RT
131

Software Maintenance Tool

By
Rabe'ah H. Yassin

PROJECT

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science
at the Lebanese American University
May 1995

Signatures Redacted

Dr. Nashat Mansour (Advisor)
Assistant Professor of Computer Science
Lebanese American University

Signatures Redacted

Dr. Mohammad Kodeih
Assistant Professor of Computer Science
Lebanese American University

BUC

Abstract

This report describes a software maintenance tool that reduces the maintenance effort and cost. Its goals are: to provide consistent documentation of a software system during both its development and maintenance phases; to help maintainers understand the architecture and algorithm of a software system without having to go over the actual code or paper documents; to identify any interrelationship between the various parts of a software system so that maintainers can better determine the affected and involved parts in a maintenance change; to provide a disciplined procedure to implement any maintenance action.

An important feature of this tool is dealing with the version control problem which is needed in any baseline, where multiple versions of the same software and its parts are present, facilitating the identification of each configuration of a software system and its constituent parts at any one time.

Acknowledgment

First of all, I would like to thank my advisor, Dr. Nashat Mansour, for his grateful guidance and advice. Many thanks go too for my second reader Dr. Mohammad Kodeih for his precious comments. Also I would like to thank Mr. Kamal Haidar and Mr. Jalal Kawash for their assistance in using the Oracle tools.

Last, but not least, I would like to thank the Natural Science Division chaired by Dr. Ahmad Kabbani; and Mr. Vatche Papazian and the ACC staff for the facilities they offered me that helped in developing this tool in an acceptable working environment.

Table of Contents

Chapter 1

Introduction

The Waterfall model is a widely used software development life cycle model. It consists of the following phases: Problem Definition, Requirement Specifications, Analysis and Design, Implementation - Coding, Testing (Integration and System), and Maintenance.

Software maintenance consists of those activities required to keep a software system operational and responsive after it is accepted and placed into production. It can also be defined as the set of activities which result in changes to the originally accepted (baseline) product set [Longstreet 1990]. Software maintenance falls into three categories: Corrective, Adaptive, and Perfective, which consist of modification created by correcting, inserting, deleting, extending, and enhancing the baseline system.

Software maintenance represents 60-70% of the total cost of software. Perfective maintenance (changes, enhancements, and extensions, etc.) comprises approximately

60% of the software maintenance costs. Adaptive maintenance and corrective maintenance are each approximately 20% of the total [Longstreet 1990].

Hence, maintenance is the most difficult of all aspects of software production because it incorporates aspects of all the other phases as well as being the costliest phase.

Not only software maintenance is important and needs consideration from the software development managers' part, but also software documentation and configuration management. A key criterion to a better maintenance is good Documentation. Without it, there is little assurance that the software satisfies its stated requirements or that the organization will be able to maintain it later on. Therefore, the software development manager should pay a tremendous attention to the preparation, structure, content, and presentation of the software documentation.

The successful realization of a software product requires the strictest control over the defining, describing and supporting documentation and the software code constituting the product. It is inevitable that this documentation will be subject to change over the life cycle of the product due to correcting some bugs or errors, introducing improvements or responding to the evolving requirements of the marketplace. Configuration Management provides the disciplines required to prevent the chaos of uncontrolled change [Thayer 1992]; e.g., people forgetting to do something they meant to do, making changes that interfere with the progress of the others, or performing a change that has not been approved yet, etc. Hence, Configuration Management can be defined as the art of identifying, organizing, and controlling modifications to the software being built by a programming team [Babich 1986] or already operational in the market.

It is important to prepare for maintenance both during initial development and subsequent evolution of a software system. This implies full consistent documentation, change control procedures, version and release management, regression testing, etc. It is difficult to see how large-scale software projects can succeed without strong management to implement these techniques [Bennett 1991]. The use of software engineering tools can make a substantial contribution to producing better software systems.

This report presents a Software Maintenance Tool (SMT) that provides constant documentation to a software system during its development and maintenance phases, organizes the steps required to follow in implementing a maintenance action, and solves the version control problem. It is based on COMFORM [Capretz and Munro 1992] which is a maintenance tool, that provides guidelines and procedures for carrying out a variety of maintenance activities through a systematic approach. COMFORM accommodates a change control framework called the Software Maintenance Model (SMM), around which the Software Configuration Management (SCM) discipline is applied. The aim is to exert control over an existing software system while maintaining it, and at the same time redocumenting it.

In this tool (SMT) the proposed (SMM) phases are implemented fully along with the (SCM) discipline, in addition to the following amendments. The documentation of the development phase and the solution of the version control problem.

The importance of such a tool is to keep consistent documentation both in the development and maintenance environments so that the ultimate software product will reflect the specified system requirements. Also it helps in applying any maintenance activity by allowing the maintainer to query the original or last changes to the software as a whole or to any piece of its source code instead of going over the paper documentation to find what was the last change done, or how was that piece of source code designed and implemented. Such a tool is also important in identifying the configuration of a product at any one time; i.e., to let the programmers/maintainers know at any one time which version of the product they are currently working on or that needs maintenance, and which revision/variations of each piece of source code - of that particular version - of the product are included in the corresponding configuration.

This report is organized as follows. Chapter 2 summarizes previous and related work. Chapter 3 presents the specifications and goals of the tool. Chapter 4 explains the high-level design of the tool. Chapter 5 describes the major algorithms of the tool, states some test cases, and gives the software and hardware requirements needed to run it. Chapter 6 presents suggestions for further work and a conclusion. Appendix A is the user manual.

Chapter 2

Related Work

Several tools have been developed in order to automate the activities of software maintenance. Each of these tools is unique and special in the goals it serves and in its orientation towards a specific language. In this chapter, we present a brief review of some of these tools and their goals.

1. Visual Interactive FORtran (VIFOR)

VIFOR [Rajlich et al. 1990] is a tool oriented towards maintenance of medium-to-large Fortan 77 programs. In most cases when programmers come to modify a program, they have first to understand it. To do so they have only one way: The actual source code because it is the only "Exact" reflection of the program requirements unlike the paper documentation - if available.

With large programs where complicated interrelationships among its different components, it is not easy to read and understand the code. From here came the idea of creating a tool that allows the programmer to deal with program architecture directly, without having to extract it repeatedly and manually from the code. Another improvement was to represent the architecture graphically making the program easier and more understandable.

VIFOR is this tool that maintains the graphical representation of a program and provides the programmer with a visual editor to build and modify the program. It also stores the relations in a database to help the programmer understand the code and to follow the ripple effects of the modifications.

In VIFOR a program is represented in two ways: the source code and the graphical one. VIFOR also contains transformation in both directions, i.e., from code to graph and from graph to skeletons of code. Hence, it is suitable for re-engineering and maintenance of

existing code. Specially designed browsers implement the graphical interface. VIFOR contains a database that is based on a simple but effective data model of Fortran programs. The model contains only four entity classes and three relations, which make the tool small, and easy to implement and use. A simple query language allows browsing through the database.

A prototype of a similar tool for the C language has also been implemented and called VIC.

2. Maintainer's Assistant (MA)

The Maintainer's Assistant (MA) is a research environment implemented for the maintenance of software systems written in the C language. The environment includes tools to support analyzing system architectures, making structured changes to programs, and checking the adequacy of test cases [platoff et al. 1991]

.

These tools use cross reference and data and control flow information to support the analysis and understanding of existing systems. These systems can be edited by applying structured transformations to an integrated representation that presents views of the source text, syntax, static semantics, and control and data flow of software systems.

The Maintainer's Assistant consists of a variety of tools to support the software maintenance process:

- Arch: a tool for discovering and critiquing system architectures.
- The Change Assistant: assists in the process of changing a system through its various views.
- Tactic: a test analysis and coverage tool implemented for C.

In addition, an integrated program representation underlying the Maintainer's Assistant is multi-faceted. It provides views of the syntax of the programs, and the static symbol table and cross reference information, and the control and data flow information. All of these views are linked to support the requirements of the M.A. tools.

The program transformation toolkit concentrates ont he following tasks:

- Porting
- Global re-structuring
- Re-structuring control and data flow
- Introducing new abstractions
- Instrumenting code

These transformations allows maintainers to work in higher levels of abstraction, such as editing program architectures at the module level, or by introducing new abstractions in the coding, data structure, and algorithm domains.

This environment supports all of the C language, including the 'C' pre-processor. It provides a pattern matching language with C-like syntax and a facility for generating transformations by example.

3. A Tool for the Maintenance of C++ Programs

This tool is developed to help programmers understand Object-Oriented software systems written in C++. This task is accomplished by providing information about the set of classes and files comprising the system and the relationships among them [sametinger 1990].

The tool eases the process of navigating through the files and classes, and helps the user to get any needed information in a fast and easy way. To do so, the files of a C++ program are divided into little pieces of information, managed together with their relations among them.

In order to enhance the readability of the source code, the user can define global styles for different syntactic constructs, e.g., comments, keywords, etc. If a short description exists for an identifier then this description together with some other useful information can be shown at any place this identifier is used.

The user interface concept - present in this tool - is based on modern application frameworks amd the supported concepts thereof. It provides two selection lists, an editor window, an icon bar containing several browsing tools, a menu bar, and two information bars.

This maintenance tool offers the possibility to easily browse through the system by means of the existing relations. Also, useful information is displayed to protect the user from getting lost in the complex information web.

4. The Incremental Software Maintenance Manager (ISMM)

ISMM [Ryder 1989] is a prototype software maintenance tool which uses incremental static analysis to assess the scope of proposed source code changes. These effects can be predicted a priori, that is, without actually having to perform the software change, thus enabling maintainers to choose between alternative anhancements or bug fixes on the basis of their predicted system impact.

Incremental analysis efficiently updates data flow information describing the definition, use and sharing of data in an evolving software system, keeping this information consistent with current system state. The goal of ISMM is to demonstrate the feasibility and practicality of using incremental static analysis to aid in the maintenance phase of the software life cycle.

A software system is a dynamic entity; even well designed systems evolve over time, if not of bug fixes, then as enhancements are added. Incremental analysis allows us to effieciently maintain consistency between the current structure of the system and our descriptions of it.

The research is primarily aimed at large systems which involve large, distinct groups of people in specification, design, implementation, testing, and maintenance. No one person has a full understanding of the entire system.

Data flow analysis algorithms gather facts about the definition and use of data within a program. Interprocedural data flow analysis algorithms summarize the behavior of each procedure, its effects on its parameters and on global variables, and thus effectively document a software system. A full re-analysis after a system change is expensive; the described incremental analysis algorithm provides the manitenance programmer with the ability to examine the scope of change effects.

These incremental techniques efficiently document the current structure of the system, providing details which allow re-structuring to increase encapsulation in data use and to achieve a better organization than that which has grown over time.

5. Concluding Remarks

The four maintenance tools discussed above all deal with the problem of maintenance. As is obvious, they all tend to ease the understanding of a program by some means, e.g. some by presenting the source code in a graphical way, and some try to document any proposed changes, etc. But the main difference, among them, is that no one tool is addressed for more than one language; e.g., one tool is for Fortran like languages, one for

C, and even one for C++, etc. Hence, this specification is a hindrance for the tool's public usage and those software companies which work under different languages and environments have to own many maintenance tools, in order to decrease the maintenance problems.

Chapter 3

The Tool's Specifications

The basic idea of this tool is to structure the maintenance process into sequenced steps or phases (SMM phases). Each phase is represented by a form, the outcome of which is the source of documentation of that maintenance action. Change requests for any maintenance action had to go through this procedure, (SMM) steps. No step is performed unless its parent step is fully accomplished. This requirement makes it impossible for a change request to be processed without an authorized approval, and without some other authorized staff member specifying the change requirements needed.

1. The Software Maintenance Model

The following steps are the proposed ones for the maintenance procedure:

- Change Request

- Change Evaluation
- Maintenance Design Specification
- Maintenance Design Redocumentation
- Maintenance Implementation
- System Release

1.1 Change Request

Any reported change should be formally announced through a change request form. It is the first step in the maintenance procedure and the only trigger (i.e., no change should be considered if no formal form for that change was submitted).

1.2 Change Evaluation

This request for a change is then evaluated by authorized staff member to see if this change is urgent or if needed at all. This phase shows the effects of the modifications on the existing software system in terms of cost and schedule, resulting in approval or rejection.

Only upon approval, the following steps are implemented.

1.3 Maintenance Design Specification

Specifications of the proposed and approved changes are determined forming the structure of the modifications. Also, how the software components have to be modified is clarified (design issues). In this phase, analysis of the side-effects of the changes is also done ensuring that the modules' functionality is kept consistent. Integration and system tests need to be planned too.

1.4 Maintenance Design Redocumentation

This phase requires redocumenting the software components that have been subject to modification. During this phase the algorithms and the behavior of procedures for both normal and exceptional cases are explained.

1.5 Maintenance Implementation

In this phase the designs of the modifications are implemented to the corresponding affected source code modules.

1.6 System Release

The last phase is the system release where validation of the overall system is achieved by performing the planned module regression, integration, and system tests on the system as a whole.

The equivalent phases of the above maintenance procedure in the SMM are the following forms:

1) Change Proposal
2) Change Approval
3) Maintenance Specification
4) Module Design
5) Module Source Code
6) Configuration Release

The output of these phases are forms which form the *baselines* of the software maintenance process, and offer objective visualization of the evolution of that process. In other words, a baseline is a configuration, a set of versions of all modules in the product [Capretz and Munro 1992]. When trying to correct a fault, the maintenance programmer puts copies of any needed modules into his or her own *private workspace*, where any changes to the modules in this private workspace do not result in any changes in the original baseline version.

1.7 Change Proposal

A change proposal form is the first form in the model. It is the form that is filled-in data from the change request form. If the proposed change is for corrective maintenance, then a complete description of the circumstances leading to that error must be included. For other types of maintenance, an abbreviated requirements specification must be submitted.

1.8 Change Approval

For the change evaluation phase a change approval form exists which is one of the documents used as the basis for planning the system release. It is a vehicle for recording information about a system defect, a requested enhancement or quality improvements. By documenting new software requirements or requirements that are not being met both the change proposal and change approval forms become the contract between the person requesting the change and the maintainers who work on the change.

In this phase the approved changes are ranked and selected for the next release. But first the work required by the proposed change is classified as perfective, adaptive, or corrective. In addition, every software component involved in the proposed change must be known at this stage. The inadequacies, or unfulfilled requirements described in the change proposal form are identified in the existing software system. This identification involves different aspects of software which depend on the type of the change required (the three types mentioned above).

1.9 Maintenance Specification

The maintenance specification phase is related to the maintenance design specification step in the proposed maintenance procedure. This form contains all the information described in the equivalent phase i.e., a complete, consistent, and comprehensible common specification of all the changes proposed and approved for a planned and a scheduled release. The affected software components are listed, the side-effects of the changes are identified and entered, Integration and system tests are specified.

1.10 Module Design

The module design form is the one associated with the maintenance design redocumentation phase. It aims at documenting the maintenance action on a software component of an existing software system. The forms will be filled-in when the corresponding software component has to be modified (since for each change request many software components may be affected, hence each has a different form).

In this form the module's purpose is mentioned along with its algorithm outline and the interface definitions (what tables it uses and what other source codes it calls). In addition, the tests planned for each of the changed or implemented software components are specified. This form captures the highest level documentation of each of the source code modules/software components a software system.

1.11 Module Source Code

Module source code is the form associated with the maintenance implementation phase where the actual change in the source code occurs to the affected modules. There is a one-to-one relationship between each source code form related to each software component. While the module design form is aimed at keeping general and stable information for a software component, its corresponding module source code form aims at keeping the information pertaining to modifications on the components (e.g., tests' outcomes, comments, etc.)

1.12 Configuration Release

System release is the last phase of the proposed software maintenance model. Validation of the overall system is achieved by performing the integration and system tests on the software system. Once modifications on the system have been performed under the configuration control function (exerted by the Software Configuration Management -

SCM discipline), the task at this stage is to certify that all baselines have been established.

The configuration release form contains details about the new configuration. It is the software system release planning document which aims at keeping information pertaining to the history of the whole corresponding maintenance phase.

2. The Software Configuration Management

The SCM discipline applied to (SMT) consists of the following four functions:

- Software Configuration Identification
- Software Configuration Control
- Software Configuration Status Accounting
- Software Configuration Auditing

The purpose of the first function is to identify the parts of a software system in a manner that makes explicit the relationship between these parts [Capretz and Munro 1992].

The role of the second function is to ensure that any change required in a software system is defined and implemented by following the SMM phases properly.

The third function aims at recording and reporting the current status as well as the evolution of existing software system.

The auditing function comprises the processes of verification and validation. It ensures that related phases are consistent in some of the information, that no other information is missing from any form and that all phases have been performed before a maintenance activity can be announced as DONE.

Although COMFORM is dealing with the software maintenance and documentation problems, yet it does not solve the version control problem associated with each maintenance action.

3. The Tool's Specifications

This tool is composed of two components: Development and Maintenance. First of all, this tool will serve as the database storing place for the documentation of any software system under development and for its evolving maintenance problems and enhancements later on.

During the Development phase, the user can, as a first step, define the name of the software system under development as well as its modules, submodules, and programs, etc. Also the user can define all the files, tables, etc. used by the software system during its development.

Definitely, relationships between the software system and its constituent parts are defined in order to identify which modules and components belong to what system; in short, the hierarchy of the system can be drawn easily - for maintenance purposes.

Each part of the software, including the software system itself, has a place where it is described in full details including nature, purpose, algorithm, comments, parameters passed (in and out), any useful comments, and estimated and actual time resources spent in developing each module or source code, etc. This serves as the basic and original documentation for each part of the software system.

At this stage, the maintainer/developer can query or view the description of any piece of source code, or even the description and structure of each file or table used. This reduces the manual work of going back to paper documentation in order to understand a piece of code.

The available reports at this stage are:

- List of all source code modules related to a particular software system
- List of all other components related to a particular software system

- Query the Status of a module
- List the components used by a piece of source code
- List the names of the staff members who developed a certain piece of code
- List of the callees of a particular source code
- List of the Input/Output parameters used by a certain source code
- List the structure of a certain table or file
- List the authorized grants given to a staff member for a certain table/file
- List of the Tests performed, and their Status, on a certain piece of code
- List of the Estimated and Actual time taken to develop a certain piece of code
- List the name of softwares that are written in a particular Language
- List of all the staff members working in the software company
- List of all the Client Institutions to which software are distributed
- List of the Employees at the Client Institutions
- Date of delivering a particular software system to a certain Institution

The maintenance phase is initiated upon the arrival of a bug/enhancement report. Then the SMM phases have to be followed in order for the change to be implemented. Each phase is a form where the name of the software to be maintained - along with its version number - is required. The bug/enhancement report is transformed to a more formal form which is the Change Proposal Request, where the change Description and the Reason for change are required.

This proposal is studied, by an authorized staff member, and it is either Approved or Rejected. Maintenance Change Specifications are prepared for the Approved changes only. The type of maintenance, how urgent it is, the consequences of implementing the change, and the modules Affected, etc. are identified at this stage before the Change Designs are prepared (for the Affected modules). Once ready, the designs are first entered into their proper form in the tool, assuring the action of Re-Documenting only those Affected modules.

The actual implementation of the changes are therefore performed on the actual code with the supervision of some authorized staff members, whose names appear on the corresponding form. At the end, a new system configuration is released producing both a new Version Number for the software just being maintained and at the same time a new Revision Number for the modules affected by the change.

Hence, the original designs are not touched, each is stored in a different record and we will have in our database more than one Version of the same product each with different Revision Numbers of some modules (the affected ones by the maintenance action).

The available reports at this stage are:

- List of all source code modules affected by the change
- List of all other components affected by the change

- Query the Version Number of a particular software system
- Query the Revision Number of an affected source code
- List the names of the staff members who maintained a certain piece of code
- List of the new Input/Output parameters used by a certain source code
- List of the Module Regression Tests performed, and their Status, on a certain piece of code
- List of the Estimated and Actual time taken to maintain a certain piece of code
- List the new Algorithm of an affected source code
- List the new components that it uses
- List the new callees that it calls
- List the time needed to maintain a certain piece of source code and the total time needed to accomplish the whole maintenance action

etc.

The same reports, of that of the Development Phase are found in the Maintenance phase. Hence, this tool ensures at each step a proper documentation, as well as takes care of the Version/Revision control problem by assigning a new number for each module affected by any change.

