Management of Computing

Robert W. Zmud Editor

Departmentalization in Software Development and Maintenance

Exploring the strengths and weaknesses of three alternative bases for systems staff departmentalization suggests the benefits of an organizational form in which maintenance is separate from new system development.

E. Burton Swanson and Cynthia Mathis Beath

Software maintenance—the correction, adaptation, and perfection of operational software [37]—has been a relatively neglected subject in the literature of software engineering and management. Attention has instead focused primarily on improved techniques for new system development. The virtues of these techniques are often held to include ease of maintenance on implementation. However, such claims are seldom validated through empirical study. It has not been shown that the maintenance burden is reduced by user involvement, prototyping, or the use of fourth generation development techniques. Systems beget systems; better systems generate more systems, subject data bases, and strategic information systems. The installed software base grows larger and more diversified as end user developed systems, third party developed systems, and purchased packages are added. The maintenance burden grows too. (For background, see [10, 11, 30, 35, 47].)

The allocation of organizational resources to new system development and installed system maintenance has rarely been studied as a joint problem. Among the few studies which touch upon this issue are those of Lientz et al. [28] and Lientz and Swanson [26, 27], who report, based on their surveys of application software maintenance, that information systems (IS) organizations generally devote about the same amount of effort to maintenance as to new system development. Lientz and Swanson [26] also report that the expenditure of staff time to maintain a system tends to increase with both its age and size. Further, growth in size averages a substantial 10 percent per year, a finding which closely parallels that of Belady and Lehman [6] in their classic study of the growth of Operating System/360 over successive releases. In both studies, the provision of additional features and functionality is found to largely account for the common pattern of growth. Thus, older systems tend to be larger and harder to maintain; one reason for the increased difficulty is they have been enhanced to meet the needs of their users. (See also related studies by [20, 21, 42].)

The mature IS organization is therefore responsible for a substantial accumulation of installed application systems, which undergoes continuous growth and "evolution" [6, 7, 25], and which must be managed in conjunction with the acquisition and development of those new systems to which the organization also commits itself. With growth in the size of the IS organization often limited by management fiat despite continued growth in the size of the application system portfolio, the proper organization of work to carry out the joint tasks of maintenance and development is a subject of substantial management interest (see [15, 46]).

IS productivity in system development and maintenance is recognized to be a major concern [23]; the typical organizational backlog of programs to be written stretches to a period of three years or more [24, 32]. The business risks of failure both in development and maintenance are also significant. A notorious case of such failure is Bank of America's attempt to develop a new trust accounting system at an estimated cost of $80 million; among many difficulties the staff bore the brunt of working concurrently on both the new system and the older operational system it was designed to replace [16]. The frequently reported failures of critical operational software also illustrate the risks involved (see [22]).

Two issues form the crux of the organizational problem: The first is whether the individual professional analyst or programmer should divide his or her time between maintenance and new system development work. Here, the matching of the motivating potential of

the work to the "growth need strength" of the individual is an important consideration in any assignment, according to Couger and Colter [13], who found that development work has higher motivating potential than does maintenance work. The second issue, which arises only where it is decided that an individual should not divide his or her time between the two tasks, is whether or not maintenance staff should be organized as a separate department. In this case, considerations of productivity gains through specialization, efficiency of communication, and management control have been suggested to be of primary importance [38]. This article will focus on the departmentalization issue.[1]

We begin by presenting three alternative bases for departmentalization of the systems staff. The strengths and weaknesses of the three alternatives are discussed. Then, data taken from a set of twelve case studies are used to describe current departmentalization practice. In subsequent discussion, an historical interpretation of the pattern of practice is suggested, and it is argued that a life cycle based organizational form, in which maintenance is organized separately from new system development, deserves closer scrutiny by IS management. Implications for current practice and further research are drawn in conclusion.

ALTERNATIVE BASES FOR DEPARTMENTALIZATION

Issues in the organization of work have long been studied by management and organizational researchers, as well as by other social scientists interested in the impacts of organizational practice on society. Classical management theory, dating from the early 1900s, originating in Adam Smith's 18th century work, provides much of the contemporary theoretical vocabulary. (For a review, see Galbraith [17].) Among its basic concepts are the horizontal division of labor among workers based upon specialization; and the vertical division of labor between workers and managers, and among levels of management, typically based on principles of command and control. Closely related is the concept of departmentalization, the aggregation of work roles to form groups, units, departments and divisions. Galbraith [17], whose information-processing theory of organization design we draw upon here, suggests three fundamental bases for departmentalization: input resources (grouping by function, technical specialty, or process), outputs (grouping by product, market, or customer) and physical location.

Our own analysis of application software development and maintenance suggests that three particular bases for the division of labor and the departmentalization of systems staff are of importance:

Work type: systems analysis versus programming

Application type: application group A versus application group B

Life cycle phase: development versus maintenance

Here the term "versus" indicates simply that a formal distinction is made for the purpose of organizing work. As we shall indicate, these bases relate closely to organizing around inputs or outputs as described by [17].

Division of labor by work type implies job specialization according to distinctive work skills. Historically, the most common work type distinction has been between systems analysis and programming, with systems analysts specializing in the functional specification of a system, and programmers in its computer-based implementation [12]. Where departmentalization is also based on this specialization—systems analysts and programmers are organized into separate departments—the organization may similarly be said to have a work type (W-type) form (also referred to in the general management literature as a "functional" form; see [46]). The W-type form corresponds to the concept of departmentalizing around input resources. Its distinguishing feature is concurrent multidepartment responsibility for a system's development or maintenance. This is necessary even though systems analysis precedes programming within the task sequence because analysis as a whole is iterative and continuous.

