The Influence of the Information Systems Development Approach on Maintenance

By: Sasa M. Dekleva
School of Accountancy
De Paul University
25 East Jackson Boulevard
Chicago, Illinois 60604 U.S.A.

Abstract
This is an exploratory field study that examines the influence of selected system development methodologies on maintenance time. A number of factors related to the early stages of information systems development and to information systems maintenance were investigated: development methodology, maintenance time and its allocation, number of users, their understanding and involvement, system documentation, software quality, system characteristics, project controllability, system size and age, organization of the maintenance function, use of tools, ability of personnel, stability of organization, and others. The survey findings do not support the proposition that the application of modern information systems development methodology decreases maintenance time. However, some benefits are identified. Time spent on emergency error correction, as well as the number of system failures, decreased significantly with the application of modern methodology. Systems developed with modern methodologies seem to facilitate making greater changes in functionality as the systems age.

Keywords: Information systems, software maintenance, software engineering, systems development, systems analysis, systems maintenance.


Introduction
Dissatisfaction with the effectiveness of the traditional systems development life cycle (SDLC) approach to building information systems (IS) during the late sixties and early seventies led to a search for better methodologies. Several have been proposed to improve development productivity and quality. There is, however, a discomforting disparity between the proposed benefits of modern methodologies and the findings of empirical studies. There is little empirical evidence linking the use of modern IS development methodologies with improvements in development productivity and quality.

This study evaluates the benefits of modern IS development methodologies from the perspective of systems maintenance. Analysts define and document a system’s functions and, to a large extent, its structure. Modern methodologies give more weight to early phases of IS development and are believed to improve the process and outcome of systems analysis. Both the completeness of functional specifications and improved qualities of a system’s structure and documentation are often expected to reduce maintenance time. This study questions this expectation. Maintenance time is a predominant component of the maintenance cost structure. Maintenance, in turn, is the most expensive aspect of the IS life cycle (Parikh and Zvegintzov, 1983). Therefore, improvement in maintenance activities are even more important than enhancements of the development process.

Background
Much of the past research on the influence of different IS development techniques and methods was centered around the programming phase. Program development techniques have been shown to influence maintainability. In one study, 89 percent of modular programming users reported improved maintainability of programs (Hoskins Systems Research, Inc., 1973). Another claimed that the use of structured programming methodologies can reduce subsequent maintenance costs and effort by a 3:1 ratio (Lyons, 1981). Some studies, however, have been less supportive of the linkage between advanced development methodologies and main-
tenance time. A study at IBM (Fjeldstad and Hamlen, 1979) was unable to confirm the relationship between use of these techniques and maintenance. The respondents, however, were of the unanimous opinion that improvements are or should be significant. Investigators who evaluated the studies of the benefits of structured programming conclude that while the principles of structured programming are forceful, the empirical evidence to date is ambiguous (Vessey and Weber, 1984). The results of others investigating the benefits of analysis and design methodologies, in general, have implications for maintenance. Projects using a structured analysis and design methodology have been found to be less productive than projects following the traditional approach. However, according to Banker, et al. (1991), "Many of the presumed benefits of using a detailed methodology that requires a lot of documentation are not observed until the follow-on projects, when enhancements or repairs need to be made to the system" (p. 13). Better structure is expected to benefit maintenance by decreasing the ripple effect of changes (Gibson and Senn, 1988). Similar claims are made by Grudnitski (1984) and Martin and McClure (1983).

In other words, if development productivity does not benefit from the use of modern IS development methods, perhaps maintenance will. We also anticipate variations in benefits depending on the type of maintenance and that the time allocation for various types of maintenance may differ. For example, corrective maintenance may consume the time gained by the decreased need for corrective maintenance.

The most extensive study of IS maintenance so far was performed by Lientz and Swanson (1978; 1979; Lientz, et al., 1978). They noticed that IS development techniques may influence maintenance and called for related research (Lientz and Swanson, 1978). The correlation between system development and maintenance was identified also by Martin and McClure (1985), who called for program design that enables control over maintenance requirements and procedures. They proposed that maintenance time could be reduced by more than half with the implementation of modern IS development methodologies. Similar findings or conclusions were reported by Parnas (1979), Hetzel (1978), Heninger (1980), Higgins (1981), and Parikh (1982). Much of this research looked at the relationship between programming style and maintenance. The particular focus of this study is on the analysis phase of IS development as it influences maintenance.

Research Propositions, Model, and Variables

The anticipated contribution of modern methodologies to the improvement of the IS development has not yet been demonstrated. Perhaps that failure has come from looking for benefits in the wrong place. This study looks for those benefits by looking for improvements not in IS development but in the ensuing maintenance activities. We first ask whether modern methodologies benefit maintenance by lowering the maintenance time requirements. The high costs typically associated with the maintenance phase justified attention to this issue. We then ask if and how the allocation of maintenance time changes as a consequence of development methodology.

Stated more formally, the two propositions are:

1. Maintenance of IS developed by the modern development methodologies consumes less time than maintenance of systems developed using the traditional approach.

