Software maintenance in fourth-generation language environments

by PAUL C. TINNIRELLO
AT&T Communications
Piscataway, New Jersey

ABSTRACT

It is often asserted that fourth-generation languages will resolve the problems associated with software development in traditional languages, and in particular the technical and morale problems of software maintenance. The analysis of this paper suggests that fourth-generation languages do not solve all of the present problems of maintenance, and indeed they can introduce problems of their own. The successful user of fourth-generation languages will be the organization that takes appropriate countermeasures.
INTRODUCTION

The evolution of software technology coupled with the demand for more productivity from data processing organizations has prompted a widespread appeal for fourth-generation languages (4GL). The term fourth-generation language is applied to a class of DP languages developed in the mid-1970s that offer simplified expressions for common DP tasks. These languages allow for system development in significantly less time than third-generation languages such as COBOL, FORTRAN, and PL/I. Advocates of these new languages are confident that they will lessen many of the problems that have burdened traditional language environments.1,2 Such problems include a heavy backlog of requests, lack of maintainability, lack of adaptability, and human resource issues. High expectations, however, especially in the area of software maintenance, could lead to disappointment for many professionals who are seeking to escape the frustrations encountered in third-generation systems.3 The overconfidence in a fourth-generation language’s ability to eliminate most of the software maintenance issues could seriously jeopardize the recent efforts to improve software maintenance attitudes. An untimely eagerness to abandon concern for software maintenance could also compound maintenance problems in systems using fourth-generation software.

This paper focuses on several fundamental issues of software maintenance that will continue to exist in many fourth-generation language environments. It is not the intention of this paper to critique the overall effectiveness of fourth-generation languages or to evaluate the necessity of their use, but rather to discuss the impact of these new languages on the software maintenance process.

FACTORS INFLUENCING THE USE OF FOURTH-GENERATION LANGUAGES

Before examining maintenance issues in fourth-generation language environments, it would be advantageous to review the following factors that are promoting the spread of these new language systems:

1. System development problems
2. Maintenance problems and request backlogs
3. Increased DP knowledge by users
4. Pressure for productivity/accountability

System Development Problems

Apportioning of monetary resources and time has been an inherent consideration in traditional system development. Software costs may account for as much as 80% of the total cost of system development.4 Much of this cost is attributed to escalating programmer salaries; further, many user groups resent department budget cuts because of the high expenses incurred by data processing.5 Often the time required to replace and/or develop a system is much longer than is acceptable to end users. Fourth-generation languages offer system development with less effort than traditional development techniques, thus offering a savings in time and cost.

Maintenance Problems and Request Backlogs

Software maintenance problems have been a well-known stumbling block for years. Negative attitudes about maintenance work are held by DP managers, programmers, and end users.5 These attitudes inhibit the necessary effort needed to perform maintenance work successfully. Maintenance problems also include unmaintainable, unadaptable programs and systems. Some systems do not fulfill user requirements and specifications. Numerous corrective measures within these systems have left them in an unmaintainable state. Other systems have been poorly designed, and modifying or enhancing their capability is only possible through rewriting large portions of the system. Software maintenance accounts for as much as 50%—80% of the software activities performed by programmers.6

System maintenance problems have created a flood of user requests. Often users submit requests for replacement systems even though there are already numerous outstanding requests for maintenance work on these systems. Request backlogs in some companies may be as high as 2 1/2 years.3,7,8 Vendor advertising of fourth-generation systems emphasizes the vast improvement in application development over traditional programming methods. These advertising claims support the premise that maintenance problems and lengthy request backlogs, which are attributed to traditional programming, will be reduced.

Increased DP Knowledge by Users

With new technological applications in industry, many end-user professionals have sought to become more computer literate. Much of this need for literacy is due to an increased number of automated business functions that require data processing knowledge to use with them. In other instances, professionals have educated themselves in preparation for the new technology of the personal computer. With this increased knowledge of computing technology, end users are becoming less dependent on data processing professionals and computing resources.5 Many users want to apply their new found
computing knowledge at the workplace in an attempt to make their jobs easier. Numerous vendors claim that their fourth-generation language allows writing a program to be a simple and uncomplicated task. This simplicity gives the non-data-processing professional the ability to use computing resources advantageously. 9

Pressure for Productivity/Accountability

The problems of development, maintenance, and request backlogs have been apparent to many non-data-processing professionals in the computer-based organization. User groups are expressing dissatisfaction with budgetary spending as more funds are being allocated to data processing departments rather than to user departments. The reaction from upper management in response to this dissatisfaction has been to allow end users to write their own application programs. Upper management would ultimately be making end-user departments more accountable and productive in relation to data processing activities. It has already been recognized that end users have a tremendous optimism toward the use of fourth-generation software products. 7 Vendors of these products propose that the simplicity and ease associated with programming will allow the end user the time to write more programs and thus increase productivity.

