To:	MTB Distribution
From:	Gary C. Dixon
Date:	April 23, 1982
Subject:	Multics C&F Maintenance Study

INTRODUCTION

This memo describes the strategy currently employed by the Multics Development Center (MDC) to correct software problems. The strategy is composed of two parts: methods for fixing reported problems; and methods for getting the fixes to customer sites.

In the past, customers have identified problems with the maintenance strategy, both in the level of maintenance provided for the software and in the frequency at which fixed software was provided to the field. To address these problems, MDC is starting to use a revised software maintenance strategy.

Several aspects of the strategy are experimental in nature. During the next six (6) months, these experimental approaches will be scrutinized for their effectiveness in dealing with C&F problems. This MTB proposes criteria for evaluating effectiveness.

A subsequent MTB will describe the observed effectiveness of the experimental approaches. Publication of the results MTB is scheduled for October, 1982. It will either recommend permanent adoption of the experimental approaches used during the study, or it will recommend changes to the studied approaches (and perhaps further study) based upon the results achieved during the six month study period.

METHOD FOR FIXING PROBLEMS

MDC software maintenance procedures stem from the development process used in the Multics Development Center. For each set of software modules, one group of people handles all aspects of the development process, including maintenance. Such development teams are always small, often consisting of only one person.

Software maintenance is an integral part of the software development process. Maintenance activities fall into two categories: maintenance of software whose development is continuing (active software); and maintenance of software no longer under development (dormant software). This MTB primarily addresses the maintenance of dormant software, although maintenance of active software is briefly described.

Maintenance of Active Software

Maintenance of an active software product is performed by the development team for that product. Problems identified during one development cycle are corrected during the next cycle. Development cycles are kept short so that the few problems which reach the field are corrected in a timely manner. For example, in Multics Release 10, problem fixes will be released every 6 months as part of a complete Multics release package.

The cost of product maintenance is minimized by having the development team perform maintenance activities as part of the normal development cycle. This strategy also provides incentives for the development team to avoid introducing new problems.

Maintenance of Dormant Software (An Experimental Approach)

Maintenance of dormant software is handled by a special Software Maintenance team within the Multics Development Center. This team corrects problems in one of two ways: maintenance may be performed on behalf of the maintenance team by someone from the original development team; or it may be performed by a maintenance team member whose primary task is correction of problems in dormant software.

In 1982, the Software Maintenance team was shifted from Cambridge to Phoenix, and more resources were applied to address the maintenance of dormant software. As part of their support for new hardware, Phoenix personnel have increased their expertise in hardcore supervisor and related functions. This added expertise, plus their knowledge of the user environment, makes these people well-prepared to deal with the software maintenance function.

EFFECTIVENESS OF DORMANT SOFTWARE MAINTENANCE

This approach to maintaining dormant software is not inherently experimental. We've used a similar approach for several years. However, both customers and management have become concerned over the increasing number of dormant software problems.

MDC has increased the resources applied to correction of errors in dormant software. For example in 1981, MDC applied less than .5 man year to such maintenance. In 1982, this figure has been increased to 2.5 man years, distributed among 5 people in the MDC C&F project.

Increasing resources applied to dormant software maintenance is an experimental approach. MDC will evaluate the effectiveness of this approach using several criteria:

(1) Currently, there are about 450 Trouble Reports (TRs) relating to dormant software. 200 of these report problems and the remaining 250 suggest enhancements to dormant software. The maintenance strategy will be judged effective if 100 of these TRs (63 problems and 37 suggestions) can be resolved by the October 1, 1982.

(2) If the rate of entry for new problems is faster than the rate at which the problems can be corrected, then MDC will still not be solving the maintenance problem with the added manpower. The maintenance strategy will be judged effective if more TRs on dormant software are corrected than are entered during the period from March 1, 1982 through October 1, 1982.

METHOD FOR DISTRIBUTING FIXES

In parallel with increased manpower applied to fixing problems, MDC is trying several experimental methods to improve distribution of fixes to the field. These methods are aimed at providing better maintenance support of Multics software to customer sites.

The Distribution Problem

In the past several years, MDC has distributed fixes to customers only as a part of each major Multics software release (eg, MR7.0, MR8.0, MR9.0). Because major releases are separated 12-18 months apart, the fix cycle (from the time a problem is reported until the fix reaches the customer site) for problems could be a year or more. Such lengthy fix times have led to customer dissatisfaction with the maintenance level of Multics software.

