# CONFERENCE ON SOFTWARE MAINTENANCE-1985

F-16

# CONFERENCE ON SOFTWARE MAINTENANCE-1985

**Sheraton Inn Washington-Northwest November 11-13, 1985**

ISBN 0-8186-0648-7
IEEE CATALOG NUMBER 85CH2219-4
LIBRARY OF CONGRESS NUMBER 85-62325
IEEE COMPUTER SOCIETY ORDER NUMBER 648


IEEE COMPUTER SOCIETY PRESS

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.
IEEE

SPONSORS:

Data Processing Management Association (DPMA)

IEEE Computer Society - TC on Software Engineering

National Bureau of Standards (NBS)

In Cooperation With:

ACM/SIGSOFT - Special Interest Group on Software Engineering

Association for Women in Computing (AWC)

SMA - Software Maintenance Association

Published by IEEE Computer Society Press
1730 Massachusetts Avenue, N.W.
Washington, D.C. 20036-1903

COVER DESIGNED BY JACK I. BALLESTERO



**ISBN 0-8186-0648-7 (paper)**
**ISBN 0-8186-4648-9 (microfiche)**
**ISBN 0-8186-8648-0 (case)**
**IEEE Catalog Number 85CH2219-4**
**Library of Congress Number 85-62325**
**IEEE Computer Society Order Number 648**

Order from: IEEE Computer Society
Post Office Box 80452
Worldway Postal Center
Los Angeles, CA 90080

IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854


THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Nicholas Zvegintzov
General Chair

## Message from the General Chair

As General Chair of the Conference on Software Maintenance 1985 (CSM-85), I find myself sitting in an honorable seat. CSM-85 is the second open research conference in the topic of software maintenance, following the pioneer steps of the Software Maintenance Workshop of December 1983, chaired by Norman F. Schneidewind.

CSM-85 defines software maintenance as the enhancement, restructuring, and correction of software in production use. Everything in our society, from national defense to children's toys, runs on software. The job of maintaining and enhancing that software is an important trust. It is also a daunting intellectual problem, and an organizational challenge for an era of mature software.

The logo of CSM-85 shows the world held in a human hand. The ancient Greeks thought of the world as an immense weight on the shoulders of a giant, Atlas. In 1985 we see the world as small and fragile. It is in our hands.

The logo was designed by Washington, DC area artist Gabriella D'Andrey. She told me: "I looked at the emblems of other computer organizations and they were mechanical. I feel that the ideas in software maintenance are very emotional and I wanted the logo to match that feeling."

I think Ms. D'Andrey's observation gives an idea of the excitement and dedication with which everyone approached CSM-85. I cannot express how much has gone into this conference on the part of 6 members of the Conference Committee, 17 members of the Program Committee, 69 referees, one keynoter, 53 paper authors, 28 panelists, 11 vendors, the staff and members of the sponsoring and cooperating organizations, and the attendees.

Thank you all. It was worth it.

Robert S. Arnold
Program Co-Chair

Roger J. Martin
Program Co-Chair

## Message from the Program Co-Chairs

The Conference on Software Maintenance 1985 (CSM-85) has created a forum in which software maintenance researchers and practitioners may talk and share, and in which the latest progress on software maintenance is made readily available to the computing community. We hope that CSM-85 has achieved its goal of advancing the state of the software maintenance art and of its practice both for the CSM-85 attendee and for the reader of this volume.

Creating CSM-85 has been like constructing a large office building. The CSM-85 Conference Committee, consisting of Robert Arnold, Roger Martin, Wilma Osborne, Donald Parker, Norman Schneidewind, and Nicholas Zvegintzov, conceived the conference in early 1984. The IEEE Computer Society and the Data Processing Management Association, as financial sponsors, supplied the "venture capital." The IEEE Computer Society suggested a conference site. The Conference Committee created a conference architecture and a construction plan. The authors of papers and proposers of panel sessions supplied the "raw materials." The Program Committee and the referees acted as quality control. The Conference Committee, the Program Committee, and many others helped fill the conference with worthwhile sessions and oversee their success. Employers of the CSM-85 volunteer workers provided much appreciated support in allowing their employees to help. The National Bureau of Standards was particularly generous in its support. To all parties involved, thank you.