Division of labor by application type constitutes the second basic alternative. Here, the distinction is between individuals being assigned work on one system or group of systems versus their being assigned work on another system. Specialization is in the knowledge associated with the domain of application, rather than in certain work skills. Such a domain often, though not always, maps closely to one or more user departments. (In some instances, the domain may be integrative across departments.) Where departmentalization is also based on application domain, the organization may similarly be said to have an application type (A-type) form which corresponds closely to the "product" form discussed in the general management literature. The A-type form corresponds to the concept of departmentalizing around outputs. Its distinguishing feature is that a single department is responsible for the development and maintenance of a system over the system's life.

A third alternative is the division of labor by life cycle phase. The distinction here is typically between development work on new systems or on new versions of installed systems, and maintenance work on installed systems [37]. Specialization is in the skills and management of the development or maintenance phases of the life cycle. Where departmentalization is also based on this distinction—developers and maintainers are organized into separate departments or work units—the organization may similarly be termed a life cycle based (L-type) organization. Less obviously than the A-type form, the L-type form also corresponds to the concept of departmentalizing around outputs. In this case, the development unit focuses upon software products as its output, while the maintenance unit con-

[1] While users have significant roles to play in both development and maintenance activities, we are concerned here with the standing organization of the technical staff. Of course, if technical staffs are decentralized to user organizations, similar questions of organization structure might arise and could be addressed using the logic presented in this article.

centrates on service to users of installed systems. The important distinguishing feature of the L-type form is the transfer of responsibility for a system's development and maintenance between departments at the time the system becomes operational.

The L-type form, based on a distinction between development and maintenance work, is an idea that has been around for years (see [34]). It may be employed in several variations. Among these are the centralized development unit with decentralized maintenance at multiple installation locations; the location of a "fire fighting" maintenance team within the computer operations unit [38]; and the integration of maintenance into the user organization [8].

The choice of any particular organizational structure necessitates trading off the strengths and weaknesses of the three alternatives, (i.e., departmentalization by work type, application type, or life cycle phase). In Table I we summarize these basic trade-offs. Following Galbraith's [17] theory of organization design, focal strengths and weaknesses of the three forms are presented in terms of their knowledge development and information processing implications. Uncertainty in the development and maintenance tasks is understood to be the basis of information and communication requirements within the IS organization, both laterally and hierarchically. Increased uncertainty requires that the IS organization find ways to increase its capacity to process information, or reduce its need for information. The preferred IS organizational form is in general that which is best matched to the task uncertainty faced. (See [17, 45] for details.)

The unique strength of the W-type form, in which programmers and systems analysts are organized into separate departments, is the development and specialization of programming skills. Here, the role of the systems analyst is to buffer the programmer from the user, allowing the programmer to focus on translating specifications into software. Where programming is a formidable task—as it is where machine constraints are tight and the programming language is close to that of the machine—the development of programming knowledge and skills may be critical and the advantage of the W-type form decisive. However, the weakness of the W-type form is also significant. Costs of coordination between systems analysts and programmers may be substantial for two reasons: the software specification bears the burden of formally mediating work between the two departments, often with considerable difficulty; and resulting interdependency problems between the departments may require frequent and costly resolution within the management hierarchy. These costs may, of course, be moderated where software specification is relatively straightforward and few problems need resolution. However, where software specification is particularly problematic and subject to ambiguity or instability, the weakness of the W-type form may be its undoing.

The A-type form in which departmentalization is based on application type has its own unique strength and weakness. Its strength is the development and specialization of application knowledge [15]. In contrast to the W-type form in which specialization focuses on inputs, such as programming skills, specialization in the case of the A-type form focuses on outputs, (i.e., upon the functionality of the applications). To the extent that applications share substantial communality within areas, such specialization is likely to be particularly important. However, there is a weakness here regarding the cost of coordinating and integrating across application areas. An overall information architecture may be needed to formalize application interdependence and autonomy. A strategy may also be required to establish priorities for resource claims across domains. However, such formal mechanisms will not always suffice. Where application areas are strongly interdependent, the resulting frequent conflicts, problems, and ambiguities will require resolution by the management hierarchy, at significant further costs.

In the case of the L-type form, in which development and maintenance are organized as separate departments, the particular strength is in the specialization and provision of services in support of day-to-day business operations which rely on installed information systems. These services, which focus on improving the value of installed systems to their users, have historically taken a back seat to new system development. In particular, the system development life cycle has focused almost entirely upon aspects of the task that precede system installation and maintenance. In the IS organization, maintenance has been a background rather than a foreground task. But where the installed base of systems is large and mature, the advantages of specializing in the maintenance and enhancement of their services may be substantial. Like the other forms, the L-type form has a weakness. It is in the cost of coordination during implementation and in transferring

TABLE I. Trade-Offs Among Alternative Organizational Forms

W-Type	Departmentalization by work type (systems analysis versus programming)
	Focal strength: development and specialization of programming knowledge and skills
	Focal weakness: costs of coordination between systems analysts and programmers
A-Type	Departmentalization by application domain (application group A versus application group B)
	Focal strength: development and specialization of application knowledge
	Focal weakness: costs of coordination and integration among application groups
L-Type	Departmentalization by life-cycle phase (development versus maintenance)
	Focal strength: development and specialization of service orientation and maintenance skills
	Focal weakness: costs of coordination between development and maintenance units

TABLE II. Twelve Cases in Information Systems Organization

Org	Staff	Dep	Span	Form	M-Sys	D-Sys	R-Sys	M-Role	D-Role	S-Role
1	148	6	6.4	A-L	51	16	4	38%	59%	3%
2	102	4	7.5	L-A	81	22	9	28%	62%	10%
3	7	1	2.5	L-A	33	11	11	100%	0%	0%
4	67	6	10.2	A	14	3	0	34%	16%	42%
5	46	3	5.6	A-L	103	15	10	26%	48%	26%
6	266	18	13.8	A	89	21	11	42%	31%	27%
7	45	4	2.8	A	28	5	3	31%	38%	31%
8	19	1	5.3	A	33	6	3	26%	16%	58%
9	117	9	12.0	A	22	0	0	100%	0%	0%
10	102	8	11.8	A	30	3	2	0%	0%	100%
11	118	5	8.8	A-W	25	10	6	44%	28%	28%
12	108	4	3.5	A-L	171	0	0	93%	7%	0%