The unaffected modules are kept under the same "old" revision number, but they are related somewhere to the software system with the new Version Number. Because previously all source code and components belong to the Original software system of

Version Number "ZERO". Later, and as this number increases, all its parts are also kept related to the same software name but of different Version Number and hence all original designs and algorithms can be seen by entering the proper Version/Revision numbers. Any new changes can also be viewed - for any piece of source code - by entering the new Revision Number and the New Version Number.

Ultimately, this tool will be able to store the documentation of each part of a software system regardless if its revision number. At any one time, the user can view the original documentation of any source code module even if the system has been released for ten times; i.e., it now holds the version number 10 and maybe revision number 5 for the module. The same applies for the files and tables related to the same software system. The user can also view any changes on a specific source code module or other component by entering - in the right place - the name of the Design Change of that specific source code module or component as well as its revision number and the correct software version number.

4. Assumptions

Following is a list of the assumptions used in this project and some terms that need be clarified from the user's part before doing any further reading of this report.

4.1 Terminology

- Source Code Module: This term refers to the actual piece of source code related to a software system; e.g., a procedure or function.

- Components: This term refers to the components other than the source code modules related to a software system; e.g., a table, file, etc.

- Software system or project: This term refers to any software developed by the company owning this tool.

4.2 Assumptions

- This tool is meant for third generation languages; i.e., procedural languages.

- Each software system is identified in this tool by a unique name. The same for the source code modules and other components.

- The software systems are assumed to be of *LEVEL ONE* and all the *functional* source code modules are of *LEVEL TWO*, whereas the *actual* source code modules are of *LEVEL THREE*. The other components have no level and are documented from separate screens.

- All software systems newly developed start with a version number of *ZERO*. The same for source code modules and other components, they start with revision number of *ZERO*.

- Only upon modifications - maintenance actions - that the source code modules and the other components are given new revision numbers (incremental and generated by the tool). Any modification done during the development phase does not lead to a new revision number.

- The same for the software systems themselves. Upon a maintenance release a new version number (also incremental and generated by the tool) is given to the software system.

- A company may have more than one name, but only one should be declared as Effective.

- Prior to coding the Status of a source code can be: Under-Development, but after it is finished it should be declared as: Complete or done, etc. with any modifications to the already entered designs and algorithms, in order to have a consistent documentation.

- On amending a component - adding/deleting/changing functionality or description of one of its fields - all its modified and unmodified fields have to be defined and described again as if for the first time.

- A Change Proposal can have a Change Approval and - at the same time - a Change Rejection. Definitely each should be with a different date. Hence, the most recent one is the effective; i.e., if the Change Approval date is greater than the Change Rejection date then the Proposal is considered as Approved. Otherwise, it is considered as Rejected.

- A Change Proposal can be Canceled any time before its Maintenance Specifications are prepared by assigning a Change Rejection to it. But after the Maintenance Specifications are entered to the tool, a Change rejection will not do the job of canceling the proposal.

- No Change Request or Proposal on a component; i.e., components can not be declared as Involved in the changes - from the Change Approval form - but they can be declared as "Software Affected" - Module Design form.

- "Software Involved", hence, are the source code modules that the authorized maintainer, who approved the change, thinks are involved.

- "Software Affected" are the source code modules and/or the other components that the authorized maintainer, who prepared the maintenance specifications of the change, believes are affected by the change; i.e., that are ultimately subject to change in their design in a way or another.

- After the release of any software system, the user has to declare the software's status as "Complete" or so.

- If the maintenance action is Perfective or adaptive new source code modules need be implemented, then the user has to do the following actions in order:

N.B.: the user has already filled-in a Bug, Proposal, and Approval forms.

1- Define the names of the source code modules and/or any components related,

2- Relate them to the corresponding software system (of *CURRENT* version and revision *ZERO*),

3- Describe each according to its corresponding level,

4- Fill-in the corresponding Maintenance Specifications form declaring the just-defined source code modules as Affected,

5- Fill-in the corresponding Module Design form with a 'Y' in the Perfective field,

6- Proceed in the maintenance steps as usual, but without specifying any new algorithm, new specsifications, new estimates, new tests, etc.

7- Remove the newly implemented source code or component from the new software version by performing Remove Source Code from aVersion option.

The above sequence of steps allows new source code modules - of revision number *ZERO* - to be added to the *CURRENT* version of the software system, to which the perfective maintenance actions are to be performed, in the same ease of defining and describing any

original source code modules during the development phase, not forgetting to perform all the maintenance procedure steps - SMM phases. In addition to a small difference which is the omission of the Design Changes, on that source code module, that are entered from the Module Design form after pressing [Next Record] key and going to the corresponding form (depending whether the source code module be of Level One, Level Two, or Level Three).

- In any phase - form - all data is obligatory to be filled-in. This is part of the Auditing Process (completeness checking) in the SCM discipline.

- Time resources estimates in Maintenance Specifications serve as the Total time resource estimates for the whole maintenance action. The actual time resources are entered from the Maintenance Configuration form.

- If the change corresponds to a change in the software as a whole, then these estimates - and actual - will be considered as the Level One time resources, but for the new version.

- For the time resource estimates for any source code module during its maintenance, they are entered from the Module Design form, while the corresponding actual time resources are defined from the Mtacual form.

- The same for the source code modules during their Development. The estimates are defined from the Level3 form while the actual are defined from the Actual form.

- No Change Request is allowed on an old version of any software system.

Chapter 4

Design

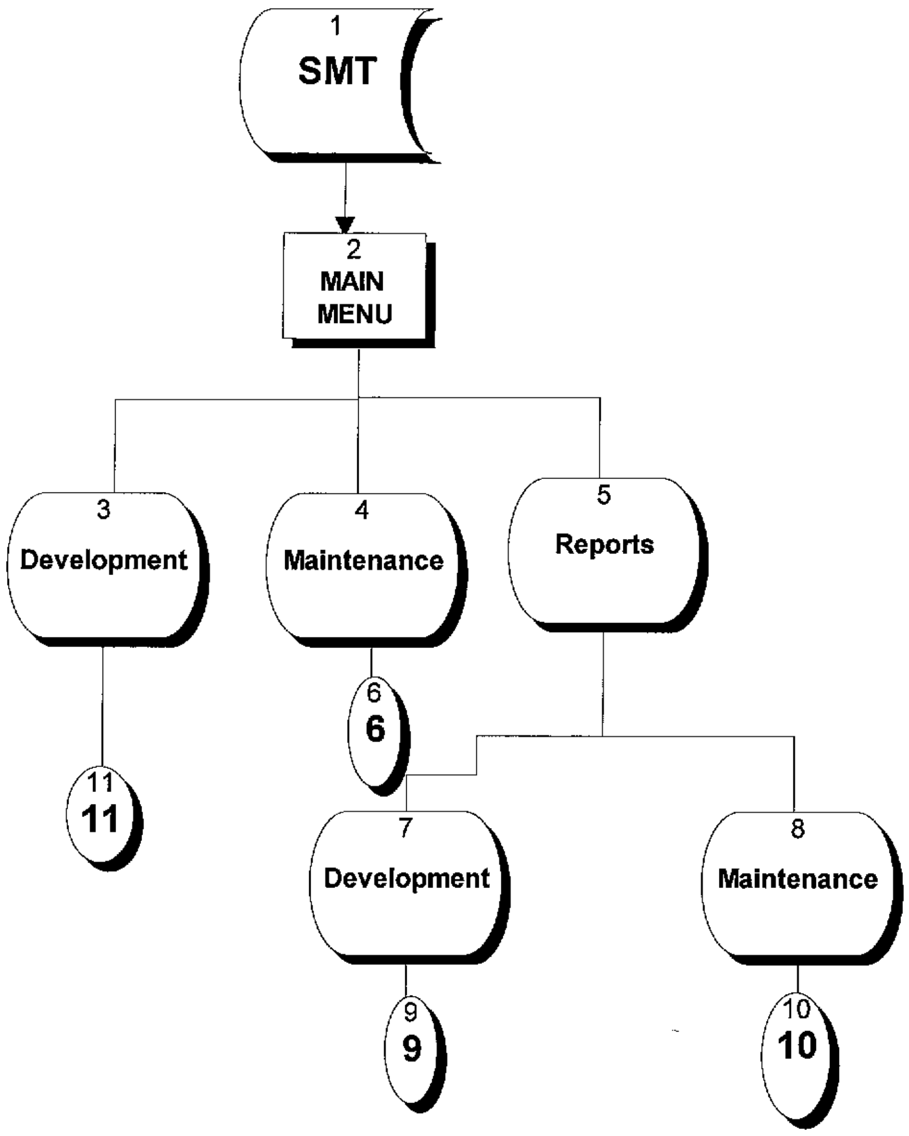
This chapter presents the design of the input forms and data base tables used in the tool. The tool is divided into three items:

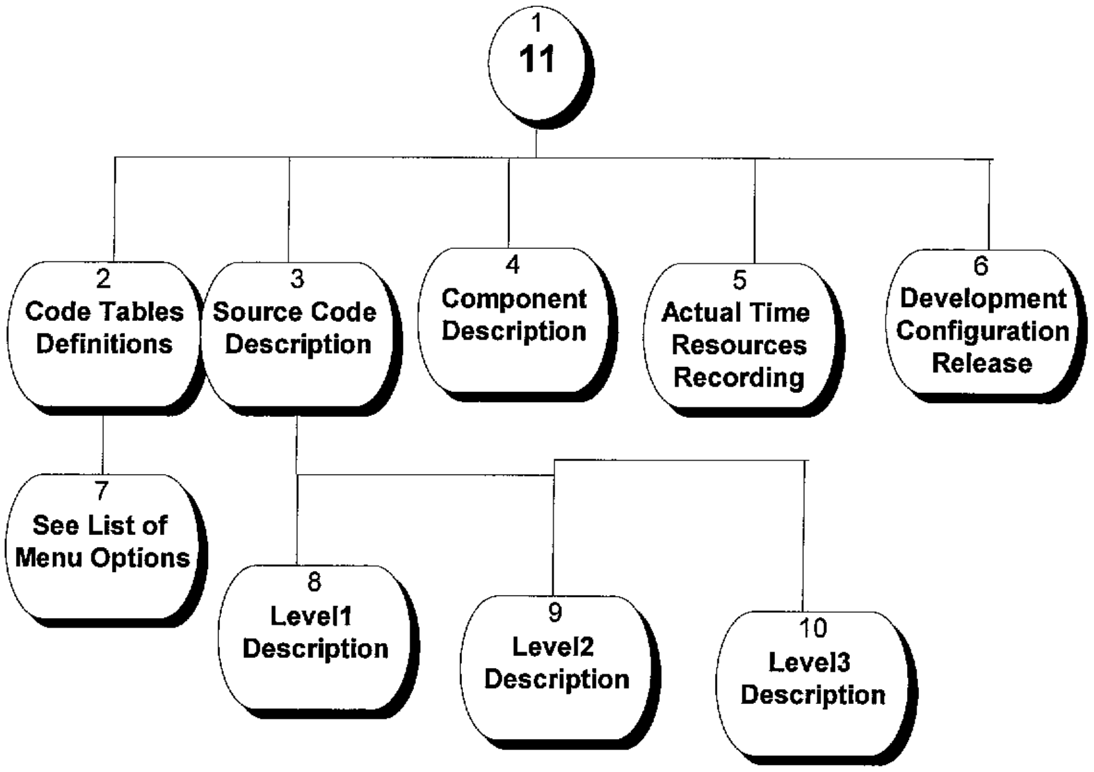
Development, serves as the place where the user defines the documentation of a piece of software down to the lowest level of its parts.

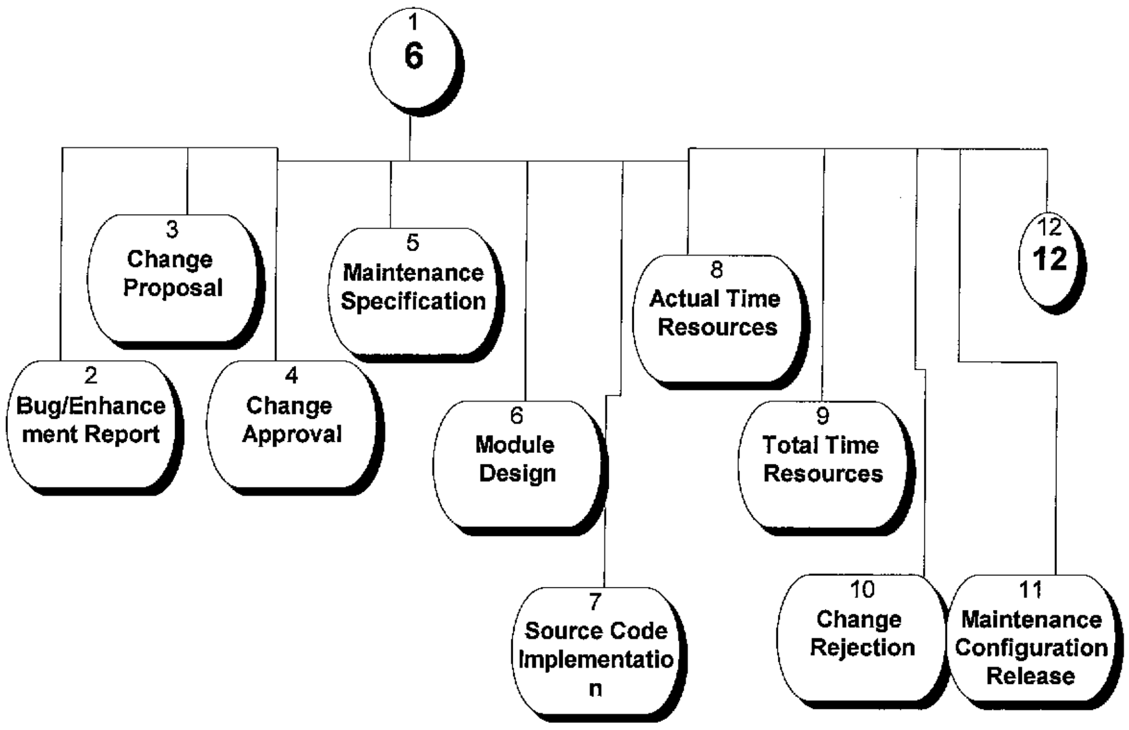
Maintenance, is where a proper procedure for maintenance is implemented. Any maintenance action has to go through the assigned menu options in the sequence provided.

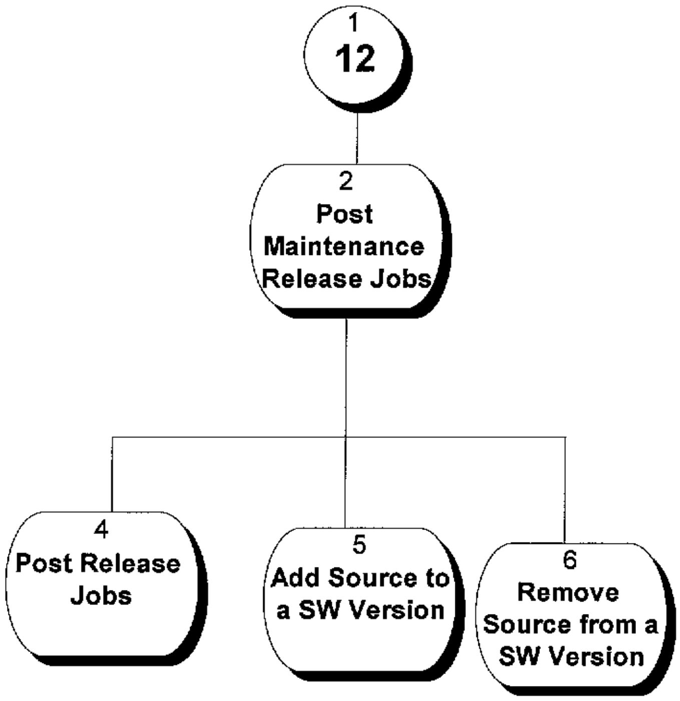
Reports, is where the statistical reports from the tool's data are provided. It is divided in its turn into two parts: Reports from the Development phase and Reports from the Maintenance phase.

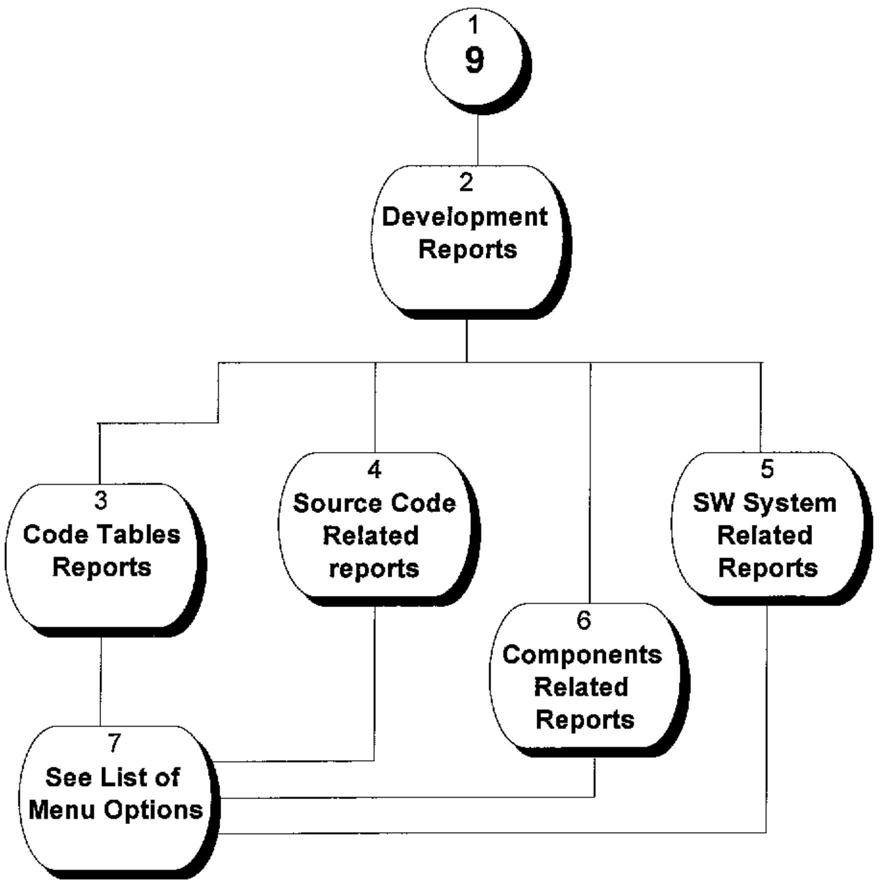
Following is a structure chart for the whole tool.

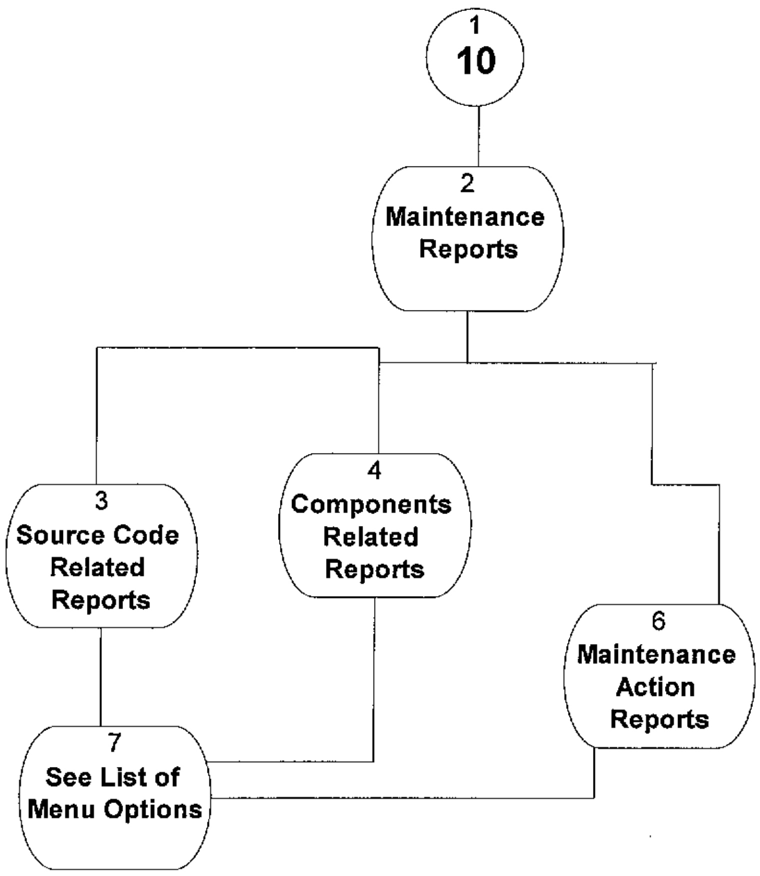



1
11
2
Code Tables Definitions
3
Source Code Description
4
Component Description
5
Actual Time Resources Recording
6
Development Configuration Release
7
See List of Menu Options
8
Level1 Description
9
Level2 Description
10
Level3 Description


1
6
3
Change Proposal
5
Maintenance Specification
8
Actual Time Resources
12
12
2
Bug/Enhancement Report
4
Change Approval
6
Module Design
9
Total Time Resources
7
Source Code Implementation
10
Change Rejection
11
Maintenance Configuration Release


1
12
2
Post Maintenance Release Jobs
4
Post Release Jobs
5
Add Source to a SW Version
6
Remove Source from a SW Version


1
9
2
Development Reports
3
Code Tables Reports
4
Source Code Related reports
5
SW System Related Reports
6
Components Related Reports
7
See List of Menu Options


1
10
2
Maintenance Reports
3
Source Code Related Reports
4
Components Related Reports
6
Maintenance Action Reports
7
See List of Menu Options

1. Design of Menus and Input Forms

This section discusses in detail the design of the tool starting with its menus and the forms associated with each menu option. It also states the order to be followed in using them, and the flow of data entered from the forms into the tables.