We thank all people who submitted papers and panel ideas to CSM-85. Recognizing these papers' significance, we tried to create a fair evaluation process. The papers were refereed without the referees knowing who authored them, which reduced potential bias in referees' reviews. The papers were selected by the Program Committee based on the referees' ratings and the relevance of the paper to software maintenance. We originally received 52 papers for consideration, of which 27 were accepted and are published here.

We thank Laszlo A. Belady for being our keynote speaker and Professor Ben Shneiderman for sharing his and his colleagues' current research in their invited paper.

We look forward to the next CSM and further significant progress in making the need for "maintenance" all but disappear. The victory of the CSMs will be when the need for software maintenance will be no more. Until then, we hope the reader finds in this volume valuable ideas for placing software maintenance under ever firmer control.

## Conference Committee

**General Chair**
Nicholas Zvegintzov
*Editor, Software Maintenance News*

**Immediate Past General Chair**
Norman F. Schneidewind
*Naval Postgraduate School*

**Program Co-Chairs**
Robert S. Arnold
*MITRE Corp.*

Roger J. Martin
*National Bureau of Standards*

**Caucus Chair**
Wilma M. Osborne
*National Bureau of Standards*

**Publicity Chair**
Donald A. Parker
*NASA Goddard Space Flight Ctr.*

## Program Committee

David Bellin
*William Paterson College*
Bruce Blum
*Applied Physics Laboratory*
Mel A. Colter
*University of Colorado*
John C. Dodd
*Computer Sciences Corp.*
William C. Hetzel
*Information Mgmt. Institute*
David Marca
*Digital Equipment Corp.*
Tobey B. Marzouk
*Rothblatt & Marzouk*
James A. McCall
*Science Applications International Inc.*
Steven W. Oxman
*OXKO Corp.*
David E. Peercy
*The BDM Corp.*
Gary L. Richardson
*Texaco Inc.*
Gary C. Sackett
*Hughes Aircraft Co.*
Cees J. Schrama
*IBM Netherlands*
Don Shafer
*Los Alamos National Laboratory*
Elliot Soloway
*Yale University*
Barbara J. Taute
*BJT Enterprises*
Dolores R. Wallace
*National Bureau of Standards*

# List of Referees

Robert S. Arnold
Nathaniel Ballou
John C. Barnwell
David Bellin
Eugene Bellin
Andrew R. Bennett
Andre Blokdijk
Bruce Blum
Linda Brice
Ruven Brooks
Charles W. Butler
Ugo Buy
Robert Chamberlain
Ned Chapin
Mel A. Colter
Hope A. Cope
Bill Curtis
Taz Daughtrey
Paul P. Davis
Nicholas Easton
David R. Falconer
Roger U. Fujii
David Gelperin
Kenneth A. Gilbert
Charles E. Goorevich
Elwood W. Greene, Jr.
Reginald C. Grier
P. E. Hartmann
William C. Hetzel
David C. Hubbard
Dwayne L. Knirk
D. Richard Kuhn
Stan Letovsky
Dennis Lindenberg
Frank Lynch
David Marca
Roger J. Martin
James A. McCall
Fred R. McFadden
Patricia H. Morris
Christie Nichelsen
Wilma M. Osborne
Steven W. Oxman
Donald A. Parker
David E. Peercy
William E. Perry
Patricia A. Pierce
Robert P. Rich
Gary L. Richardson
H. Dieter Rombach
Gary C. Sackett
Cees J. Schrama
David J. Schultz
Don Shafer
Ben Schneiderman
Elliot Soloway
I. Strausz
Barbara J. Taute
Paul C. Tinnirello
H. J. Van Aerle
H. Van Goolen
R. L. Van Tilburg
R. Wachter
Ann K. Wagner
Dolores Wallace
Kathy Brittain White
Nancy Wogrin
S. S. Yau
Nicholas Zvegintzov

# Table of Contents

Message from the General Chair ..... iii
Message from the Program Co-Chairs ..... v
Conference and Program Committees ..... vii
List of Referees ..... ix