Notes: Organizations are numbered in the sequence in which the cases were prepared. Staff counts represent application development and maintenance personnel only. Department (Dep) counts represent units reporting to first-level managers. Span of control (Span) is the average number of staff reporting to a first-level manager or supervisor. Organizational forms refer to the predominate basis of departmentalization: the A-type organizations are departmentalized by application type; A-L type organizations include subordinated maintenance and/or new system development units; the A-W type organization includes subordinated analyst and programmer units; the L-A type organizations are departmentalized according to maintenance and new system development, with subunits subordinated according to application type. Numbers of systems maintained (M-Sys), new systems under development (D-Sys), and replacement systems among the new systems under development (R-Sys) are based on differing levels of system definition across organizations. Numbers of maintainers (M-role) spend two-thirds or more of their effort on maintenance; developers (D-role) spend two-thirds or more of their effort on new system development; others (S-role) split their efforts between maintenance and development more evenly.

responsibility for development and maintenance between departments. Here again, it will be necessary for the management hierarchy to resolve problems, especially in coordinating a system replacement; as with the A-type form, the costs may be significant.

No form is perfect. Fortunately, the three alternatives are by no means mutually exclusive as bases for organizing, and thus their strengths and weaknesses can be balanced through combination. For example, an individual work assignment may consist wholly of maintenance programming of a single application system or subsystem. In the case of departmentalization, IS organizations may combine two or even all three of the basic alternatives into a hybrid form. In deciding whether and how to combine the basic alternatives IS organizations have an opportunity to trade off the pluses and minuses of the three forms, if they so choose.

How do IS organizations currently make choices among alternative forms? What hybrid forms, if any, are employed, with what frequencies? What stresses and strains result with what implications? The design choices of 12 contemporary IS organizations will now be presented and analyzed to illuminate current practice.

A STUDY OF CURRENT PRACTICE

A set of 12 cases on application software maintenance was recently developed as part of an ongoing research project. These cases focus on the comparative maintenance environments of IS organizations, and on alternative IS management strategies for maintenance of the application system portfolio, including alternative approaches to organization design; task definition and assignment; work technique; and policies for coordination and control.

Development of the cases concluded in the summer of 1985. Among the participating organizations were four high technology manufacturers, two food and beverage producers, one oil company, a retail grocery firm, an aerospace company, a research and development laboratory, a public utility, and a university. Six of the organizations are based in southern California; the rest are dispersed throughout the U.S. Significant diversity among the participants, in terms of organizational size and type, was sought. The sample is not representative of a specific population. Rather, its purposes are intentionally exploratory and the appropriate logic of comparative analysis is that of theoretical replication, where each case is roughly analogous to a separate experiment [44], in which a single research question is studied from several different experimental vantage points. Thus, diversity among participants may be compared to variety among experimental arrangements, with the expectation that the findings should also be rich across cases.

Questionnaires, on-site interviews, and reviews of organizational documents were used in data gathering. Both quantitative and qualitative data were obtained, following a common protocol [40]. Together, the cases appear in Swanson and Beath [41]. Selected data are presented in Table II.

Across the 12 cases, classification of the organizational forms employed is based on the three basic forms discussed earlier: the work type (W-type); the application type (A-type); and the life cycle based (L-type) forms. Both pure and hybrid forms are identified, with the latter described in terms of principal and subordinate bases of departmentalization as explained in Table II. Remarks on each of the cases included in Table III explain each instance of the forms and provide additional perspective. Considerable diversity is seen to exist, and it is further apparent that various temporal and contextual factors (e.g., company reorganization or

TABLE III. Remarks on Organizational Forms

Org	Form	Remarks
1	A-L	Maintenance has a significant profile in this organization. Of six departments, four are responsible for application systems grouped by user functional areas. A fifth is responsible for development of a major new system. A sixth provides planning and technical support. Within departments, staff are assigned to either new system development or maintenance. A few years ago, maintenance was centralized as a separate department. Now, management is weighing whether to return to this arrangement.
2	L-A	Maintenance is a major function in this organization. Of four departments, two work primarily on new system development, with division of labor by user area. A third maintains systems developed by the first two. A fourth develops and maintains systems for another area of use. Management considers the arrangement cost-effective, but worries about the "stigma of maintenance" and its effects.
3	L-A	A single small department works wholly on maintenance and local plantsite support. Within the department, two supervisors are responsible for distinct application groups. New systems are developed and supplied by the parent organization. A major group of Manufacturing Resource Planning (MRP) systems is soon to be implemented, and one supervisor is working full time on the project.
4	A	Of six departments in this organization, four develop and maintain systems according to area of application. Within one of these, a team of five individuals maintains a major system. However, this reflects the scale of the system, more than a commitment to maintenance organization. Two other departments develop and maintain supporting common systems. Management envisions a transition to a distributed processing environment with user-developed applications.
5	A-L	Management in this organization believes that maintenance and development appeal to different types of people. Maintenance was once centralized. Now two departments develop and maintain systems for distinct user divisions. Within each, separate staffs are associated with maintenance and new system development. Each has its own supervisor. A third department develops and maintains office systems.
6	A	This large organization is departmentalized by groups of applications. Eighteen units are formed within five departments under one manager. There is no formal distinction between maintenance and new system development staff. Enhancement of existing systems occupies a substantial proportion of staff time, and the age and maintainability of applications is a concern. User dissatisfaction with the backlog of work is also a problem.
7	A	In this organization, four departments develop and maintain systems grouped by area of application. Responsibility for production systems, as opposed to new systems, is a formal requirement for career advancement. However, there is no division of labor based on maintenance versus development, and, in fact, the distinction is blurred. Responsibility for production systems includes their further development.
8	A	In this organization, the systems staff is departmentalized by area of application. Each of three groups has full responsibility for development and maintenance within its area of jurisdiction. Necessarily, more time is spent on maintenance than on development, but an attempt is made to spread development opportunities equitably. Responsibility for the maintenance of several systems critical to business operations is reserved for senior people.
9	A	This new organization is responsible for the development and maintenance of manufacturing information systems, and reports directly to the manager or manufacturing. It has just been spun off from a large, centralized IS function. New system development is frozen during the present transition period. Staff include programmers as well as analysts who formerly worked in user departments. All now work as programmer/analysts. Departmentalization is by area of application.
10	A	This organization is departmentalized by area of application into two major groups. Each unit within a group is responsible for both development and maintenance. Staff tend to split their efforts evenly between the two tasks. Nearly half the systems in the current portfolio are more than ten years old. New system development is motivated by changes in the core business technology supported.
11	A-W	This organization consists of five departments, four of whose supervisors divide development and maintenance responsibility for the applications portfolio. Systems are allocated to supervisors by client area, for the most part. Separate staffs for programming and systems analysis exist within each department. Analysts tend to work in a liaison role between users and programmers.
12	A-L	New system development is currently frozen in this large organization, while the business seeks a major new government contract. Three departments are responsible for systems grouped by area of application. A fourth department provides system support, which will shortly include acceptance of all systems put into production, as well as primary responsibility for corrective maintenance in the event of operational difficulties.