The first menu is the Main Menu composed of two components.

Main Menu

1- Development

2- Maintenance

3- Reports

4- Exit

This menu divides the tool into two main phases: Documenting a software system under development, or Re-documenting a software system being maintained.

1.1 The Development Option

We will start with the development option. Choosing it leads to the following menu:

1- Code Tables Definitions

2- Source Code Description

3-Components Description and Structure

4- Actual Resources Recording

5- Development Configuration Release

6- Previous Menu

1.1.1 Code Table Definitions option leads to the following menu:

1- Nesting Levels

2- Systems

3- Source Code Defintions

4- Components Definitions

5- Development Phases

6- Institutions

7- Employees

8- Maintenance Team

9- Maintenance Types

10- Components Types

11- Tables Grant Types

12- Status

13- Company's Name

14- Tests

15- Field Types

16- Previous Menu

In the following section, we describe each option in more details explaining its function and the data flow in the tables triggered by some keys.

1.1.1.1 Nesting Levels

a. Function

This option allows the development team to define how large and nested the software system under development is. It defines the software's hierarchy by assigning, in decreasing order, its levels (e.g., Project, System, Subsystem, etc.) Since this tool identifies three levels which are primarily: Level 1, for the system or project; Level 2 which contains all theoretical and functional levels (i.e., subsystem, module, submodule, etc.); and Level 3 which consists of the actual pieces of source code. Hence, it is up to the defining user to assign one of the three numbers (1, 2, and 3) for each level defined in order to tell later on what 'Description' screen a specific level needs. The form associated with this step is:

<u>(Name of the Development company)</u>

<u>Define the Nesting Levels</u>

<u>Name</u>	<u>Assign a level (1-3)</u>
System	1
Subsytem	2
Module	2
Submodule	2
Program	2
Subprogram	2
Procedure	3
Function	3

Of course, internally the system generates unique codes for the items being defined in this form and in all the other forms (discussed below).

b. Tables Affected

Table Levels is the only table to be affected. Each entered hierarchy level will cause a new record to be opened in the Levels table, as well as generating for it a unique code. By this, all internal reference to a level in the other tables will be through the code and not the name. The assigned number for each level is entered in the Levels table in the field "levelnbre".

1.1.1.2 Systems

a. Function

This option allows the users of the development team to define all the systems that they are about to develop. The form associated with this option is the following:

(Name of the development company)

Software Systems Entry

Name

BUC_PROJECT

LAU_PROJECT

DAR_AL_HANDASAH

SOLIDERE

etc.

b. Tables Affected

Table Systems is the first table to be affected. Each entered or defined system will have a new record created for it, as well as a unique code. All internal reference to any system, among the other tables, will be through the system's code.

Table Systemchart is also affected in the following manner: as a new record is created in this table for each defined system assigning the field "syscode" in the latter table the value of the "code" field just created in the Systems table. The "modcode" field in the Systemchart table is used to assign a unique code for each item within a system. It is going to have the value of Zeros for the records created for the systems defined in this form.

1.1.1.3 Source Code Definition

a. Function

This option allows the development team to relate a piece of source code to a software system. This option does not lead to any other menu, but directly to the following form:

(Name of the development company)

Source Code Definition Form

Level1 Name:	Version:
Source Code Name:	Revision:

Parent Source Code Name:

Source Code Level Type Name:

b. Tables Affected

The Systemchart table, which is one of the most important tables in this maintenance tool, is the one affected by the data entry in this form. Once a new piece of source code gets defined through this form, a new record for it is created in this table; the system will generate a unique code for it and assigns it to the field "modcode"; it also reads the name of the system that this code belongs to and gets its corresponding code from the Systemchart table and assigns this code to the field "syscode"; the name of the parent piece of the code is also read and from the Source table the code is determined and assigned to the field "parentcode"; the last thing read is the source code level name which is for example "subsystem" or "submodule". The name is read and the tool will search in the Levels table to get the corresponding level code and the value got is the one assigned to the field "levelcode".

1.1.1.4 Components Definitions

a. Function

This option allows the development team to define the tables, files, etc. used in developing a software system. The form associated with this option is the following:

(Name of the development company)

The Components' Definition

Name
StdDecision
In_File1
Student_Info
Student_Gpa

etc.

b. Tables Affected

Table Component is the only table affected. A new record is created for each defined component assigning to it a unique code to be used later on for internal references.

1.1.1.5 Development Phases

a. Function

This option allows the development team to define the phases that their company is applying to achieve a proper development of a software system. Of course, it has a unique form associated with it:

(Name of the development company)

Development Phases Entry

Name

Analysis

Design

Coding

Testing

etc.

b. Tables Affected

The table affected is the Develophases table. Each entry defined will have a new record created for it in the table as well as generating for it a unique code. This code will serve as the foreign key in other tables (of course internally speaking, because once the user needs to refer to a specific phase, all he/she has to do is to type the actual name, or choose from a pop up menu.)

1.1.1.6 Institutions

a. Function

This option allows the user to define the client institutions that the development company will be selling their softwares to. The associated form with this option is the following:

(Name of the development company)

Define the Client Institutions' Names

Institution Name	Type of business
BUC	ACADEMIC

AUB	ACADEMIC
AUH	MEDICAL

etc.

b. Tables Affected

The table Institutions is the one affected. It creates a new record for each institution getting defined, giving it a unique code as well. This code will serve as the foreign key in the other tables once referenced. The institution name defined will be assigned to the field "institution_name", the same happens for the "typeofbusiness" field in the same table.

1.1.1.7 Employees

a. Function

This option allows the development company to define ultimate users in the client institutions, who are going to request the changes or report any errors, later on. The form associated with this option is the following:

(Name of the development company)

Define the Client Employees' Names

Institution Name

Buc

Employee Name	Position
Randa Gharzeddeen	Registration officer
Nada Badran	Admission officer

etc.

b. Tables Affected

The Employees table is the one affected. For each new employee name defined the system will generate for it a unique code and a new record as well is inserted in the corresponding table. The system will also pick up the institution's id, from the Institution table, that corresponds to the institution's name just entered as the working place for the defined employee. This institution code is assigned to the field "institutionid" of position column.

1.1.1.8 Maintenance Team

a. Function

This option allows the development company to define its staff members who are doing the development of the softwares and who are going to perform the maintenance actions, ultimately. The form associated with this option is the following:

(Name of the development company)

Staff Members' Names

Name	Position	Authorized
Kamal Haidar	General Manager	Y
Jalal Kawash	Consultant	Y

b. Tables Affected

Mtceteam table is the one affected by the process of defining employees' names who belong to the development company. Each name will have its own and unique code.

1.1.1.9 Maintenance Types

a. Function

This option allows the development company to define the types of the maintenance, e.g.: Corrective, Perfective, etc. The form associated with this option is the following:

<u>(Name of the development company)</u>

<u>Software Maintenance Types</u>

<u>Maintenance Types Names</u>

corrective

adaptive

perfective

b. Tables Affected

Mtcetypes table is the one affected. Each type defined will have its unique record and code that is going to be used later on as the foreign key in other tables.

1.1.1.10 Components Types

a. Function

This option allows the development team to define the possible types of the components related to the software systems, in general. The associated form with this option is the following:

(Name of the development company)

The Components' Types Definition

Components types

index

view

table

b. Tables Affected

The table Component_Type is the one affected. For each defined type there is a unique record as well as a unique code; hence, the code will serve as the foreign key in the other tables that need reference a component type.

1.1.1.11 Tables Grant Types

a. Function

This option allows the development team to define the grant types of the tables used in the developed softwares; i.e., each table or file created by some of the development team might be granted to some other staff member in the development team by some condition, the grant type. Hence, some have the right to update the contents of a table, while some others have the right of only viewing its contents (query).

<u>(Name of the development company)</u>

<u>Defining the Grants' Types</u>

<u>Grants' Types Names</u>

Query

Update

Query/Update

etc.

b. Tables Affected

The Grantype table is the one affected by this form's entry. It makes a new record for each grant type defined as well as a unique code for it.

1.1.1.12 Status

a. Function

This option allows the development team to define all possible status conditions, that the source code under development might undertake, or the outcome of each test, etc. The associated form looks like this:

<u>(Name of the development company)</u>

<u>Status Conditions' Entry</u>

Status Description

effective

under-development

in_process

o.k.

frozen

complete

b. Tables Affected

The Status table is the one affected by this form. Each defined status condition gets its own record and code. This code is going to be the foreign key in other tables.

1.1.1.13 Define the Development Company's Name

a. Function

This option allows the software company owning this maintenance tool to define its name to the tool, in order to have it as a heading in all of the tool's forms. The screen associated with this option is the following:

(Name of the development company)

Define the Name of the Owner Company

Define the company's name	Effective (Y/N)
New Dimensions	N
Logos	Y

Only one name should be effective at any one time and this is why the tool sets all Effective fields to 'N' upon defining a new name with 'Y' as effective.

b. Tables Affected

Only table Company_Name is the one affected. For each defined name a new record is created with a unique code number. The name defined is assigned to the field "name" and the flag (Y/N) is assigned to the field "effective". The "code" field is assigned the value generated by the system as the record number.

1.1.1.14 Tests

a. Function

This option allows the users to define the names of the tests used in testing all the software systems they are developing - or already developed. The screen associated with this option is the following:

(Name of the development company)

Tests Names' Entry

Tests' Names

Reg_Test1

Admission_Test1

Integration_Test4

etc.

b. Tables Affected

Only table Tests is the one affected. For each defined name a new record is created with a unique code number. The name defined is assigned to the field “name”. The “code” field is assigned the value generated by the system as the record number.

1.1.1.15 Field Types

a. Function

This option allows the users to define the types of the fields that might occur in a table or file. For example, a filed in a table can be of “character” type or “integer” or “alpha-numeric”, etc. The screen associated with this option is the following:

<u>(Name of the development company)</u>

<u>Data Types Entry</u>

<u>Data Names</u>

Character

Number

Date

Alpha-Numeric

b. Tables Affected

Only table Tests is the one affected. For each defined name a new record is created with a unique code number. The name defined is assigned to the field "name". The "code" field is assigned the value generated by the system as the record number.

1.1.1.16 Previous Menu

a. Function

This option leads the user one menu backwards; i.e., to the development option items.

b. Tables Affected

NONE !!!!

1.1.2 Source Code Description

a. Function

This option allows the development team to describe a piece of software that has already been defined. This option leads to another menu, the following:

1- Level 1 Description

2- Level 2 Description

3- Level 3 Description

4- Exit

1.1.2.1 Level 1 Description

a. Function

Choosing this option leads to the following form that allows the user to describe a piece of code which is of level 1, mainly a System or a Project. It is the description of the total software system under development.

(Name of the development company)

Level1 Description Form

Level 1 Name: Date:

Version:

Status:

Language:

Nature:

Development Phase Resources Estimates

Hardware Requirements

Constraints

---

b. Tables Affected

Table Level1desc is the one affected. First, the "level1 id" item on the form is read and table Systems is searched to find the system code, to relate the description to it. This code is assigned to the field "syscode" in the table. Each description creates a new record in the table with a unique code. The status condition code, selected from the Status table, is assigned to the "status" field, the same for the field "date", and "Languagetype" field means the type of language that the software system will be written in, whose value is the name entered by the user.

Tables Nature, Hardware_Requirements, and Constraints are affected in the same way. For each line of text entered in one of these blocks on the form, a new record is inserted in the corresponding table with the generation of a unique record code number assigned to the field "code". The field "name" is assigned a line of text, "flag" is assigned the value of 'L' meaning on development of Level1, "oldverno" and "newverno" are assigned the same value of the Version number found on the screen, "tabcode" is assigned the unique code that identified the software system.

Table Compare_Resources is affected as well. There are going to be as many records for each Level1 description as there are defined development phases for this specific software system; For each record created the "levelflag" field is assigned the value of "1", which is the current level number. "Flag" field is assigned the value of 'L' meaning Levels Description. The "tabcode" field is assigned the value of the current Level1desc record number. The "phasecode" field is assigned the value of the corresponding development phase (e.g., "1" for design, "2" for coding", and "3" for testing, each constitute a record on its own.) The "estimates" field is assigned the values entered from the screen for each phase. "Oldverno", "newverno" are given the value of the Version number defined in this form. All Level1 items do not have Revision numbers - only version numbers - hence, "oldrevno" and "newrevno" are not defined in this context. "Mdcode" field is used only for the records entered from a maintenance action. "Actual" field is left empty to be filled out at the end of the development through the configuration release form.

1.1.2.2 Level 2 Description

a. Function

This option allows the user to describe a piece of "theoretical" source code; i.e., a module's description, a program's description, etc. The following form is the one associated with this option:

(Name of the development company)

Level2 Description Form

Level 1 Name:		Date:
Level 2 Name:		
Version:	Revision:	

Status:

Purpose

Description

b. Tables Affected

Table Level2desc is the one of three tables affected. A unique code is generated for each record inserted in this table which will serve as the unique code for the unique combination of a software system and a piece of source code. We need the Level1 Name in order to tell to which system or project this piece of software, being described, belongs. Level2 Name is needed to identify the piece of source code being described. The codes - of the two levels' names - are sought from the Systems and Source tables, respectively, and inserted in the equivalent fields in the Level2desc table ("syscode" and "modcode"). The same for the "status" field where its code is brought from the Status table; the "datte" field is given the value the user has entered.

The second table affected is the Purpose table. The text entered in the corresponding block is inserted as records in the Purpose table where each line of text forms one record with a unique code and where the "tabcode" field is assigned the unique code generated for the Level2desc table. The field "name" gets a line of text, "code" gets the generated record code number, “levelflag” gets the value of 2, “flag” gets the value of ‘L’ meaning that the record belongs to a Level 2 Description. “Oldverno”, “newverno” are assigned the value of the Version number entered in the form, “oldrevno” and “newrevno” are assigned the value of the revision number.

The third table affected is the Description table where the “description” item on the screen is also a text entered and inserted as records in the corresponding table. Each record is given a unique code assigned to the table field “code”, the field “name” gets a line of text, and the field “tabcode” is assigned the value of the unique code generated for the Level2desc, in order to tell later on to which piece of source code this description belongs. The table field “flag” is assigned the value of ‘L’ meaning that the record belongs to Level 2 Description. “Oldverno”, “newverno” are assigned the value of the Version number entered in the form, “oldrevno” and “newrevno” are assigned the value of the revision number.

1.1.2.3 Level 3 Description

a. Function

This option leads to the following form. It allows the user to describe a piece of source code which is of level 3; i.e., actual code. It has the following form:

(Name of the development company)

Level3 Description Form

Level1 Name:		Date:
Level3 Name:		
Version:	Revision:	

Status:	
Developed by:	Starting Date:

Purpose

Algorithm Description

Comments

Interface Definitions

Uses(Components)	Calls(Source Code)	
	Level1 Name	Source Code Name

Resource Estimates for Development

Development Phase	Estimates

Input Parameters

Output Parameters

b. Tables Affected

The following tables are all affected: Level3desc, Developedby, Purpose, Algorithm, Comments, Uses, Calls, Compare_Resources, Input_Parameters, Output_Parameters.

For each source code description of this level a new record is created and inserted in table Level3desc with a unique code. Level1 Name is needed to tell to which software system this piece of source code being described belongs to. The name of the software system is replaced by its code sought from the Systems table, as well the code for the Level3 Name itself assigned from the Source table. The "status" field is assigned the code of the status condition entered from the form, and the "datte" field is assigned the same value entered by the user.

In table Developedby a new record is inserted for each staff member defined in this form. The code of the staff member is assigned to the table field "developedby", the field "startdate" is assigned the value the user has just entered, "tabcode" field is assigned the value of the unique code generated for Level3desc table, "flag" field is assigned the value of 'L' meaning that this recorded is inserted upon 'Level3' description and not Maintenance. "Oldverno", "newverno" are assigned the value of the Version number entered in the form, "oldrevno" and "newrevno" are assigned the value of the revision number. N.B.: more than one person may be involved in the implementation of a certain piece of source code.

Tables Purpose, Algorithm, Comments, Input_Parameters, Output_Parameters are affected the same way as table Purpose is affected in the Level2 description form, but the "tabcode" field is assigned the unique code generated for the Level3desc table, and the table field "levelflag" is assigned the value of 3 - to identify the records as belonging to Level 3.

In table Uses a new record is created for each component name entered in the corresponding block. The "tabcode" field is assigned the unique code generated for Level3desc table. The name of the component is read first and then its code is determined from the Component table assigned to the field "compcode" of the Uses table; by this it relates a component to a piece of source code by a 'Uses' relationship. Also "oldverno" and "newverno" are assigned the value of the version number just being defined, and "oldrevno" and "newrevno" are assigned the value of the revision number of the source code in question.

In table Calls two fields are the first to be assigned a value which are the "sys2code" and "mod2code". The first is the code of the software to which the 'called' source code belongs assigned from the Systems table, and “mod2code” is the code of the ‘called’ source code assigned from the Source table. The callee’s code, i.e., the code of the source code name , described in this form and generated for the Level3desc table, is assigned to the field “tabcode”. Also “oldverno” and “newverno” are assigned the value of the version number just being defined, and “oldrevno” and “newrevno” are assigned the value of the revision number of the source code in question.

For each development phase defined in this form a new record is created in the table Compare_Resources. “Tabcode” field is assigned the value of the unique code generated for the Level3desc table, “phasecode” is the code of each development phase entered by the user, “estimates” is assigned the value entered by the user - to estimate the time resources spent in each phase. The field “flag” is assigned the value of ‘L’ meaning that this record is entered upon Level3 description and not maintenance, “levelflag” is assigned the value of ‘3’ to identify that this record is for the Level3 description form - since this table is shared by all the three levels and is accessed from different forms. Also “oldverno” and “newverno” are assigned the value of the version number just being defined, and “oldrevno” and “newrevno” are assigned the value of the revision number of the source code in question. Only the estimates are entered where as the actual time resources are left till the end of each piece’s development where from the Actual Resources Recording form they are entered.

1.1.3.4 Previous Menu

a. Function

Returns control to the previous menu.

b. Tables Affected

NONE !!!!

1.1.3 Component Description and Structure

This option leads to the following menu:

1- Component Description

2- Component Structure Definition

3- Previous Menu

1.1.3.1 Component Description

a. Function

This option allows the development team to relate the already defined components to a software system as well as describing the purpose of each. The associated form with this option is the following:

(Name of the development company)

Component Definition Form

Component Type:

Level1 Name:	Version:
Component Name:	Revision:

Component created by:

Description

b. Tables Affected

The table Comp_Def is the one affected. A unique code is generated for each record defined; this code is assigned to the table field “code”. The code of the Level1 name is assigned from the Systems to the table field “syscode”. For the component type item on the screen, a pop-up window is available for the user to choose a type for the component. The equivalent field in the table “comptype” is assigned the code of the chosen component type - from the component_type table. Also the name of the owner who created the component is needed and the code is sought from the Mtceteam table and assigned to the table field “createdby”.