**TRACK A: Techniques, Methods, and Tools**

**Session A1**
(Session Chair: C.J. Schrama)

An Automated System for Controlling Operational Program and JCL Changes ..... 2
*D.L. Wiggins*
Automated Configuration Management on a DOD Satellite Ground System ..... 6
*K.B. Christian and S.H. Zucker*
A Database Approach to Configuration Management for Large Projects ..... 15
*B. Taylor*

**Session A2: Panel**
Putting Software Quality Assurance into Software Maintenance
(Panel Chair: R.S. Arnold)

**Session A3**
(Session Chair: D.A. Parker)

Structured Program Analysis Applied to Software Maintenance ..... 28
*J.D. Wedo*
The Automatic Restructuring of Cobol ..... 35
*E. Bush*
Maintenance and Porting of Software by Design Recovery ..... 42
*G. Arango, I. Baxter, P. Freeman, and C. Pidgeon*
Controlling the Evolution of Large Scale Software Systems ..... 50
*N.H. Minsky*

**TRACK B: Management and Education**

**Session B1: Panel**
Negotiating Software Maintenance Contracts
(Panel Chair: T.B. Marzouk)

**Session B2**
(Session Chair: D. Bellin)

Software Maintenance Management ..... 62
*M.A. Branch, M.C. Jackson, M.C. Laviolette, and E. Frankel*
The Validation, Verification, and Testing of Software: An Enhancement to Software Maintainability ..... 69
*D.R. Wallace*
A Survey of Software Quality Assurance in the Department of Defense During Life-Cycle Software Support ..... 79
*R. Day and T. McVey*

**Session B3: Panel**
The Need to Teach Software Maintenance
(Panel Chair: G. Parikh)

**Panel**
Is the Waterfall Model Sinking Software Maintenance?
(Panel Chair: P. Bassett)

**TRACK C: Metrics**

**Session C4**
(Session Chair: D.E. Peercy)

An Analysis of Software Changes During Maintenance and Enhancement ..... 92
*D.A. Gustafson, A. Melton, and C.S. Hsieh*
An Empirical Approach to the Study of Errors in Large Software Under Maintenance ..... 96
*C.K.S.C.H. Yuen*
Controlling the Maintainability of PL/I and PL/I-Like Software ..... 106
*G.P. Hill*

**Session C5: Panel**
Metrics for Software Maintenance
(Panel Chair: J.A. McCall)

**Session C6**
(Session Chair: B. Taute)

Metrics for Optimal Maintenance Management ..... 114
*H. Schaefer*
A Framework for Risk Assessment of Software Supportability ..... 120
*D.E. Peercy*
Predicting Software Customer Support ..... 128
*J. Chapin and G. Faidell*

**TRACK D: Research and Case Studies**

**Session D4**
(Session Chair: D.R. Wallace)

Display Strategies for Program Browsing ..... 136
*B. Shneiderman, P. Shafer, R. Simon, and L. Weldon*
Strategies for Documenting Delocalized Plans ..... 144
*S. Letovsky and E. Soloway*
Impact of Software Structure on Maintenance ..... 152
*H.D. Rombach*

**Session D5**
(Session Chair: S.W. Oxman)

Upgrading Aging Software Systems Using Modern Software Engineering Practices: IBM-FSD's Conversion of FAA's National Airspace System (NAS) En Route Stage A Software from 9020s to S/370 Processors ..... 162
*B. Britcher and J. Craig*
Designing Data Base Systems for Maintainability: Case Study: IDMS and IMS (Lessions Learned) ..... 171
*E.D. Zeisler*
A History of Software Maintenance for a Complex U.S. Army Battlefield Automated System ..... 181
*R. Day*

**Session D6: Panel**
Targets for Effective Software Maintenance Research
(Panel Chair: D. Shafer)

**Panel**
The Psychology of Program Documentation
(Panel Chair: E. Soloway)

**TRACK E: Tools for Understanding Software**

**Session E7**
(Session Chair: M.A. Colter)

Help! I Have to Update an Undocumented Program ..... 194
*S.D. Fay and D.G. Holmes*
Simple Tools to Automate Documentation ..... 203
*D.R. Kuhn and C.G. Hollis*
Using a Relational Query Language as a Software Maintenance Tool ..... 211
*T.G. Glagowski*

