He believed that enforcing a strict standard of documentation would delay the implementation of essential modifications to the system. Such a standard would require additional DP staff training, and such costs were not within the budget and schedule constraints.

Modifications were now being requested much more frequently and were always characterized as "urgent." The senior CSF managers therefore approved a plan that adopted an upgraded form of the previous standard of practice for documentation. They also implemented various incentive programs aimed at reducing the rate of staff turnover. The idea behind such a scheme was to save the time and effort required to train and integrate new staff. This scheme also ensured that authors of most of the system modifications would be available personally to answer any inquiries.

At the time of concluding this study, the system had been in place for 2.5 years. The turnover rate for the maintenance staff over this period had been kept to less than 8%, which was the main objective of the incentive program.

3.2.1 Discussion of the R2S2 case

In examining the R2S2 case the following conditions can be identified:

(1) The documentation requirements in the DP section of the CSF changed when the size of system being run and maintained changed from the small ARX system to the larger R2S2. The change in the size reflected a change in the number of functions performed for the CSF organization, in the number of the programmers working in the IS department, and in the number of users depending on it. The increase in the number, variety, and degree of interdependence of system components thus increased *system complexity* [11].

(2) Unlike the situation when only the ARX system was running, the new R2S2 logistics system became an essential and critical part of the day-to-day operation of the CSF. Changes in the supply procedures at CSF needed to be reflected in the R2S2 system before they could be carried out. The ARX system can therefore be viewed as a *non-backbone system* (not operationally critical), while the R2S2 can be viewed as a *backbone* (operationally critical) system. The *importance of the system to the organizational work* can therefore be a factor that may have changed the overall documentation requirements in the IS section.

(3) When the longevity of the R2S2 system was identified as fairly short (3 years), the project managers adopted a scheme that reduced the rate of staff turnover. This helped reduce volume of documentation created or revised during system maintenance. The fact that the system was temporary seems to have justified the development of this scheme and contributed to its success. Therefore, it appears that the *expected longevity* of the system can influence its documentation requirements.

(4) The reduction of the staff turn-over by various incentive policies was used by the R2S2 management as an effective, practical, and economical substitute to elaborate documentation requirements. It was therefore plausible to consider the (low) *rate of staff turnover* influenced the change in the documentation requirements that occurred.

(5) Change requests for the ARX system were usually infrequent and routine, and rarely classified as urgent. By contrast, change requests for the R2S2 system arrived at a high rate and were almost always urgently needed. The fact that the requests were infrequent and routine in one system, but frequent and urgent in the other signified that one system had *slowly evolving* requirements, while the other had *dynamically evolving* requirements. It was therefore reasonable to identify the *rate of changes in the system requirements* contributed to the change in documentation requirements.

(6) The ARX system was a batch system while the R2S2 system was an on-line system. The *type of system* thus can affect the pattern of production and consumption of documentation. However, changes made in maintaining ARX were generally routine and not urgent, while changes requested for R2S2 were increasingly urgent as schedule deadlines approached. Thus, constraints on the *users time schedule* may also be considered as yet another condition to consider.

3.3 The LOG case

This case describes a large development effort for a supply and maintenance system, called the LOG system. The system was intended to be in operation for 10 to 15 years. Further, the LOG system, upon completion of its development, would replace the temporary R2S2 system described in the previous case. The Information System's (IS) department was charged with supervising its development.

Five international software development companies were invited to research the requirements, and to propose a fixed price and time bid for its development. Although one contractor was quickly selected, the project ran into financially-related delays. Meanwhile the IS department started a new data processing (DP) center. By the time the LOG contract was finally signed, the DP center had become a well established entity that operated and developed systems for the various STBG functions, including the T&T system that was discussed earlier.