2. Allocation of time among different maintenance activities depends on the IS development methodology used in development.

The IS development methodology, which can be either modern or traditional, appears as an independent variable in both propositions. Maintenance time is a dependent variable in the first proposition, and its allocation among different maintenance activities is a dependent variable in the second. But the type of methodology is not the only factor affecting maintenance. It is necessary to control for those factors that are known to most influence maintenance or are believed to do so. Some factors have been recognized as important development or maintenance productivity factors, such as the size of the project, age of the system, and ability of personnel. Other potentially important factors were identified by 10 experts from academia and industry who participated in the survey instrument validation. Examples include an organization's
stability, number of IS users, characteristics of the application, organization of the maintenance function, and use of tools. The research model is shown in Figure 1. Each of the variables presented in Figure 1 is further described in the following section.

IS development methodologies

The classification and definition of IS development methodologies used for this study are compatible with those of Necco, et al. (1987). Four varieties of modern approaches are identified. According to Martin’s terminology, software engineering is a modernized version of the SDLC where analysis, design, and programming are performed in a structured fashion (Martin and McClure, 1985). It is a process-oriented methodology in the sense that analysis and design are centered around business processes to be automated or supported by the system. A second variety, popularized by Martin, is information engineering. In this data-oriented approach, most attention is directed toward data with an intention to implement them in a database so they can support any possible future processing requirements. A third distinctive methodology is usually referred to as prototyping. When applied during the requirements analysis phase, prototyping encourages user involvement and iterative refinement of a system model. A fourth approach seeks to automate some development tasks; it is called computer-aided software engineering (CASE). CASE is not a methodology in itself; it is an environment where one or more

Figure 1. Research Model
of the three modern methodologies is supported by computer. Integrated CASE tools support all
three approaches in a coherent fashion.

These four modern IS development approaches are often used in combination. Although they differ from each other, they also have a set of common characteristics that distinguish them from the traditional SDLC. The traditional IS analysis method does not produce an implementation-independent logical representation of a system's functions. All modern analysis methodologies do. Anticipated effects or properties of modern methodologies include (DeMarco, 1978):

- improved communication between users and analysts
- improved system structure
- improved system documentation
- increased importance of early development stages
- richer definition of system functions
- sequenced tasks and defined intermediate results

These effects justify the popular claim that modern IS development methodologies improve many aspects of systems, including their maintainability. An early understanding of system functions by IS users, for example, should decrease requests for changes and enhancements after the system is implemented, thus decreasing the demand for system maintenance. Modern methods generally stress the importance of system structure and provide measures for its evaluation, such as coupling and cohesion. A structure whose parts are unfunctional and as independent from each other as possible is considered superior because of an increased likelihood that the impact of a change is confined to a single module and that any undesirable ripple effect is limited (DeMarco, 1978). Improved documentation is also expected to make corrective as well as other maintenance procedures easier, more accurate, and, thus, faster (Yourdon, 1989). Increased attention to the early phases of IS development is believed to assure that system requirements are identified more completely, thus decreasing the need for later enhancements (DeMarco, 1978). Well-defined intermediate development results should improve the manageability of projects, including quality control, verification, and project estimation (DeMarco, 1982). All these aspects are in turn expected to improve the reliability and maintainability of a newly developed IS and therefore decrease the burden of maintenance.

Consequently, all observed systems developed with any one of the modern approaches or a combination were categorized together. These systems developed with a modern methodology were then compared with systems developed with the traditional SDLC approach.

**Maintenance time and its allocation**

"Software maintenance," according to one expert, "is the performance of those activities required to keep a software system operational and responsive after it is accepted and placed into production" (Osborne, 1985, p. 1). Maintenance includes studying and documenting existing systems, enhancing existing functions, adding new functions, detecting and correcting defects, answering users' and computer operators' questions, training new users and IS personnel, rewriting, restructuring, converting, program tuning, program management, and other activities related to an operational IS. This definition was printed at the beginning of the survey questionnaire to ensure that all respondents reported compatible time measurements. The first dependent variable is the average time spent monthly to maintain a particular IS.

IS maintenance is often structured into three categories of activity: correcting, enhancing, and perfecting (Osborne, 1985). Such a classification, however, is too broad to provide a good understanding of maintenance time consumption; other categories are needed. Implementing mandatory changes, normally imposed by external factors, was introduced as an additional category. Supporting users was also added because this important function does not fit into any of the above three categories. It has been reported that evaluation of change requests consumes a significant part of maintainers' time, so it was introduced as a separate category (Connell and Brice, 1984). The allocation of effort among the following seven maintenance activities was collected (listed according to general sequence of activities):
• implementing mandatory changes
• correcting errors
• implementing functional enhancements
• supporting users
• evaluating change requests
• rewriting, converting, rejuvenating
• other