**TRACK F: Diagnosing Software Problems**

**Session F7**
(Session Chair: D. Marca)

Software Maintenance Criteria for Small Microprocessor-Based Systems ..... 222
*P.P. Howley, Jr., and G.W. Reimer*
Upgradeability: A Measurement of Quality ..... 227
*A.W. Brehl*
Systematic Problem Solving: The Link to Maintenance Solutions ..... 231
*J.D. Wedo*

**Panel**
Software from Here to Eternity
(Panel Chair: N. Zvegintzov)

**Author Index** ..... 239

# Track A: Techniques, Methods, & Tools

## Session A1

**Session Chair**
C.J. Schrama
*IBM Netherlands*

# AN AUTOMATED SYSTEM FOR CONTROLLING OPERATIONAL PROGRAM AND JCL CHANGES

David L. Wiggins

Texaco, Inc. CISD - P.O. Box 37327 - Houston, TX 77237

## Abstract

The need to automate control of operational program and JCL changes arose due to two factors. First, it was impractical to establish a control group to handle the change activity for the DP shop which is dispersed into two geographical areas. Second, the production program controls at the application level were inadequate and nonstandard as an internal EDP audit revealed. These factors caused a project to be established that sought to correct the deficiencies of existing control procedures and standardize the change activity between the application areas.

This paper will discuss specific strategies used, products developed in-house, products purchased from vendors, and special interfaces between in-house and purchased products.

## Environment

This control system is currently operating in an IBM JES3/(MVS-MVS/XA) environment. The system is written in PL/I and Assembler languages and makes use of the IBM Interactive System Productivity Facility (ISPF) - Dialog Management Services. It is designed to interface with the PANVALET program product from Pansophic, Inc. as well as other software products which will be discussed later.

## Background

An internal audit of DP activities severely criticized the procedures used by application support personnel to modify program code and JCL. The auditors, in examining Job Control Language (JCL), found that programs were being executed from non-standard execute libraries such as load libraries cataloged under private TSOids. Another critical issue identified was that the source code for the execute module could not always be found and when present was not necessarily the version executed in the production environment.

## Library Management Project

Based on the audit report and the desire to standardize change procedures within operational programming (maintenance) application areas, a project called the Library Management System (or LIBMAN, for short) was initiated. Development of the control system was complex to a degree that it was implemented in two phases. The first phase addressed controlling the program code changes of operational execute (load) modules and JCL changes to PROCLIB. The second phase addressed changes to the operational JOB decks, notifying operations automatically when changes are made to JOB decks, generating special scheduling instructions to be transmitted to operations, and supports the release concept of installed application system changes.

## Phase I

### Controlling Program Code

Company standards required that all production source code reside on PANVALET libraries. Prior to the implementation of LIBMAN, all production source code resided on a single PANVALET library. However, in the audit report it was noted that there was inadequate security on individual members of the PANVALET library. LIBMAN increased access security by splitting code into multiple PANVALET libraries that were set-up by functional areas, using existing member name standards. Next, a standard set of dataset names was established for the execute libraries with a one-to-one correspondence between the PANVALET and execute libraries. Each functional area within LIBMAN has a designated person who serves as a system integrity manager. This person is allowed to bypass the LIBMAN software for updating execute libraries

in special circumstances. An example would be copying vendor-supplied load modules when source code is not provided.

The ACF2 security software product allows controlled updating of the execute libraries via its program-pathing facility. This facility allows write-access to the execute library only if the updating is done by a program executed from a specific library. The updating is done in a batch environment with a job generated using the Dialog Management Services of ISPF. The batch job executes a program that causes a member to be extracted from the PANVALET library, compiled and link-edited into the execute library. Other functions of this program are enforcement of standards for member names, validation of linkage editor control statement formats, and writing the compiler and linkage editor messages to a disk file. After compilation, the modified production code and the previous production code are compared using the IBM product SUPERC. The differences are also recorded to the disk file. This disk file is later processed for creating tapes for use by the company document processing department to create microfiche for audit purposes. Capabilities exist to compile multiple PANVALET members using the same compiler or mixing compilers, to conditionally link-edit execute modules, to link-edit into IMS-DC/DB execute libraries, and to analyze COBOL code using the software product SCAN/370 (which checks for unencountered procedures, builds hierchary charts, etc.).