Because fixes were distributed only with major releases, there was little that MDC could do to reduce fix cycle times seen by the customer. Even when a problem fix was available on System M one month after being reported, the fix did not reach the customer until the next major release.

Of course, certain critical problems encountered by customers had to be fixed immediately to keep the customer systems running. Customers reported such problems directly to MDC. Problem fixes were generated by developers (interrupting their normal development activity) and transmitted from System M to the customer site via dial-out (or in rare cases via a revised system tape). However, the high cost of this approach (in developer interruptions and direct site support interfacing) restricted use of this approach to the small set composed of the most critical problems.

Even such highly-personalized attention in providing critical problem fixes was not totally acceptable to customers. Critical fixes were developed and distributed to customers as they encountered the problem. There was no automatic mechanism to notify customers that a critical problem existed and to provide the problem fix. So different customers would encounter a given problem, spend site resources trying to diagnose or bypass the problem, and then would become disconcerted that they had not been notified of the problem or its fix.

A Possible Solution

In the past several months, MDC has considered several possible solutions for the fix distribution problem. Until now, the most promising was to ship a Bug Fix Release (BFR) tape to each site on a quarterly basis.(1) The tape would include problem fixes for the most recent (major or minor) Multics software release (eg, MR9.1). While this scheme is appealing, it suffers from two flaws which prevent its adoption at the current time.

First, creation of BFR tapes would require significant MDC manpower (to identify and separate problem fixes from new functionality, to store problem fixes separately, to generate and test installation instructions for the BFR tape, etc). It was felt by many that such manpower could be more effectively applied to fixing additional problems, rather than to providing BFR tapes.

The second flaw is that fixes on a BFR tape would not receive any significant testing or exposure in the environment of the most recent release. LISD does not have a Multics system which runs the most recent Multics release. System M and the MIT Exposure Site are continually updated with software for the next release. So there is no system available to MDC on which testing and exposure of BFRs could be performed.

(1) See MTB-539 for a complete discussion of Bug Fix Release tapes, as well as other methods of distributing problem fixes.

Because the Bug Fix Release tape strategy will not work, MDC has decided to experiment with separate approaches for distributing critical and non-critical problem fixes.

Distributing Non-Critical Fixes (An Experimental Approach)

To address the long fix cycle times for non-critical problem fixes, MDC will ship problem fixes with major Multics releases (eg, MR10.0) and in minor Multics releases (eg, MR10.1, MR10.2). For a variety of reasons (smooth implementation staging for new software, ease of conversion to new features, etc), it is desirable to ship a complete set of system software (rather than just incremental changes) with each minor release. By including problem fixes in minor releases, these problem fixes will be available to customer sites at six-month intervals. This approach will significantly reduce fix cycle times.

The approach is experimental because MDC is not certain how it will be received by customers. In the past, some customers (who run unmodified system software) have asked for more frequent Multics releases so they can get problem fixes more quickly. Other customers (who tailor system software to meet special site needs) prefer less frequent releases because applying site modifications to a new release requires significant site manpower. Reducing the frequency of new releases reduces the conversion manpower required at such sites.

In an attempt to satisfy both types of customers, MDC is coupling the approach of releasing complete software in minor releases with a policy of allowing sites to skip some minor releases when upgrading from one release to another.

EFFECTIVENESS OF NON-CRITICAL FIX DISTRIBUTION

The effectiveness of shipping problem fixes more frequently must be evaluated in the context of the overall strategy to ship complete system software with minor releases. The basic criteria for evaluation (customer acceptability) will be measured subjectively from customer feedback received throughout the MR10 time frame. Shipment of problem fixes cannot be separately evaluated because customer acceptance will probably be based upon other factors which overshadow the fix distribution issue.(1) In

---

(1) Factors which customers might consider in evaluating the ship strategy include: costs of installing releases more frequently; problems encountered during installation because a prior minor release was skipped; and difficulties associated with unavailability of problem fixes or new features because a site decided to skip a minor release.

addition, MDC will evaluate the internal impact of the software ship strategy on development cycles, cost of preparing more frequent releases, etc. Evaluation of such general factors is outside the scope of this study. Thus, it will not be possible to evaluate the effectiveness of shipping problem fixes with each minor release.

