Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1995 Proceedings

Americas Conference on Information Systems (AMCIS)

8-25-1995

Distributed Computer Environment Software Maintenance:System Complexity Versus Component Simplicity

Scott L. Schneberger
Georgia State University

Distributed Computer Environment Software Maintenance: System Complexity Versus Component Simplicity

Scott L. Schneberger
Georgia State University

The business computing world is undergoing a significant transition from centralized computer architectures, typified by single mainframe computers, to non-centralized or distributed computer architectures typified by client-server systems. Numerous industry trade journal articles tell of an increasing number of companies moving to distributed computing environments and the wide range of issues they face in doing so. Of the many technical and management issues surrounding this architectural transformation, one of the most significant involves software development and maintenance. Software maintenance--correcting coding mistakes, adding or improving software functionality, and perfecting software to make it more maintainable--is particularly crucial to industry. Software maintenance has been repeatedly identified as the largest single life cycle cost of information systems (Jones, 1994) as over $100 billion is spent annually on software maintenance in the U.S.

A paradox appears, however. In terms of hardware and software, there is an emerging perception that computer systems based on smaller, cheaper computers, operating systems, and applications will have correspondingly lower software maintenance costs (Bozman, 1993). But other authors express concern that these very same features will *increase* software maintenance costs due to system complexity. A 1994 survey of *InformationWeek* 500 IS executives revealed that 45% believed that client-server computing environments will save their company money, while 32% said they would not (Caldwell, 1994).

This exploratory research was centered on the issue of whether software maintenance is more difficult for distributed computer operating environments than for centralized environments. It uses cues from trade journal articles that this issue revolves around the characteristics of two key diametrics of information system architectures: *component simplicity* and *system complexity*. If the simplicity (and therefore, ease of use) of components like PCs and PC operating systems has a greater effect on software maintenance than the overall complexity of the system, then it is reasonable to assume software maintenance costs will be *reduced* with increased componentization and simplification. If, on the other hand, the complexity of system architectures has ascendancy over component simplicity, then software maintenance costs will *increase* with increases in system complexity.

The deeper, theoretical issue is that of system composure; defining information system complexity, identifying relative strengths of system complexity factors, and identifying relative strengths of components. The primary research question is: *What are the perceived explanatory factors of software maintenance difficulty in terms of the*

computing environment? Answering this question will better enable IT executives to manage distributed computing environments (and their budgets) by maximizing beneficial factors (like component simplicity) while minimizing detrimental factors (like system complexity).

Research Theoretical Bases

This research was based on the theory that operational computing environment factors affect software development and maintenance, the factors of information system complexity, and information system components--as explained below.

Computing Environment Effects. Previous software maintenance research could be grouped into three main causal factors: the characteristics of the software itself (such as its size, modularity, and complexity), the development environment (people, tools, and methodologies), and user needs and involvement. But related research in information systems, organizational theory, and operations research suggests a fourth key factor in software maintenance that has not been fully researched: the operational computing environment--the environment in which system and application software executes--as highlighted in Figure 1. It is postulated that this factor has not been well-researched because the computing environment--until recently--has been relatively stable based on the centralized, mainframe environment predominated by a few vendors. Revolutionary movement to distributed, more vendor-diverse environments such as client-server, however, brings this factor to the forefront for research.

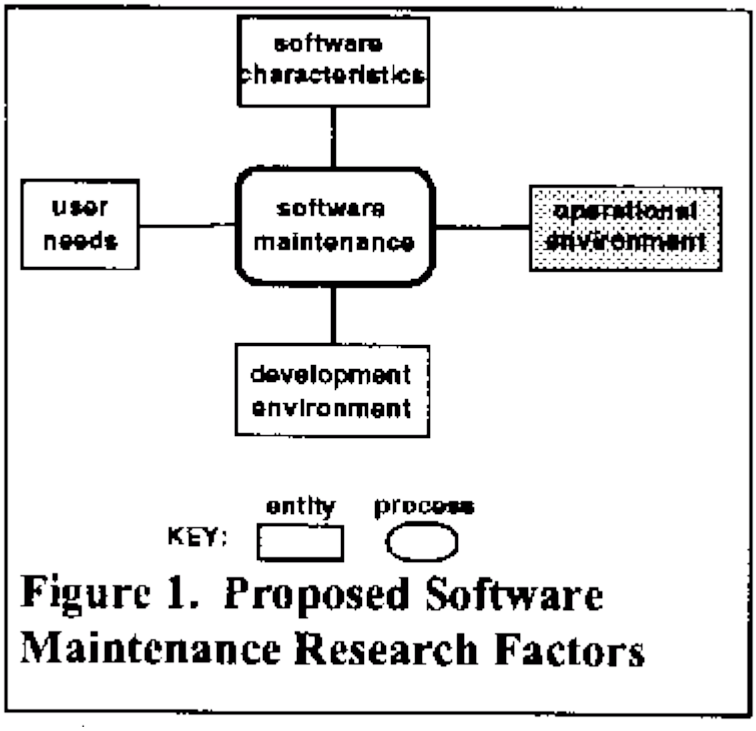

Figure 1. Proposed Software Maintenance Research Factors

Computing Environment Complexity. In general, most people agree that the higher the *number* of components (and therefore also the number of interrelationships), the higher