Thus, a typical change process for program modification would be as follows:

1. Using LIBMAN, retrieve current source from the production center and place it on the work library.
2. Edit the code, making the required changes.
3. Compile, link-edit, and test the program using test libraries and data.
4. Assuming successful testing, store changed code back on the work library.
5. Again using LIBMAN, fill in the appropriate responses, indicating that the production code is to be updated using the work library source, compiled, and linked to the production execute library.

## Controlling PROCLIB JCL

It is a company standard that all job decks should execute catalog JCL procedures (PROCS). However, in the audit report it was noted that there was inadequate security on individual members of the PROCLIB library. For performance reasons, LIBMAN uses a different approach than that used to increase the security on program code. LIBMAN, instead of creating multiple PROCLIBs, set up an ACF2 resource database that contains the first four characters of the member name. The company has been using a standard scheme for these characters for several years. The ACF2 security software product then allows controlled updating of the PROCLIB library via its program-pathing facility. The updating is done in a batch environment by a JOB generated using the Dialog Management Facilities of ISPF. The batch job executes a program that checks the resource rules to see whether the programmer who submitted the job is authorized to update that member. In addition, any STEPLIB JCL statement found is checked to ensure that an approved execute library is specified. Any changes made to the JCL are recorded into a disk file which is used as an audit trail of requested changes. A later enhancement to this phase involves the use of a product from Diversified Software Systems Inc. called JOBSCAN (formerly PRESCAN). This product scans the updated PROC to check the syntax of the JCL. It also has a facility for loading a in-house written routine which is used to check standards for disk dataset allocations, EXPDT coding for tape datasets , and MSVGP values according to each JES Global site.

## Phase II

### Controlling JOB Deck Changes

Phase II of LIBMAN seeks to control job deck changes. However, in the case of job decks, the update will not be performed automatically since operations personnel is delegated final control over the production JCL decks. In order to notify operations that a job deck change is ready for implementation, an inter-office memo is generated and sent to a printer located in the operations area. Phase II also generates similar memos for special scheduling requests when required.

### Supporting the release concept of system change

Phase II of LIBMAN also was designed to handle a release concept of program code changes. The release concept, as defined for our purposes, involves the packaging of requests for modification to installed applications. To support the concept, an additional level of PANVALET and execute libraries was created. Using this strategy, it is possible for support personnel to back-out a release quickly and easily without having to restore

the previous source code and recompiling that code. Now the previous production module can be executed by simply changing execute library references. After a release is successfully installed and is now to be the production version, the system integrity manager has the capability to move all source and execute modules from the release libraries into the production libraries with a single request.

## Management Reporting

Phase II of LIBMAN also was designed to provide daily change activity reports for system integrity managers and supervisors. Reports are created listing all programs that have been compiled and linked, all JOB deck changes that have been requested, and all JOB scheduling changes that have been requested.

## Additional Enhancements

The following enhancements have been made to the LIBMAN system since Phase II was implemented:

- When changes are made to a productive job deck, the resultant job deck is scanned using the JOBSCAN product previously mentioned.
- Test load modules can be generated for the work PANVALET libraries.
- Management reports from different production centers have been combined into a single report which resides on a disk file.
- After CICS and COM-PLETE were installed, application program libraries were placed under LIBMAN control.
- ADF source code updating is now controlled by LIBMAN.
- Interpretative code (SAS, RAMIS, etc.) is placed into a protected PDS by LIBMAN after retrieval from PANVALET.