The center had some 20 programmers and analysts and was divided into three sections: operations, development and maintenance, and quality assurance. It had a set of standards and procedures that were a subset of those imported under a previous contract from a major aerospace company. The standards called for a *sign-off* of the requirements by a user representative, and for a formal design *walkthrough* with the quality assurance section before coding would start. The standards detailed which documents were to be produced and in what format. However, there were no formal procedures for reviewing their completeness or consistency. In most situations, programmers that participated in the development effort were designated as the maintainers of the system after its development. These programmers conducted or supervised the maintenance functions even though they may be involved in other development efforts at the center.

The LOG system had a star architectural structure with minicomputers distributed in the remote sites, and one mainframe in the center to coordinate common functions. A project officer was assigned to the project from the IS department. His main task was to coordinate with the supply and maintenance departments. Although most of the real end-users were geographically dispersed, both the contractor and the project officer considered the supply and maintenance departments at the STBG main corporate offices as their main users. It was the responsibility of these departments to coordinate the requirements of all the remote sites.

The new supply system was to adopt a completely new logistical system philosophy known as a *push* system, as opposed to a *pull* system that was previously used at STBG.[1] This meant that totally new supply and maintenance procedures had to be realized and written. Although the supply department had gained some experience in automating the supply functions during the R2S2 implementation, the *push* philosophy was a completely new concept. The adoption of this new philosophy entailed the redefinitions of boundaries within the supply and maintenance departments as well as between them. It is in this context that the sign-off of requirements and other documentation by the users took a political flavor. The documentation of the system became a platform for airing an old inter-departmental rivalry in addition to describing the system behavior.

The contractor had its own organizational standards and procedures. These standards adopted a formal and hierarchical management structure which encouraged centralization of decisions. This management style might have had its advantages. However in practice, many staff indicated that it caused delays in resolving problems, and occasionally worked against advancing the project that was under fixed time and price constraints.

The contract was designed to link payments to a list of deliverable documents (e.g., System Requirements, Software Specifications, etc.). This was an article the STBG negotiators added to the contract as one way to relate the contractor's progress to some tangible product. In some particular cases, the contractor had to receive the approval sign-off from the users on certain documents (e.g., Software Specifications, Testing Procedures) before payment. These documents therefore assumed a new meaning to the contractor since it became the unit of measure of progress, hence payment. One senior technical person indicated that the list of documents in the contract was their "methodology." The development team, with the blessing of senior management, thus became more interested in producing the documents on time rather than following a particular IS engineering method.

As it turned out, this project suffered numerous delays and difficulties. Although signing-off specifications and requirements by users was standard practice within the STBG company, the sign-off of LOG requirements and specifications were not easy for the contractor. First, it was an external contractor and the users were more suspicious of the contractor's intentions. Second, the sign-off was contractually related to the payment made to the contractor and hence was contractually binding. This was in contrast to the more accommodating atmosphere of the IS data processing center where documentation mistakes were addressed with little or no embarrassing questions. Third, the LOG system was introducing entirely new supply and maintenance concepts that were changing

[1]A push system detects spare parts requirements and pushes them to the appropriate site. A pull system requires a site to request the type and quantity of the spare parts that it needs.

many of the work arrangements that had been in place for many years. This meant that the users needed more time to research and understand the proposed system. It also meant that arguments over the different interpretations of the requirements, and invariably requirements changes, were a common feature of this development effort. These difficulties persisted even into the coding phases of development. Requirements changes became a major problem and were only checked after a senior STBG manager issued a directive freezing requirements and forbidding any changes until the system was fully developed.

3.3.1 Discussion of the LOG case

Documentation in this case was clearly needed as a means of advancing the development effort. It seems plausible that the contractor recognized from the outset that the list of deliverable documents was only a list of milestone products for the development process. Accordingly, this list did not necessarily reflect the actual documentation required to support the development process. The case also reveals that it is difficult to accurately predict at the outset of a project the actual iterative sequence of development phases. Furthermore, predefining this sequence and adhering to it in a very strict way, as the contractor did with the list of deliverable documents, can give rise to serious difficulties during development.