the level of complexity (Bunge, 1963). Indeed, one of the earliest writings about computing environment complexity describes it in terms of the number of components and their interactions (Langefors, 1973). But the *variety* of the components--and the effort required to deal with their differences--has also been cited as a complexity factor (Welke, 1983). In the same vein, it is proposed that the *variety of interactions* is also a factor of computing environment complexity. Finally, this paper proposes that the *rate of change* in components and interactions would be an additional complexity factor; the higher the rate of change, the more difficult to understand a system. Taken together, this paper suggests that computing environment complexity is based on *the number and variety of components and their interactions, and the overall rate of change* as graphically shown in Figure 2.

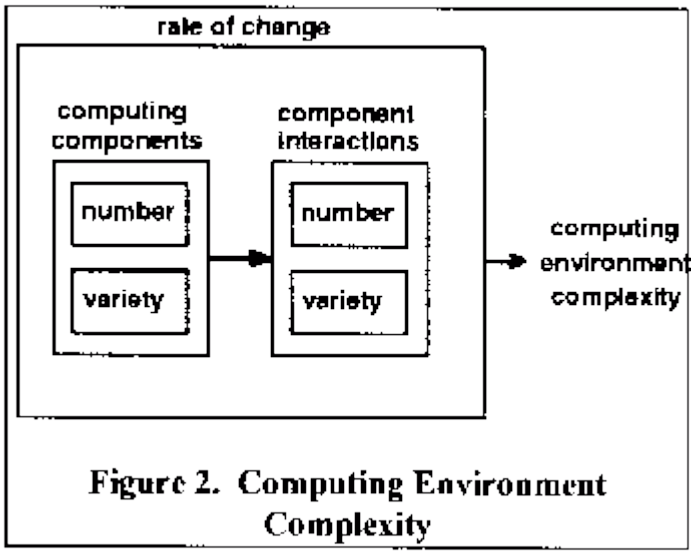

Figure 2. Computing Environment Complexity

Computing Environment Components. There is ample literature revealing the wide range of possible computing components from hardware to software to data. In general, many IS textbooks categorize these information system components as shown in Table 1.

Research Methodology

Since this was largely exploratory work in a new area, empirical data collection began with a pilot study to validate perceived factors, identify new significant variables, refine a survey instrument, and gain insights from qualitative interviews with programmers, designers, and project managers. A survey was given directly to 155 selected maintenance programmers in four large corporations: IBM Rochester (MN), IBM Austin, BellSouth Information Systems Atlanta, and AT&T Network Systems, Atlanta--professionals chosen by supervisors to maximize the breadth of software maintenance experience in both centralized and distributed computing environments. The survey sought to collect their professional perceptions on (1) the effects on maintaining software due to increases in the computing environment complexity factors (as well as ranking them) using Likert scales to indicate the relative degree of increased difficulty from "no difference" to "much more difficult," and (2) the relative ease of use of specific

distributed computing environment components versus centralized components (with rankings). Demographic data was also collected on each individual and the development environment.

HARDWARE	SOFTWARE	INFORMATION	PROCEDURES
processores	systems	systems	data preparation
storage devices	applications	applications	system operations
input/output devices			user
telecomm. devices			

Table 1. Information System Components

In-depth, qualitative interviews were also conducted with 30 systems analysts, designers, programmers, testers, installers/operators, and customer service representatives seeking direct answers to the research question and mitigating factors perhaps not identified in the formal survey. The interviews were process-based; the interview flow was based on the maintenance process (i.e., analysis, design, coding, testing, and implementation).

Data and Analysis

Survey data was statistically analyzed using standard t-tests for individual factors and Hotelling's T2 for grouped factors (differences between the means and null means of "no difference"). Rankings were used to check the overall reliability of the mean scores. Qualitative data was categorized in terms of computing environment factors, and summarized.

Surveyed and interviewed people indicated (all survey results significant at =.05):

1. Distributed architectures are generally more complex and more difficult for software maintenance than centralized architectures--but involve components that are easier to deal with individually. The net effect is that the complexity of present systems overwhelms the simplicity of their pieces--making maintenance harder overall. Moreover, software maintenance difficulty increases at a much higher rate from increases in system complexity than software maintenance decreases from increases in component simplicity.