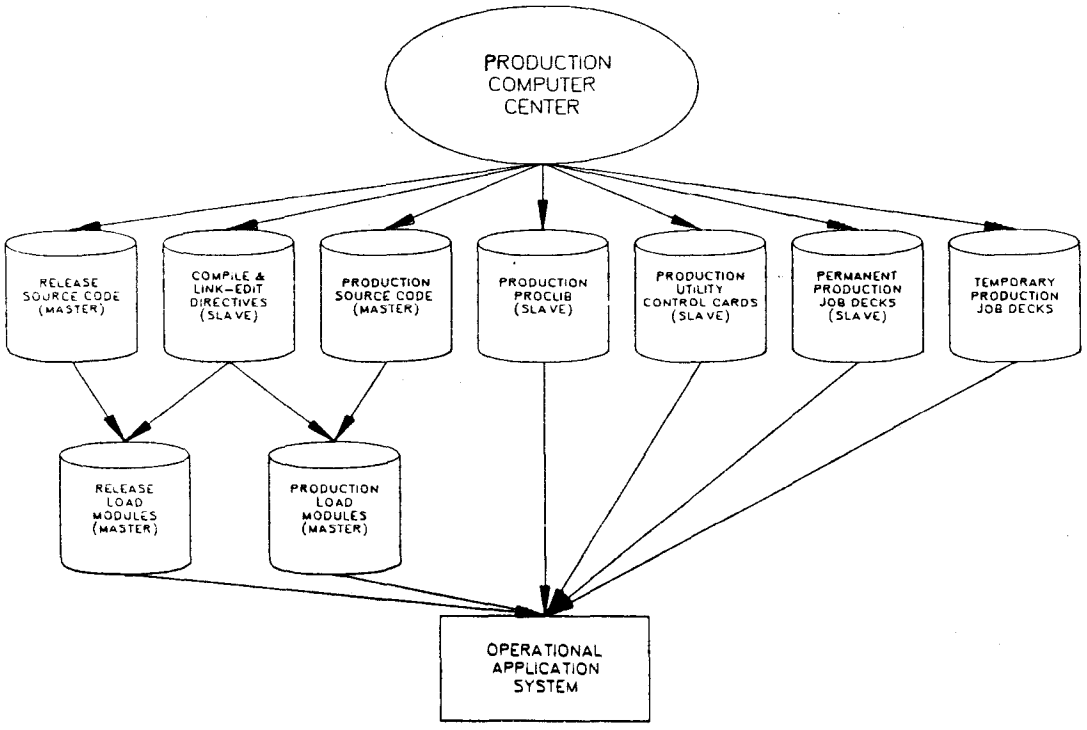



Figure 1. LIBMAN Logical View – PRODUCTION

## Summary

Figures 1 and 2 show the logical views of datasets controlled and used by the LIBMAN software in the Production and Development Computer Centers. When the Production and Developmental systems are physically the same hardware, some of the master-slave relationships no longer are required and the master dataset is the only one that exists.

The LIBMAN software was able to address and correct most of the deficiencies reported by the internal EDP audit department. Use of LIBMAN has standarized the procedures for making program code, job deck JCL, and PROCLIB JCL changes. This standarization has resulted in an additional benefit of eliminating the learning curve that previously existed when each maintenance area had their own method of controlling program updates. This has facilitated the rotation of personnel within the operational programming areas. Its use has also extended into the transition for turnover of applications from developmental project teams to operational programming teams.

## Vendor Software Products

Further information about certain software products mentioned may be obtained by writing:

- Internatinal Business Machines Corp. (IBM) concerning ISPF and SUPERC,
- Diversified Software Systems Inc. (DSSI) concerning JOBSCAN,
- Group Operations, Inc. concerning SCAN/370,
- Pansophic, Inc. concerning PANVALET, and
- Cambridge Systems Group concerning ACF2 which was developed by SKK, Inc. of Rosemont, Il.

