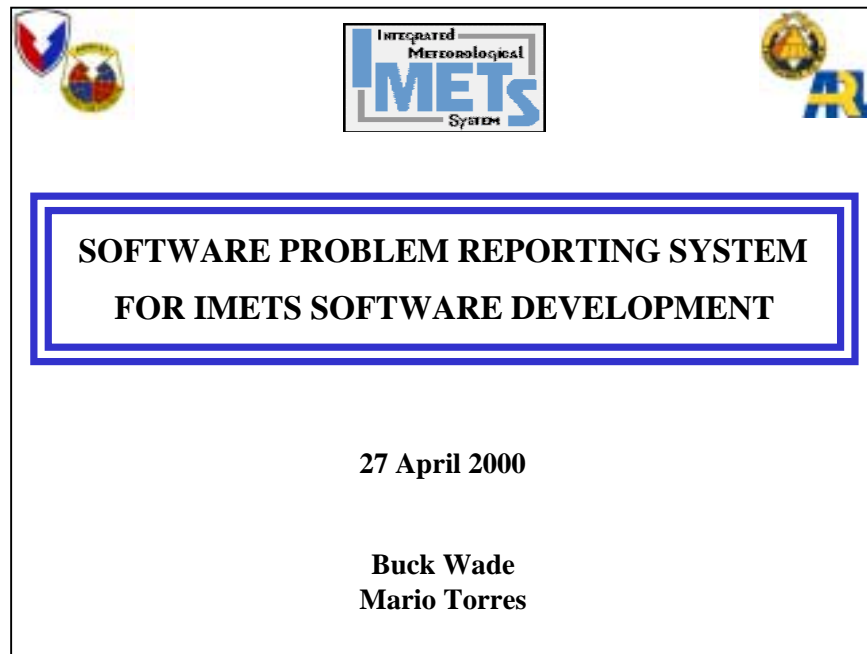


Software Problem Reporting System **For IMETS Software Development**


BY BUCK WADE & MARIO TORRES

APRIL 27, 2000



Good afternoon. My name is Buck Wade, and my partner in preparing this presentation, who is presently sitting in the audience, is Mario Torres. We are from the Battlefield Environment Division which is part of the Army Research Laboratory, or ARL, located at White Sands Missile Range in New Mexico. Today, I will attempt to describe the evolution of the software problem reporting system, or SPR system, that we use today to record problems found in the software developed by my colleagues at ARL for the Integrated Meteorological System or IMETS. I will review the woes we encountered in using a manual, hand written SPR system and the benefits we gained by using an almost fully automatic SPR system like the one we developed for use in IMETS. Perhaps this presentation will allow you to compare our system with the one used by your agency, and may even convince you to adapt our system to your needs.

What is software problem reporting? It is the process of bringing errors, discrepancies, or desired changes in software programs to the attention of the software developer with the desired effect of having the software corrected.




**WHAT IS SOFTWARE
PROBLEM REPORTING ?**

- IDENTIFIES & DOCUMENTS PROBLEMS IN SOFTWARE
- IDENTIFIES NEEDED IMPROVEMENTS
- SEEKS SOLUTIONS

- KEY ELEMENT OF CONFIGURATION MANAGEMENT (CM)

Software problem reporting is a key element of software configuration management and is one of the major factors around which software configuration management was developed. Configuration management originated in the hardware and construction industry around the need to make changes to design blueprints, to notify all users of the blueprints about the changes, and to keep track of those changes. Software problem reporting is similar in that it is the means for requesting software changes, whether it be for correcting an error or for making a desired improvement, for notifying the software users and developers about the desired changes, and for keeping a record of all changes made.





CONFIGURATION MANAGEMENT


- ORIGINATED IN INDUSTRY
- ESTABLISH REQUIREMENTS & DESIGN
- FOLLOW DEVELOPMENT GUIDELINES
- CONTROL CHANGES
- VERIFY REQUIREMENTS ARE ACHIEVED
- TRACK STATUS
- MAINTAIN CONFIGURATION RECORDS

Configuration management is a discipline for applying technical and administrative direction and surveillance to:

- * Identify and document the functional and physical characteristics of a configuration item.
- * Control changes to those characteristics (SPR system).
- * Control the development to make it follow established goals and procedures.
- * Record and report the processing status of the change.
- * Verify compliance with specification and other requirements (evaluation).
- * Maintain records of all development and changes from original design.



**SPR SYSTEM**



PROBLEMS WHEN NO CM

<ul style="list-style-type: none">• REQUIREMENTS NOT MET• NOT UP TO SPECS• NO QUALITY CONTROL• DIFFERENT MODULES ---- SAME NAME• APPROVED CHANGES NOT INCORPORATED• DIFFICULT TO EVALUATE• REPEATED MISTAKES• HIGH RATE OF DISCARDS	<ul style="list-style-type: none">• NO TRACKING OF<ul style="list-style-type: none">- CHANGES- PROGRESS• CONFLICTS IN<ul style="list-style-type: none">- DESIGN- CHANGES- DEVELOPMENT- SCHEDULES• DELAYS DEVELOPMENT• MISS DELIVERY SCHEDULES
--	--