The table Component_Description is affected by the number of records inserted in the Description item on this form. For each line or record a unique code is generated assigned to the table field “code”, “flag” is assigned the value of ‘C’ meaning normal entry not upon maintenance, “tabcode” is assigned the value of the code generated for the Comp_Def table. Also “oldverno” and “newverno” are assigned the value of the version number just being defined, and “oldrevno” and “newrevno” are assigned the value of the revision number of the source code in question.

1.1.3.2 Component Structure Definition

a. Function

This option allows the development team to define the structure of the already defined components, as well as describing the purpose of each field in the structure. The associated form with this option is the following:

<u>(Name of the development company)</u>

<u>Component Structure Definition Form</u>

Levell Name:	Version:
Component Name:	Revision:

<u>Authorized user</u>	<u>grant type</u>

Field Name:

Field Length:

Field Type:

Field Description

b. Tables Affected

Two tables are affected: Component_Structure and Authorized_Grants. The first table will have a new record inserted for each field defined in the form and where a unique code is generated for it and assigned to the table field "fldid". "Fldname" is assigned the value of the form item Field Name, "fldtypecode" is assigned the code of the value chosen in the form item Field Type, "fldlength" is assigned the value of the form item Field Length.

"Flag" is assigned the value of 'C' meaning upon creation of the component and not modification, "tabcode" is assigned the unique value generated for the table Comp_Def. Also "oldverno" and "newverno" are assigned the value of the version number just being defined, and "oldrevno" and "newrevno" are assigned the value of the revision number of the source code in question.

In table Authorized_Grants the following fields are assigned the same value as the Component_Structure table: "oldverno", "oldrevno", "newverno", "newrevno", "flag", and "tabcode". "Staffcode" field is assigned the code of the Authorized user's name just defined through this screen and the "grantcode" field is assigned the code of the grant chosen.

1.1.4 Actual Resources Recording

a. Function

This option leads to a form that allows the developers to enter the actual time resources spent in developing a certain piece of source code in all its phases (design, coding, testing, etc.) Of course, prior to this stage, at the Level3 Description form where the developers enter only the estimated time resources. The associated form is the following:

<u>(Name of the development company)</u>

<u>Actual Resource Recording</u>

Level1 Name:	Version:
Level3 Name:	Revision:

Development Phase	Estimates	Actual
Tests Done	Tests Outcome	Date
Source Code Developed by	Start Date	Finishing Date

b. Tables Affected

Tables Compare_Resources, Tests_Outcome, and Developedby are the ones affected. This form fills up the missing information in the tables Compare_resources and Developedby, since at source code description time (Level3 Description) the other fields were filled-in, the "phasename", the "estimates", and the "startdate", while the actual resources and finishing date are kept till the end of the development of each piece of source code to actually be able to determine them. The tool will read the name of the source code in question, from the form, get its code from the Source table, and the corresponding information is sought from the Compare_Resources table, displayed, and the “actual” field is then filled-in from the form field Actual.

The second block of the form is used to allow the user define the names, outcome, and date of performing tests that assure the correctness and validation of the source code in question. Hence, for each test a new record is created in the table assigning its following fields the following values: “code” the code of the source code got from the Level3desc table, “flag” the value of ‘L’ meaning during development and not maintenance, “datte” the date entered from the form, “testcode” the code of a test name selected from the Tests table, “testout” the code of the outcome of the test selected from the Status table. Also “oldverno” and “newverno” are assigned the value of the version number just being defined, and “oldrevno” and “newrevno” are assigned the value of the revision number of the source code in question.

In table Developedby the Ending Date item on the screen will tell the date that the piece’s development is finished. It is entered in the above mentioned table as the “enddate” field.

1.1.5 Development Configuration Release

a. Function

This option leads to a form that allows the development company to produce a configuration release for a software system that has been developed and in its way to be distributed to some client institution. It has this following form:

<u>(Name of the development company)</u>

<u>Development Configuration Release</u>

Level1 Name: Version:

Configuration Release Identification: Date:

Status: Released by:

<u>Development Phase</u>	<u>Estimated Time</u>	<u>Actual Time</u>

Integration & System Tests	Tests Outcome	Date

Configuration Distributed to	Distribution Date

Unfinished Work

b. Tables Affected

Table Configrelease is affected. There is a new record created for each initiation of the form, of course with a unique code. CRID item is assigned to the field "crid" in the table, where as the generated record code goes to the "code" field. The "syscode" field has the value of the code of the Level1 Name. The "status" field is assigned the code of the value entered in the form, and "date" field is assigned the value defined in the form. "Verno" is assigned the value of the Version number defined in the form, "releasedby" is assigned the code of the staff member whose name appears in the form field Released by, "flag" is assigned the value of 'D' meaning it is a Release on Development not on Maintenance.

The Compare_Resources table is also affected. The actual resources for development for all the defined phases (in Level1 Description), for the whole software system, are defined in this form.

Table Tests_Outcome is also affected. New records are inserted for each test defined in this form stating the names of the tests performed on the whole software system to ensure that it conforms to its requirements specifications.

Table Distributed is affected as well because the above form assigns a configuration release to an institution and hence in this table the "syscode" and the "institutionid" fields are assigned respectively the values: code of the software system and code of the institution name for which the software is released. The "crid" field in this table is assigned the value of the corresponding Configrelease generated record code.

Table Unfinished_Work is also affected if and only if the user enters some text in the last block stating what unfinished work is left undone or for the next release. A unique code is generated for each line of text entered assigned to the field "code", "flag" is assigned the value of 'L' meaning unfinished work on Development, "oldverno" and "newverno" are assigned the value of the Version number of the software system defined in the form, and "tabcode" is the unique code generated for the Level1desc table to identify the software system being released in this form.

1.1.8 Previous Menu

a. Function

This option brings the user backwards to the main menu.

b. Tables Affected

NONE !!!!

1.2 The Maintenance Option

This option contains all the necessary forms needed to have a controllable framework to implement any maintenance action; starting from the Bug/Enhancement report and ending with the Maintenance Configuration Release form.

Choosing the maintenance option leads to the following menu:

1- Bug/Enhancement Report

2- Change Proposal

3- Change Approval

4- Maintenance Specification

5- Module Design

6- Source Code Implementation

7- Maintenance Configuration Release

8- Change Rejection

9- Actual Resources Recording for Maintenance of Source Code

10- Actual Resources Recording for Total Maintenance

11- Post Maintenance Release Jobs

12- Previous Menu

1.2.1 Bug/Enhancement Report

a. Function

This option allows the development team to store, into the tool's database, any user requests concerning any changes or enhancements. By this, the development company would oblige its client institutions to fill in this form - on paper - in order to assess later the request. The form associated with this step is the following:

<u>(Name of the development company)</u>

<u>Bug/Enhancement Report</u>

Level1 Name:

Version:

Report Identification:

Related Bug/Enhancement report (B/E): Date:

<u>Reported by</u>	<u>Institution</u>	<u>Employee</u>

<u>Description</u>

---

b. Tables Affected

Reports table is the one affected. For each new report received, in this form, a new record is created in the table giving each record a unique code. The Level1 Name is read to know to which software system the report belongs in order to get from the Systems table the "syscode" value that need be inserted in the Reports table. The "B" or "E" character read is used to assign it to the field "flag" in order for us to know if the report is a bug or an enhancement report. The "reportid" field in the table gets the value entered by the user, which is an identification (name) for the bug/enhancement report. The field "date" is assigned the value from the form field Date, "verno" field is assigned the value of the Version number on the form - this is needed to know in which version the error occurred. The field "reportedby" is assigned the unique code of the user in the client institution who requested the change. It is selected from the Employee table where a match is found with the name of the employee and the name of the institution as entered by the user of the form.

Table Repdesc is also affected. For each line of description in the form a record is inserted into this table assigning to it a unique code and assigning the other fields the following values: "repcode" the code generated for the Reports table to identify to which report this line of description belongs, "flag" is assigned the value of 'R' meaning that this description is defined from the Reports form.

1.2.2 Change Proposal

a. Function

This option leads to a form that is filled in by one of the maintenance team. It is a more "technical" version of the bug/enhancement report. The associated form is the following:

(Name of the Development company)

Change Proposal Form

Change Proposal Identification: Date:

CP status:

Related Bug/enhancement report ID: Version:

CP Proposed by:

CP description

Reason for change

b. Tables Affected

Table Change_Proposal is the first table to be affected by the initiation of this form. The user has to give a name or identification to the change proposal to identify it later on, and it is assigned to the field "Cpid". The tool, on the other hand, generates a unique code for the record just created for the new proposal and it is assigned to the table field "code". The other table fields are assigned the following values: "datte" the date entered by the user in the form, "status" the code selected from the Status table where the name is equivalent to what the user has entered, "proposedby" the code of the staff member selected from the Mtceteam table where the name is equivalent to the name defined in the form as to whom proposed the change, "verno" is assigned the value of the Version number and "reportid" is assigned the code selected from the Reports table where the report name is equivalent to the one just defined by the user as Related report. "Approved" is left empty till a Change Approval is done then it is flagged 'Y' or a Change Rejection is done and then it is flagged 'N', "crid" is also left blank until a configuration is released that includes this change proposal and hence the configuration release code is assigned to it.

Table Repdesc is affected by the text entered in the Change Proposal Description block. For each line a new record is inserted - with a unique code - in the Repdesc table same as in the Reports form except that the flag is set to 'C' in this form to distinguish the lines of text of a bug report from those of a change proposal.

Table Change_Reason is also affected by the text entered in the Reason for Change block. For each line a new record is inserted - with a unique code - in the Change_Reason table with its fields having the following values: "cpcode" the code of the change proposal generated previously in this form for the Change_Proposal table, "name" is assigned the text entered in this block.

1.2.3 Change Approval

a. Function

This option leads to a form initiated only in the case a change proposal has been studied and approved. The form associated with this option is the following:

(Name of the development company)

Change Approval Form

Change Approval Identification: Date:

Related Change Proposal Identification: Version:

CA Authorized by:

CA Baseline Established by:

CA Status:

Type of Change:

Identification of Change

Priority of Implementation:

Consequences if not Implemented

Define the Involved Software

Name	Revision Number

b. Tables Affected

Table Change_Approval is the first table to be affected. The user chooses an identification name for the approval form and the tool generates a unique code for each record. This code is assigned to the primary key "code" of the Change_Approval table, while the "caid" field is assigned the value of the identification entered and chosen by the user. The "authorized" field is assigned the code of the name of the authorized person of the maintenance team that has approved the change. The "status" field is also assigned the status condition code entered by the user of the form about the status of the change approval. "Cpid" field is assigned the code of the change proposal identification chosen by the user as the Related CP. "Datte" is assigned the Date value entered by the user in the form, "baselineby" field is assigned the code of the staff member who established the baseline, "typeofchange" field is assigned the code of the maintenance type chosen by the user, "verno" is assigned the Version number selected in the form.

Since many change approvals may lead to only one maintenance specification, hence the field “msid” will be left null at this stage waiting for the corresponding Maintenance Specification step to be taken in order to be able to assign a value to this field.

Table Idofchange is affected by the text entered in the Identification of Change block. For each line of text a new record is inserted in the table with a unique code given to the record and with its fields assigned the following values: “code” the unique code generated for this table, “flag” is assigned the value of ’C’ meaning this identification belongs to a Change Approval, “tabcode” is assigned the code generated for the Change Approval table, and “name” is assigned a line of text from the block.

Table Priority is affected by the text entered in the Priority of Implementation block. For each line of text a new record is inserted in the table with a unique code given to the record and with its fields assigned the following values: “code” the unique code generated for this table, “flag” is assigned the value of ’C’ meaning this identification belongs to a Change Approval, “tabcode” is assigned the code generated for the Change Approval table, and “name” is assigned a line of text from the block.

Table Consequences is affected by the text entered in the Consequences if not Implemented block. For each line of text a new record is inserted in the table with a unique code given to the record and with its fields assigned the following values: “code” the unique code generated for this table, “flag” is assigned the value of ’C’ meaning this identification belongs to a Change Approval, “tabcode” is assigned the code generated for the Change Approval table, and “name” is assigned a line of text from the block.

Table Swinvolved is also affected and in this way: For each change approval record created in the Change_Approval table, a new record is created as well in this table assigning as its record code the change approval record code. For each source code involved the tool will search in the source table to get the source code piece “modcode”, the field “verno” is assigned the Version number value defined in the form, and the field “revno” is assigned the Revision number of the source code piece, also defined by the user in the form.

1.2.4 Maintenance Specification

a. Function

This option leads to a form that allows the authorized maintainer to enter the maintenance specifications needed by a maintenance action and that might include more than one approved change proposal. The associated form is the following:

(Name of the development company)

Maintenance Specifications Form

Maintenance Specification Identification: Date:

MS Formulated by:

MS Baseline Established by:

MS Status: Version

Related Change Approval Identification: ----------------- ---------- ---------

---

Identification of change

Consequences of the Change

Define the Affected Software

Flag	Source Code Name	Revision

Tests Required

Estimated Resources for the Change

Phase Name	Estimates

b. Tables Affected

The following tables are affected: Mtcespecs, Change_Approval, Idofchange, Consequence, Swaffected, Tests_Outcome, and Compare_Resources. In table Mtcespecs a new record is created for each specification entered; i.e., for each initiation of the above form. The name of the form entered by the user is assigned to the field "msid" in the table, while the tool will have generated a unique code for the record assigned to the table field "code". Formulated by item in the form is used to state the name of the person from the development company that has formulated the specifications. Its code is selected from the table Mtceteam and assigned to the table field "formulatedby". The Date item on the form is assigned to the table field "datte"; Status item on the form is the status condition of the specification where its code is selected from the Status table and assigned to the table field "status". 'Baseline Established by' item on the form is the name of the staff member who initiated the baseline, in preparation to implement the specifications, where its code is selected from the Mtceteam table. The table field "verno" is assigned the value of the Version number found in the form.

Change_Approval is affected only in the following manner: For each change approval identification defined in the above form as related to this maintenance specification its table field "msid" - which was left empty on inserting the record on Change Approval step - is assigned the code generated for the table Mtcespecs above. By this the m-to-1 relation is done between a change approval step (Change Approval Form) and a maintenance specification that implements the approved change (Maintenance Specification Form).

Idofchange table is used to store the text entered by the user in the corresponding item on the above form. A unique code is generated for each line of text inserted in the table and assigned to the table field "code". The field "flag" is assigned the value of 'M' meaning these are the change identifications of the Maintenance Specifications and not for the Change Approval. When more than one form is inserting to the table a flag is needed to differentiate between the texts of each form. "Tabcode" field is assigned the value of the generated code for the Mtcespecs table. Of course, the field "name" is assigned the value of the text entered.

The table Consequence is affected in the same way as the Idofchange table. This tables' fields are assigned the same values as the above table in the exception for the "name" field where it contains the text entered in the Consequence of the changes item on the form.

Table Swaffected is also affected in the following manner. The table field "msid" is assigned the code generated for the new record of Mtcespecs and that is to relate a maintenance specifications to the pieces of software that are affected by this change. The second field is the "modcode" that is assigned the code of the source code name defined as affected by the user. The "flag" field is assigned a value of either 'S' or 'C', respectively according to the user's choice ('S' if the affected is source code and 'C' if the affected is a component).

Table Tests_Outcome is affected in the following manner: for each test name entered as required the tool will create a record in the former table with the table fields assigned the following values: "mdcode" is assigned the code generated for the table Mtcespecs, "testcode" is assigned the code of the test name selected from the Tests table, "testout" is assigned the code of the test outcome selected from the Status table, "datte" is assigned the Date value found on the form, “flag” is assigned the value of ‘S’ meaning that the record belongs to a Maintenance Specification form, “levelflag” is assigned the value of zero because this form talks about the whole maintenance specification and not about a specific level source code. Also “oldverno” and “newverno” are assigned the value of the version number just being defined, and “oldrevno” and “newrevno” are assigned the value of the revision number of the source code in question. The field “code” is left blank because it is accessed only from the Development option of this tool, while the Maintenance option deals with the “mdcode”.

Finally, table Compare_Resources is affected in the following manner. For each maintenance phase entered by the user as a step towards implementing the change, a record is inserted in the table with its fields assigned the following values: “phasecode” the code of the phase name defined by the user selected from the Develophases table, “estimates” the values entered by the user on the form, “levelflag” is assigned the value of zero, “mdcode” is assigned the value of the code generated for the Mtcespecs table, “oldverno” and “newverno” are assigned the version number found on the form, “flag” is assigned the value of ‘R’ to identify it as belonging to a maintenance specification. The “flag”, “levelflag”, and “mdcode” identifies the records as belonging to this specific maintenance specification identification.

1.2.5 Module Design

a. Function

This option leads to a screen (or form) that allows the maintainer to enter the "new" designs of a piece of source code (or even a component). The following form is the one associated with this option:

(Name of the development company)

Module Design Form

Level1 Name:	Version:
Is Second Level Source or Component (S/C):	Revision:
Second Level Name (if available):	

Module Design Identification:	Date:
Module Design Designed by:	
Module Design Baseline Established by:	
Module Design Status:	
Related Maintenance Specification Identification:	

Is the Related Design for a Perfective Maintenance:

Press PageDown to move to the corresponding Level Description where you can define the new designs for the affected software piece you have just identified in this form.

b. Tables Affected

Table Module_design is the only one affected. For each initiation of this form, a new record is created in the former table with its fields assigned the following values: "code" the unique code generated by the form that uniquely identifies the record in the table, "mdid" the name defined by the user in the form field Module Design Identification, "datte" the value of the Date defined in the form, "status" the code of the status condition defined by the user in this form - selected from the Status table, "msid" the code selected from the Mtcespecs where the table field "msid" is equivalent to the name identified by the user as the Related Maintenance Specification Identification, "flag" the value entered by the user - 'S' if source code, 'C' if component, and null if the Module Design's purpose is to re-design the whole software system, "oldverno" the Version number found in the form.

Also "oldverno" and "newverno" the value of the version number just being defined, and "oldrevno" and "newrevno" the value of the revision number of the source code in question, "designedby" the code of the staff member who designed the change selected from the Mtceteam table, "Baselineby" the code of the staff member who prepared the baseline selected from the Mtceteam table, "levelid" the unique code that identifies a certain piece of source code, "levelflag" either of the values [0,1,2,3]; Zero if the module to be designed is a component, the other values represent the Levels of the source code which means the tool will be able to get the level of the source code in question.

1.2.5.1 New Design Documents for Level1

a. Function

This form is called if the source code module to be re-designed is simply the whole software system. The associated form is:

<u>(Name of the development company)</u>

<u>New Design Documents for Level1</u>

Level1 Name: Version:

Module Design Identification:

Nature

Hardware Requirements

Constraints

Estimated Time Resources

Maintenance Phase Name	Estimates

Tests to be Performed

b. Tables Affected

This form affects five tables: Nature, Hardware_Requirements, Constraints, Compare_Resources, and Tests_Outcome.

Table Compare_Resources is affected as well. There are going to be as many records for each Level1 description as there are defined development phases; For each record created the "levelflag" field is assigned the value of "1", which is the current level number. "Flag" field is assigned the value of 'D' meaning Description on maintenance. The "mdcode" field is assigned unique code that identifies the module design record which includes the name of the software system. The "phasecode" field is assigned the value of the corresponding development phase (e.g., "1" for design, "2" for coding", and "3" for testing, each constitute a record on its own.) The "estimates" field is assigned the values entered from the screen for each phase. "Oldverno", "newverno" are given the value of the Version number defined in this form. All Level1 items do not have Revision numbers - only version numbers - hence, "oldrevno" and "newrevno" are not defined in this context. "Actual" field is left empty to be filled out at the end of the development through the configuration release form.

Tables Nature, Hardware_Requirements, and Constraints are affected in the same way. For each line of text entered in one of these blocks on the form, a new record is inserted in the corresponding table with the generation of a unique record code number assigned to the field "code". The field "name" is assigned a line of text, "flag" is assigned the value of 'D' meaning on maintenance of Level1, "oldverno" and "newverno" are assigned the same value of the Version number found on the screen, "mdcode" is assigned the unique code that identifies the module design record which includes the name of the software system.