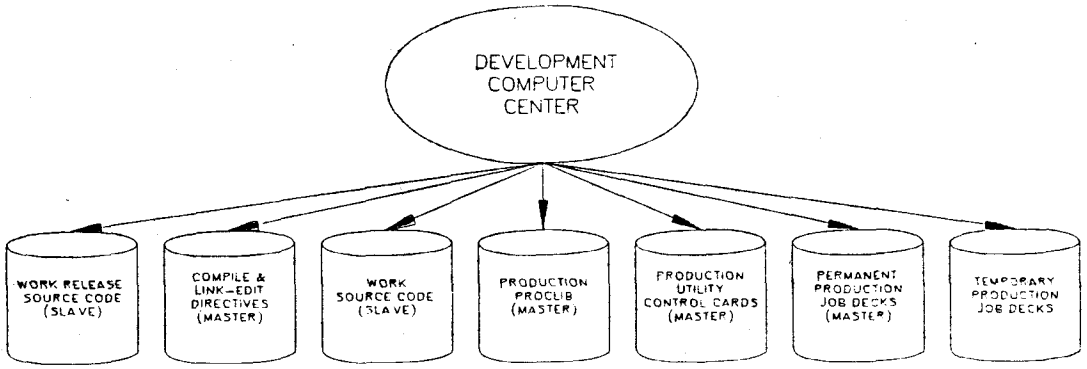



Figure 2 LIBMAN Logical View – DEVELOPMENT

# AUTOMATED CONFIGURATION MANAGEMENT ON A DOD SATELLITE GROUND SYSTEM

Kathleen B. Christian  Sandra H. Zucker

General Electric Space Systems Division
P. O. Box 8555
Philadelphia, Pennsylvania 19101

## ABSTRACT

An automated Configuration Management System (CMS) was developed at General Electric and is in use on a DOD Satellite Ground System maintenance contract. CMS improves and enhances the manual techniques for project tracking and change control and allows for reliable management of large projects. Because CMS works with any file, it can perform Configuration Management on all items in a configuration. The CMS Bookkeeping and Status Accounting forms are displayed on a terminal and the user is guided into filling them out correctly. New configuration items or changes can be entered into the system only after approval has been supplied by the proper authority. Since a common project data base is built by CMS, visibility of current system status is available to those who are permitted project access. Standard report forms as well as user defined report forms are used when viewing the current or historic system information. CMS not only controls the configuration, but also the paperwork, change approval cycle, and the quality of the product.

## INTRODUCTION

When a DOD Satellite Ground System went into the maintenance phase, and multiple versions of the Ground System software were required, it was decided to use the automated Configuration Management System (CMS). Although the Ground System software was developed for the MODCOMP computer and the CMS program ran on a Digital Equipment Corporation (DEC) VAX 11/780, it was expected that enough benefits would be derived from the control of source code files and Configuration Management (CM) paperwork to warrant the acquisition and the use of the VAX tool.

A prime concern of the software maintenance phase was the control of software modifications made as a result of customer requested enhancements, upgrade of Vendor hardware and software, and software rework for the correction of problems found at the field sites. This was a very difficult CM chore since each of the Ground System field sites contained different hardware and software configurations. Each site had approximately 1000 source modules and over 300,000 lines of code. The controlled master copies of all the Satellite Ground System software versions as well as associated documentation and manuals had to be stored and maintained. An automated Configuration Management System was needed to facilitate and control the process of analyzing and approving problem reports and enhancement requests, integration of multiple modifications into a software release, validation of new versions of software, and tracking associated documentation. We believe CMS is the only software program that integrates code control and forms control for Configuration Management. It merges the paperwork with the control so that changes cannot be made without the proper approvals. By combining the two into a data base, we are able to generate unique status reports on this information.

## AUTOMATED CONFIGURATION MANAGEMENT SYSTEM

CMS was developed at GE under an IR&D project in 1982. The objective was to automate the manual procedures used for change control, configuration item (CI) control and identification, and status accumulation and reporting for software projects. The result was a program that can perform the major functions for identifying, controlling and tracking of any CI. A paperless system where forms are filled out at a terminal, where users are aided by system prompts and help responses to questions, simplifies CM procedures.

Because large projects require a high degree of coordination, a single master data base accessible to all project personnel via a computer terminal is a necessity. CMS has a master data base which is protected and cannot change unless the proper approval has been entered. The status data base is available to enhance project visibility, aid in coordination of activities, accumulate historical data and improve product quality.