The features of modern methodologies lead us to anticipate a number of influences on each of the above maintenance tasks. Improved documentation and structure should increase the speed of evaluating change requests, implementing mandatory changes, correcting errors, implementing functional enhancements, and supporting users. Early development of system documentation in collaboration with users should decrease the time spent evaluating change requests and making functional enhancements. Structured methods should also improve the ability of developers to manage complexity, thus decreasing the time spent in correcting errors. Improved systems structure, resulting in lower coupling among subsystems as well as higher cohesion, should also speed up the tasks of locating and correcting defects as well as enhancing and modifying functions. More understandable documentation should decrease user support requirements while simultaneously improving the quality of that support. Similar effects can be anticipated in the training of new users and IS personnel; the need to rewrite and restructure programs should be lower as well. Better knowledge of functional requirements should also improve the initial design of performance-critical parts of the system, thus lessening the demand for later tuning. Better documentation and improved structures should similarly ease conversions and facilitate the management of programs.

It was anticipated, based on the above, that virtually every aspect of maintenance should benefit from the application of modern IS development methodology. It was impossible to predict which of the above seven maintenance tasks would be influenced more or less than the others. Different allocation of maintenance time among these tasks was considered quite probable and was therefore investigated as a second dependent variable.

Control variables

Based on reports from previous investigations, published common beliefs, expert opinions, preliminary interviews, and general experience, several factors were expected to have an influence on maintenance time and its allocation independently of the development methodology. It was necessary to control for their possible influences. The control variables observed included:

• system size
• number of users
• system class
• system age
• maintenance organization
• use of software tools
• ability of maintainers
• organizational stability

Although some researchers (e.g., Christensen, et al., 1981; Wrigley and Dexter, 1991), found a very high correlation between the number of executable lines of code (ELOC) and more sophisticated software engineering measures of program size, number of ELOC is generally considered an inappropriate indicator of system size (Jones, 1986). To avoid dependence on only the number of ELOC, two other indicators of size were also used: the number of different data elements and the number of programs. All three variables were strongly correlated. For analysis purposes, the three measures of system size were combined with equal weights into a derived measure.¹

It was suspected that the number of system users may influence maintenance time. The number of change requests, for example, may be proportionate to the number of users. User support time will also likely increase with the growing number of users.

Grouping of IS into (1) data processing systems, (2) management information systems, and (3) decision support systems (Sprague, 1980) enabled us to control for the different types of IS. Maintenance time was expected to depend on the class of IS. Decision support systems, for example, are less well-defined than data process-

¹ The Cronbach alpha reliability coefficient for this composite scale was 0.90.
ing systems and may require more frequent modifications. The characteristics of the three system classes were described on the questionnaire. Managers were asked to categorize a system according to its predominant features.

It is commonly believed that a system’s structure deteriorates over time and that older systems require more maintenance than younger ones (Flaatlen, et al., 1992). It is also known that the maintenance type labeled as “repair” is most intensive just after the system’s implementation or major enhancements. The system later stabilizes. Although all systems surveyed had been operational for at least six months, age might still influence the maintenance time independently from development methodology. The number of months since the system became operational was used as a measure of system age.

Some software maintenance groups are organized as separate organizational units from those that are developing new systems. Swanson and Beath (1989a; 1990) observed a potential for quality, productivity, and user service improvement by the separation of maintainers from developers and suggested an innovative way to separate them. For this study, a categorical variable separated cases with maintenance organized separately from those cases with joint organization.

Many software tools have been developed to support system development as well as maintenance. They are promoted as a means of dramatically increasing productivity. The survey included a question asking, “To what extent are maintenance activities supported by the use of software tools?” Available responses were very great, great, small, very small, and don’t know. Nobody selected the last option. The “use of tools” thus became a categorical variable with four values.

Large differences in programmers' productivity levels have been recognized and measured by many researchers. Order-of-magnitude differences in productivity between so-called “super programmers” and marginal ones have been reported (Boehm, 1987). Managers were asked to evaluate the ability of the personnel who maintain the described system by responding to the question: Regarding these maintenance personnel, how would you rate their talents?

- extremely talented ___%  
- above average ___%  
- average ___%  
- below average ___%

Percentages provided by respondents are weighted by factors 3, 2, 1, and 0, respectively, and summed. The control variable “ability of maintainers” therefore was a continuous variable in the range of 0 to 300.

Conceivably, events such as extensive fluctuation of managers, mergers, changes of organizational structure, and changes of product line or service may be management/human resource factors that may influence the need and demand for IS maintenance. Assessments of organizational stability were gathered from IS managers using a question in which all the above events were mentioned as examples of instability. A five-level Likert-type scale was used for categorization of this control variable, called “organizational stability.”

### Methodology

One purpose of this study was to develop a new theory related to IS development and maintenance. The design of the survey instrument was notably influenced by that used by Lientz and Swanson (1980). In the pretest, the draft instrument was subjected to qualitative testing. Ten experts in academia and industry from several areas, such as research methodology, information systems, and software maintenance, were asked to individually evaluate the instrument. To further validate the instrument, a pilot survey of 15 managers of larger IS departments from two different geographical areas was carried out. Eight responded and also provided comments regarding the questionnaire. All suggestions were related to the form rather than the substance of the questions.