In analyzing the LOG case, the following conditions can be identified:

(1) The LOG documentation process and products were always overshadowed by the fact that the development was a contracted, *out-of-house* effort. This meant that almost all life cycle documents were a contractually deliverable product. This contractual arrangement emphasized that people involved in approving the documentation were accountable for their decisions. This also attached a special meaning to documentation and imposed a particular organizational formalism in the production and consumption of documentation. In contrast, accountability and deliverable documents were not particularly emphasized for systems developed by the IS department for the *in-house* use of the STBG corporation. Consequently, the two types of development efforts pursued a completely different set of documentation processes and products. It is therefore reasonable to claim that *the type of development effort* (in-house vs. out-of-house) affects how and what documents will be produced, as well as how they will be consumed.

(2) The adoption of the push supply philosophy meant that the supply department was unfamiliar with the proposed concepts of the system under development. This was the situation even though they had some experience during the R2S2 automation effort. With respect to the maintenance department, the LOG system was their first automation effort and hence they were unfamiliar with the new concepts and powers that such an automated system offers. This user unfamiliarity seemed to be a major factor in the long delays in signing-off the requirements and other delivered documents. It also seemed to be the main

reason behind the almost constant stream of system changes that were characteristic throughout all the development phases. It is therefore reasonable to suggest that (the lack of) *user familiarity with similar automated systems* can affect the documentation requirements in the setting.

(3) The contractor was an international software company that had its own well-established management standards and procedures. These standards relied upon a hierarchical management structure and centralized decision making process. This management style set a rather formal method of communication within the developer's own organization. This may imply that *the extent of formalism in interorganizational transactions* affects the documentation requirements of a system.

4 Comparative Analysis and Hypothesis Construction

The three preceding case studies collectively identify twelve variable conditions that affect the documentation products and processes (see Table 1). The discussion following each case shows how each of these variables is developed from particular circumstances in the case study. A close look at the twelve variables reveals that they may be classified along the technical and organizational attributes of the system, of its developers, and of its users.

To further investigate the relationship between documentation products, processes, and their organizational settings, we select one *representative* variable from each class. This allows us to examine each of these variables across the three cases, without an individual examination of each of the twelve variables. Such an examination would require lengthy analysis, well beyond the practical limits of this paper. Thus, we choose the following three variables, having selected one from each category identified in Table 1:

1. The importance of the system to the overall organizational work
2. The type of development effort
3. User familiarity with similar automated systems

The argument for selecting a particular variable to represent its class is not critical since this classification is not unique. The set of relationships constructed here are therefore only a sample of the possible relations that may be developed. Thus, other variable selections can accomodate different analytical topics for investigation. This in turn can add further substance and depth to the set of IS documentation requirements we present in the remainder of this paper. However, the challenge faced in selecting any group of variables from the set is to determine whether the conclusions or implications reached contradict those derived from some other variable group. If such contradictions are

Variables relating to the software system itself

- Type of system (case 2)
- Importance of system to overall organizational work (case 2)
- Rate of change to the system (case 2)
- System longevity potential (case 2)
- Complexity of development effort (case 2)
- Interconnectivity of system with other system (case 1)

Varibles relating to the IS developers

- Type of development effort (case 3)
- Relation between development and maintenance units (case 1)
- Rate of staff turnover (case 2)
- Extent of formalism in organizational transactions(case 3)

Variables relating to the IS users

- User familiarity with similar automated systems (case 3)
- User time schedule constraints (case 2)

Table 1: Classifying documentation variables into three classes

found, then the analysis of the individual cases in conflict is misstated or wrong. While not a true validation, our anecdotal result is that we have examined the complete set of variables and we could find no such contradictions. Nonetheless, the reader can independently review the data presented, the variables identified, and determine whether or not our result holds.

Table 2 summarizes the relations that we found from further examining and comparing the three case studies. Our objective now is to present these hypotheses by relating the three selected variables to various documentation products and processes. As such, what follows are the comparative case analyses we used to develop these hypotheses.