strategic realignment) also shape the organizational design choices. (Zmud, [46], makes a similar argument.)

Overall, the A-type form is represented in all 12 cases, and is the primary basis of departmentalization in 10 instances. The L-type form is currently represented in five cases, though most frequently as a subordinate basis of departmentalization. The W-type form appears only once in a subordinate role. While the A-form dominates current practice, the less-recognized L-form is well represented.

The cases provide a good illustration of the ways in which IS organizations seek to strike a balance among the advantages and disadvantages of the three principal forms. Half the cases present hybrid forms. In several cases, transitions between forms occurred prior to or subsequent to the data gathering. Organization 1 has recently changed from L-A form to A-L form. In the words of one manager, however, this "was probably a mistake." (A task force was appointed at this site to examine the issue, and subsequent to our study, the department returned to an L-A form.) Organization 5, which used an L-A form for about 14 years, recently changed to A-L, retaining many desirable features of their L-A form in the process. Organizations 11 and 12 are both in the process of establishing groups dedicated to maintenance. Small groups dedicated to the maintenance of an important or problematic system were found at other organizations.

On balance, the predominant trade-off that concerns IS managers is between (a) forms in which most of the application staff members split their time between maintenance and development (the A and A-W forms

in our cases), and (b) forms in which development and maintenance staff are separated to some extent (the L-A and A-L forms in our cases).

We analyzed the data of Table II to compare these two alternatives—the more traditional A or A-W forms with the two L-forms (L-A and A-L). Specifically, we asked whether separating maintenance and development (as in the two L-forms) was related to (i) organizational size; (ii) management span of control; (iii) application portfolio maturity, as indicated both by systems maintained as a proportion of the total under new development and maintenance, and by replacement systems as a proportion of the total under new development; and (iv) the percentages of staff effort allocated to development and maintenance. No statistically significant differences were found, except for the percentages of staff effort allocated to development and maintenance.[2] Here it was found that relatively more staff time was allocated to development work, on average, where development and maintenance were organized separately. Figure 1 summarizes.

Interpretation of Figure 1 is by no means straightforward. Our own interpretation provides for an almost paradoxical finding. While caution is in order, given the sample, we believe that separating maintenance from development is associated with a focusing of managerial attention on maintenance, with the result that the maintenance staff is more efficient and more productive at maintaining systems. Therefore, more resources are available for development work. Of course, other interpretations are conceivable. The firms in our sample with separate maintenance departments may simply be understaffing maintenance, or they may have relatively smaller maintenance burdens for reasons we have not considered. In any case, a more systematic study of this issue is clearly needed.

How do we account for the pattern of organizational practice found here? Why does the A-form dominate while the W-form is barely represented? What features of the L-form account for its apparent efficiencies, and what are its overall benefits and costs when examined in more depth? What are the likely implications for the future? We will now turn to these questions.