EXAMINING THE MAINTENANCE ISSUES

In a previous paper, the author suggested that the software maintenance process could be segmented into the three following areas: maintenance management, maintenance programming, and maintenance attitudes. 5 Maintenance management is defined as the management process necessary when performing maintenance tasks. Maintenance programming is defined as the technical methodology in which a correction, modification, or enhancement takes place. Finally, the maintenance attitude is defined as the position that programmers, managers, and users take toward maintenance tasks.

The fourth-generation language environment will not solve all the problems associated with the three areas of the maintenance process. More specifically, the remaining problem areas of the fourth-generation language environment with respect to the three maintenance segments are the following:

1. software ownership responsibility
2. documentation
3. software selection and quality assurance
4. product releases and software warranty
5. software standards

Software Ownership and Responsibility

Software ownership/responsibility is defined as a policy for maintaining the programs and systems written in fourth-generation languages. 5 The need for such a policy becomes evident when consideration is given to several facts that will be present in the fourth-generation language environment. First, there will be a diverse population of potential language users that include data processing and non-data-processing professionals with varied technical skills. Second, there will be a need to make changes to fourth-generation application programs for product release changes, for business changes, and for ensuring hardware efficiency. Although the last issue was present in third-generation systems, it was usually resolved by a narrower population of language users—programmers. With a broader population of fourth-generation language users, there is concern as to how capable end users are in making the changes to application programs described above. 3, 8

Software ownership/responsibility can be categorized as part of maintenance management. It will require the cooperation and coordination of upper management, data processing, and end-user departments. Without formal agreement on software ownership/responsibility, the maintenance management function would be ineffective and more complex. In addition, required changes to fourth-generation language programs might be circumvented by the data processing and end-user departments as a result of the conflict over which group is better prepared to perform maintenance tasks. Numerous companies who have already implemented fourth-generation language systems may still be in the process of developing ownership/responsibility policy. 10

One of the more popular ownership/responsibility methods is found in the information center concept that numerous companies are implementing. 10 In this method, the information center serves both as a product support group for vendor changes and as a consulting group for end-user applications. While this approach has merit, there is still a need to manage the end-user application system more carefully. A suggested method of ownership/responsibility that will help the maintenance management function is the designation of end-user department specialists. These individuals will manage the use of the software product or group of software products at the department level. Operating within an established set of company standards for a given product, these specialists will help ensure the quality of code, documentation, and standards in end-user systems. The end-user department specialists will need to have skills in both the department business function and the application software product.

A potential danger of such a position is the reliance of the ownership/responsibility process on one individual. These highly specialized end users could find their combination of business and data processing skills very marketable. It is possible that they would begin the migratory habits that have long been associated with data processing professionals. This might create a kind of maintenance problem similar to the one that exists with programmers in traditional language systems. Perhaps assigning several people as end-user specialists could alleviate this problem.

Another alternative for software ownership/responsibility is the formation of an end-user product group. This group would serve all end users of a specific software product in the computer-based organization. The end-user product group would oversee all development and maintenance work performed with a particular language system. The group would also be responsible for reviewing documentation, release lev-
Documentation

Documentation problems are not unique to data processing departments. Many end-user departments, which also require retained written information specific to their business functions, suffer from poor or too little documentation. Documentation activities are often the last phase of a business development project, and frequently this phase is hastily completed in an effort to meet project deadlines. In traditional data processing systems, maintenance programmers found that documentation problems accounted for much of the difficulty in maintaining a system. Poor documentation has caused programmers to spend hours tracking errors that could have been taken minutes to locate. Inadequate documentation has also made it difficult to locate key system areas where modifications and/or enhancements are required. Although the fourth-generation language concept deemphasizes the need for elaborate documentation, there are still several issues about these new languages that make documentation vital to the software maintenance process.

The first issue is that some fourth-generation languages are not as self-documenting as the software vendor would have end users believe. In fact, some end users find that many new languages are complex and not so user-friendly. This complex command structure, combined with potential use of complicated end-user logic, could make the maintenance process in new language systems as difficult as it is in third-generation systems.

Another issue concerning documentation is evident when considering the life expectancy of systems developed with fourth-generation software. In several companies, new language systems have been running in a production mode for as long as five years. It would seem unlikely that the applications developed with fourth-generation software would immediately be replaced when the next software evolution occurs. Such is the case for many third-generation systems, which will probably continue to run for the next decade. Given this anticipated longevity of new language systems, it is only practical to document them.