BENEFITS OF CM ---ELIMINATE THESE PROBLEMS

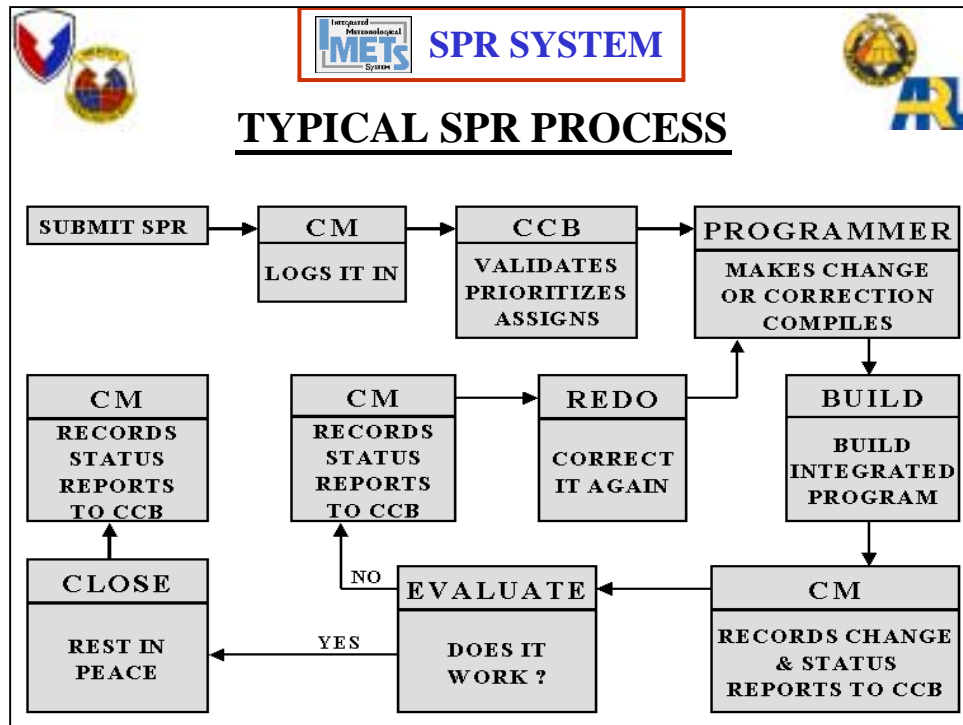
Without configuration management, the following problems are common:

- * Different software modules with the same name.
- * Software modules are not developed to specifications.
- * Approved changes are not incorporated.
- * Software modules do not meet customer requirements.
- * Software modules that have been developed cannot be defined.
- * High rate of quality problems. High rate of discards.
- * Minor changes cause major problems.
- * Difficulty maintaining delivery schedules
- * Difficulty maintaining of modifying delivered software.

The benefits of applying configuration management are:

- * Allows for correcting mistakes without causing major problems.
- * Avoids repeating mistakes.
- * Keeps track of all changes.
- * Allows for reproducing or duplicating.
- * Keeps developments in line with requirements.
- * Reduces scheduling conflicts.




- * Evaluates impact of changes.
- * Verifies and validates operability of product.



As a tool of configuration management, a software problem reporting system provides a means of making and controlling changes to the software being developed, avoids repeating mistakes, and allows for verification and validation of the operability of the developed software. Typically, SPRs are written by hand and given to the project configuration manager (CM) for processing. SPRs consist of discrepancy reports (DRs), configuration change requests (CCRs), and document (user's manuals) DRs and change requests. After collecting several SPRs, The CM will call a meeting of a configuration control board (CCB) which consists of team leaders, project leaders, and selected developmental programmers. The CCB members then discuss the SPRs, prioritize them, and assign them to programmers to write the needed corrections or changes. When corrected, the program that contains the correction or change is given to a team of testers who run and test the program to determine if the defect has indeed been corrected or that the requested change has been implemented to the satisfaction of the requester and the CCB. If all is well, the program is returned to normal use; otherwise, it begins a test-fix-test cycle until it works properly. As the SPRs are being processed, the CM must keep a record of their status by manually tracking their progress and recording the current stage of their correction on a status report form. He then reports this status at each CCB meeting.

The initial SPR system used in the development of software programs for the IMETS was just such a manual reporting and correction system. It was cumbersome and required the use of lots of forms and lots of time to process. Each discrepancy or change request required a different form. Each request was manually entered into a database and was printed out on two forms for reporting the correction status to the CCB. Also, there was no automatic method of keeping




track of the changes made to the IMETS programs. The current IMETS build was kept on a development computer where the changes were made. Each new build represented the latest changes, and this build was transferred to an evaluation computer where the evaluations were conducted. Builds were not made for every change, but only after several changes accumulated so that a new version of the integrated IMETS program was created at the build. This way, the requested changes as well as new ideas by the developers were incorporated in the new builds. Development could continue while the last build was being evaluated. The process was slow, but allowed for changes and kept track of them, but it was inefficient.



CHANGES TO SPR PROCESS (INITIAL)

- **RCS**
- **CONSOLIDATE FORMS**
- **AUTOMATION**

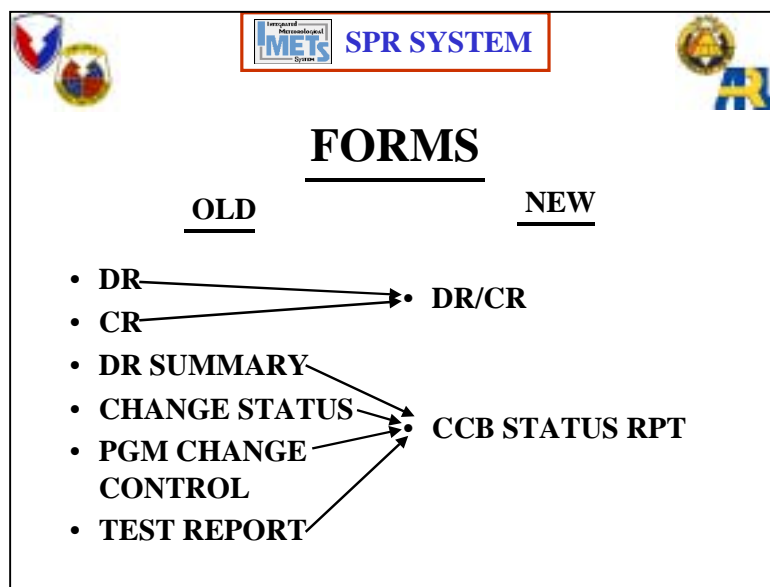
It had to be changed. I was the configuration manager of the system, so the first change I pursued was an automatic system for keeping track of the changes in each module of the IMETS software. The second was an effort to reduce the number of forms used for reporting discrepancies and tracking their status. The third was to convert the manual use of forms and configuration control board meetings into an automated system that would eliminate the forms altogether and reduce the requirement for configuration control board meetings.