H1: The R2S2 case indicates that informal set of documentation standards that supported the ARX software were no longer adequate when the R2S2 software became operational. The new documentation standards were needed to maintain accountability of, and coordination between, system changes. Since both software systems were run in the same DP center, it seemed plausible that the reason for this change in documentation requirements was inherent in the differences between the purpose of each system. The ARX system was a

1. Backbone systems require a more formal documentation process than non-backbone systems.
2. Backbone systems require documents with a high degree of traceability
3. Out-of-house development/maintenance efforts undergo documentation processes that are more formal than those of in-house.
4. Out-of-house developments require a high degree of traceability
5. In-house developments can accommodate low consistency and completeness but require accurate details
6. In-house development efforts require less documentation than those of out-of-house development efforts.
7. Users unfamiliar with similar automated systems follow a different documentation process than familiar users.
8. Users unfamiliar with similar automated systems require documents that are more consistent, complete, and traceable

Table 2: Summary of constructed hypotheses.

non-backbone system that performed market analysis to detect specific market trends. The R2S2 system automated all the issuing, receiving and distributing functions of the CSF and hence was a *back-bone* system. It seemed significant that accountability and coordination appeared to have become questionable only after a back bone system started running in that DP center. The issue of accountability was also prominent in the LOG case, which was also a back bone system. This may indicate that *backbone systems require a more formal documentation process than non-backbone systems.*

H2: Another feature that was common in both the LOG and R2S2 cases was how some system functions were used by more than one organizational unit. Changes to such functions must therefore be coordinated with all related users. This would require that functional requirements documents can easily be traced through the various documentation products to their programming modules (and vice versa). The assertion that *backbone systems need documentation products with a high degree of traceability* therefore seems plausible.

H3: In the LOG case, the customer/contractor relationship created additional significance to the documentation of the system. For the contractor, documentation products became a contractually required deliverable in addition to (or rather than) a product that supports a systematic IS development process. For the customer, the documentation products became a measure of the contractor's progress. These additional meanings imposed a set of documentation processes and products that were fundamentally different, and more formal from those required for in-house developments in the same DP center. It is therefore reasonable to claim that *out-of-house development/maintenance efforts undergo documentation processes that are more formal than in-house efforts.*

H4: In the LOG case, the contractor was obliged to show that its IS had contractually satisfied the system's requirements. Sound project management would therefore require the tracking of how particular sets of requirements are satisfied through a set of designs, programs, and tests. Accordingly, the contractor would not want to develop more (or less) than what was contractually required. Therefore, it is reasonable to assert that *out-of-house developments require documentation products that are characterized by a high degree of traceability.*

H5: All three DP centers examined in the cases were (at some point in time) engaged in developing and maintaining systems for their organization's use. Programmers that were involved in development efforts generally stayed in the DP centers to perform maintenance functions and/or to become involved in new developments. In such centers, the interactions between developers and users create a supplementary tacit knowledge base that captured the history and circumstances of the development process. This kind of informal undocumented knowledge is difficult in practice to record in documents. This is the kind of knowledge that nobody took the time to write down and incorporate into the extant documentation products. Instead, it was preserved in the collective minds of those who were involved. In the R2S2 case study, this was the knowledge base that management sought to preserve when they transferred 20 programmers (together with the R2S2 software) from one DP center to another. In the T&T study case, it was the loss of this knowledge base, as NRP moved from one DP to another, that most probably initiated the additional documentation upgrade. This undocumented knowledge base seems to offer an important means of filling in gaps and/or resolving inconsistencies in the documents describing software system structure, behavior or operation. It may therefore be reasonable to assert that *written documents produced by in-house development/maintenance efforts can accommodate low consistency and completeness characteristics when undocumented knowledge can be sustained.* However, the documents that are produced necessarily need to *accurately capture details* that are easily forgotten.