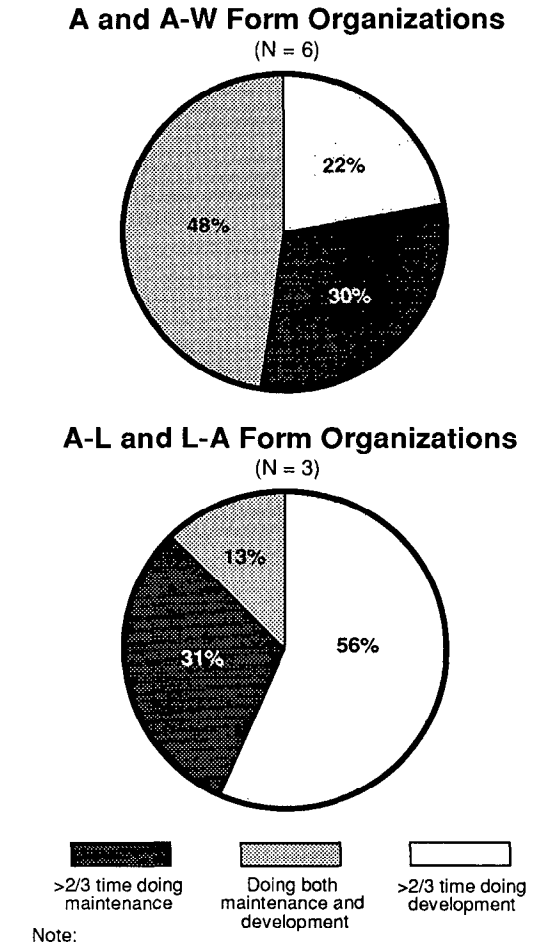
DISCUSSION

We suggest a process of historical change underlies current IS departmentalization practice. During the early years in which IS organizations emerged, almost three decades ago, division of labor was frequently based on work type, by distinguishing between the programming and systems analysis tasks [43]. Because systems were typically constructed in lower-level languages (frequently, assembler languages) and utilized an expensive and scarce computing resource, the programmer's task was to work close to the machine, while systems analysts inherited the problem of mediating between the programmers and users who knew little if anything of computing technology. Overall task uncertainty derived in large part from the computing resource and its efficient deployment, and the development of computing expertise was critical.

Over the years, however, as user organizations have become more computer-sophisticated, and as computer technology itself has become more user-friendly, the systems analysis task (and with fourth-generation tools, even the programming task) has increasingly been shared between IS and user organizations. Moreover, because programming work no longer takes place so close to the machine, and because the mediating role of

FIGURE 1. Comparison of Allocation of Effort

When maintenance receives managerial attention, in A-L or L-A form organizations, relatively more resources are available to be used for development than in A or A-W form organizations.



Note:
Cases 3, 9 and 12 are excluded here. Case 3 (L-A form) does only maintenance work; development is done at corporate headquarters. Case 9 (A-form) and Case 12 (A-L form) are currently doing only maintenance due to other organizational factors.

[2] Unpaired T-tests were used to compare six A and A-W form cases to three L-A and A-L form cases. Three cases were excluded from the analysis—Organizations 3, 9, and 12—because in each case no development was being performed at the time due to various organizational exigencies. T-values and significance levels were: Development work (D-role plus half of S-role), $T = -4.68$. $p + .001$; Maintenance work (M-role plus half of S-role), $t = 5.62$, $p = .001$.

TABLE IV. The Life Cycle Based on Organizational Form (Departmentalization by Maintenance and Development)

Overall Strengths
1. Clear accountability for both maintenance expenses and the investment costs of new system development.
2. Buffering of new system development personnel from the intermittent demands of maintenance.
3. Facilitation of software quality assurance, in that the maintenance unit is motivated to require a meaningful acceptance test prior to accepting responsibility for a system.
4. Supports a focus on improved level of service to the user, by means of maintenance specialization.
5. Increased productivity in maintenance, through concentration of system familiarity.
Overall Weaknesses
1. Potential status differential between development and maintenance units, with consequential degradation of maintenance work and demotivation of those who perform it.
2. Loss of knowledge about system in transferring it from development to maintenance.
3. Costs of coordinating between the development and maintenance units during implementation period, especially where new systems are replacement systems.
4. Increased costs of system acceptance.
5. Possible duplication of communication channels to user organizations.

systems analysis spans less of a knowledge gap, the distinctive features of both jobs have become blurred even within IS; apparently this basis for the division of labor has eroded. Increasingly, more individuals possess the skills necessary for both roles, and the integrated programmer/analyst job is now a realistic alternative. For example, at Organization 9 a recent reorganization has merged programmers with analysts, who formerly worked in user departments. The new department hopes for "synergism in the programmer/analyst activity."

Also, since the early years, the IS organization has gradually shifted its task focus to its application outputs. Beyond the automating of routine accounting procedures, the application domain has been extended to almost all corners of the enterprise. Certain applications have further come to be recognized as strategic [33]. Therefore, task uncertainty for the systems staff has increasingly derived less from computing resources than from complexities and risks associated with applications. Development of these applications has accordingly required the nurturing of application expertise among systems staff, and as a substantial and diversified base of applications has been accumulated, increased specialization by application type has been pursued. Thus, the application domain has come to constitute the predominant basis for departmentalization in today's IS organization.

Nevertheless, as depicted earlier, a significant number of organizations do make use of the life cycle based alternative in which a distinction between system development and maintenance forms at least some basis for departmentalization. Use of this alternative may also be on the rise. A recent study in the UK (cf. [36]) found the percentage of IS departments that organize maintenance into separate groups was significantly higher (40 percent vs. 16 percent) than the level found by Lientz and Swanson [26]. Why should use of the life cycle based alternative persist? In our view, L-forms (here, the L-A and A-L forms; more generally, any form that separates developers from maintainers) offer certain important advantages for the contemporary IS organization. Diversity among applications is not the only source of task uncertainty for the systems staff; a more important source of uncertainty may lie in the immediate organizational impacts of installed applications.

This suggests that the overall strengths and weaknesses of L-forms should be examined more closely. Based on the present study, in addition to drawing from other related work, we present our own assessment. As summarized in Table IV, five overall strengths and five weaknesses are suggested. We discuss each briefly. As a matter of practical interest, we include mentioning ways in which adopters of L-forms may attempt to compensate for its weaknesses. In doing so, we do not mean to suggest that L-forms are in general superior, but rather they need to be more deeply examined and more broadly understood.

Strengths

A first strength of the L-forms is *clear accountability for both maintenance expenses and the investment costs of new system development*, which has long been a problem for IS management. When personnel are assigned both maintenance and new system development work, they have some discretion in the charging of their time between the two classes of activity. Our case observations suggested that under- or overcharging to maintenance is commonplace and follows the management pressures of the moment. Campaigns to "reduce the time spent on maintenance" or "meet those development targets" may thus be deceptive in their appearance of success. L-form departmentalization puts an end to this situation, and gives better visibility to both maintenance expenses and the costs of new system development.