REVISION CONTROL SYSTEM

- **GNU PROGRAM**
BY FREE SOFTWARE FOUNDATION, INC
- **AUTO RECORD CHANGES**
- **RETURN TO ANY VERSION**
- **WEEKLY SEMI-AUTO BUILD**

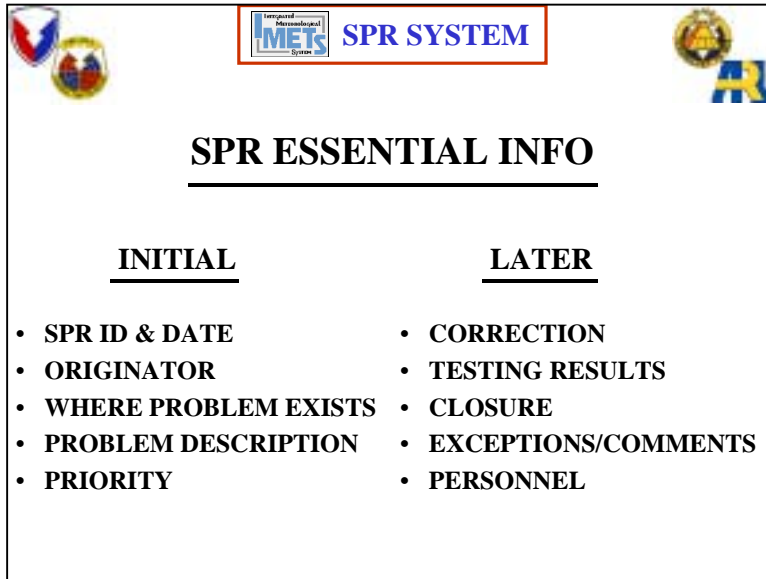
For the change, I formed a committee of four people to consider different methods of recording changes to the software. The method chosen was one called the Revision Control System or RCS, which is a program by the Free Software Foundation, Inc. I had two project system administrators at that time, and they emptied the development computer of all IMETS programs, installed the RCS, and the reinstalled each program, one at a time, under the control of the RCS. Thereafter each change to individual modules was recorded in the RCS, thus eliminating the dependency of relying on new builds to keep track of the changes. Programs requiring evaluation could then be built more rapidly and transferred to the evaluation computer without the involved formal process of creating new versions. As time went on, I had one of the system administrators write a script for making the build semi-automatic, and instituted the procedure of making a new build once per week. This allowed all new developments as well as corrected discrepancies and requested changes to be incorporated in the weekly build.



We were using separate forms for discrepancy reports and for configuration change requests, and we were using four different forms for tracking status and test results. For the second change, I consolidated the discrepancy report and configuration change request into one form, eliminated a discrepancy report summary, a change status request form, a program change control form, and a test report form, and created a new consolidated CCB status report form to summarize all changes for the CCB. This simplified the entry of information and the reporting of changes to the CCB. These new forms were used for quite awhile and worked very well within the manual system.

Two big problems remained, however, with this manual system. Evaluators and programmers were reluctant to enter discrepancies and change requests on a hand written form. Thus, not many SPRs were written. Worse yet was the fact that too many of the CCB members were constantly TDY. It was very difficult to convene a CCB meeting to review the SPRs that were written and to start corrective action on them. Delays were sometimes two to three months in length. A better method was needed.

My biggest change to the SPR process came when I designed and implemented an SPR system that resided on one of the UNIX developmental computers. This system was based on a set of directories which would automatically show the status of the progress in correcting the problem. Thus the need to keep status forms was eliminated. It was an ASCII system where a user, evaluator, or programmer could call up a blank SPR form from any UNIX developmental computer in the IMETS developmental network and enter the report data with a text editor. A report could also be generated on a PC and FTPed to the UNIX computer where the SPR system resided.



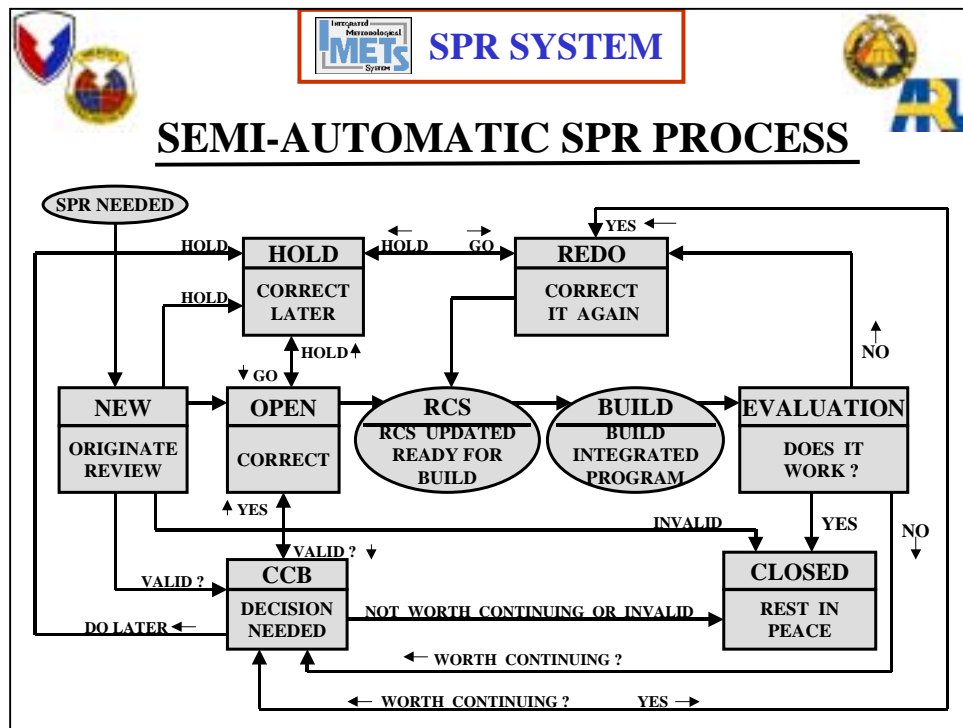
The image shows a screenshot of a terminal window titled "SPR SYSTEM". At the top, there are several logos, including the IMETS logo and a circular emblem. The main content is titled "SPR ESSENTIAL INFO" and is organized into two columns: "INITIAL" and "LATER".