H6: A common feature that was seen in both the T&T and the R2S2 cases was the programmers' frequent use of a particular subset of the system's documentation. This subset was the most preferred documentation. It was regarded as the most accurate and consequently was kept reasonably up to date. This indicates that *in-house development/maintenance efforts require less documentation than out-of-house efforts.*

H7: In the LOG development effort, particular difficulties were encountered in defining and freezing a set of system requirements. One can argue whether such changes should be allowed at all. However, these changes were made and are indicative of the difficulties that the user had in defining the system's requirements. The changes occurred as the details of the system unfolded during the specification and the design phases, and persisted even through the coding phase. It seems that as the implications of initial decisions became clear, the users found it necessary to either redefine requirement details or to re-interpret them. The fact that the users were generally unfamiliar with similar software systems seemed to be the underlying factor for these changes. As the users interacted with the developers, they became more familiar with the powers and limitations of the proposed system. They were then able to better evaluate and state their requirements. It is therefore reasonable to claim that *users unfamiliar with similar automated systems require documentation production/consumption patterns that are different from those of users that are familiar with automated systems.*

H8: Following from (H7:), unfamiliar users need documents that are consistent, complete, and traceable when compared to users of similiar systems. It may also be plausible to claim that because of these frequent change requests, *maintaining the consistency and completeness* of documentation products may become troublesome. *Traceability* of the functional requirements throughout the various system development life cycle transformations may also be a valuable characteristic to keep track of such changes.

The preceding eight hypotheses form a basis for identifying the requirements for more effective IS documentation products and processes. They also indicate the salience of certain mixes of documentation product and process features may vary in predictable ways in different organizational settings. This is another way of saying that our hypotheses can be experimentally tested in other settings with IS development efforts underway as a way to further confirm or refute their validity. Such predictions, which represent the conditions for validating the accuracy and robustness of the hypotheses, therefore serve as the basis for establishing testable requirements for software system documentation that can either be empirically substantiated and refined, or else unconfirmed or refuted.

5 Conclusions and Implications

We introduced this paper with a review of the many problems and proposed solutions of software system documentation. These solutions generally represent a growing set of attributes that IS development documents should possess. However, through our case studies, we found that these documentation product attributes are not all equally applicable or critical in all settings for all kinds of software systems. Instead, through a comparative case study and analysis, we found a set of relations that interlink documentation products, processes, and settings. Thus, our resulting hypotheses suggest a more optimal refinement for mixing and matching IS documentation product attributes with documentation processes and settings. Furthermore, from a practical standpoint, our hypotheses may suggest that producers of custom systems or "off-the-shelf" software packages should consider that they may best satisfy their customers by providing different kinds of documentation, or different documentation "views" according to the particular hypotheses that most closely reflect their customers situation.

The eight hypotheses we identify in the preceding section do not represent a complete set of relationships between IS documentation products, processes, and settings. They do however constitute the beginnings of a new or more refined understanding of what the requirements are for developing documentation and documentation processes in ways that are tied to how the associated information systems are being developed and used. Such an understanding in turn outlines a more practical set of requirements for what kinds of documentation tools, techniques, and standards may be most applicable and effective in different organizational settings.

While we hold that our "requirements as hypotheses" are substantiated through their constructive empirical grounding, they are intended to be extended and refined through further empirical experimentation and measurement, following the tradition of grounded theory research methods [4, 13]. In this sense, our *hypotheses are the requirements* and *the requirements are hypotheses* for resolving or clarifying the litany of IS documentation problems we cited in the Introduction. As hypotheses, they can be treated as testable predictions that can be confirmed, revised or refuted in other IS development efforts. As requirements, they stipulate the conditions or constraints of how software or IS documentation should be produced so as to best facilitate its consumption. In both situations, the documentation that results from a software or IS development effort will serve as the data that (i) substantiates or refutes the hypotheses, and (ii) successfully or unsuccessfully validates that the requirements have been fulfilled.