---

1 Since this is an extension rather than a repetition of their study, only two questions are exactly the same. Fourteen (or 30 percent) of the questions are based on their instrument but are slightly modified; 32 (or 66 percent) of the questions were developed specifically for this study. Some questions were intentionally redundant and placed at different parts of the questionnaire. Correlation analysis showed consistency among responses to similar questions. The instrument is available from the author.
A revised version of the seven-page questionnaire was mailed to 1,000 randomly selected IS managers under the auspices of the International Data Processing Management Association. Of these, 122 returned the questionnaires. This discouraging response provided an additional opportunity to strengthen the questionnaire. The survey form was reduced to four pages, polished, and restructured.

The final questionnaire consisted of two sections. For the first, respondents answered questions about a particular system they were familiar with. This was to be a system developed in their department that had been operational for at least six months. The second section contained questions related to the IS department and the organizational environment.

The research questions were examined using data collected from Fortune 500 companies. Large firms were chosen because the first survey revealed that modern methodologies were still not widespread. Other researchers have found that Fortune 500 companies are more likely than others to be using modern IS development approaches (Jenkins, et al., 1984). Five hundred individuals were surveyed, one per company. Questionnaires were mailed out to these IS managers in the summer of 1985.

One hundred twelve managers (22.4 percent), responded. Table 1 presents some demographics about the responding environments.

**Results**

In order to classify a system as developed with a traditional or modern analysis methodology, several indicators were analyzed. These factors included:

1. Respondent's classification into one of the following methodologies:
   - traditional SDLC
   - software engineering
   - information engineering
   - prototyping
   - computer-aided software engineering
   - other

2. Whether the logical system model was built
3. The name of the methodology
4. Whether the methodology was purchased
5. The type of documentation techniques used

The consistency of the above indicators was then examined. The system development methodology was accepted to be a modern one only when the respondent classified it as such and when the logical system model was developed before the system was designed. Four respondents answered that their system was developed without any methodology at all. Twenty-five respondents described their methodology as "unknown." Six cases had conflicts between responses to the two questions used as a classification rule. All these cases (35, or 31 percent), were excluded from the analyses comparing modern with traditional development methodologies. Of the remainder, 44 systems were classified as traditional and 32 as modern.

Multiple regression analysis was used to examine the predictability of maintenance time based on the independent variable, methodology, and the control variables system size, number of users, system class, system age, maintenance organization, use of software tools, ability of maintainers, and organizational stability. Table 2 presents descriptive statistics for the five continuous variables. Regarding system class, 48 systems classified as data processing systems, 18 as management information systems, and nine as decision support systems. Only five of 76 companies had maintenance organized separately from the development. Figures 2 and 3 present frequencies for the remaining two categorical variables, use of software tools and organizational stability.

Results presented in Table 3, which gives the analysis of variance procedure for testing the

---

1 Technical Publishing, the publisher of Datamation and other magazines, provided the mailing labels. Names were randomly selected from the publisher's database out of those who classified their position as data processing manager.

4 One of the responses was clearly recognized as an outlier (it described a large commercial communication software product and was exceptional not only by its size but also by its nature) and was eliminated from the analysis.

5 Notice large differences between median and mean values for maintenance time and number of users due to their skewness; system size was analyzed in its derived and logarithmic form.
Table 1. Sample Demographics (N = 111)

<table>
<thead>
<tr>
<th>Size (Employees in IS Unit)</th>
<th>Number</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 or less</td>
<td>22</td>
<td>19.8</td>
</tr>
<tr>
<td>7 to 12</td>
<td>26</td>
<td>23.4</td>
</tr>
<tr>
<td>13 to 45</td>
<td>21</td>
<td>18.9</td>
</tr>
<tr>
<td>46 to 100</td>
<td>21</td>
<td>18.9</td>
</tr>
<tr>
<td>Over 100</td>
<td>17</td>
<td>15.3</td>
</tr>
<tr>
<td>Not available</td>
<td>4</td>
<td>3.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Size (Programmers/Analysts)</th>
<th>Number</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 or less</td>
<td>21</td>
<td>18.9</td>
</tr>
<tr>
<td>4 to 6</td>
<td>20</td>
<td>18.0</td>
</tr>
<tr>
<td>7 to 20</td>
<td>25</td>
<td>22.5</td>
</tr>
<tr>
<td>21 to 50</td>
<td>23</td>
<td>20.7</td>
</tr>
<tr>
<td>Over 50</td>
<td>18</td>
<td>16.2</td>
</tr>
<tr>
<td>Not available</td>
<td>4</td>
<td>3.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Position in Organizational Structure</th>
<th>Number</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headquarters</td>
<td>62</td>
<td>55.9</td>
</tr>
<tr>
<td>Group</td>
<td>5</td>
<td>4.5</td>
</tr>
<tr>
<td>Division</td>
<td>39</td>
<td>35.1</td>
</tr>
<tr>
<td>Product</td>
<td>2</td>
<td>1.8</td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
<td>0.9</td>
</tr>
<tr>
<td>Not available</td>
<td>2</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Table 2. Descriptive Statistics of Continuous Variables From the Regression Model