<u>INITIAL</u>	<u>LATER</u>
• SPR ID & DATE	• CORRECTION
• ORIGINATOR	• TESTING RESULTS
• WHERE PROBLEM EXISTS	• CLOSURE
• PROBLEM DESCRIPTION	• EXCEPTIONS/COMMENTS
• PRIORITY	• PERSONNEL

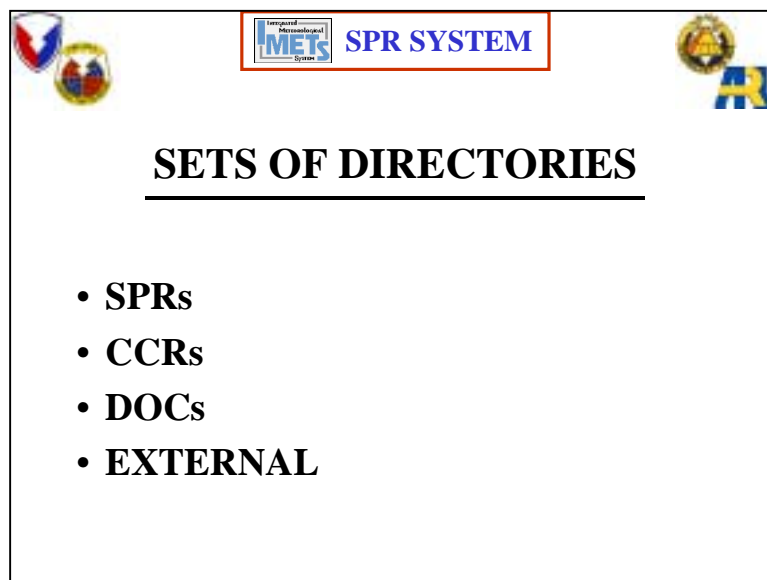
Blank SPR forms were kept in the "New" directory, each prenumbered so that no SPRs would have duplicate numbers, and so they could be tracked. The basic essential information required to be entered on the form was the date of the report, who was writing the report, in which program did the problem exist, a description of the problem, and the suggested priority that should be placed on the problem. Later, the corrective action and the testing results needed to be entered along with the names of the person doing the correction and testing. Lastly, applicable comments and the closing date needed to be entered, or, if not closed, an explanation needed to be entered.

Once written, an SPR would remain in the "New" directory where it could be reviewed by everyone. The responsible project leader would assign a final priority and a programmer, thus relieving the need to call a CCB meeting. He would update the SPR form with this information and then move the form to the "Open" directory where the programmer could review it and start corrective action. When corrected, the programmer would update the SPR form with his correction and the date of the correction and then move the SPR form to the "Testing" directory. A tester would evaluate the correction, enter the date of the test and the testing results on the SPR form, and move it to the "Closed" directory if the correction worked. The process was very similar to the one shown previously, but without the involvement of the formal CCB except on rare occasions. Other directories existed for exceptions to the normal flow. If a correction didn't work, the SPR would be placed in the "Redo" directory, and the correction process would start



over. If for some reason, the problem could not be corrected until later, the SPR would be placed in the "Hold" directory. Finally, if there was reason to question the validity of the SPR,



or it might tie up too many resources, or there might be a scheduling conflict in getting the correction completed, the SPR might require a CCB review and decision. In this case, the SPR would be placed in the CCB directory to await a formal CCB meeting that would decide its fate. These exceptions could be implemented by programmers, evaluators, or project leaders at any time it became necessary.



Within this process, there are four sets of directories to keep the types of SPRs separated and easier to manage. There are ones for discrepancies (SPRs), change requests (CRs), document changes (DOCS), and problems for which other agencies, or contractors, are responsible for correcting., One of the major contractors on the IMETS program is Logicon Advanced Technology, or LAT. Formerly its name was Research and Development Associates or RDA.