2. Effects from the *variety* of components outweigh the effects of component *number*. In particular, the variety of *processors, system software programs,* and *telecommunications devices* led all component variety and number complexity factors.

3. The computing environment complexity factor that makes software maintenance the most difficult is the *rate of change*. The factor with the least effect is the number of interactions or transactions.

4. The top three (out of 12) individual distributed computing environment components that made distributed computing software maintenance the easiest were personal computer (PC) development tools, PC interfaces, and PC software packages. The individual distributed computing environment component that made software maintenance the most difficult was the rate of technological change associated with PCs.

5. There was no significant relationship between individual job title, years professionally programming, or years programming in either centralized or distributed environments with the results noted above; the findings held for all groups.

Conclusions and Implications

These findings suggest a new model (Figure 3) for distributed computing environment software maintenance (and, by relation, development). As computing environments become more distributed, the individual system components become less complex and easier to deal with, but the system as a whole becomes more complex and harder to manage. The paradox of opinions on distributed computing noted earlier is likely due to one's perspective; those focusing on distributed computing pieces see decreasing costs while those thinking of the system as a whole see increasing costs.

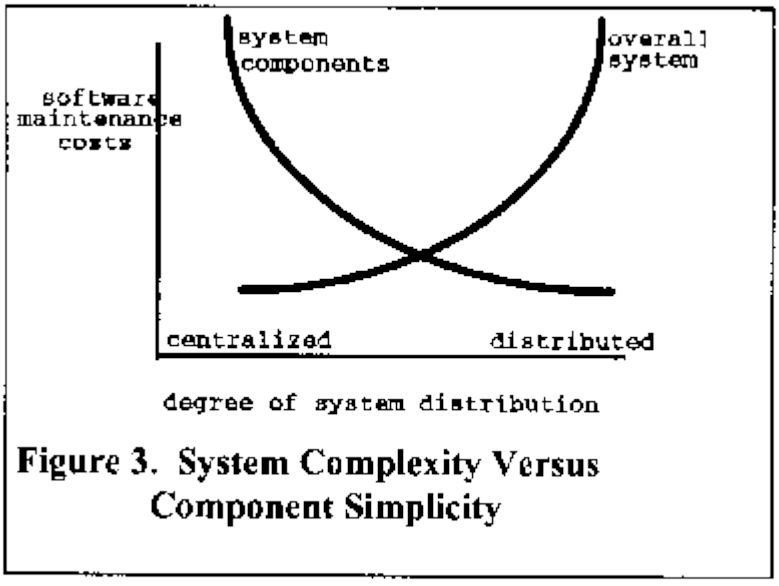

Figure 3. System Complexity Versus Component Simplicity

This model suggests that IS managers who want the benefits of distributed computing (such as architectural flexibility and scalability with many "smaller," cheaper, and readily available components) but who also want to minimize overall system costs have two basic choices: (1) lower the system component and overall system cost curves or their slope, or (2) adjust the degree of system distribution.

The findings suggest that a manager can lower the component cost curve through minimizing component change rates (e.g., lengthen the time between new versions), and by taking advantage of PC graphical development tools, graphical interfaces for applications, and packaged software. Managers can lower the system cost curve through increased system standardization by minimizing component variety (especially processors, system software programs, and telecommunications devices), and by decreasing the rate of system change (e.g., lengthening the time between major hardware and software changes).

Further Research

The perceptions of professional software maintainers need to be empirically tested with actual field observed costs or laboratory experimentation. In particular, data is needed to confirm the concluding management model, to quantify the component and system cost curves in relation to the degree of distribution based on the complexity model proposed, and to determine if the effects of the two curves are additive or multiplicative. Empirical data from case study attempts to lower the component and system cost curves--and the effects on overall costs--is needed.

References

Bozman, Jean S. "Grocer Buys Into Open Systems." *Computerworld* 27:12 (March 22, 1993): 57, 59.

Bunge, Mario. *The Myth of Simplicity*. Englewood Cliffs, NJ: Prentice-Hall, 1963.

Caldwell, Bruce. "Looking Beyond the Costs." *InformationWeek* (January 3, 1994): 50-56.

Langefors, B. *Theoretical Analysis of Information Systems.* Sweden: Auerbach, 1973.

Welke, Richard. "IS/DSS: DBMS Support for Information Systems Development." In C.W. Holsapple and A. B. Whinston (eds.), *Proceedings of the NATO Advanced Study Institute,* Estoril, Portugal (June 1-14, 1981). Holland: D. Reidel, 1983.