Still another documentation issue is changes in the business environment that will probably occur during the life cycle of an end-user application system. End-user professionals are subject to promotions, career changes, and relocation changes. Without documentation, systems written by these individuals become extremely difficult to maintain from the viewpoints of both business function and programming function.

Finally, the ad hoc development technique that many envision as commonplace in the fourth-generation language environment may create an ad hoc attitude about documentation needs. Thus, end users may become lax in their creation of meaningful program and system documentation.

Several documentation practices can be implemented by the computer-based organization to help minimize software maintenance problems. One practice would be for the end user to document an application system on a business level. Included in this business application description is a section that identifies special algorithms or formulas that are used within the end user program. A business application description is similar to the high-level functional overview found in traditional documentation, except that it reflects more of the business functions than technical functions. This may encourage end users to document more thoroughly, since the style of documentation is in business terminology. Another documentation technique is the establishment of standards that require documentation to be written on the basis of the length of the end-user program or the number of executable commands. This may vary from one software language to another, depending upon the clarity of the command language. Each technique, of course, will require the review of either the end-user departmental specialist or the end-user product group. Whichever documentation plan is chosen, it is important that the issue of documentation be recognized as an integral part of both the software maintenance process and the successful use of the end-user application system.

Software Selection and Quality Assurance

Selecting a fourth-generation language system that will serve the broad needs of the computer-based organization warrants careful consideration. Besides finding agreement among end users on application needs, the advertising strategy used by software vendors makes the selection process difficult even for data processing professionals. An important aspect in software selection is acquiring a language system that will fit the needs of end-user applications. Without a close matching of application needs and language capabilities, end users will struggle with programming logic in an effort to achieve the desired result. Usually the struggle in language usage results in the use of trick code techniques. Often found in third-generation programs, trick code is extremely difficult to correct, modify, or enhance, because the logic does not follow the intended vendor system design. Therefore, the fourth-generation software selected should be readily adaptable to the present business environment.

A variety of techniques can be used to survey the list of potential software products; however, the best method for deciding on the final end-user product is to pilot the system within a typical application environment. The software piloting phase can best determine the true application capabilities, as well as help establish a set of language standards to be followed by end users when the language system is finally installed. The software selection process will affect both the maintenance management and maintenance programming segments of the software maintenance process.

Quality assurance is as vital a process in fourth-generation language environments as it is in third-generation systems. The quality assurance function should certify that the final end-user system performs all the functions for which it was designed. Quality assurance should also review the efficiency
of the end-user application programs, since fourth-generation languages can make a heavy demand on hardware. Quality assurance affects the software maintenance process in several ways. First, poorly designed systems will eventually require corrective action if the system is to remain functional to the end user. As with traditional systems, corrective maintenance can be an ongoing process. Second, inefficient programming techniques and poor program design will complicate modifications and enhancements made to a system. Though modifications and enhancements may be accomplished with greater ease in fourth-generation language systems, the possibility of producing unmaintainable systems can still exist if intended structured procedures are not followed. Finally, the potential problems of trick code from either poor product selection or unique application requirements can make system maintainability poor. This can be even more critical in fourth-generation language systems, because vendors may not be providing diagnostic tools.

The quality assurance function should be addressed within the software ownership/responsibility phase of the new language implementation process. A central quality assurance group could be formed with both data processing and end-user professionals. Aside from exercising their normal function, the quality assurance group would educate end-user specialists who are unfamiliar with programming logic and design. It would be unfair to expect all end users to possess the design and logic expertise acquired by data processing professionals. The end result of this education process would help prevent maintenance problems that are created as a result of poorly designed programs and systems.

Product Releases and Software Warranty

The appeal of new software language systems has created a very competitive environment for software vendors of fourth-generation languages. Many data processing professionals recognize that the software marketplace is flooded with products claiming to have fourth-generation technology. This highly competitive environment has created two important issues that will affect the software maintenance process.

The first issue is that of vendor product releases. When traditional languages systems such as COBOL and FORTRAN underwent release changes, certain difficulties were encountered. In numerous companies, conversions from one release level to another took months and perhaps years to complete, even when vendors provided conversion tools and aids. Although the release changes were supposed to provide upward compatibility, there were countless programs that required line-by-line examination for conversion conflicts.