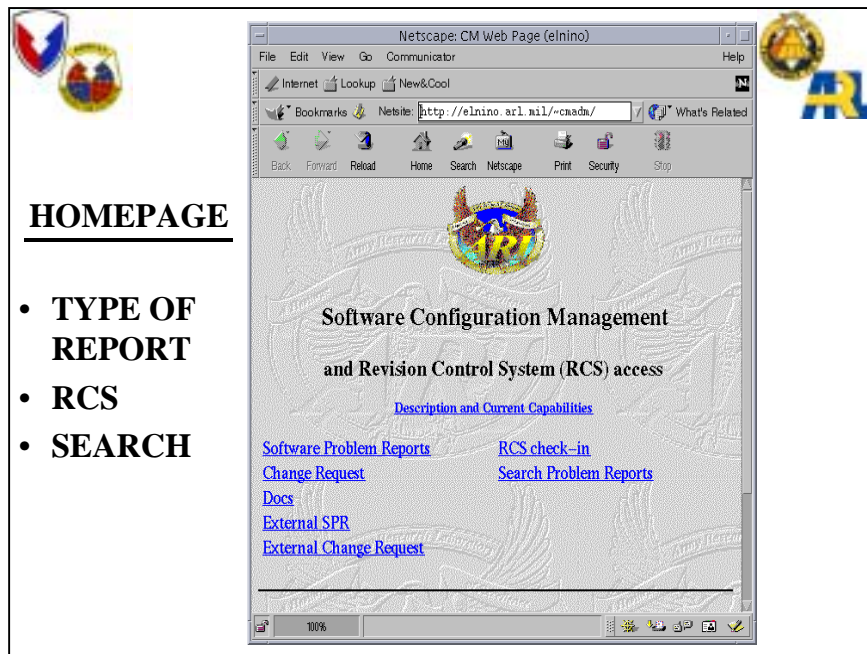


**SEMI-AUTOMATED SYSTEM
PROBLEMS**

- **TEXT IN ASCII**
- **MANUAL MOVES BETWEEN DIRECTORIES**
- **STAGNATION**
- **UNCONTROLLED CHANGES**
- **NO LINK TO RCS**

This “semi-automated” process was a tremendous improvement over the manual SPR system and increased the number of SPRs written and processed, most likely to near the number that would have been written had the users not been reluctant to hand write SPRs. This system was used for a long time, but three problems still existed. The ASCII form had to be filled out with an ASCII editor, either VI in UNIX, or a windows editor on a PC. An automated form was needed where the users only had to fill in blocks that automatically expanded to the size of the contents, and which would automatically move into the appropriate directory by clicking on the name of the directory desired. Also, project leaders were sometimes slow to look at the “NEW” directory, again because of TDYs or other workload and did not always review or assign a priority or programmer in a timely manner. Another method was needed to speed up the process. Finally, for a long time, the programmers had been making changes or improvements to their programs without any written requirement or SPR to work against. They would think of a new idea or innovation and would immediately start to code it. The RCS system would record the changes, but they would not be linked to a requirement or SPR. I, and the Project Director of IMETS, had always emphasized that no change should be made without a written requirement.

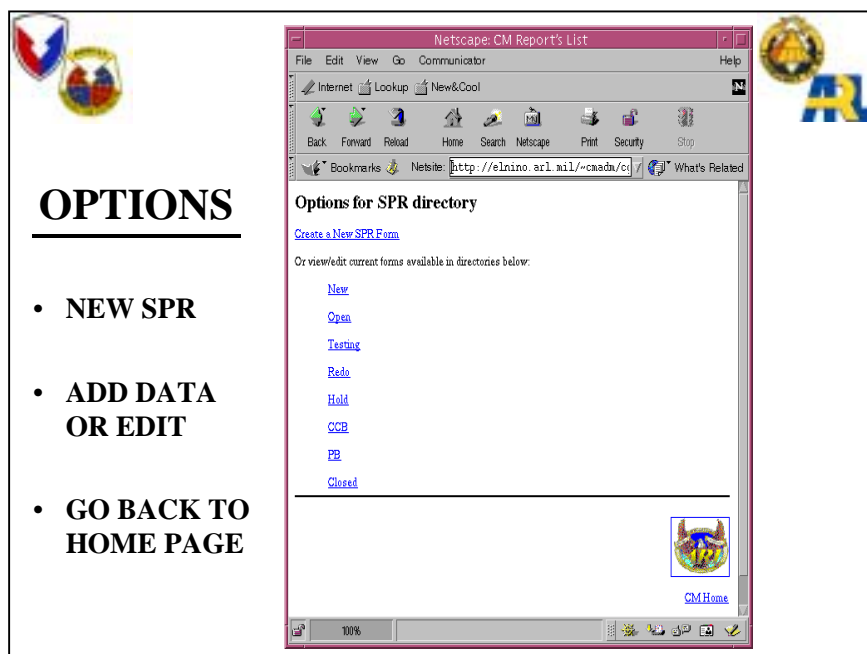
To improve the semi-automated system, I started a project to make an automated form that would address the shortcomings mentioned previously. Two choices were available, a database system such as Microsoft Access with an automated SPR entry form, or a web based system which could have more automated features than a database system. The CCB vote went for a web based or homepage system. Mario Torres did the programming of this system. I just provided guidance on the content. We call this new system the Configuration Management Homepage.



HOMEPAGE

- TYPE OF REPORT
- RCS
- SEARCH


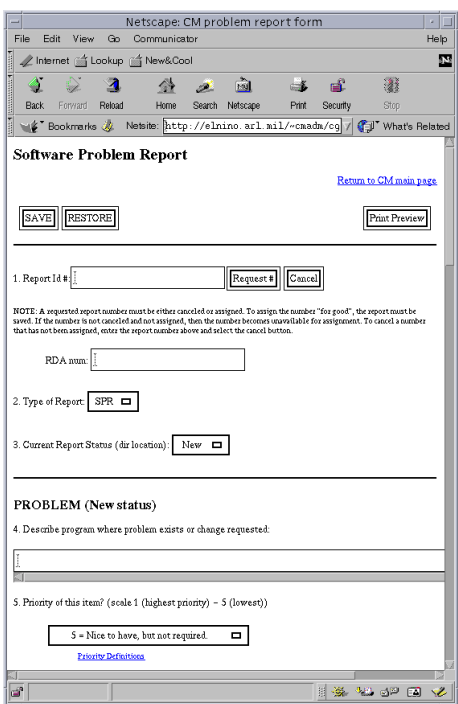

This is our starting page. This page and all the successive pages you will see were developed to work on Netscape. To create a new discrepancy report, we click on one of the five choices shown on the left side of the slide. These five choices represent the different sets of directories that we used in the semi-automatic system for SPRs, change requests, document problems, and problems to be corrected by an external agency. Upon clicking on one of these choices, a new page will appear.