<table>
<thead>
<tr>
<th>Variable</th>
<th>No. of Cases</th>
<th>Median</th>
<th>Mean</th>
<th>St. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintenance time (hours/month)</td>
<td>76</td>
<td>14</td>
<td>106</td>
<td>214</td>
</tr>
<tr>
<td>System size (derived measure)</td>
<td>76</td>
<td>924</td>
<td>992</td>
<td>222</td>
</tr>
<tr>
<td>Number of users</td>
<td>75</td>
<td>25</td>
<td>308</td>
<td>1,075</td>
</tr>
<tr>
<td>System age (months)</td>
<td>76</td>
<td>24</td>
<td>30</td>
<td>34</td>
</tr>
<tr>
<td>Ability of maintainers (scale 0-300)</td>
<td>76</td>
<td>200</td>
<td>211</td>
<td>70</td>
</tr>
</tbody>
</table>

significance of the independent and control variables in the model, show no support for the first research proposition. The class of methodology used for systems development does not seem to influence the time of maintenance. In other words, the use of modern IS development methodologies does not appear to decrease the overall maintenance time.

But the analysis does show that three of the control variables significantly affect maintenance time at or below the probability level of 0.05. These are: number of users, system size, and system age. As expected, it is more time consuming to maintain a large system than a small one.

The results also support the expectation that maintenance time increases with the increasing number of users. While the size of a system has long been recognized as a major factor influencing development and maintenance time, the
The number of users has not been previously related to maintenance time. These two variables are not highly related. Results show that number of users should be further tested as a predictor of maintenance time.

As anticipated, maintenance time increases with increasing system age. The correlation is low when all systems are grouped together, having a Pearson correlation coefficient equal to 0.28. When systems developed with modern methodologies and those with traditional methodology are analyzed separately, correlation coefficients between system age and maintenance time increase somewhat to 0.53 and 0.43, respectively. This indicates that maintenance time in each class correlates differently with system age.

Figure 3 shows smoothed lines representing maintenance time by system age for the two classes of methodologies.

Figure 4 shows that maintenance time for systems developed with the traditional approach slowly and steadily increases after they are used.
Table 3. Analysis of Variance Results for Testing the First Proposition

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>SS</th>
<th>df</th>
<th>f</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methodology</td>
<td>0.07</td>
<td>1</td>
<td>0.04</td>
<td>0.85</td>
</tr>
<tr>
<td>System size</td>
<td>11.10</td>
<td>1</td>
<td>5.39</td>
<td>0.03</td>
</tr>
<tr>
<td>Number of users</td>
<td>11.56</td>
<td>1</td>
<td>5.61</td>
<td>0.02</td>
</tr>
<tr>
<td>System class</td>
<td>0.51</td>
<td>2</td>
<td>0.13</td>
<td>0.88</td>
</tr>
<tr>
<td>System age</td>
<td>8.40</td>
<td>1</td>
<td>4.08</td>
<td>0.05</td>
</tr>
<tr>
<td>Maintenance organization</td>
<td>0.84</td>
<td>1</td>
<td>0.41</td>
<td>0.53</td>
</tr>
<tr>
<td>Use of software tools</td>
<td>0.10</td>
<td>3</td>
<td>0.05</td>
<td>0.83</td>
</tr>
<tr>
<td>Ability of maintainers</td>
<td>4.52</td>
<td>1</td>
<td>2.19</td>
<td>0.15</td>
</tr>
<tr>
<td>Organizational stability</td>
<td>19.92</td>
<td>4</td>
<td>2.42</td>
<td>0.07</td>
</tr>
<tr>
<td>Error</td>
<td>61.80</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Squared Multiple R = 0.604</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

for about three years. (Note that the lines are moderated with the use of a logarithmic scale.) More surprising is the line showing maintenance time for systems developed with modern methodologies. Maintenance time for systems of this class is indeed initially lower than that of the other group but increases drastically after about a year and becomes much greater than that for the traditional group. This outcome is certainly counterintuitive and will be further elaborated later.

The other control variable influencing maintenance time at somewhat higher probability level (p = 0.07) is organizational stability. As expected, an unstable organizational environment appears to increase maintenance time, but the effect is small. The correlation between the organizational stability and maintenance time is very weak, with a Spearman correlation coefficient of only -0.08. Separate analysis of each of the two groups does not change it much.

Four other controlled variables did not show significant influence on maintenance time. These are: system class, maintenance organization, use of software tools, and perceived ability of maintainers.

The second research proposition evaluated differences of maintenance time allocation between the two groups of systems. The results of this analysis are summarized in Table 4.