Buffering of new system development staff from the intermittent demands of maintenance is a further advantage of L-forms, our studies suggest. Maintenance problems are by their nature largely unpredictable, and they play havoc with the more orderly process of new system development. A manager at Organization 10 complained that a major problem for developers was being drawn into "answering user questions" about installed systems, which he noted, was "the biggest chunk of maintenance." At Organizations 1, 2 and 5 we were told that maintenance had been separated from development in part to buffer the development staffs from the demands of maintenance, allowing the developers to concentrate on their projects. Many failures to de-

liver new systems on schedule have been attributed to the siphoning off of staff time to meet the exigencies of maintenance. With an L-form this drain is much more apparent and can be more easily resisted.

Facilitation of software quality assurance is also supported by the L-forms. The maintenance unit is clearly motivated to require a meaningful acceptance test prior to assuming responsibility for a newly-developed system. At Organization 12, an elite group "possessing the best talent within IS," is being formed to provide the first line of support for installed systems. Of necessity, this group is expected to design and administer acceptance criteria for new systems. These criteria may be more or less formal, our case studies indicate, depending upon the working relationships between the maintenance and new system development work units. Acceptance tests provide leverage for assuring that quality standards are not unduly compromised by the pressures of new system development schedules and budgets.

A focus on an improved level of user service, particularly regarding responsiveness to user requests for maintenance, is also encouraged by L-form departmentalization. In contrast to new system development, which orients itself more toward delivering a software product, maintenance is by its nature a service activity, undertaken largely in response to a continuing stream of user requests. Thus, maintenance managers at Organization 5 pointed with pride to the "big in-basket and big out-basket" aspect of maintenance. Centralization of maintenance also allows users and IS to more closely evaluate responsiveness in meeting these IS requests.

Where user service is particularly important, pockets of staff dedicated to maintenance are often found in an overwise A-form organization. At Organization 8, for example, the IS department's principal mission is to provide high reliability operations of a few critical applications in a highly dynamic business environment. Maintenance of critical systems is reserved for a small group of highly skilled senior people who have developed elaborate and reliable techniques for making changes and retesting these systems while providing continuous service to their users.

Finally, *productivity gains in maintenance* may also apparently be achieved through specialization and L-form departmentalization, as originally suggested by Mooney [31]. Why might this be the case? Recall that familiarity with systems is fundamental to maintenance [1, 26]. Where staff split their time between maintenance and new system development, more staff must generally be assigned to maintain a given portfolio of systems with the consequence that their collective familiarity is more fragmented. We suspect fragmented familiarity is difficult to deploy and sustain. Loss of efficiency follows.

Separate maintenance groups may also be more efficient because their analysts and programmers develop expertise in maintenance and because their managers learn how to manage maintenance. At Organization 9, for example, after the manager began distributing a simple report of processing times and maintenance assignments for a group of systems, processing times for the systems were cut by half. Centralization of maintenance also makes it easier to justify investment in modern maintenance tools, such as restructuring or code analyzer packages.

Weaknesses

These strengths of L-forms are clearly compelling. However, against these strengths there are a number of weaknesses.

A potential status differential between the new system development and maintenance units is a first concern. Among the most discussed subjects in the literature is the low status and motivating potential of maintenance work ([13, 29, 34]). To the extent that maintenance is seen as undesirable work, an L-form is disadvantaged, in that it creates two classes of citizens in system development and maintenance. When an L-A form was adopted 14 years ago at Organization 5, it was feared that everyone might quit. They did not. But at Organization 2, which uses an A-L form, the maintenance manager's principal worry is morale.

Some argue that maintenance is inherently less motivating than new system development, because, for example, the hours are unpredictable and it involves routine work with older technology. Others disagree, arguing that maintenance offers at least as great a work challenge, involving, for example, expert troubleshooting under substantial operational pressure requiring a sensitive, experienced touch. Among our cases, for instance, at Organizations 5, 11 and 12, management's experience is that some people prefer maintenance work to development work. Our own view, based on the case studies, is that managerial attitudes and traditional IS career paths may explain much of the present motivational differential where it exists. At Organization 10 (A-type), for example, the IS manager told us he believes that maintenance work is not as challenging as development; his lower level managers told us that development assignments had more impact on upward career movement. Similarly, at Organization 2 where the maintenance manager worries about morale, IS management's attention is on development, and a common career path is from operations, through maintenance, and "up" to development.

Too many careers begin or end in maintenance. Newcomers are often initiated in maintenance before advancing to new system development. Similarly, old-timers skilled in earlier technologies too frequently find themselves retired to the maintenance pastures. At Organization 11 (A-W form), managers try to avoid the "second-class citizen syndrome" by rotating maintenance assignments, but they are reluctant to do this if it means sacrificing in-depth knowledge of a system. So, some maintenance assignments drift on indefinitely.

Career paths in which responsibility for a major installed system is recognized as a significant and necessary mid-career achievement might do much to alleviate the current motivational issue. At Organization 7,

an A-type organization where management nevertheless devotes attention and resources to maintenance, a period of responsibility for a "production" (installed) system is a prerequisite for advancement to management. (See [2, 4, 5, 18].)

Investment in new maintenance technology should also contribute to motivation, in that staff skills may be upgraded. In fact, managerial attention to maintenance in almost any form seems to alleviate some of the morale problem.

Loss of knowledge about systems in transferring them from new system development to maintenance is a second potential disadvantage of L-forms. As discussed earlier, knowledge about a system is fundamental to its efficient maintenance. To mitigate against this loss, the use of maintenance escorts is recommended [26]. At Organizations 2 and 5, maintainers sometimes participate in development projects and then rejoin the maintenance staff to maintain the system. An even simpler approach, also used at Organization 5, is for developers to take a few days to teach the maintainers the general data flow of the new system.