OPTIONS

- NEW SPR
- ADD DATA OR EDIT
- GO BACK TO HOME PAGE

Let's say we clicked on "Software Problem Report". This is the page that we will see with a list of the directories, but with a new choice: "Create a New SPR Form". This is the same page we will see for the other sets of directories, but the top option line will have the name of the directory set that was chosen. To create a new SPR, we may click on the "Create a New SPR Form" option. To enter additional data in an already existing SPR, we click on the directory name where we know the SPR resides. To return to the homepage, we can click on the icon in the bottom right corner of the page. There is an extra directory here that you did not see in the semi-automatic system. It is the "PB" directory. This is for SPRs written against our Platinum Build which is the final build of the current version of IMETS software being delivered to Force XXI at Ft Hood, Texas for use in the Army Battle Command System.

DATA ENTRIES


1 - 5

- **REPORT NO.**
- **TYPE OF REPORT**
- **CURRENT STATUS**
- **WHERE DOES PROBLEM EXIST?**
- **PRIORITY**

This data entry form is the form currently being used for all SPRs on the IMETS developmental system. The specific data entries are shown on this slide and next four slides. They are an expansion on the essential data entries I mentioned earlier. These five slides are successive portions of the form that we see by scrolling down through it. The first entry is the SPR ID number. Just click on "Request #" to have a number automatically assigned to the SPR. The "RDA Num" is for an external agency. It is the number that LAT or other agency assigns to the SPR when it gets it and starts processing it. If this is a new SPR, line 2 and 3 are filled in automatically. If you are editing an existing SPR, you may click on these buttons to choose a new entries in the pull-down menus. In line 4, name or describe the program where the problem exists. In line 5, click on the button and choose the applicable priority from the pull down menu.

The priorities we use are extracted from MIL-STD 498. This MIL-STD has been replaced by an industrial standard, but the priorities still apply. Priority 1 is a problem that can cause a mission to fail with no possible recovery. A priority 2 problem has a very serious adverse effect on the

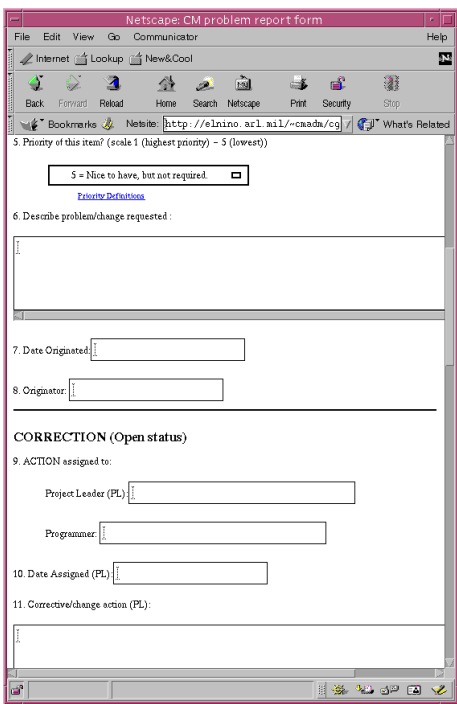
mission, and the program has no chance of being replaced by a work-around. A priority 3 problem also has a very serious adverse effect on the mission, but the program with the problem



PRIORITY

- 1. DOESN'T WORK OR CRASHES - CASTOSTROPHIC
- 2. ADVERSELY AFFECTED - NO WORK AROUND
- 3. ADVERSELY AFFECTED - HAS WORK AROUND
- 4. INCONVIENCE TO OPERATOR
- 5. NICE TO HAVE, BUT NOT NECESSARY

has a work-around, or an alternative method of accomplishing the mission. Priority 4 problems Are those which should be corrected or changed to relieve annoyance to the operator or to operate in accordance with standard accepted procedures and produce standard results. A priority 5 problem is anything else that has no effect on the mission, but would be nice to have.



DATA ENTRIES
6 - 10

- **DESCRIBE PROBLEM**
- **DATE ORIGINATED**
- **ORIGINATOR**
- **ASSIGNMENTS**
 - **PROJ LEADER**
 - **PROGRAMMER**
- **DATE ASSIGNED**



Line 6 is where the originator describes the problem. It is important to be clear and concise about the problem, even getting help from the programmer to write it up accurately. This block, like all the others in which one enters text, automatically expands to accept all the text that is entered. The originator then enters the date of the report and his name.

The image shows a Netscape browser window displaying a web form titled "Netscape: CM problem report form". The browser's address bar shows the URL "http://elino.srl.nsl/cnadm/cg". The form contains several sections:

- DATA ENTRIES**
22 - 23
- A list of instructions:
 - EMAIL ADDRESSEE
 - EMAIL MESSAGE
- THEN**
- SAVE OR RESTORE**

The form also includes a "Date:" field, a "To:" field for email recipients, a "From:" field, and a large text area for the message. A note states: "Note: All names (to and from) not containing the '@' directive, will have the '@bmsou-elny2.srl.nsl' address appended. Be sure to be explicit if you want it to be otherwise. The '@srl.nsl' will also be replaced by the same to expedite delivery." At the bottom of the form are "SAVE" and "RESTORE" buttons. The browser window also shows a "CM Home" link and a small logo in the bottom right corner.