Table Tests_Outcome is also affected. New records are inserted for each test defined in this form stating the names of the tests performed on the whole software system to ensure that it conforms to its New requirement specifications.

1.2.5.2 New Design Documents for Level2

a. Function

This form is called if the source code module to be re-designed is a functional module; i.e., of Level 2. The associated form is:

(Name of the development company)

New Design Documents for Functional Modules

Level1 Name:	Version:
Level2 Name:	Revision:
Module Design Identification:	

Purpose

Description

---

b. Tables Affected

This form affects two tables: Purpose and Level2_Desc.

For each line of text entered in the Purpose block a new record is inserted in the Purpose table with a unique code and where the "mdcode" field is assigned the unique code that identifies the module design record which includes the name of the source code module. The field “name” gets a line of text, “code” gets the generated record code number, “levelflag” gets the value of 2, “flag” gets the value of ‘D’ meaning that the record belongs to a maintenance Module Design. “Oldverno”, “newverno” are assigned the value of the Version number entered in the form, “oldrevno” and “newrevno” are assigned the value of the revision number.

The third table affected is the Description table where the “description” item on the screen is also a text entered and inserted as records in the corresponding table. Each record is given a unique code assigned to the table field “code”, the field “name” gets a line of text, and the field “mdcode” is assigned the value of the unique code that identifies the module design record which includes the name of the source code module. The table field “flag” is assigned the value of ‘D’ meaning that the record belongs to a maintenance Module Design. “Oldverno”, “newverno” are assigned the value of the Version number entered in the form, “oldrevno” and “newrevno” are assigned the value of the revision number.

1.2.5.3 New Designs Documents for Level 3

a. Function

This form is called if the source code module to be re-designed is of Level 3. The associated form is:

<u>(Name of the development company)</u>

<u>New Designs for Source Code Modules</u>

Level1 Name:	Version:
Level3 Name:	Revision:

Module Design Identification:

<u>Purpose</u>

<u>Algorithm Description</u>

Maintained by: Starting Date:

Interface Definitions

Uses(Components)	Calls(Source Code)	
	Level1 Name	Source Code Name

Resource Estimates for Maintenance

Maintenance Phase Name	Time Estimates

Input Parameters

Output Parameters

Comments

b. Tables Affected

The following tables are all affected: Purpose, Algorithm, Developedby, Uses, Calls, Compare_Resources, Input_Parameters, Output_Parameters, Comments.

Tables Purpose, Algorithm, Comments, Input_Parameters, Output_Parameters are affected in the same way. For each line of text entered in the corresponding blocks, a new record is inserted in the tables with a unique code and where the "mdcode" field is assigned the unique code that identifies the module design record which includes the name of the source code module. The field "name" gets a line of text, "code" gets the generated record code number, "levelflag" gets the value of 3, "flag" gets the value of 'D' meaning that the record belongs to a maintenance Module Design. "Oldverno", "newverno" are assigned the value of the Version number entered in the form, "oldrevno" and "newrevno" are assigned the value of the revision number.

In table Developedby a new record is inserted for each staff member defined in this form. The code of the staff member is assigned to the table field "developedby" selected from the table Mtceteam, the field "startdate" is assigned the value the user has just entered, "mdcode" field is assigned the value of the unique code generated for Module_Design table, "flag" field is assigned the value of 'D' meaning that this recorded is inserted upon

Module Design for Level3 description. “Oldverno”, “newverno” are assigned the value of the Version number entered in the form, “oldrevno” and “newrevno” are assigned the value of the revision number. N.B.: more than one person may be involved in the implementation of a change of a certain piece of source code.

In table Uses a new record is created for each component name entered in the corresponding block. The “mdcode” field is assigned the unique code generated for the Module_Design table. The name of the component is read first and then its code is selected from the Component table assigned to the field “compcode” of the Uses table; by this it relates a component to a piece of source code by a ‘Uses’ relationship. Also “oldverno” and “newverno” are assigned the value of the version number just being defined, and “oldrevno” and “newrevno” are assigned the value of the revision number of the source code in question.

In table Calls two fields are the first to be assigned a value which are the “sys2code” and “mod2code”. The first is the code of the software to which the ‘called’ source code belongs assigned from the Systems table, and “mod2code” is the code of the ‘called’ source code assigned from the Source table. The table field “mdcode” is assigned the unique code generated for the Module_Design table. Also “oldverno” and “newverno” are assigned the value of the version number just being defined, and “oldrevno” and “newrevno” are assigned the value of the revision number of the source code in question.

For each development phase defined in this form - as a needed step in implementing the maintenance action - a new record is created in the table Compare_Resources. “Mdcode” field is assigned the value of the unique code generated for the Module_Design table, “phasecode” is the code of each development phase entered by the user, “estimates” is assigned the value entered by the user - to estimate the time resources spent in each phase. The field “flag” is assigned the value of ‘D’ meaning that this record is entered upon Module Design (maintenance), “levelflag” is assigned the value of ‘3’ to identify that this record is for source code of Level3 - since this table is shared by all the three levels and is accessed from different forms. Also “oldverno” and “newverno” are assigned the value of the version number just being defined, and “oldrevno” and “newrevno” are assigned the value of the revision number of the source code in question. Only the estimates are entered where as the actual time resources are left till the end of the piece’s maintenance where from the Maintenance Actual Resources Recording Form they are entered.

1.2.5.4 New Design Documents for Components

a. Function

This form is called if the item to be re-designed is a component . The associated form is:

<u>(Name of the development company)</u>

<u>New Design Documents for Components</u>

Level1 Name:	Version:
Component Name:	Revision:

Module Design Identification:

Component Description:

<u>Authorized user</u>	<u>grant type</u>

Field Name:

Field Length:

Field Type:

Field Description

---

b. Tables Affected

Three tables are affected: Component_Description, Component_Structure, and Authorized_Grants. The table Component_Description is affected by the number of text lines inserted in the Description item on this form. For each line of text a record is created in the table with a unique record code generated and assigned to the table field "code", "flag" is assigned the value of 'D' meaning text entry upon Module Design , "mdcode" is assigned the unique code generated for Module_Design table. Also "oldverno" and "newverno" are assigned the value of the version number just being defined, and "oldrevno" and "newrevno" are assigned the value of the revision number of the source code in question.

The second table - Component_Structure - will have a new record inserted for each field defined in the form and where a unique code is generated for the field and assigned to the table field “fldid”. “Fldname” is assigned the value of the form item Field Name, “fldtypecode” is assigned the code of the value chosen in the form item Field Type, “fldlength” is assigned the value of the form item Field Length. “Flag” is assigned the value of ‘D’ meaning upon Module Design (modification) of the component, “mdcode” is assigned the unique value generated for the table Module_Design. Also “oldverno” and “newverno” are assigned the value of the version number just being defined, and “oldrevno” and “newrevno” are assigned the value of the revision number of the source code in question.

In table Authorized_Grants the following fields are assigned the same value as the Component_Structure table: “oldverno”, “oldrevno”, “newverno”, “newrevno”, “flag”, and “mdcode”. “Staffcode” field is assigned the code of the Authorized user’s name just defined through this screen and the “grantcode” field is assigned the code of the grant chosen.

1.2.6 Source Code Implementation

a. Function

This option also leads to a form that allows the maintainer from entering the details of the implemented changes. The associated form is the following one:

<u>(Name of the development company)</u>

<u>Source Code Implementation Form</u>

Source Code Identification: Date:

Source Code Status:

Source Code Implemented by:

Source Code Understood by:

Source Code Baseline by:

Module Design Identification: Ver#: Rev#:

<u>Tests Performed</u>	<u>Tests Outcome</u>	<u>Date</u>

Comments

---

b. Tables Affected

The following tables are affected: Source_Code, Tests_Outcome, and Comments. For each initiation of this form a new record is created in the Source_Code table with a unique code number and with its fields assigned the following values: "scid" the source code identification defined by the user, "datte" the value of the Date item in the form, "status" the code of the status condition selected from the Status table, "implementedby" and "understoodby" the codes of the staff members who respectively, implemented and understood the implemented changes. These codes are selected from the Mtceteam table. Also "oldverno" and "newverno" are assigned the value of the version number just being defined, and "oldrevno" and "newrevno" the value of the revision number of the source code in question.

Since there exists a 1-to-1 relationship between Source_Code and Module_Design tables, the user is prompted to enter or choose - from a pop up menu - a module design identification that is related to the current source code implementation. The chosen identification's code - selected from the Module_Design table - is assigned to the foreign key "mdid" in the Source_Code table.

In table Tests_Outcome, the tests declared as required in the related Module Design form are displayed in this form where the user is requested to enter the outcome and date of each test. Hence, for the corresponding records in the Tests_Outcome table the corresponding fields are updated: "testout" is assigned the value of the status condition selected from the Status table, and "datte" is assigned the Date form field defined by the user.

For each line of text entered in the Comments form field, a new record is created in the Comments table generating a unique code for each and assigned to the table field "code". The table field "name" is assigned a line of text, "flag" is assigned the value of 'S' meaning the record is related to the Source_code table whose code is assigned to the table field "mdcode", "oldverno" and "newverno" are assigned the Version number found in the form, and "oldrevno" and "newrevno" are assigned the value of the source code's Revision number also found in the form.

1.2.7 Maintenance Configuration Release

a. Function

This option leads to a form that allows the maintainer from producing a maintenance configuration release after implementing the proposed and approved changes. It is up to the user to decide if a new release number is required. The associated form with this option is the following screen:

<u>(Name of the development company)</u>

<u>Maintenance Configuration Release Form</u>

Level1 Name:	Version Number:
Configuration Release Identification:	Date:

New Version Number:

Confirm the Process (Y/N):

Configuration Release Status:

Configuration Released by:

Baseline Established by:

Related Change Proposals:

Configuration Distributed to

Institution Name	Distributed Date

Actual Time Resources Recording

Maintenance Phase	Estimates	Actual

Tests Performed	Tests Outcome	Date

Unfinished Work

b. Tables Affected

The following tables are affected: Configrelease, Versions, Change_Proposal, Compare_Resources, Tests_Outcome, and Unfinished_Work.

In table Configrelease the same process occurs as in the Development Configuration Release form except for assigning the table field "flag" the value of 'M' meaning a release upon Maintenance.

The form field Confirmed should contain either characters 'Y' or 'N' depending on the user's choice. If the answer is 'Y' then the software system is given a new number and hence the Versions table is updated as follows: first, in the record where the field "syscode" is equal to the software code being released and "oldver" equal to the old version number found in the form, the table field "newver" is assigned the new version number generated by the form - increasing the old version number by one - and the table field "crid" is assigned the value of the unique code generated for the record of the table Configrelease. Second, a new record is inserted with the table field "syscode" assigned the same value of the software code and the field "oldver" assigned the new version number just given to the software system.

Change_Proposal table is only updated by assigning to the table field “crid”, where the field “code” is equal to the code of the change proposal identifications declared in the form as included in the release, the value of the record created in the Configrelease table.

Table Distributed is also affected by creating a new record for each institution the new release is distributed to. Its fields as assigned the following values: “configcode” the value of the code generated for the record for the table Configrelease, “instid” the code value of the institution name selected from the table Institutions, “flag” the value ‘M’ meaning distributed upon a maintenance release, “verno” the new version number of the software system, and “datte” the Date of distribution.

Compare_Resources and Tests_Outcome tables are only affected if and only if the set of the maintenance changes include the re-development of the whole software system; i.e., in the Module Design Form the name of the software system was mentioned and a new definition of it was declared through the next form called ‘New Design Documents for Level1’. Otherwise, the corresponding two blocks are skipped. If they are not skipped, then the already entered fields in the latter form are displayed and only the Actual time resources are entered - for the Estimated time resources block - and the Tests outcome and date for the second block.

Table Unfinished_Work is affected if any text is entered in the corresponding form block. For each line of text a new record is inserted in the table with the its fields are assigned the following values: “code” the unique code generated for each record, “name” the line of text, “flag” the value ‘D’ meaning a maintenance unfinished work, “oldverno” and “newverno” the old version number of the software system - and not the new one, and “tabcode” the code of the configuration release record.

1.2.8 Change Rejection

a. Function

This form allows the user to enter a rejection action of a proposed change request. The associated form is the following:

(Name of the development company)

Change Rejection Form

Change Rejection Identification: Date:

Change Rejection Authorized by:

Change Rejection Status:

Related Change Proposal:

Reason for Rejection

---

b. Tables Affected

Tables Change_Rejection, Change_Proposal, and Rejection_Reason are affected. For each initiation of this form a new record is created in table Change_Rejection with a unique code number. The user should assign a name for the change rejection that is assigned in turn to the table field "crjid". "Authorizedby" field is assigned the value of the code of the authorized staff member, of the maintenance team that has evaluated the proposal and decided on the rejection action, whose name is entered in the form field 'Change Rejection Authorized by'. "Status" field is assigned the code of the 'Change Rejection Status' field on the screen, the "date" as well is assigned the form field 'Date'. The record in the table Change_Proposal, where the name of the change proposal identification is equal to the one assigned by the user as a related one, is updated by assigning an 'N' to the "approved" flag; i.e., this change proposal is not approved.

For each line of text entered in the Reason for Rejection block on the screen, a record is inserted in the Rejection_Reason table with a unique code generated for the record and assigned to the table field "code". The field "crjid" is assigned the code of the change rejection record just generated for the Change_Rejection table, and "name" is assigned a line of text.

1.2.9 Actual Resources Recorded During Maintenance of Source Code

a. Function

This option leads to a form that allows the maintainers to enter the actual time resources spent in maintaining a certain piece of source code in all its phases (design, coding, testing, etc.) Of course, prior to this stage, at the Module Design phase the maintainers should have entered only the estimated time resources for the maintenance. The associated form is the following:

(Name of the development company)

Actual Resource Recorded During Maintenance

Level1 Name:

Level3 Name:		
Version:	Revision:	
Module Design Identification:		
Maintenance Phase Name	Estimates	Actual
Maintained by	Start Date	Finishing Date

b. Tables Affected

Tables Compare_Resources and Developedby are the ones affected. This form fills up the missing information in the tables Compare_Resources and Developedby, since at Module Design Re-Description time the other fields were filled-in, the "phasename", the "estimates", and the "startdate", while the actual resources and finishing date are kept till the end of the maintenance of each piece of source code to actually be able to determine them. The tool will read the name of the source code in question, from the form, get its code from the Source table, and the corresponding information is sought from the

Compare_Resources table, displayed, and the "actual" field is then filled-in from the form field Actual.

In table Developedby the Ending Date item on the screen tells the finishing date of implementing a maintenance action on a piece of source code.

1.2.10 Actual Time Resources Recorded for Total Maintenance

a. Function

This option leads to a form that allows the maintainer from entering the actual time resources spent in implementing a specific Maintenance Specification, as well as specifying the outcome of the complete set of tests performed on the implemented changes - previously defined the Maintenance Specification Form - and the date of performing each test. The associated form with this option is the following:

(Name of the development company)

Total Time Resources and Tests' Outcome

Maintenance Specification Identification:

Maintenance Phase Name:	Estimates	Actual

Tests Performed	Tests' Outcome	Date

b. Tables Affected

The following two tables are affected: Compare_Resources and Tests_Outcome. In the first table, the maintenance phase names and estimates are already entered from the Maintenance Specification Form, and hence the maintainer here has to only record the actual time taken during each phase.
The same is true for the Tests_Outcome table where the names of the tests are already defined from the Maintenance Specification Form and here the maintainer has to record only the outcome of each test as well as the date of performing each.

1.2.11 Post Maintenance Release Jobs

This option leads to the following menu:

1- Post Maintenance Release Jobs

2- Add a Source Code/Component to a Software Version

3- Remove a Source Code/Component from a Software Version

4- Previous Menu

1.2.11.1 Post Maintenance Release Jobs

a. Function

This option leads the user to a form that must be initiated only after a maintenance configuration release. The function of this form is to update all the tables, related to a certain release, to be related to the new version number release. The associated form is the following:

<u>(Name of the development company)</u>

<u>Post Release Jobs</u>

Configuration Release Identification:

Level1 Name:

Version:

Source Code & Components have been assigned to the new version:

Delete any Source Code from the New Version, Give Name:

Delete any Component from the New Version, Give Name:

Do You Confirm the Process of Updating all the Tables:

b. Tables Affected

All the tool's data base tables are affected. The same process goes for all, and it is the following: For every source code (component) affected by a maintenance change - just being implemented for the latest release - its table fields "newverno" and "newrevno" are incremented by one. Hence, the "newverno" will be equal to the new version number of the software and the field "newrevno" will be increased by one declaring that the source code (component) has been changed in the course of a maintenance action.

For those source code modules and components not affected, their “newverno” is only increased by one, retaining their old “newrevno”. This means that they have not been affected by the latest change but that they now belong to the version number release.

Just in case the maintainer feels that one of the source code or components need not be included in the new version release, then the user has to give the names and the tool will delete the corresponding records from the Source_Version and Component_Version. This step is performed before the final update of the complete set of the tool’s tables. If the given names are affected by the latest changes then they are not deleted and a message is printed to notify the user.

1.2.11.2 Add a Source Code Module/Component to a Software Version

a. Function

This option leads to a form where in some cases the maintainer or developer feels the need to relate a source code module or a component to a specific version - originally not related to. The associated form with this option is the following form:

(Name of the development company)

Add a Source Code Module/Component

Software Name:	Version:
Source Code Module or Component (S/C):	
Source Code/Component:	Revision:

b. Tables Affected

Either tables are affected: Source_Version and Component_Version. If the item to be added is a source code then the former table is affected, otherwise - if a component - then the latter table is affected.

For either table affected the same process occurs. For each item to be added, its code is selected from the Source (Component) table and is assigned to the table field "modcode" ("compcode"). The table field "syscode" is assigned the software code selected from the Systems table.

Also the table field "verno" is assigned the New version number - to which the source code (component) is being related - and which is entered by the user in the form. The same for the table field "revno" which is the revision number of the source code (component) - as well entered by the user in the form. Hence, creating a new record for the item just being added in the corresponding table.

1.2.11.3 Remove a Source Code Module/Component from a Software Version

a. Function

This option leads to a form where in some cases the maintainer or developer feels the need to remove a source code module or a component from a specific version - originally related to. The associated form with this option is the following form:

(Name of the development company)

Remove a Source Code Module/Component

Software Name:	Version:
Source Code Module or Component (S/C):	
Source Code/Component:	Revision:

b. Tables Affected

Either tables are affected: Source_Version and Component_Version. If the item to be removed is a source code then the former table is affected, otherwise - if a component - then the latter table is affected.

For either table affected the same process occurs. For each item to be removed, its code is selected from the Source (Component) table. The software code is also selected from the Systems table. The tool hence deletes from the Source_Version (Component_Version) table the record where the table field "syscode" is equivalent to the software's code and "modcode" ("compcode") is equivalent to the source code's code (component's), and "verno" equals to the form field Version and "revno" equals to the form field Revision. By this the source code (component) will no longer be related to the software name of this specific Version number.

1.2.12 Previous Menu

a. Function

This option leads us back to the main menu.

b. Tables Affected

NONE !!!!

1.3 The Reports Option

Choosing this option leads to the following menu:

1- Development Reports

2- Maintenance Reports

3- Previous Menu

1.3.1 The Development Reports Option

This option states the reports that can be drawn from the Development Phase. Such reports are found in the following menu:

1- Code Tables Reports

2- Source Code Related Reports

3- Components Related Reports

4- Software System Related Reports

5- Previous Menu

1.3.1.1 Code Tables Reports

This option leads to another menu where the following set of reports are generated:

1- Software Names

2- Source Code Module Names

3- Component Names

4- Nesting Levels

5- Staff Member Names

6- Development Phases Names

7- Status Conditions

8- Test Names

9- Component Types

10- Field Types

11- Maintenance Types

12- Grant Types

13- Employees Names in Client Institutions

14- Client Institutions Names

15- Hierarchy of a Software

16- Institutions Distributed to per Software

17- Previous Menu

1.3.1.2 Source Code Related Reports

This option produces reports related to source code modules. It leads to the following menu:

1- Source Code Names per Software

2- Maximum Revision Number of a Source Code

3- Revision Numbers of all Source Code Modules

4- List of Test Names Performed on a Source Code

5- Time Resources Spent on Developing a Source Code

6- List of I/O Parameters Used by a Source Code

7- List of Source Code Names Called by a Source Code

8- List of Components Used by a Source Code

9- Names of Staff Members who Developed a Source Code

10- Previous Menu

1.3.1.3 Components Related Reports

This option also leads to another menu where reports related to components are produced.