The differences in percent of time spent on emergency error correction and on evaluation of change requests are significant (p < .05). The second research proposition, that maintenance time allocation depends on methodology, was thus supported. The finding related to the emergency error correction is particularly encouraging: it indicates that systems developed with modern methodologies are breaking down less frequently and/or are easier to repair. Table 5 shows descriptive statistics of maintenance time allocation. Zeros in the median columns show that at least half of the respondents did not allocate any time to related activities.

Data on mean times between failures (MTBF) and mean times to repair (MTTR) provided additional insight. Results of the analysis of variance between methodology groups for these two variables are shown in Table 6. These results substantiate the finding that modern methodologies decrease the time spent on emergency repairs. They suggest that modern methodologies improve reliability. Allocation of time to repair tasks, on the other hand, does not depend on development methodology.

The difference in time spent on evaluation of change requests, although statistically significant, is not very large. This result nevertheless favors modern development methodologies. Some other results are intriguing. How maintenance work is organized significantly influences the time spent in supporting users. Separate maintenance groups spend a greater proportion of time spent in maintenance activities on supporting users. This conclusion, though in har-

---

7 The number of cases is lower because of some missing data and three cases where the respondents reported that their systems never break down. They were treated as missing data and excluded. All three of these systems were developed with modern methodologies.
mony with Swanson and Beath (1989b), can only be tentative because such separation of duties was implemented in only five cases.

The ability of maintainers, as perceived and assessed by IS managers, shows a strong relationship with the time spent on emergency error correction. The greater their ability, the less time spent. Several explanations are possible. More capable people may need less time to repair a faulty system. It is noteworthy, however, that perceived personnel abilities do not significantly influence other maintenance activities. This explanation suggests that system repairs be assigned to particularly capable personnel, leaving other maintenance activities to less qualified performers. On the other hand, repair work is usually associated with emergency situations. Maintainers who are effective in repairing defects may be perceived by managers as highly capable, while those who perform other tasks may not get the same level of recognition. A third explanation might be that work under pressure motivates maintainers to perform the best they can, while they may handle other tasks in a more relaxed manner. This question deserves further investigation.
Table 4. ANOVA Results (f-values) for Testing the Second Proposition

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>df</th>
<th>IMC</th>
<th>PEF</th>
<th>IFE</th>
<th>SU</th>
<th>ECR</th>
<th>RCR</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methodology</td>
<td>1</td>
<td>0.3</td>
<td>7.2**</td>
<td>0.3</td>
<td>2.7</td>
<td>8.4***</td>
<td>0.0</td>
<td>0.6</td>
</tr>
<tr>
<td>System size</td>
<td>1</td>
<td>3.0*</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
<td>2.9*</td>
<td>0.2</td>
<td>7.3***</td>
</tr>
<tr>
<td>Number of users</td>
<td>1</td>
<td>0.0</td>
<td>3.8*</td>
<td>2.3</td>
<td>0.2</td>
<td>0.2</td>
<td>0.0</td>
<td>2.9*</td>
</tr>
<tr>
<td>System class</td>
<td>2</td>
<td>0.7</td>
<td>0.4</td>
<td>0.6</td>
<td>0.9</td>
<td>1.5</td>
<td>0.6</td>
<td>2.6*</td>
</tr>
<tr>
<td>System age</td>
<td>1</td>
<td>3.0*</td>
<td>2.9*</td>
<td>1.6</td>
<td>0.0</td>
<td>2.9*</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Maintenance organization</td>
<td>1</td>
<td>0.9</td>
<td>0.2</td>
<td>2.0</td>
<td>4.4**</td>
<td>0.5</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>Use of software tools</td>
<td>3</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td>2.2</td>
<td>0.7</td>
<td>0.2</td>
</tr>
<tr>
<td>Ability of maintainers</td>
<td>1</td>
<td>0.4</td>
<td>7.2***</td>
<td>0.6</td>
<td>0.7</td>
<td>0.6</td>
<td>0.9</td>
<td>1.1</td>
</tr>
<tr>
<td>Organizational stability</td>
<td>4</td>
<td>0.8</td>
<td>0.5</td>
<td>0.6</td>
<td>0.6</td>
<td>1.6</td>
<td>1.7</td>
<td>1.7</td>
</tr>
</tbody>
</table>

Legend:

IMC = Implementing mandatory changes
PEF = Problem/emergency fixing
IFE = Implementing functional enhancements
SU = Supporting users
ECR = Evaluating change requests
RCR = Rewriting, converting, rejuvenating
Other

Table 5. Descriptive Statistics of the Percent of Maintenance Time Allocation Among Different Maintenance Activities