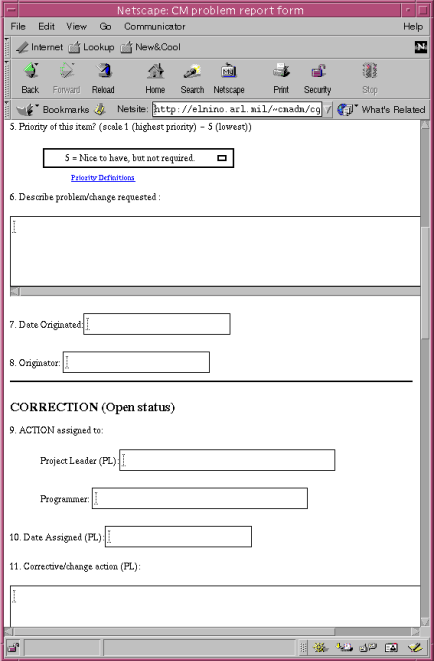
At this point, the originator goes to the top of the page, makes sure that "New" is entered in line 3, the "Current Report Status" block, and then goes to the bottom of the page to prepare an email message to the project leader for the program with the problem and the programmer. He enters the addressees, his name, and writes a message about the problem. He then clicks on the "Save" button. The SPR is automatically routed and saved in the "New" directory, and the email is automatically sent to the addressees. No more VI editor or UNIX commands to move the SPR into the desired directory, and no more writing of messages in a separate email program.



DATA ENTRIES

6 - 10

- DESCRIBE PROBLEM
- DATE ORIGINATED
- ORIGINATOR
- ASSIGNMENTS
 - PROJ LEADER
 - PROGRAMMER
- DATE ASSIGNED



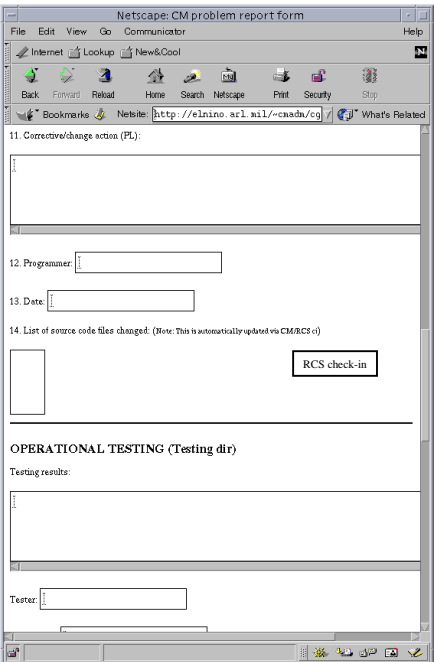
From this point on, we will be editing the SPR. After the originator enters the new SPR in the "New" directory and sends out the email notification, the project leader for the program with the problem will look at the SPR, enter his name, assign a programmer, enter the date he was assigned, and review the priority. He may leave it the same, or change it to a higher or lower priority. He then prepares his own email notification, primarily to the programmer with copies to everyone. He goes back to line 3, selects the "Open" directory, and then clicks on the "Save" button. The additions are saved and the SPR is automatically moved to the "Open" directory.


DATA ENTRIES

11 - 16

- CORRECTIVE ACTION
- PROGRAMMER
- CORRECTION DATE
- SOURCE CODE FILES
- TESTING RESULTS
- TESTER




The programmer is the next person in line to look at the SPR. He makes his correction and then enters what he did in the "Correction" block. As did others before him, he enters his name, the date of the correction, and prepares an email to notify the next person in line for processing the SPR. He then goes back to line 3, selects the "Testing" directory from the pull down menu, and clicks on "Save". The SPR with its new information is saved and moved to the "Testing" directory. The programmer is not yet finished, though. He must click on the block marked "RCS check-in". This is a link to the homepage button with the same name and brings up a form where he lists source code files that were changed. Saving that form automatically enters the source code file name in the block shown here under line 14..

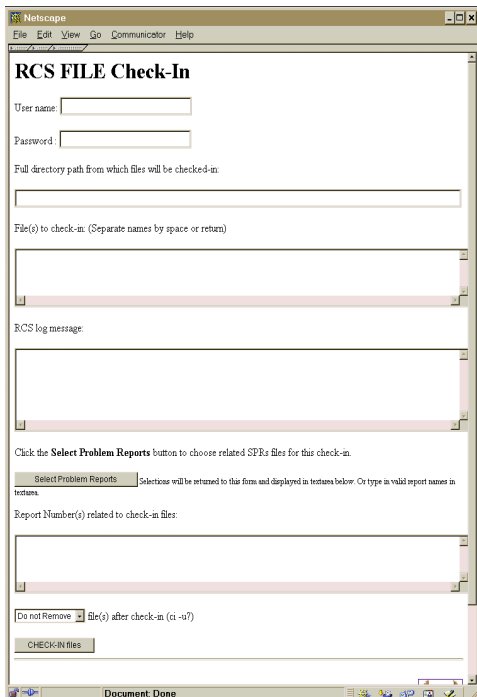


RCS CHECK-IN

- **USER NAME
(PROGRAMMER)**
- **PASSWORD**
- **DIRECTORY PATH**
- **FILES TO CHECK IN**
- **RCS LOG MESSAGE**
- **CHOOSE RELATED
SPRs**