Last, the organizational settings where IS documentation is produced and consumed shape the quality and utility of the resulting IS documents. These settings represent the workplaces where software system developers, users, and the focal software system interact with each other throughout the documentation process. Through our three case studies, we identified twelve variables that characterize the organizational settings that mediate IS documentation products and processes. Previous efforts to construct IS documentation standards or automated documentation management tools do not in general identify nor address the significance of the organizational settings of IS documentation. Yet this study corroborates the results from many other empirical studies that indicate that the statics and dynamics of organizational settings both constrain, and are constrained by, local IS development efforts. Thus to ignore these relationships when developing documentation standards or automated support tools is to miss a critical set of requirements for what can influence the quality and effectiveness of IS documentation products and processes. Instead, we recommend that such standards and tools should not be generic, but should be specialized to the settings, as well as to the documentation products and processes through which they are produced and consumed. Realizing such a recommendation we feel will increase the likelihood of producing IS documents that will be found to be of higher quality and usability with less effort.

Acknowledgements: Preparation of this report was supported through contracts and grants with AT&T, Office of Naval Technology through the Naval Ocean Systems Center, Pacific Bell, and the USC Center for Operations Management, Education and Research (COMER). However, no endorsements are implied.

References

[1] B. Blum. An approach to computer maintenance software documentation. In *Proceedings of the NBS FIPS Software Documenation Workshop*, pages 110–118. NBS FIPS, 1982.

[2] J.R. Brockmann. *Writing Better Computer User Documentation: From Paper to Screens.* Wiley-Interscience Publication, John Wiley and Sons, New York, Chinchester, Brisbane, Toronto, 1985.

[3] P.K. Garg and W. Scacchi. A Hypertext System for Managing Software Life Cycle Documents. *IEEE Software*, 7(3):90–99, 1990.

[4] B. Glaser and A. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Aldine Publ. Co., Chicago, IL, 1967.

[5] S.D. Hester, D.L. Parnas, and D.F. Utter. Using documentation as a software design medium. *The Bell System Technical Journal*, 60(8), October 1981.

[6] F. Lehner. Quality Control in Software Documentation Based on Measurement of Text Comprehension and Text Comprehensibility. *Information Processing and Management*, 29(5):551–568, 1993.

[7] T. Nakajo, I. Azuma, and M. Tada. A Case-History Development of a Foolproofing Interface Documentation System. *IEEE Trans. Software Engineering*, 19(8):765–773, 93.

[8] A.J. Newmann. *Guidlines for Documentation of Computer Programs and Automated Data Systems.* National Bureau Of Standards, Institute for Computer Science and Technology, NBS, Washington, DC 20234, 1982.

[9] A.J. Newmann. *Management Guide for Software Documentation.* National Bureau Of Standards, Institute for Computer Science and Technology, NBS, Washington, DC 20234, 1982.

[10] D.L. Parnas and P.C. Clements. A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*, SE-12(2), 1986.

[11] T. La Porte. *Organized Social Complexity.* Princeton, New Jersey, 1975.

[12] R. Ravichadran and H. Banerjee. Support for Information System Usage in Banks. *Intern. J. Information Management*, 14(1):5–12, 1994.

[13] A. Strauss and J. Corbin. *Basic of Qualitative Research: Grounded Theory Procedures and Techniques.* SAGE Publications, Newbury Park, CA, 1990.

[14] D. Teichroew and E. A. Hershey III. PSL/PSA: A computer-aided technique for structured documentation and analysis of information systems. *IEEE Transactions on Software Engineering*, SE-3(1):41–48, January 1977.

[15] United States Department of Defence. *Automated Data Systems Documentation Standards*, 1977. DoD Standard 7935.1-s.

[16] R. Williamson. *Software Documentation Support System.* PhD thesis, Computer Science Department, University of Southern California, Los Angeles, 1985.