<table>
<thead>
<tr>
<th>Maintenance Activity</th>
<th>Modern Methodologies N</th>
<th>St. Mean</th>
<th>St. Dev.</th>
<th>Median</th>
<th>Traditional Methodology N</th>
<th>St. Mean</th>
<th>St. Dev.</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementing mandatory changes</td>
<td>32</td>
<td>21.9</td>
<td>25.9</td>
<td>10.0</td>
<td>41</td>
<td>18.0</td>
<td>15.0</td>
<td>10.0</td>
</tr>
<tr>
<td>Problem/emergency fixing</td>
<td>32</td>
<td>10.1</td>
<td>16.4</td>
<td>7.5</td>
<td>43</td>
<td>17.6</td>
<td>17.5</td>
<td>10.0</td>
</tr>
<tr>
<td>Implementing functional enhancements</td>
<td>31</td>
<td>29.6</td>
<td>28.2</td>
<td>20.0</td>
<td>43</td>
<td>23.0</td>
<td>22.4</td>
<td>20.0</td>
</tr>
<tr>
<td>Supporting users</td>
<td>32</td>
<td>18.7</td>
<td>20.4</td>
<td>10.0</td>
<td>43</td>
<td>22.7</td>
<td>17.0</td>
<td>20.0</td>
</tr>
<tr>
<td>Evaluating change requests</td>
<td>32</td>
<td>6.1</td>
<td>7.4</td>
<td>5.0</td>
<td>43</td>
<td>8.1</td>
<td>5.4</td>
<td>10.0</td>
</tr>
<tr>
<td>Rewriting, converting, rejuvenating</td>
<td>31</td>
<td>6.3</td>
<td>9.3</td>
<td>0.0</td>
<td>43</td>
<td>9.3</td>
<td>14.7</td>
<td>3.0</td>
</tr>
<tr>
<td>Other</td>
<td>31</td>
<td>2.3</td>
<td>5.6</td>
<td>0.0</td>
<td>42</td>
<td>2.1</td>
<td>4.4</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 6. Descriptive Statistics and Analysis of Variance of MTBF in Days and MTTR in Hours

<table>
<thead>
<tr>
<th></th>
<th>Modern Methodologies</th>
<th>Traditional Methodology</th>
<th>Stand. Dev.</th>
<th>t</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTBF</td>
<td>N=21 Median=90 Mean=119 Dev=105</td>
<td>N=34 Median=30 Mean=68 Dev=66 df=2 d.f=53 t=2.219 p=0.031</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTTR</td>
<td>N=23 Median=24 Mean=27 Dev=28</td>
<td>N=41 Median=24 Mean=28 Dev=57 d.f=53 t=0.126 p=0.901</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

366 MIS Quarterly/September 1992
The data in Table 4 show that there is no statistically significant difference in the time spent implementing functional enhancements for systems developed with modern methodologies versus traditional methodology. This outcome is counterintuitive for two reasons. It is generally believed that systems developed with modern methodologies need less enhancements because the requirements are initially better analyzed. It is also believed that it is easier to enhance such systems because of their better structure and documentation. This brings us back to the surprising observation of maintenance time as a function of system age (see Figure 4).

Figures 5 to 7 show similar smoothed lines of times allocated to implementing mandatory changes, problem/emergency fixing, and implementing functional enhancements, respectively, for the two groups of systems.

Plots of allocated times to other maintenance tasks are not included but are all similar in shape to the one in Figure 6. They show a rapid decrease of allocated time for the systems developed with modern methodologies in about 20 months or so. Figures 5 and 7 show the opposite trend. This offers a new explanation of a sharp increase of time required to maintain

---

**Figure 5. Smoothed Percentage of Time Spent on Mandatory Changes by System Age**

- ○ = IS developed with modern methodologies
- △ = IS developed with traditional methodology

*MIS Quarterly/September 1992* 367
systems developed with modern methodologies are operational for about one year, as shown in Figure 4. It leads to a conclusion that the implementation of mandatory changes and functional enhancements and not the other maintenance tasks may be the reasons for the apparent intensification of system maintenance by age for this group.

Why do modern methodologies increase the maintenance requirements of older systems? One possible explanation is that modern methodologies improve the relationship between users and system professionals. Consequently, users understand systems better and also are better served and supported. They readily ask for the enhancements because they find system professionals approachable and responsive. Their requests may also be more reasonable, based on a more complete understanding of the system. While the implementation of major changes is prohibitively disruptive and time-consuming in systems developed with traditional methodology, it is feasible where modern methodologies are used for system development. Their implementation still consumes a lot of maintainers' time, but this kind of productive, "enhancing" maintenance improves the system's value to the organization. In other words, modern methodol-
Figure 7. Smoothed Percentage of Time Spent on Functional Enhancements by System Age

ologies might not decrease the maintenance time overall but may enable the implementation of major changes and, therefore, improve the service provided to the users.

The survey data reveal additional, although limited, insights into this relationship. Table 7 shows descriptive statistics of the number of annual functional enhancement change requests. The data suggest that a slightly larger number of such requests is received annually for systems developed with modern methodologies but that a lower number of requests is actually completed.