1- Names of Components per Software

2- Maximum Revision Number of a Component

3- Revision Numbers of all Components

4- List of Authorized Grants for a User on a Component

5- List of the Structure of a Component

6- Previous Menu

1.3.1.4 Software System Related Reports

This option also leads to another menu where reports related to a software system are produced.

1- Maximum Version Number of a Software System

2- Version Numbers of all Software Systems

3- List of Tests Names Performed on a Software System

4- Time Resources Taken to Develop a Software System

5- List of Software Systems Written in a Specific Language

6- Previous Menu

1.3.1.5 Previous Menu

This option leads us back to the Reports main menu.

1.3.2 The Maintenance Reports Option

This option states the reports that can be drawn from the Maintenance Phase. Such reports are found in the following menu:

1- Source Code Related Reports

2- Components Related Reports

3- Maintenance Action Related Reports

4- Previous Menu

1.3.2.1 Source Code Related Reports

This option leads to the following menu:

1- Names of Source Codes Affected by a Maintenance Action

2- Names of Staff Maintaining a Source Code

3- New I/O Parameters for a Source Code

4- New Algorithm for a Source Code

5- New Components Used by a Source Code

6- New Source Codes Called by a Source Code

7- List of Module Regression Tests Performed on a Source Code

8- Time Resources Spent on Maintaining a Source Code

9- Name of Source Code with Highest Number of Modifications

10- Previous Menu

1.3.2.2 Components Related Reports

This option leads to another menu where components related reports can be drawn.

1- Names of Affected Components by a Maintenance Action

2- Name of the Component with Highest Number of Modifications

3- Previous Menu

1.3.2.3 Maintenance Action Related Reports

This option leads to another menu where reports about any maintenance change can be drawn.

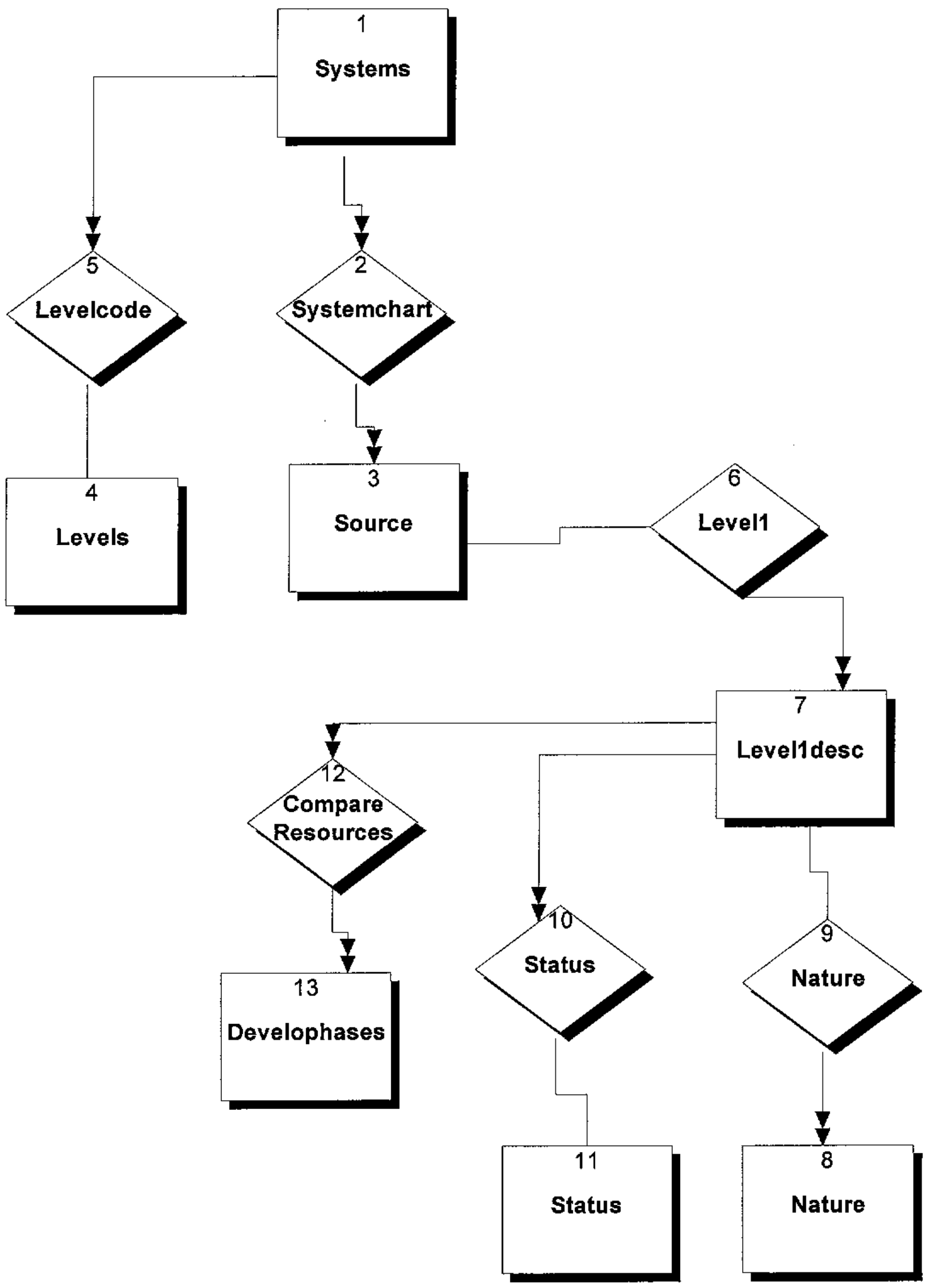
1- Names & Number of Bug Reports per Software
2- Total Number & Names of Change Proposals
3- Change Proposals not Released Yet
4- Approved Change Proposals without Specifications
5- Rejected Change Proposals
6- List of all Change Rejections
7- Time Resources Taken to Perform a Maintenance Action
8- Names of Tests Performed during a Maintenance Action
9- Number of Change Proposals per Maintenance Category
10- Previous Menu

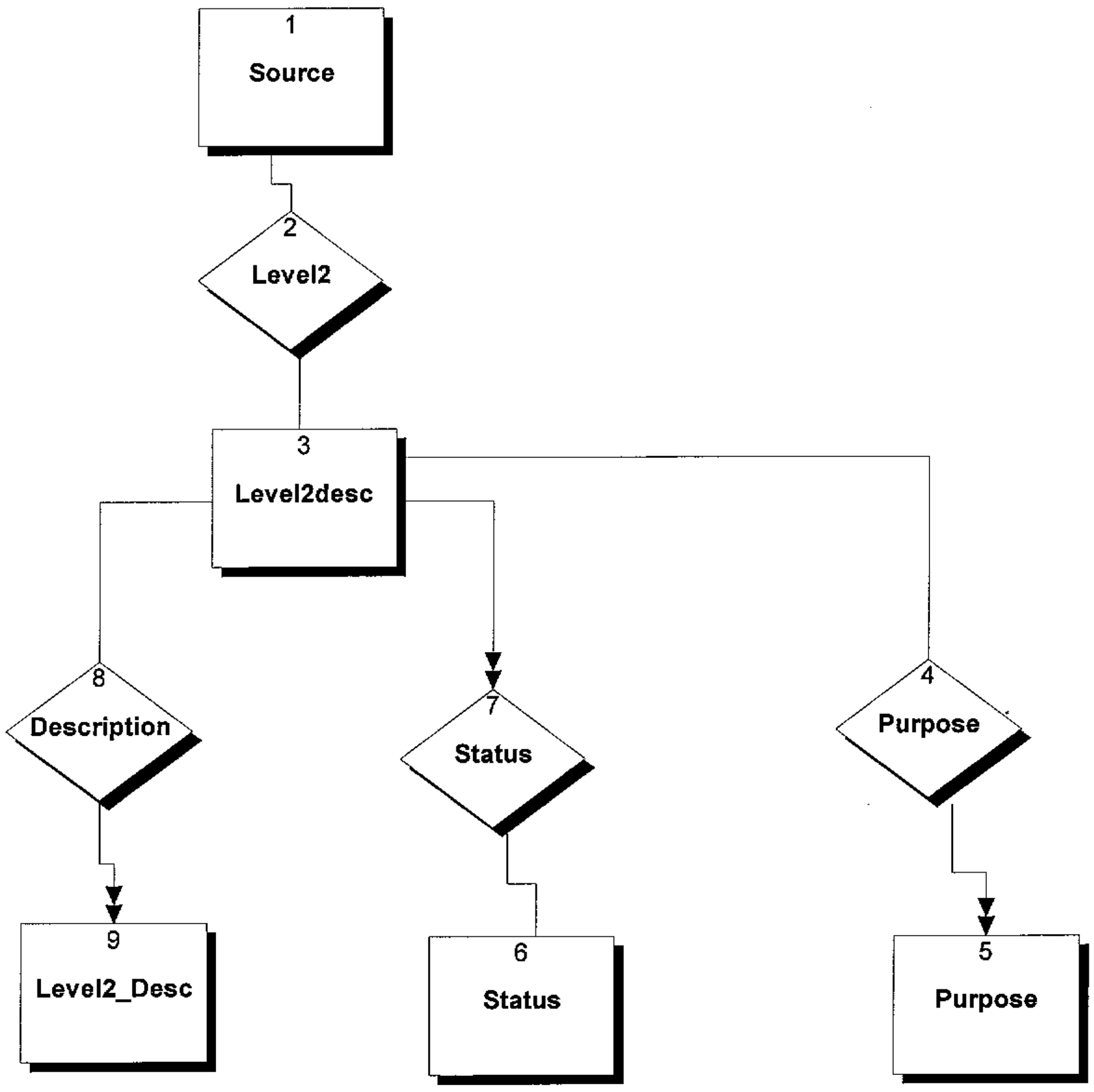
1.3.2.4 Previous Menu

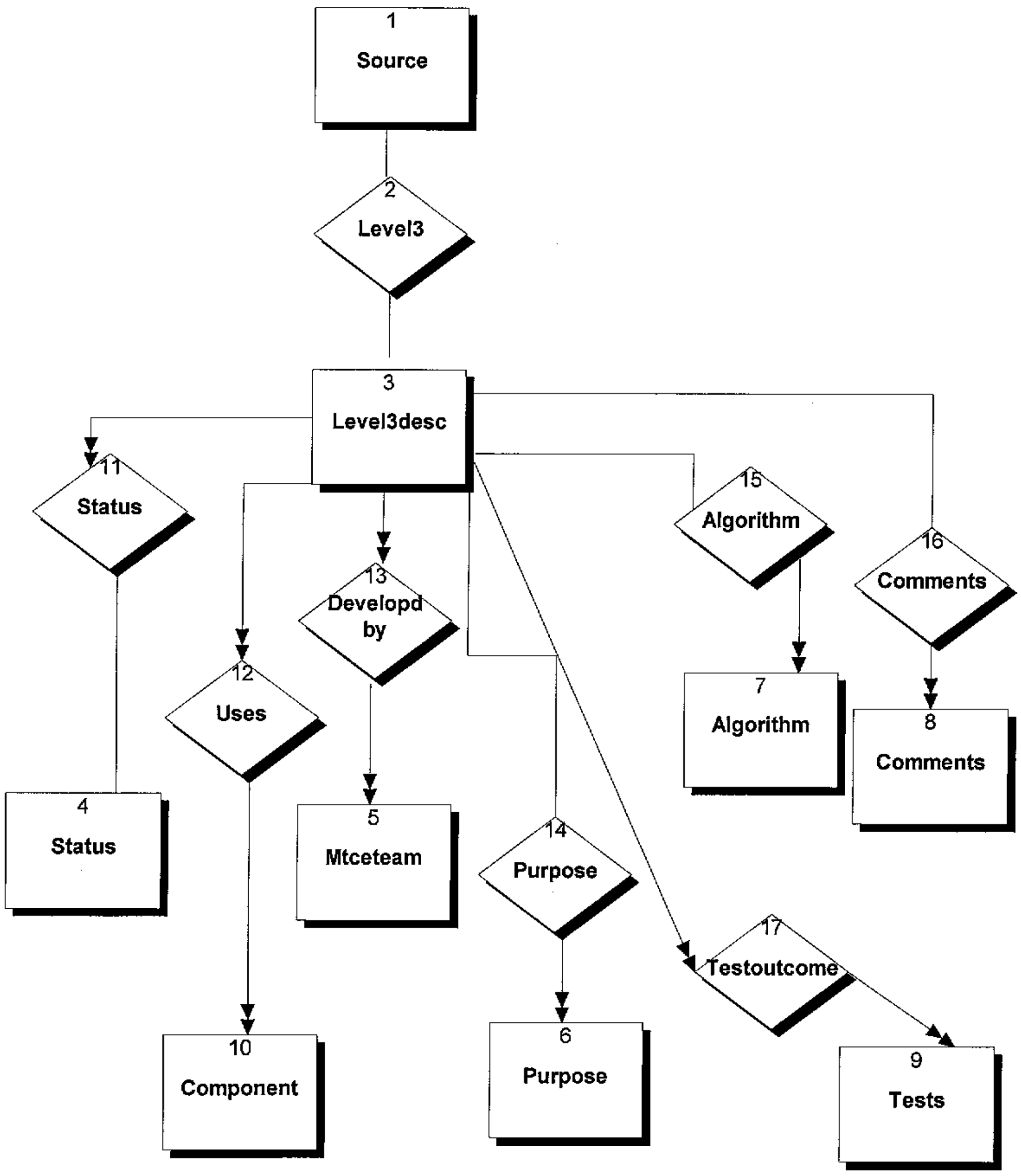
This leads us back to the Reports main menu.

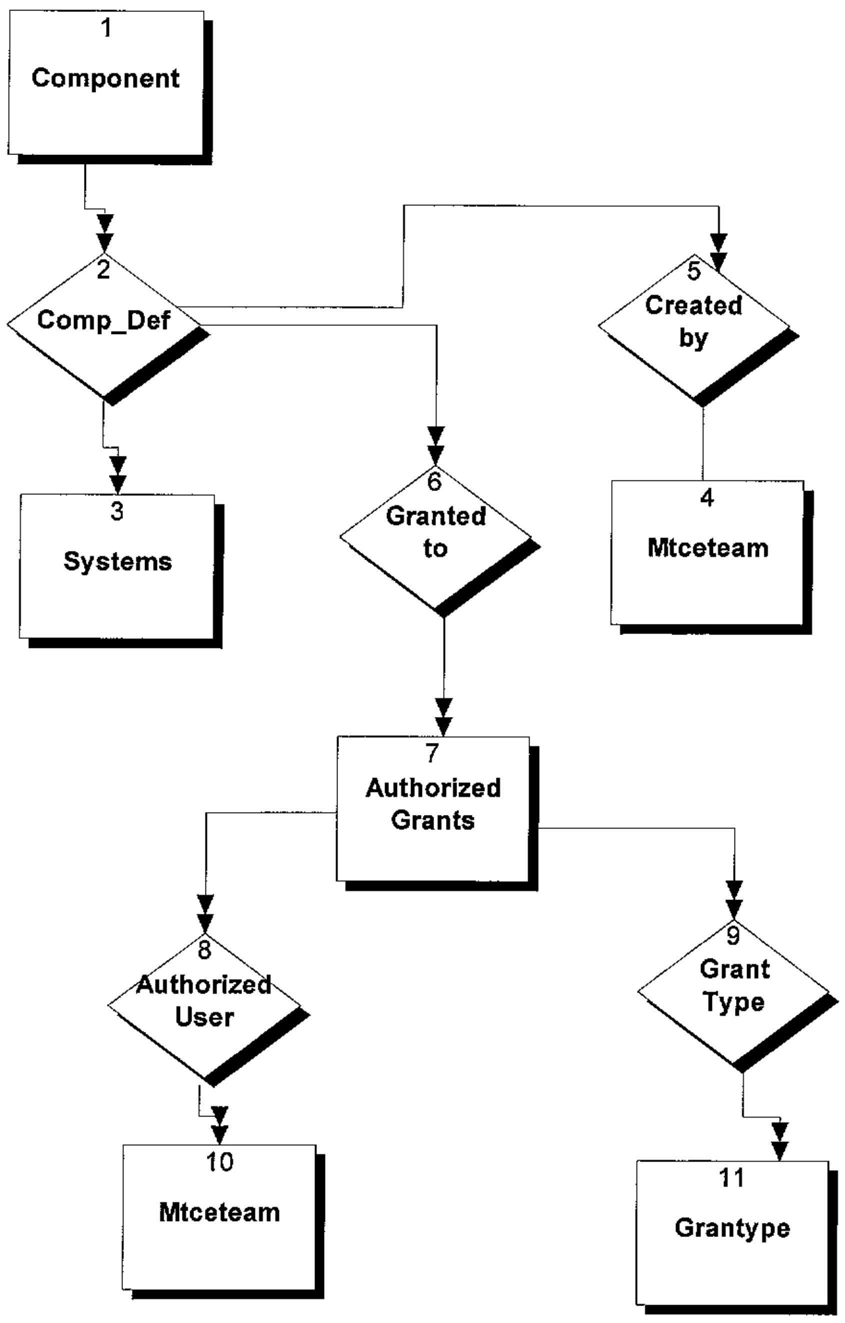
2. Tables Description

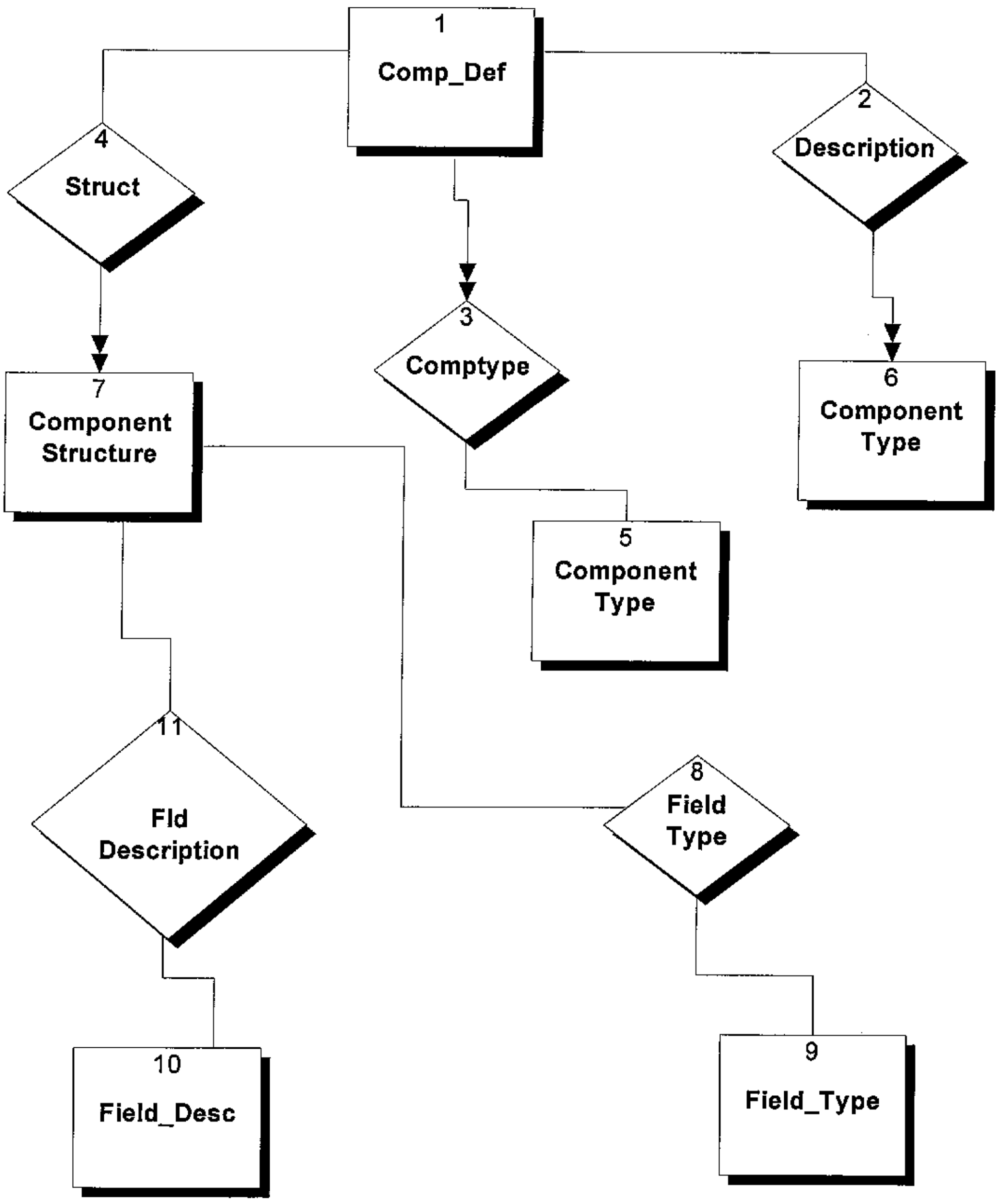
Following is the Entity_Relationship (ER) diagram that specifies the Low-Level design of the SMT tool. Next comes the listing of the tables used in the tool along with their attributes data types.