If MDC decides to continue shipping complete software for each minor release in the MR11 time frame, then problem fixes will be shipped with these releases (primarily because of the difficulty in separating problem fixes from new functionality, a problem which stems from fixing problems as part of the normal development process).

Distributing Critical Fixes (An Experimental Approach)

To address the need for automatic site notification and distribution for critical problem fixes, a database describing all known critical problems and their associated fixes will be maintained online, on system M. Fixes will still be generated by MDC developers as new, critical problems are encountered at a site. However, once a problem fix is known, it will be placed in the online data base as notification to other sites.

A representative from each site will have access to this data base. He can review the list of known problems, and select fixes needed by his site. The fixes are presented by comparing the original and fixed source. These comparisons are usually small enough to be printed on a terminal, and then applied to the remote source module by hand. Alternately, MDC can provide dial-out or tape transfer facilities for larger changes.

Appendix A provides a complete description of the online database.

EFFECTIVENESS OF CRITICAL FIX DISTRIBUTION

The effectiveness of the online database approach for notifying sites of critical problem and distributing fixes will be evaluated subjectively, based upon comments from the site representatives who use the database during the study period (March 1 through October 1, 1982). The approach will be judged effective if it appears from these comments that critical problems are being effectively resolved. Comments from site representatives will be summarized for inclusion in the results MTB to be written in October.

APPENDIX A

DISTRIBUTION METHOD FOR CRITICAL PROBLEM FIXES

In order to address the problem of distributing fixes for critical problems to all sites (not just to the site which encountered the problem), the following approach will be employed.

(1) A new directory, >udd>SysMaint>fixes, will be created on the Root LV. This directory will contain a forum meeting which announces existence of a critical fix (see item 2), and will contain subdirectories which hold the actual fixes (see item 3). Only SysMaint will have sma on the directory. All other projects having a need to examine fixes (eg, Multics, MCOBOL, Doc, Pubs, SiteSA, TR) will have s access to the directory.

(2) A forum meeting, Critical_Fixes (fixes), will be created in >udd>sm>fixes, with links to it placed in >udd>SiteSA>sam (where uclog lives). Only SysMaint will be able to add transactions to this meeting. All others in the list above will be able to read these transactions, but not add to the meeting.

All transactions in the meeting will announce a critical fix to be distributed to the field. The transaction number for a fix announcement will be used to identify that critical fix.

Comments/discussion of any given fix will occur in the uclog meeting, to which SiteSAs and others have write access.

(3) For each fix, the announcing transaction will identify the following items:

- symptoms of the problem (using wording to allow keyword searches)
- description of the actual problem
- releases in which problem is known to exist
- modules involved in fix
- type of fix (eg, cpa, new source module, etc)
- releases to which fix applies
- pathname(s) of the fix directory

(4) Subdirectories will be created under >udd>sm>fixes for each release for which fixes are available (eg, >udd>sm>fixes>MR9.0). Access to these subdirectories will be the same as on >udd>sm>fixes.

(5) For each fix which applies to a given a release, a subdirectory will be created under that release's directory (eg, >udd>sm>fixes>MR9.0>fix_1). Access on the directory and to segments in the directory will be the same as on >udd>sm>fixes. An exec_com will be provided for creating these directories, setting ACLs/IACLs, etc. All segments associated with the fix (eg, cpa output, copy of the original source, copy of the modified source, etc) will be placed in that directory. If a fix applies to several releases, links to the fix directory in one release directory can be placed in other release directories.

(6) After the first such fix is announced, I will place a transaction in uclog announcing: the existence of the Critical_Fixes meeting; and ground rules for its use and for commenting on transactions contained in the meeting. I will also update the SiteSA guidelines mail which we send to new SiteSAs to include a discussion of this fix distributio mechanism.

(7) One of our concerns is that developers occasionally encounter problems (on System M, at MIT, at CISL, or in changing the code themselves) which would warrant use of the above mechanism (critical fix distribution to all sites). To address this concern, an MTB discussing our overall C&F Maintenance Strategy will include a suggestion to developers to notify MSS of any changes they are making which would warrant a critical fix, in addition to their normal installation on System M for the upcoming release. I'll also include such plea in the uclog announcement, since many of the affected developers attend uclog.

(8) Frank Martinson is considering the possibility of sending a letter to all sites announcing this distribution mechanism, and encouraging their use of the SiteSA project for this and other reasons (eg, reporting site status, reporting TRs, discussing problems in uclog, etc).