**THEN
CHECK IT IN**


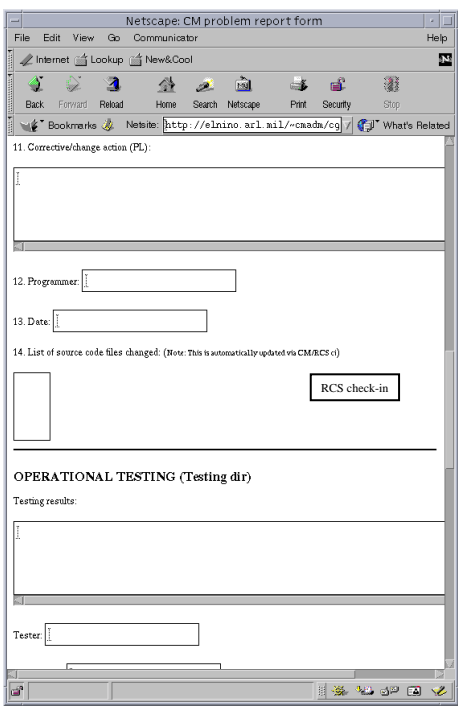





The screenshot shows a Netscape browser window titled "RCS FILE Check-In". The form contains the following fields and controls:

- User name:
- Password:
- Full directory path from which files will be checked-in:
- File(s) to check-in (Separate names by space or return):
- RCS log message:
- Click the **Select Problem Reports** button to choose related SPRs files for this check-in. Below this is a button labeled "Select Problem Reports" and a note: "Selections will be returned to this form and displayed in listarea below. Or type in valid report names in listarea."
- Report Number(s) related to check-in files:
- Do not Remove file(s) after check-in (i-i-u?)
-


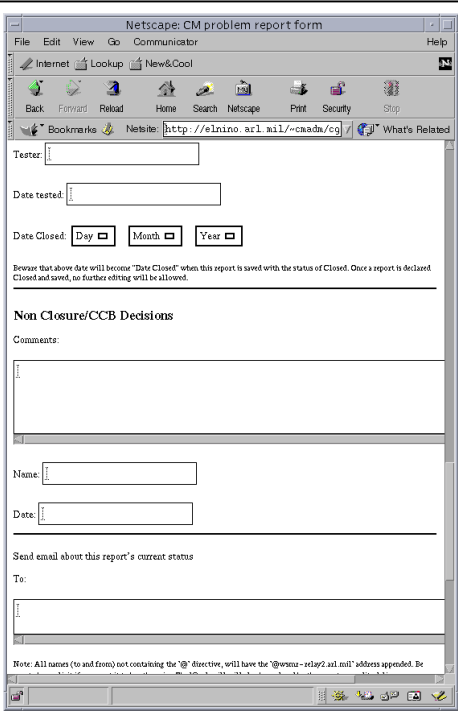

Here is the "RCS check-in" form. In it, the programmer enters his name, his password for the workstation on which the RCS system resides, the path to the directory where the changed source code files are located, the names of the changed source code files, any message he desires to enter, and the SPR number or numbers related to the check-in source code files. He may then select to either remove or not remove these files after check-in, but the prevailing option is to not remove. Finally, he clicks on the "CHECK-IN files" button. The homepage system checks to assure that the referenced SPR exists, is current, and not closed. It will then update the RCS system and the related SPR forms. In addition to working on SPRs submitted by someone else, this feature forces each programmer to write up his own proposed changes and get them approved by his project leader before he starts to code them. It links each RCS entry with an SPR and satisfies the CM requirement to have a written requirement for every program change.

DATA ENTRIES 11 - 16

- CORRECTIVE ACTION
- PROGRAMMER
- CORRECTION DATE
- SOURCE CODE FILES
- TESTING RESULTS
- TESTER

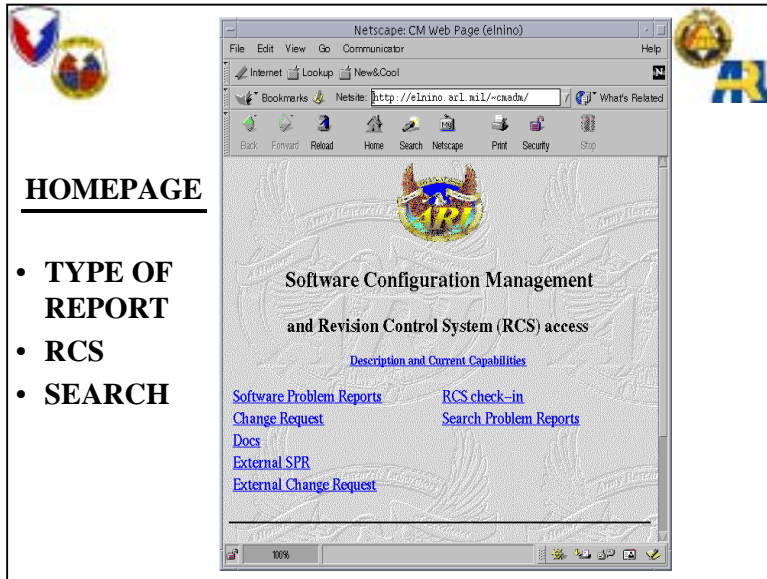
At last, the tester gets to make his inputs to the form. He tests the changes, and if they work satisfactorily, he goes to the form, enters the test results, his name, and the date of the test.

DATA ENTRIES 17 - 22

- DATE TESTED
- DATE CLOSED
- COMMENTS
- NAME
- DATE
- EMAIL MESSAGE

He also uses the pull down menus to enter the closing date. With this done, he goes back to line 3, opens the pull down menu there, and selects "Closed". This saves the new information and moves the SPR to the "Closed" directory. The SPR is finished -- UNLESS THERE ARE EXCEPTIONS. If the correction didn't work, the tester writes an explanation in the "Comments" block, enters his name and date, writes an email to applicable persons, and then goes to line 3 and selects "Redo" from the pull down menu. Clicking on "Save" saves the latest information and moves the SPR to the "Redo" directory. Other exceptions may be to move the SPR to the CCB directory or the "Hold" directory to get a CCB decision or to wait till another time to take action.



One last feature to mention is the ability to search for an SPR if one cannot remember its ID number. Go to the homepage and click on "Search Problem Reports".