CMS executes on a DEC VAX computer using the VAX/VMS version 3.5 or higher operating system and the VT100 terminal. CMS requires the use of the VAX/VMS Common Datapool Dictionary (CDD) version 3.0, the Datatrieve program version 3.0, and the Forms Management System (FMS) version 2.2. These programs are used to create, update, and access the data base for CMS. The CMS user interfaces only with the common environment of CMS so that the man-machine interface is the same for all CMS functions. CMS requires a minimum of 9000 blocks of disk space to execute. In addition to this space, CMS requires disk space needed for controlling the configuration items.

## USER FRIENDLY

Because CMS must handle many different types of users, its man-machine interface is very user friendly. It is menu driven with a HELP feature available at any prompt, and where the prompt responses are almost always obvious. Each response is assigned a priority level so that only those users having passwords which allow for this priority may execute the response. All CMS transactions may be saved on a file during a session for future retrieval and printing.

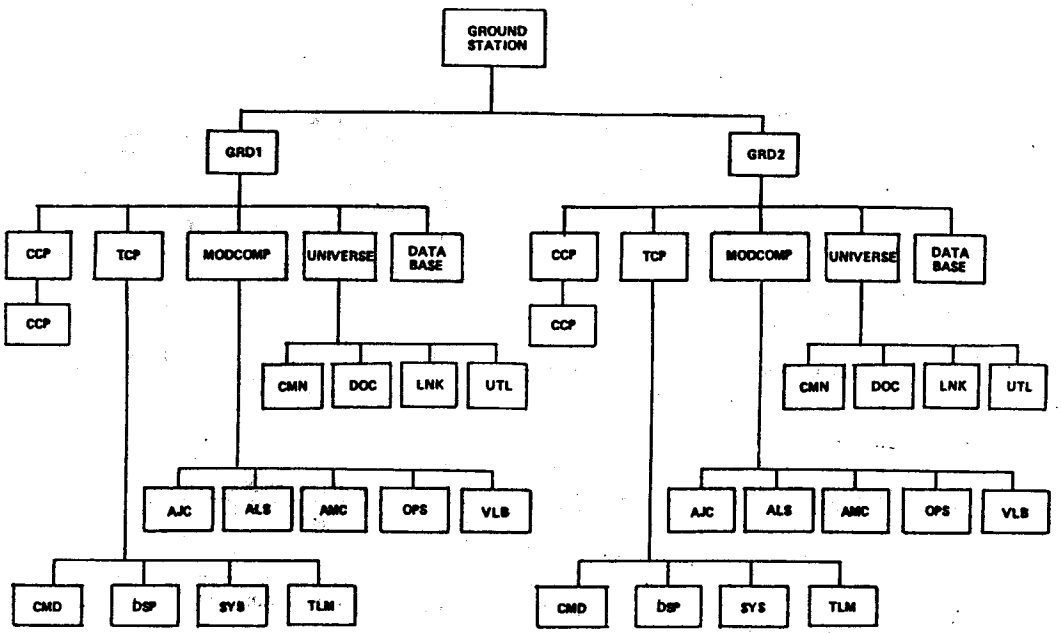
## STATUS ACCOUNTING

The bookkeeping and the related forms required by Configuration Management have been standardized and computerized for CMS. Discrepancy Reports, Change Proposals and Work Orders are entered into the system on a form displayed at a terminal. The CMS System performs error checking, fills in known data (e.g., form identification number and date), provides a HELP facility for explaining the needed data fields, and rejects records containing detected errors indicating the field in error. New CIs or changes to items already under control are accepted into the system only after the paperwork for them is complete, properly approved, and follow project standards.

## CONTROL AND TRACKING

The control and tracking of the CIs of a project is accomplished in CMS with three controlled areas of storage; Development, Test and Release. Each area contains a tree defined by the project manager which reflects the project structure. The tree is used to locate, control, and logically link the CIs of a project.

The nodes are collections of items (directories). These items may be files of source code in any language, executable code, object modules, command files, data base files, documentation files, test procedures and/or hardware descriptions. Figure 1 shows the tree built for the Satellite Ground System software. At each level a single node is taken and further defined into its next level of structure.




SATELLITE GROUND SYSTEM CMS TREE
FIGURE 1