The costs of coordinating between the new system development and maintenance units, especially for replacement systems, is a related disadvantage of L-forms. Here the problem is not so much the permanent loss of knowledge about the system in transferring responsibility from development to maintenance as it is the temporary sharing of knowledge needed to effect a smooth changeover. Where the new system replaces one or more existing systems, as it increasingly does (see Table II), this process is particularly delicate and trouble-prone. Required coordination costs may, however, be moderated by employing implementation teams as lateral integrating mechanisms [17, 38].

Increased costs of system acceptance by the maintenance unit must also be weighed against the quality benefits associated with acceptance tests. When systems are turned over to L-form maintenance units, some acceptance criteria established by that unit typically must be met. Documentation must be complete, all functions must be implemented, loose ends must be tied up. Sometimes meeting these acceptance criteria will require short-term investments which have unpredictable long-term value—documentation may never again need to be referenced or functions agreed to in specifications may ultimately not be needed. In A-type organizations these costs may be avoided or simply postponed until much later.

In L-form units, our studies suggest, acceptance costs may be moderated over time by the growth of mutual trust between departments, founded on their long-term relationship. In Organization 5, trust between units evolved over the 14 year period in which maintenance was centralized, and acceptance standards and methods were worked out in cooperation with the system development unit, eventually easing the acceptance process. Over time, the development group improved its compliance with standards and the maintenance group gained a better understanding of which standards were really important to them.

Finally, *possible duplication of communication channels to user organizations* is a concern with L-forms. Users must work with both new system development and maintenance units. Nevertheless, the costs of such duplication may be offset by other considerations. For example, the use of separate channels to resolve maintenance and new system development issues may be more effective in practice in that the integrity of each process is less easily compromised. However, we have no direct evidence that this is so from our cases.

CONCLUSION

In our view, the most significant insight in the above analysis lies in the trade-offs among the classical system parameters of quality, schedule, and budget. The most compelling advantage of the L-forms may be their potential for quality assurance and improved user service, which may have been neglected in application systems work due to the pressures of schedules and budgets associated with new system development. Adoption of an organizational form in which maintainers are separated from developers brings about a shift of emphasis to improved quality assurance and user service, we believe. This shift is also responsive to rising user expectations for information services, which have been fueled by user experiences with microcomputer products and services available in the marketplace.

Because of its quality and productivity improvement potential, we believe an L-form structure, in some variation, deserves consideration by many IS organizations, especially those with mature, well-developed application system portfolios, where services to day-to-day business operations are of central importance. At the same time, we would caution the unwary manager against seizing upon an L-form as a general solution to problems of IS productivity and service. Good organization design makes a difference in IS, as it does elsewhere, but organization design is more than organization structure [17]. The benefits of a separate maintenance unit are also dependent on good management at the head of the maintenance unit, recognition and rewards for its members, and viable systems acceptance criteria, our research suggests.

Most of the IS managers in our cases are attempting to derive some of the benefits of an L-form arrangement while maintaining the benefits they have achieved with A-form designs. Both forms are ways of departmentalizing around outputs. In the L-forms, the output focus is user service; with A-forms, it is software products. Since both of these are important outputs of the IS department, some combination of approaches seems appropriate. As possible combinations we note the various L-A and A-L forms, and also the design adopted at Organization 7, where an A-type form is combined with explicit managerial commitment to the installed system base.

Much remains to be learned about organization design for IS. Additional research is needed to investigate the relative productivity and quality benefits of A-forms and L-forms. Of particular interest may be the

user's response to differences in form. The task for IS researchers, in our view, is to accompany practitioners in their various organizational redesigns: they will observe their reforms as experiments, as originally suggested by Campbell [9], to better reveal the complex workings of the organizational process in the variety of settings in which it unfolds so that future practice may continue to be better informed.

Acknowledgments. We are grateful to Chris Kemerer and to three anonymous reviewers and the department editor for their helpful comments on earlier versions of this article.