Neither of these distributions significantly differs between both groups (p values are 0.51 and 0.52, respectively). The data nevertheless suggest that most of the requested changes are implemented with systems developed with the traditional methodology and that much fewer requested changes are completed within the systems developed with modern methodologies. Is it that only small patch-ups in systems developed with traditional methodologies are made, while substantial changes are implemented in systems developed with modern methodologies? Why is the ratio between completed and requested changes smaller with the group of systems developed with modern methodologies? These and other questions related to change requests and their implementation need to be answered.
Table 7. Annual Number of Functional Enhancement Change Requests Received and Completed

<table>
<thead>
<tr>
<th></th>
<th>Modern Methodologies (N = 26)</th>
<th>Traditional Methodology (N = 43)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std. Dev.</td>
</tr>
<tr>
<td>Change requests received</td>
<td>33.2</td>
<td>62.5</td>
</tr>
<tr>
<td>Change requests completed</td>
<td>15.3</td>
<td>24.2</td>
</tr>
<tr>
<td>Ratio (completed/received)</td>
<td>0.46</td>
<td>0.32</td>
</tr>
</tbody>
</table>

with subsequent studies. The indication that the implementation of mandatory and functional changes increases rapidly with the group of systems developed with modern methodologies after systems operate for about a year (which also causes a similar increase of the overall maintenance time) certainly calls for further study and explanation. This finding conflicts with the general belief as well as with the claims of the vendors of modern methodologies.

Limitations of the study

Since this was the first study to investigate the relationship between the system development methodology and maintenance, it was exploratory in nature. Consequently, it answered few and opened a number of questions.

The study was performed at the time when the use of modern systems development methodologies was not yet widespread. Even the companies that reported the use of modern methodologies had not been using them for long. The results can be distorted with the effects normally associated with the learning curve.

Possible benefits from the use of integrated CASE (I-CASE) tools for systems development were also not investigated. It is generally believed that prototyping, which is supported by the I-CASE technology, improves requirements analysis and specification, which in turn decreases the need for post-implementation functional enhancements. I-CASE technology also enables users to implement minor changes, such as the format of a report, the execution of a new database query, or even the generation of a new report by users themselves. Increased self-sufficiency of end users should decrease the number of service requests and, thus, the maintenance time of software professionals. A study of the effects of new I-CASE technology on software maintenance is indispensable. A degree of caution is recommended in the meantime because experience with previous generations of “breakthrough” technologies supports Brooks’ (1987) argument that “there are no silver bullets.”

The size of the sample in this study is relatively small. Another limitation is associated with the data-gathering technique. All data were reported by a single person from each company and were validated only by checking the level of consistency among redundant questions. Respondents may have reported their perceptions rather than realities. Those perceptions could be swayed as much by vendor hype and common beliefs as by the reality in the respondent’s organization.

Conclusions

This study provides no evidence that modern IS analysis methodologies decrease the overall time and cost of maintenance. However, the times spent on various maintenance activities do change. Systems developed with modern methodologies are apparently more reliable and require repairs less frequently. This effect by itself would justify the use of modern methodologies.

The study confirmed that the system’s size influences maintenance costs. Developers should try to limit the number of IS functions and exclude non-required capabilities. Complexity may breed maintenance problems, and gains achieved by such complexity may be offset by losses in productivity and increased maintenance.
The number of users also influences the overall maintenance time. Designers of systems to be used by many must carefully consider maintainability. Two other variables influencing maintenance time are system age and organizational stability.

The study indicates that systems developed with different methodologies age differently. The maintenance effort with systems developed with modern methodologies is lower during the first year of a system’s operation. After the first year it grows and surpasses the time required to maintain systems developed with a traditional methodology. The cause for this surprising effect is apparently the implementation of mandatory changes and functional enhancements. Other maintenance activities virtually disappear. Larger numbers of changes are requested but a lower number of changes are implemented. More time spent on the implementation of a lesser number of changes indicates that changes are more complex. This suggests that modern development methodologies enable the implementation of significant improvements and enhancements, while the traditional development methodology facilitates patching but discourages users to request major changes.

Modern development methodologies apparently do not yet decrease total maintenance time. Maintenance time actually increases after the systems are in operation for about two years. The use of modern methodologies does, however, lead to changes in maintenance time allocation. Less time is spent on emergency fixing, evaluating change requests, rewriting, converting, and rejuvenating the system and more on supporting users and implementing mandatory changes and functional enhancements.

To conclude, implementors of modern systems development and maintenance methodologies should not become discouraged if they experience no apparent increase in productivity or possibly even its decline. Benefits of using such methodologies appear to be significant, albeit subtle, and recognizable only with meticulous measurement and analysis.

Acknowledgements

My thanks to the contributors who assisted in this project over the years. Peter Shoderbek and Nicholas Zveigintzov provided the initial guidance, while the former and current senior editors helped with the final polishing. My special appreciation goes to the anonymous (and patient) associate editors for insightful comments and diligent readings of several versions of the manuscript.

References


About the Author

Sasa M. Dekleva is assistant professor of information systems at DePaul University, Chicago. He received his Ph.D. in management information systems from the University of Belgrade. He has also worked in industry for 11 years in the areas of systems engineering and IS management. The issues addressed in his research involve information systems development methodologies, data modeling and database design, software maintenance, and natural language querying.

372 MIS Quarterly/September 1992