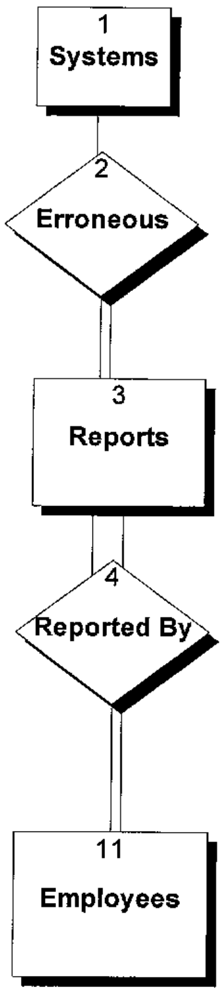

1
Systems
5
Levelcode
2
Systemchart
4
Levels
3
Source
6
Level1
7
Level1desc
12
Compare Resources
13
Develophases
10
Status
9
Nature
11
Status
8
Nature

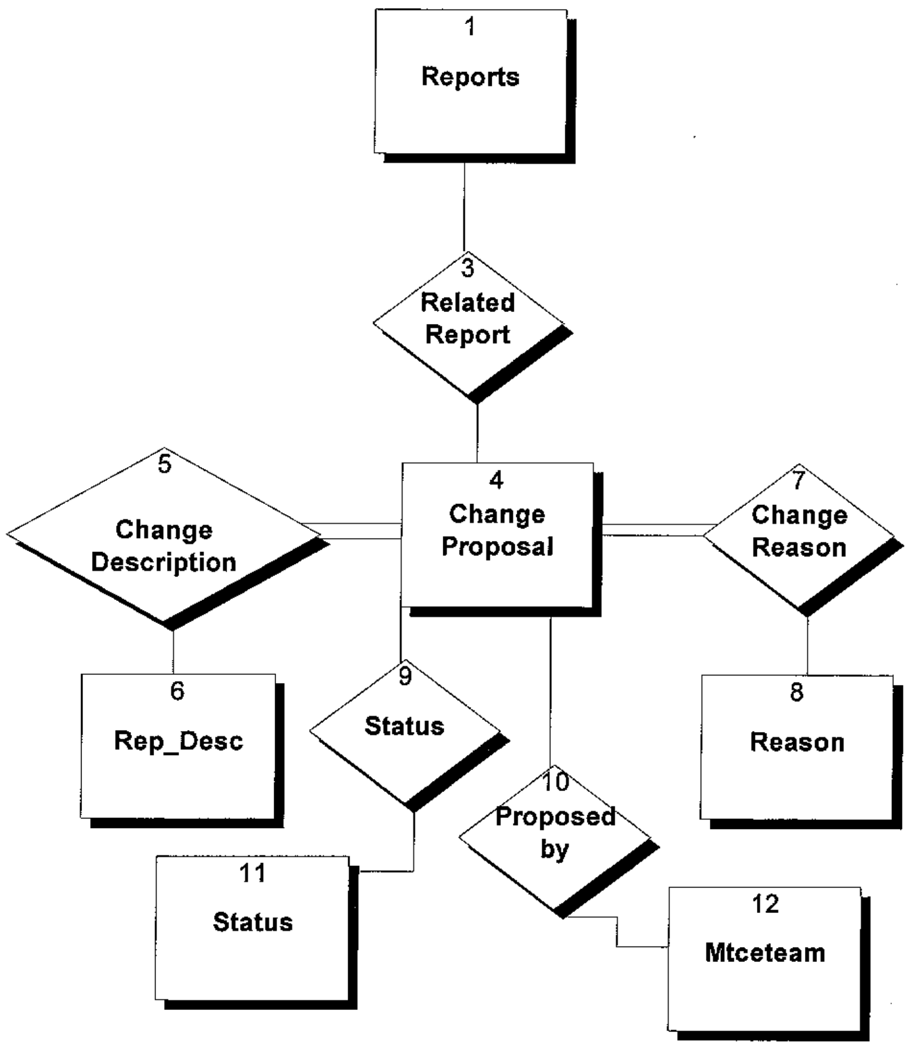

1
Source
2
Level2
3
Level2desc
8
Description
7
Status
4
Purpose
9
Level2_Desc
6
Status
5
Purpose

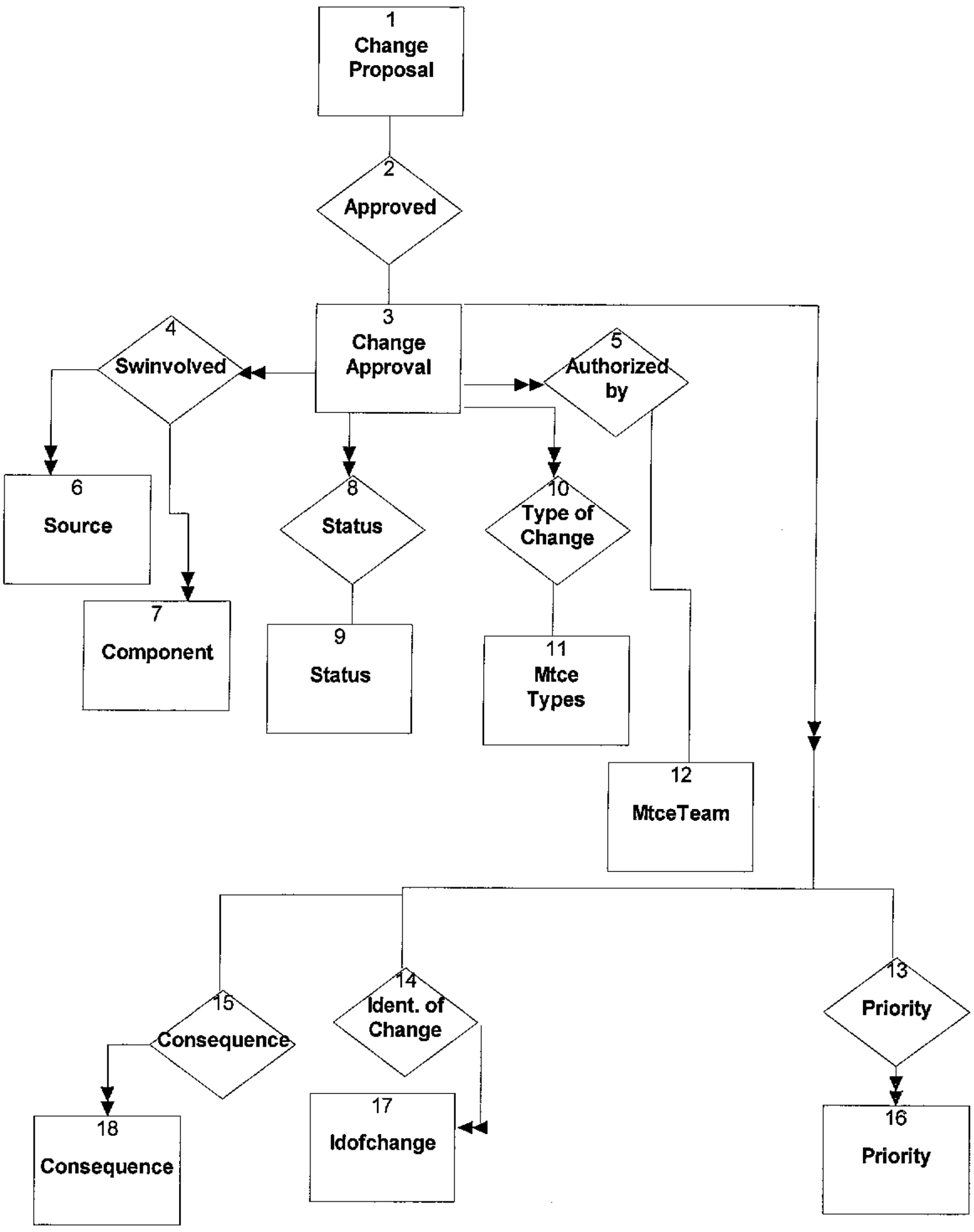

1
Source
2
Level3
3
Level3desc
11
Status
12
Uses
13
Developd by
15
Algorithm
16
Comments
7
Algorithm
8
Comments
4
Status
5
Mtceteam
14
Purpose
17
Testoutcome
10
Component
6
Purpose
9
Tests

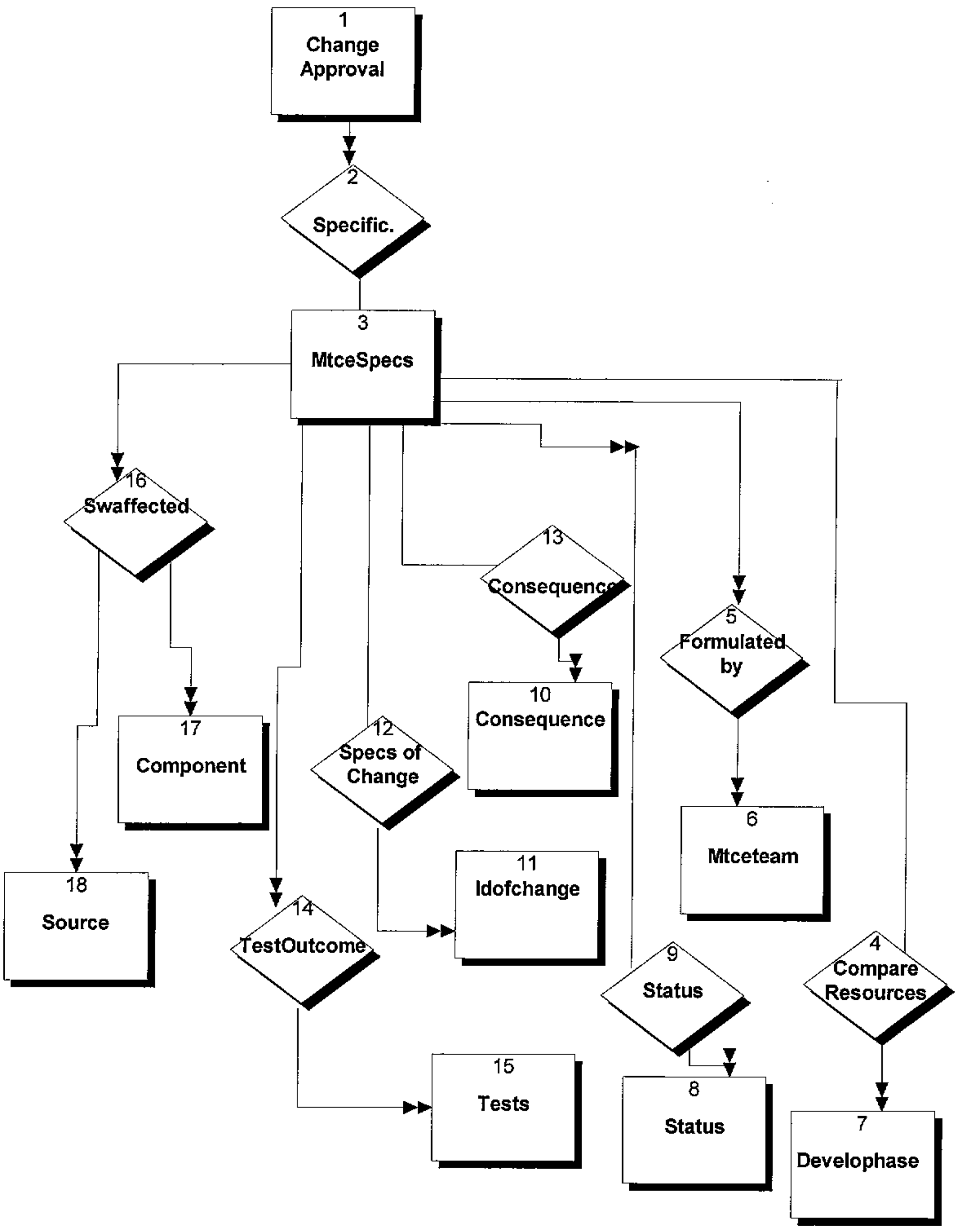

1
Component
2
Comp_Def
3
Systems
5
Created by
4
Mtceteam
6
Granted to
7
Authorized Grants
8
Authorized User
9
Grant Type
10
Mtceteam
11
Grantype

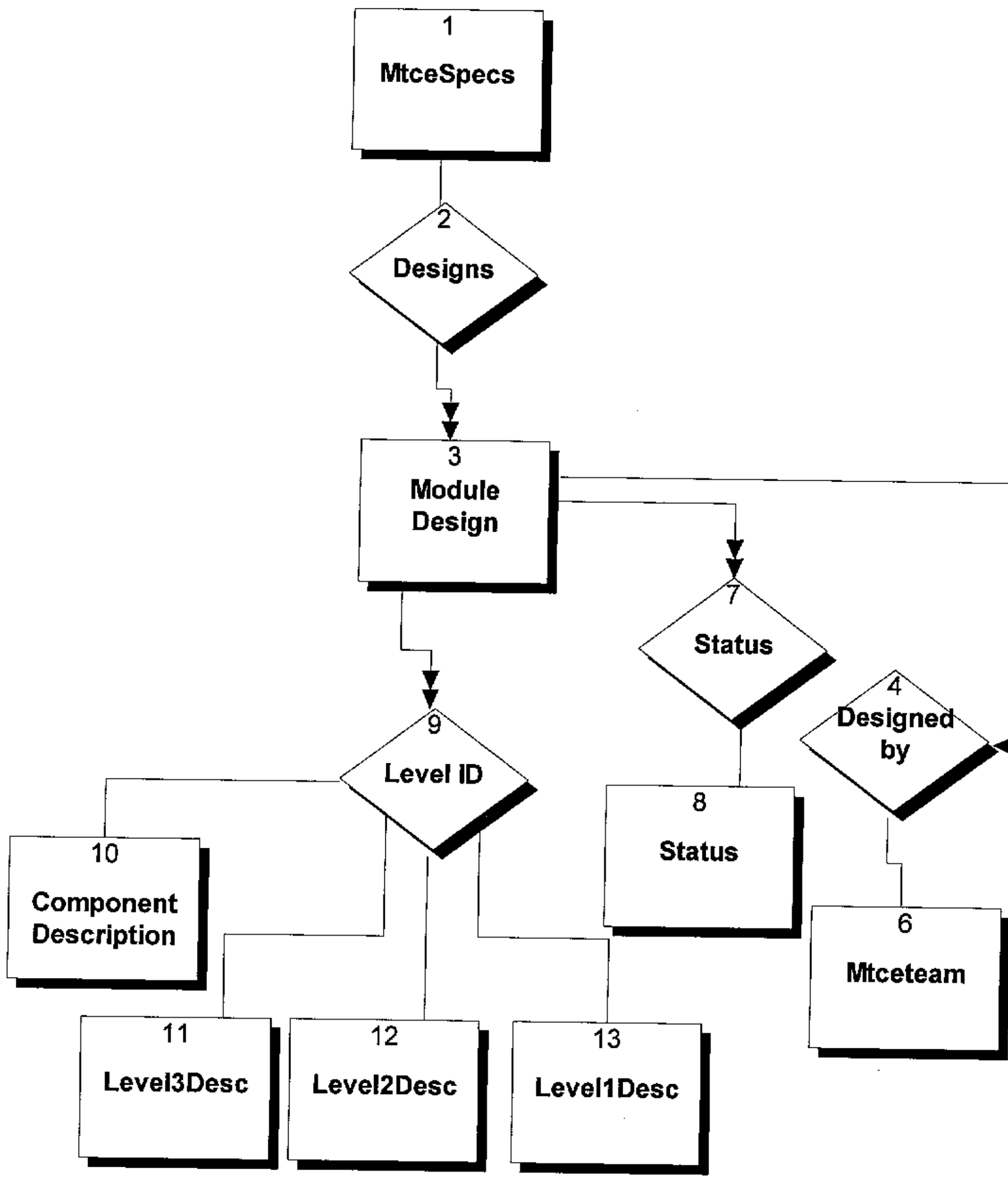

1
Comp_Def
2
Description
4
Struct
3
Comptype
6
Component Type
7
Component Structure
5
Component Type
11
Fld Description
8
Field Type
10
Field_Desc
9
Field_Type