Release changes have been infrequent in traditional languages when consideration is given to the length of time these languages have been used. There exists a strong possibility, however, that product release changes for fourth-generation language systems will be much more frequent than with traditional languages. The primary reason for these potential release changes, in the author's opinion, is the competitive environment in which software vendors must survive. When a software vendor issues a product with capabilities not found in current fourth-generation systems, there is a tendency for other software vendors to match product capabilities in order to preserve their share of the software marketplace. As mentioned previously, language releases can wreak havoc on the software maintenance process, even though a promise of upward compatibility is given by the vendor.

The second maintenance issue derived from the competitive software marketplace is that of software warranty. When vendors upgrade language capabilities as a result of competition, what guarantees are extended to the computer-based organization concerning the reliability of the new product? The traditional languages of COBOL and FORTRAN have national committees that carefully evaluate language release changes, and even then there are upgrading problems. In the competitive software marketplace, vendors may not have the time that is required to test a new release level thoroughly; the consequences to the software maintenance process and to the computer-based organization are severe. Warranty problems also occur when a software vendor quits the marketplace, leaving the product, and therefore the end users, unsupported.

Product releases and warranty issues affect all segments of the software maintenance process. Maintenance management is affected whenever product release conversions are required. Maintenance programming is required when release conversions fail as a result of special language and logic uses. Maintenance attitudes are affected by the frequency of release changes and the frustration associated with them.

An important consideration in selecting a fourth-generation language system is the reliability of the software vendor. It is beneficial to examine the business history of a prospective vendor and also to inquire about the software warranty. The time invested in selecting a competent software vendor will minimize the problems the computer-based organization will encounter through frequent and unwarranted release changes.

Software Standards

A deficiency in many fourth-generation languages is the absence of a standard set of language commands. Considering the variety of specialized software products available on the marketplace, it is probable that numerous companies will use more than one fourth-generation language system. End users, who interact with these systems, will find that a lack of standardized commands among products can be confusing and frustrating. This will be especially true when users write applications with commands that are familiar to them from one language and expect similar results when using other language systems. Nonstandard language commands can yield functional errors that will require correction, either at the time of design or through a maintenance request.

Until language standards are established, it will be important for users to gain an awareness and understanding of the possible differences that exist between new language systems. This information can be imparted by the end-user department specialist or the end-user product group. Without this awareness, much confusion will probably develop among users who work in a multiple-software-product environment.
Another important issue of standardization is the incompleteness of external language interfaces found in numerous fourth-generation languages. This problem can be subdivided into two areas. First, there are few, if any, standard interfaces among fourth-generation products from different vendors. Second, there are poor interface standards for the traditional languages of COBOL, FORTRAN, and PL/I. With the bulk of information stored within traditional systems, there will be a definite need for application programs written in fourth-generation software to interface with many of the existing systems.

For many companies that have implemented fourth-generation software, interfacing problems may have been circumvented by using data transfer programs written in traditional software languages. These data transfer programs are highly specialized and frequently require modifications whenever the data input requirements of an end-user application program change. Data transfer programs may also need modification whenever changes are made to either the new software language, through release upgrades, or to the old software system, through normal maintenance. The data transfer method used to solve the interface problems of fourth-generation software will require both maintenance management and maintenance programming activities. The maintenance management process will be further complicated by the possibility of highly dynamic data interfacing between new language systems and the existing traditional systems. Unless data interfacing requirements between systems are controlled, maintenance personnel could be spending most of their time modifying the data transfer programs.

One technique that can be used to control frequent modifications to data transfer programs is to establish a selection criterion for data items from the corporate database that will be available for end-user application programs. The criterion for selecting transferable data should consider the data items that are most often used in company business functions. Once established, this selection criterion would prohibit end users from making frequent and special requests for transferable data that are infrequently used by the majority of end-user departments. An extension of this technique might be to develop a group of data transfer programs based on different criteria, as prescribed by end-user departments. In this manner, end-user departments can access data unique to their business function and still remain within a controlled process.

CONCLUSION

The movement toward using fourth-generation languages in the computer-based organization is understandable. However, the software maintenance process, as shown in this paper, is an area that will continue to exist in the fourth-generation language environment. This critical fact should be recognized by organizations that are planning to use these new language systems. Forgoing the recognition of maintenance issues will generate unrealistic expectations in the end-user community that will eventually lead to disappointment and frustration. In addition, the efforts to improve the current maintenance process will suffer as a result of increased complexity.

As important as the recognition of the continued maintenance process is the selection of an implementation strategy that reflects the limitations and capabilities of fourth-generation languages. This strategy should include techniques, like those suggested in this paper, that help reduce the impact of fourth-generation languages on the software maintenance process. Finally, the organizations that successfully use fourth-generation software will be those that have not been deceived into thinking that technical advancements that have solved some problems have solved all problems.

REFERENCES