REFERENCES

1. Bankar, R.D., Datar, S.M., and Kemerer, C.F. Factors affecting software maintenance productivity: An exploratory study. In *Proceedings of the Eighth International Conference on Information Systems* (Pittsburgh, Dec. 6–9, 1987), pp. 160–175.
2. Baroudi, J.J. The impact of role variables on IS personnel work attitudes and intentions. *MIS Q. 9*, 4 (1985), 341–356.
3. Baroudi, J.J., and Ginzberg, M.J. Impact of the technological environment on programmer/analyst job outcomes. *Commun. ACM 29*, 6 (1986), 546–555.
4. Bartol, K.M. Turnover among DP personnel: A causal analysis. *Commun. ACM 26*, 10 (1983), 807–811.
5. Bartol, K.M., and Martin, D.C. Managing information systems personnel: A review of the literature and managerial implications. *MIS Q.*, Special Issue (1982), 49–70.
6. Belady, L.A., and Lehman, M.M. A model of large program development. *IBM Syst. J. 15*, 3 (1976), 225–252.
7. Bendifallah, S., and Scacchi, W. Understanding software maintenance work. *IEEE Trans. Softw. Eng. SE-13*, 3 (1987), 311–323.
8. Boehm, B. The economics of software maintenance. Software Maintenance Workshop, Naval Postgraduate School, Monterey, Calif., Dec. 6–8, 1983, in R.S. Arnold (Ed.), Software Maintenance Workshop Record, IEEE Computer Science Press, N.Y., 9–37.
9. Campbell, D.T. Reforms as experiments. *Am. Psych. 24*, 4 (1969), 409–429.
10. Canning, R.G., Ed. That maintenance 'iceberg'. *EDP Analyzer*, (Oct. 1972), 1–14.
11. Canning, R.G., Ed. Easing the software maintenance burden. *EDP Analyzer* (Aug. 1981), 1–14.
12. Cheney, P.H., and Lyons, N.R. Information systems skill requirements: A survey. *MIS Q. 4*, 1 (1980), 35–43.
13. Couger, J.D., and Colter, M.A. *Maintenance Programming: Improved Productivity Through Motivation*. Prentice-Hall, Englewood Cliffs, N.J., 1985.
14. Couger, J.D., and Zawacki, R.A. *Motivating and Managing Computer Personnel*. Wiley Press, New York, 1980.
15. Dickson, G.W., and Wetherbe, J.C. *The Management of Information Systems*. McGraw-Hill, New York, 1985.
16. Frantz, D. BofA's plans for computer don't add up. *L.A. Times* (Feb. 7, 1988), 1 ff.
17. Galbraith, J. *Designing Complex Organizations*. Addison-Wesley, Reading, Mass., 1973.
18. Ginzberg, M.J., and Baroudi, J.J. MIS careers—A theoretical perspective. *Commun. ACM 31*, 5 (1988), 586–594.
19. Goldstein, D.K., and Rockart, J.F. An examination of work-related correlates of job satisfaction in programmer/analysts. *MIS Q. 8*, 2 (1984), 103–115.
20. Gremillion, L.L. Determinants of program repair maintenance requirements. *Commun. ACM 27*, 8 (1984), 826–832.
21. Guimaraes, T. Managing application program maintenance expenditures. *Commun. ACM 26*, 10 (1983), 739–746.
22. Haffner, K. Is your computer secure? *Bus. Week* (Aug. 1, 1988), 64–72.
23. Izzo, J.E. *The Embattled Fortress: Strategies for Restoring Information Systems Productivity*. Jossey-Bass, San Fran., Calif., 1987.
24. Kim, C., and Weston, S. Software maintainability: perceptions of EDP professionals. *MIS Q. 12*, 2 (1988), 167–185.
25. Lehman, M.M. Programs, life cycles, and laws of software evolution. In *Proceedings of the IEEE. Special Issue on Software Engineering, 68*, 9 (Sept. 1980), 1060–1076.
26. Lientz, B.P., and Swanson, E.B. *Software Maintenance Management*. Addison-Wesley, Reading, Mass., 1980.
27. Lientz, B.P., and Swanson, E.B. Problems in application software maintenance. *Commun. ACM 24*, 11 (1981), 763–769.
28. Lientz, B.P., Swanson, E.B., and Tompkins, G.E. Characteristics of application software maintenance. *Commun. ACM 21*, 6 (1978), 466–471.
29. Liu, C.C. A look at software maintenance. *Datamation 22*, 11 (1976), 51–55.
30. Martin, J., and McClure, C.L. *Software Maintenance: The Problem and Its Solutions*. Prentice-Hall, Englewood Cliffs, N.J., 1983.
31. Mooney, J.W. Organized program maintenance. *Datamation 21*, 2 (1975), 63–64.
32. Port, O. The software trap: Automate—Or else. *Bus. Week Special Report*, (May 9, 1988), 142–154.
33. Porter, M.E., and Millar, V.E. How information gives you competitive advantage. *Harvard Bus. Rev. 63*, 4 (1985), 149–160.
34. Riggs, R. Computer systems maintenance. *Datamation 15*, 9 (1969), 227, 231–2.
35. Schneidewind, N.F. The state of software maintenance. *IEEE Trans. Softw. Eng. SE-13*, 3 (1987), 303–310.
36. Software Maintenance News. *Numbers from England*, 7 (July 1989), 8.
37. Swanson, E.B. The dimensions of maintenance. In *Proceedings of the Second International Conference on Software Engineering* (San Fran., Calif., Oct. 13–15, 1976), pp. 492–497.
38. Swanson, E.B. Organizational designs for software maintenance. In *Proceedings of the Fifth International Conference on Information Systems* (Tucson, Ariz., Nov. 28–30, 1984), pp. 63–72.
39. Swanson, E.B., and Beath, C.M. The demographics of software maintenance management. In *Proceedings of the Seventh International Conference on Information Systems* (San Diego, Calif., Dec. 15–17, 1986), pp. 313–326.
40. Swanson, E.B., and Beath, C.M. The use of case study data in software management research. *J. Syst. Softw. 8* (1988), 63–71.
41. Swanson, E.B., and Beath, C.M. *Maintaining Information Systems in Organizations*. Wiley, Chichester, England, 1989.
42. Vessey, I., and Weber, R. Some factors affecting program repair maintenance. *Commun. ACM 26*, 2 (1983), 128–134.
43. Willoughby, T.C. Staffing the MIS function. *Comput. Surveys*, 4, 4 (1972), 241–259.
44. Yin, R.K. Case Study Research: Design and Methods. Sage, Beverly Hills, Calif., 1984.
45. Zmud, R.W. Management of large software development efforts. *MIS Q. 4*, 2 (1980), 45–55.
46. Zmud, R.W. Design alternatives for organizing information systems activities, *MIS Q. 8*, 2 (1984), 79–93.
47. Zvegintzov, N. Immortal software. *Datamation* (June 15, 1984), 170–180.

CR Categories and Subject Descriptors: D.2.9 [**Software Engineering**]: Management—*productivity*; K.6.3 [**Management of Computing and Information Systems**]: Software Management—*software maintenance*
General Terms: Departmentalization
Additional Key Words and Phrases: Development, maintenance, management, organizational forms

ABOUT THE AUTHORS:

E. BURTON SWANSON is associate professor of Information Systems at the Anderson Graduate School of Management. His research interests include the organizational uses of computer-based information systems, and organizational approaches to system development, implementation and maintenance. Author's Present Address: Anderson Graduate School of Management, University of California, Los Angeles, CA 90024.

CYNTHIA MATHIS BEATH, assistant professor of Information and Decision Sciences at the University of Minnesota, is currently a visiting assistant professor at the Anderson Graduate School of Management, UCLA. Her research interests focus on the partnership between the IS function and its users. Author's Present Address: Carlson School of Management, University of Minnesota, Minneapolis, MN 55455.