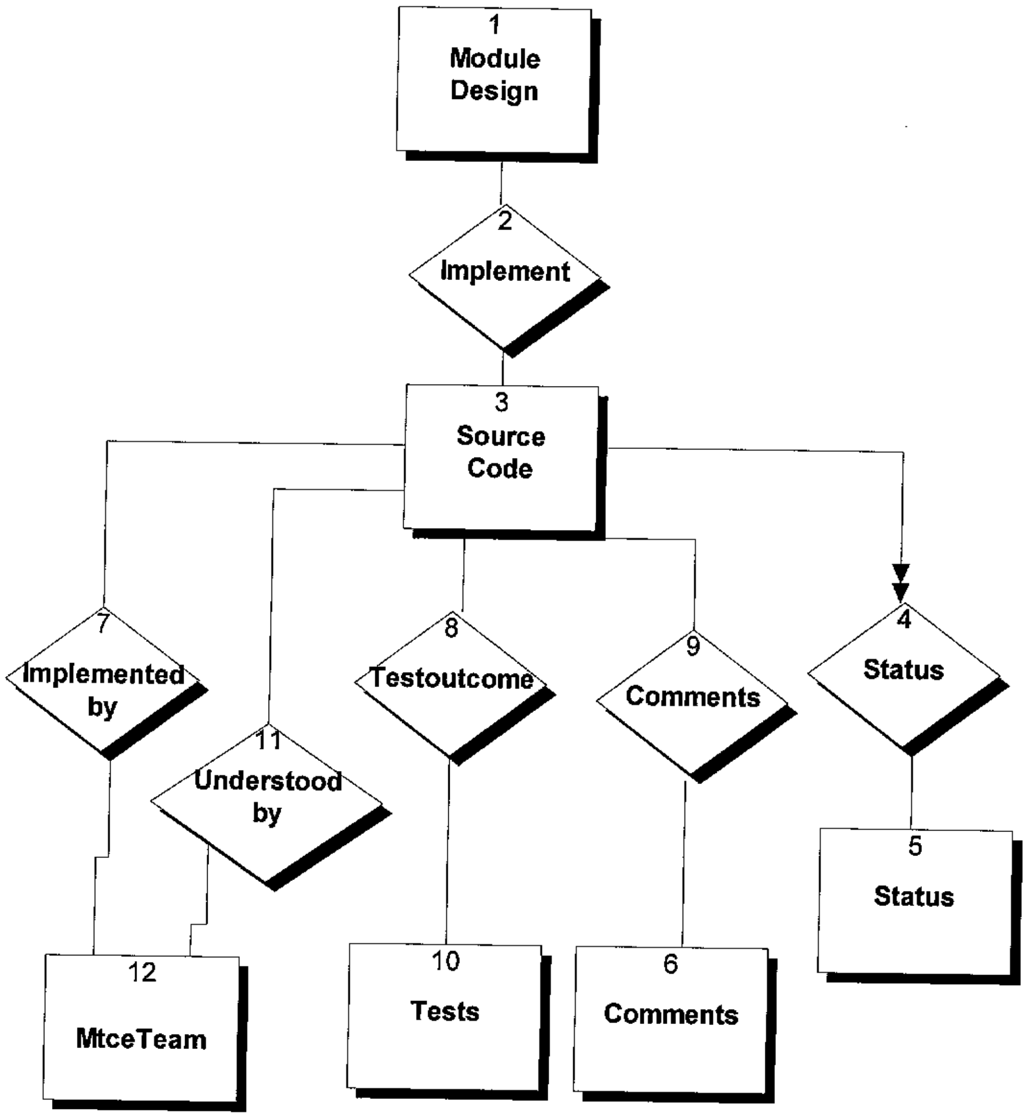

1
Systems
2
Erroneous
3
Reports
4
Reported By
11
Employees

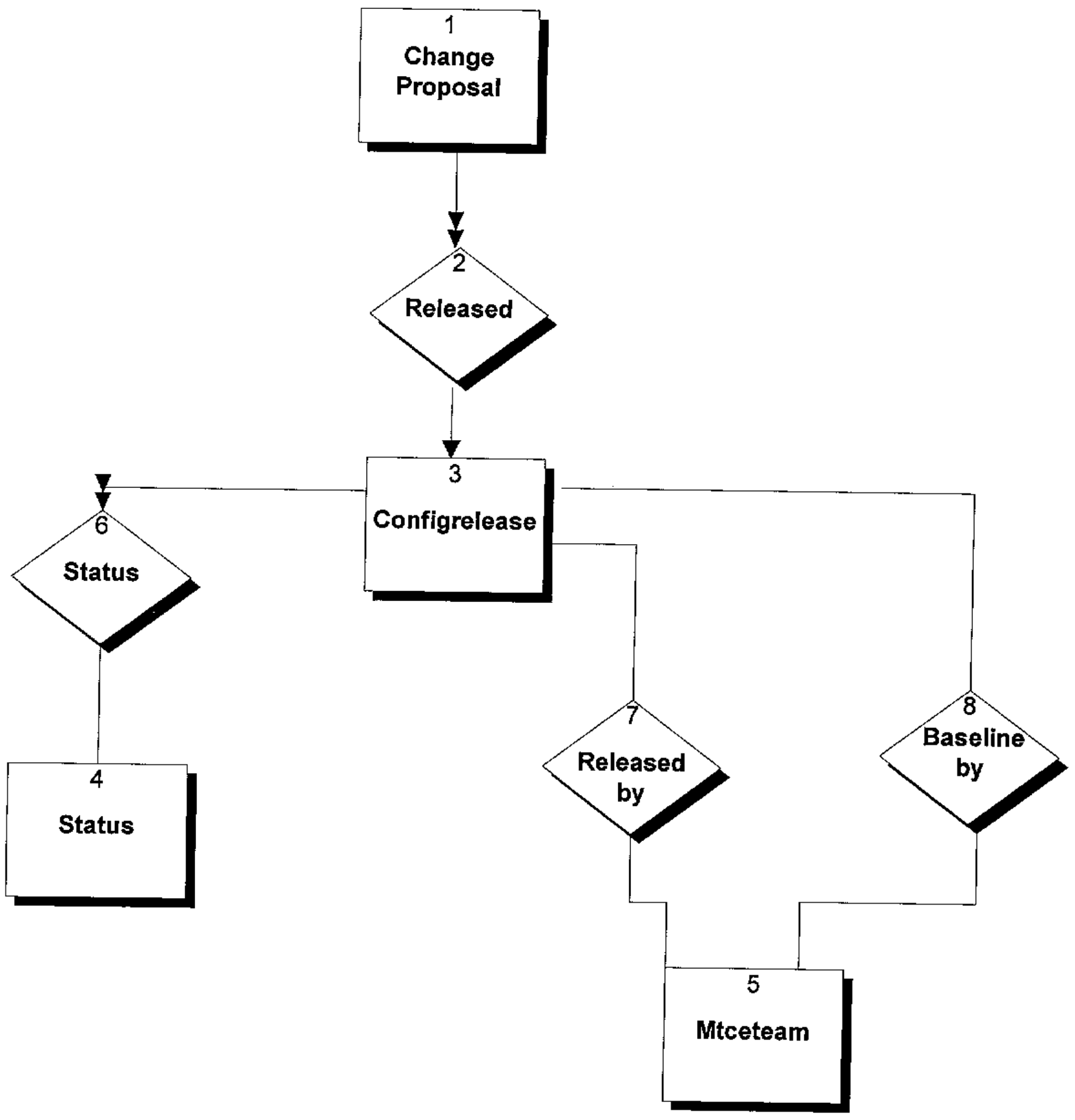

1
Reports
3
Related Report
4
Change Proposal
5
Change Description
7
Change Reason
6
Rep_Desc
9
Status
8
Reason
10
Proposed by
11
Status
12
Mtceteam

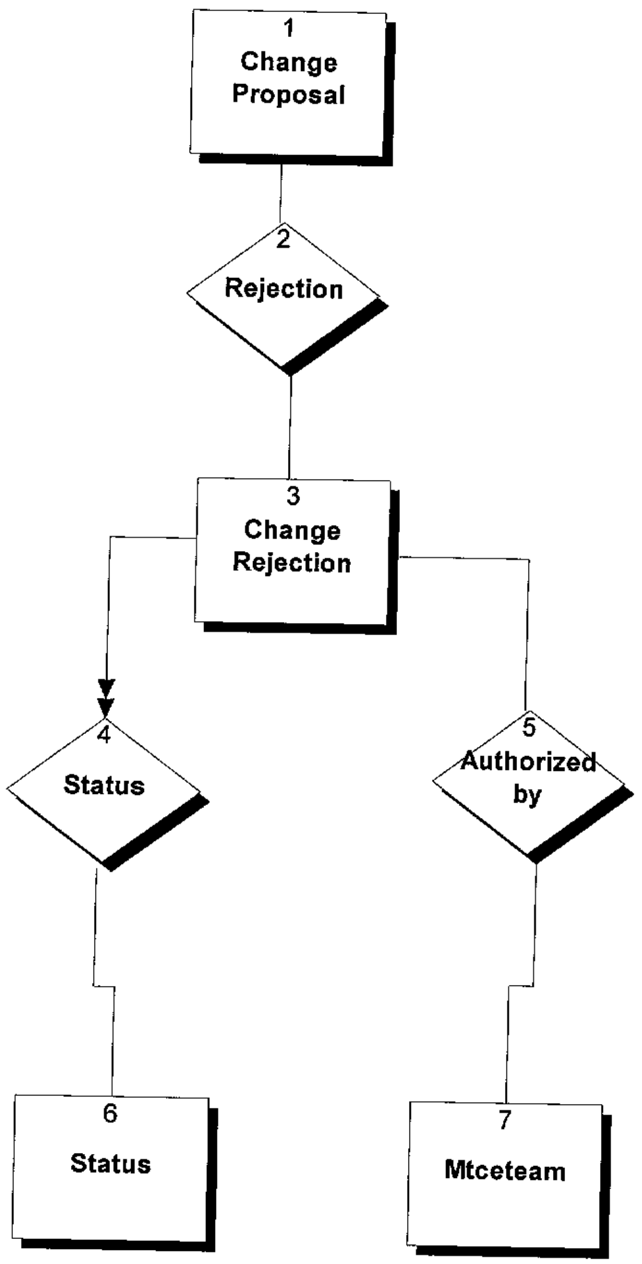

1
Change Proposal
2
Approved
3
Change Approval
4
Swinvolved
5
Authorized by
6
Source
7
Component
8
Status
9
Status
10
Type of Change
11
Mtce Types
12
MtceTeam
13
Priority
14
Ident. of Change
15
Consequence
16
Priority
17
Idofchange
18
Consequence

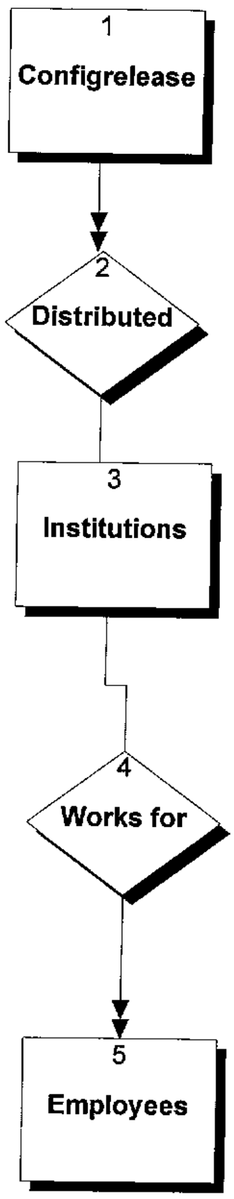

1
Change Approval
2
Specific.
3
MtceSpecs
16
Swaffected
13
Consequence
5
Formulated by
10
Consequence
17
Component
12
Specs of Change
6
Mtceteam
11
Idofchange
18
Source
14
TestOutcome
9
Status
4
Compare Resources
15
Tests
8
Status
7
Develophase


1
MtceSpecs
2
Designs
3
Module Design
7
Status
4
Designed by
9
Level ID
8
Status
6
Mtceteam
10
Component Description
11
Level3Desc
12
Level2Desc
13
Level1Desc


1
Module Design
2
Implement
3
Source Code
7
Implemented by
11
Understood by
8
Testoutcome
9
Comments
4
Status
5
Status
12
MtceTeam
10
Tests
6
Comments


1
Change Proposal
2
Released
3
Configrelease
6
Status
4
Status
7
Released by
8
Baseline by
5
Mtceteam


1
Change Proposal
2
Rejection
3
Change Rejection
4
Status
5
Authorized by
6
Status
7
Mtceteam


1
Configrelease
2
Distributed
3
Institutions
4
Works for
5
Employees

```
Table Input_Parameters
(
code          number(5)      not null,
syscode       number(5)      not null,
tabcode       number(5),
flag          char(1),
mdcode        number(5),
oldrevno      number(5),
oldverno      number(5),
newverno      number(5),
newrevno      number(5),
param         char(72));


Table Output_Parameters
(
code          number(5)      not null,
syscode       number(5)      not null,
tabcode       number(5),
flag          char(1),
mdcode        number(5),
oldrevno        number(5),
oldverno        number(5),
newverno        number(5),
newrevno        number(5),
param           char(72));

Table Rejection_Reason
(
code            number(5)       not null,
crjid           number(5)       not null,
name            char(72));

Table Consequence
(
code            number(5)       not null,
flag            char(1),
tabcode         number(5),
name            char(72));
```

```
Table Priority

(

code            number(5)       not null,

cacode          number(5)       not null,

name            char(72));


Table Idofchange

(

code            number(5)       not null,

flag            char(1),

tabcode         number(5),

name            char(72));


Table Change_Reason

(

code            number(5)       not null,

cpcode          number(5),

name            char(72));
```

```
Table Repdesc
(
code          number(5)     not null,
repcode       number(5),
flag          char(1),
name          char(72));


Table Field_Desc
(
code          number(5)     not null,
name          char(72),
flag          char(1),
mdcode        number(5),
oldrevno      number(5),
oldverno      number(5),
newverno      number(5),
newrevno      number(5),
tabcode       number(5),
fldid         number(5)     not null);
```

```
Table Field_Type
(
code            number(5)       not null,
name            char(20));

Table Algorithm
(
code            number(5)       not null,
text            char(72),
flag            char(1),
mdcode          number(5),
oldrevno        number(5),
oldverno        number(5),
newverno        number(5),
newrevno        number(5),
tabcode         number(5));

Table Authorized_Grants
(
flag            char(1),
mdcode      number(5),
oldrevno    number(5),
oldverno    number(5),
newverno    number(5),
newrevno    number(5),
tabcode     number(5)     not null,
staffcode   number(5)     not null,
grantcode   number(5)     not null);

Table Calls
(
sys2code    number(5)     not null,
mod2code    number(5)     not null,
flag        char(1),
mdcode      number(5),
oldrevno    number(5),
oldverno    number(5),
newverno    number(5),
newrevno    number(5),
tabcode     number(5));
```

```
Table Change_Approval

(

code            number(5)       not null,

caid            char(15),

datte           date,

status          number(5),

authorizedby    number(5),

baselineby      number(5),

typeofchange    number(5),

verno           number(5),

cpid            number(5)       not null,

msid            number(5));


Table Change_Proposal

(

code            number(5)       not null,

cpid            char(15),

datte           date,

status          number(5),
proposedby    number(5),
approved      char(1),
verno         number(5),
crid          number(5),
reportid      number(5)       not null);


Table Change_Rejection
(
code          number(5)       not null,
crjid         char(15),
datte         date,
status        number(5),
verno         number(5),
authorizedby  number(5),
cpid          number(5)       not null);


Table Company_Name
(
code          number(5)       not null,
name          char(40),
effective       char(1));


Table Compare_Resources

(

tabcode         number(5)       not null,

phasecode       number(5),

actual          number(5),

estimates       number(5),

levelflag       number(1),

mdcode          number(5),

oldrevno        number(5),

oldverno        number(5),

newverno        number(5),

newrevno        number(5),

flag            char(1));


Table Component

(

code            number(5)       not null,

name            char(15));
```

```
Table Comp_Def

(

code          number(5),

compcode      number(5)        not null,

createdby     number(5),

syscode       number(5)        not null,

comptype      number(5));


Table Component_Type

(

code          number(5)        not null,

name          char(25));


Table Component_Description

(

code          number(5)        not null,

flag          char(1),

mdcode        number(5),

oldrevno      number(5),

oldverno      number(5),

newverno      number(5),
newrevno        number(5),

tabcode         number(5),

description     char(72));


Table Component_Structure

(

fldid           number(5)           not null,

fldname         char(15),

fldtypecode     number(5),

fldlength       number(5),

flag            char(1),

mdcode          number(5),

oldrevno        number(5),

oldverno        number(5),

newverno        number(5),

newrevno        number(5),

tabcode         number(5));
```

```
Table Configrelease

(

code          number(5)        not null,

crid          char(15),

datte         date,

status        number(5),

syscode       number(5)        not null,

baselineby    number(5),

verno         number(5),

releasedby    number(5),

flag          char(1),

confirmed     char(1));


Table Developedby

(

developedy    number(5)        not null,

startdate     date,

tabcode       number(5),

mdcode        number(5),

flag          char(1),
oldrevno        number(5),
oldverno        number(5),
newverno        number(5),
newrevno        number(5),
enddate         date);


Table Develophases
(
code            number(5)          not null,
name            char(20));


Table Distributed
(
configcode      number(5)          not null,
instid          number(5),
flag            char(1),
verno           number(5),
datte           date);
```

```
Table Employees
(
employeeid        number(5)        not null,
employeename      char(30),
position          char(20),
institutionid     number(5)        not null);

Table Grantype
(
code              number(5)        not null,
name              char(25));

Table Institutions
(
institutionid     number(5)        not null,
institution_name  char(30),
typeofbusiness    char(20));
```

```
Table Level1desc
(
code          number(5)       not null,
status        number(5),
datte         date,
language      char(20),
syscode       number(5)       not null);

Table Level2desc
(
code          number(5)       not null ,
status        number(5),
datte         date,
syscode       number(5)       not null ,
modcode       number(5)       not null);

Table Level3desc
(
code          number(5)       not null,
status        number(5),
datte           date,
syscode          number(5)          not null,
modcode          number(5)          not null);
```

Table Level2_Desc

```
(
code            number(5)           not null,
name             char(72),
flag            char(1),
mdcode           number(5),
oldrevno          number(5),
oldverno          number(5),
newverno          number(5),
newrevno           number(5),
tabcode          number(5));
```

Table Comments

```
(
code            number(5)        not null,
name             char(72),
flag          char(1),
mdcode        number(5),
oldrevno      number(5),
oldverno      number(5),
newverno      number(5),
newrevno      number(5),
tabcode       number(5));


Table Levels
(
code          number(5)         not null,
name          char(20) ,
levelnbre     number(5) );


Table Module_Design
(
code          number(5)         not null,
mdid          char(15),
datte         date,
status        number(5),
msid            number(5),
flag            char(1),
oldrevno        number(5),
oldverno        number(5),
newverno        number(5),
newrevno        number(5),
designedby      number(5),
baselineby      number(5),
levelid         number(5),
levelflag       number(1));


Table Mtcespecs
(
code            number(5)           not null,
msid            char(15),
datte           date,
status          number(5),
verno           number(5),
formulatedby    number(5),
baselineby      number(5));
```

```
Table Mtceteam
(
code          number(5)          not null,
name          char(30),
authorized    char(1),
position      char(30));


Table Mtcetypes
(
code          number(5)          not null,
name          char(20));


Table Nature
(
code          number(5)          not null,
name          char(72),
flag          char(1),
mdcode        number(5),
oldverno      number(5),
newverno      number(5),
tabcode         number(5));


Table Purpose
(
code            number(5)          not null,
name            char(72),
tabcode         number(5),
flag            char(1),
mdcode          number(5),
oldrevno        number(5),
oldverno        number(5),
newverno        number(5),
newrevno        number(5),
levelflag       number(1));


Table Reports
(
code            number(5)          not null,
reportid        char(15),
datte           date,
reportedby      number(5),
flag            char(1),
verno           number(5),
syscode         number(5)          not null);


Table Source
(
code            number(5)          not null,
source_name     char(30));


Table Source_Code
(
code            number(5)          not null,
scid            char(15),
datte           date,
status          number(5),
implementedby   number(5),
understoodby    number(5),
baselineby      number(5),
oldrevno        number(5),
 oldverno        number(5),
 newverno        number(5),
 newrevno        number(5),
 mdid            number(5)          not null);

Table Status
(
 code            number(5)          not null,
 name            char(20));

Table Swaffected
(
msid             number(5)          not null,
modcode          number(5)          not null,
flag             char(1),
verno            number(5),
revno            number(5));
```

```
Table Swinvolved

(

caid          number(5)        not null,

modcode       number(5)        not null,

verno         number(5),

revno         number(5));


Table Systemchart

(

syscode       number(5)        not null,

modcode       number(5),

parentcode    number(5),

levelcode     number(5));


Table Systems

(

code          number(5)        not null,

name          char(20));
```

```
Table Tests

(

code            number(5)           not null,

name            char(30));


Table Tests_Outcome

(

code            number(5),

testcode        number(5)           not null,

testout         number(5),

datte           date,

mdcode          number(5),

oldrevno        number(5),

oldverno        number(5),

newverno        number(5),

newrevno        number(5),

levelflag       number(1),

flag            char(1));
```

```
Table Uses
(
 syscode      number(5)        not null,
 tabcode      number(5),
 flag         char(1),
 mdcode       number(5),
 oldrevno     number(5),
 oldverno     number(5),
 newverno     number(5),
 newrevno     number(5),
 compcode     number(5)        not null);

Table Versions
(
 syscode      number(5)        not null,
 oldver       number(5),
 newver       number(5),
 crid         number(5));
```

```
Table Source_Version

(

 syscode        number(5)          not null,

 modcode        number(5),

 verno          number(5),

 revno          number(5));


Table Component_Version

(

 syscode        number(5)          not null,

 compcode       number(5),

 verno          number(5),

 revno          number(5));


Table Constraints

(

code            number(5)          not null,

name             char(72),

flag             char(1),

oldverno         number(5),
newverno        number(5),
mdcode          number(5),
tabcode         number(5));
```

Table Hardware_Requirements

```
(
code            number(5)           not null,
name            char(72),
flag            char(1),
mdcode          number(5),
oldverno        number(5),
newverno        number(5),
tabcode         number(5));
```

Table Unfinished_Work

```
(
code            number(5)           not null,
name            char(72),
flag            char(1),
mdcode          number(5),
oldverno        number(5),
newverno        number(5),
tabcode         number(5));


Create table Temp_Rep
(
repname      char(40),
user_name    char(20),
dum1         char(40),
dum2         char(30),
dum3         char(20),
dum4         char(20),
num1         number(3),
num2         number(3),
num3         number(3),
num4         number(3));
```

Chapter 5

Implementation

This chapter includes an explanation for the algorithm of the most important forms in the tool, states some test cases, and lists the hardware and software requirements needed to run it. The forms of the Development phase are described first, and then those of the Maintenance's. Not all of the listed forms below are going to be described because only those which do more than what is explained in the User Manual need Technical Description, while for the others - simple forms - a reference to the User Manual is enough.

1. Forms' Technical Description of the Development Component

The following forms are the ones associated with the Development component of the tool. The ultimate user is not concerned with them because he/she will be dealing with the Menu Options where internally each menu option is associated with one of these forms.

1- Levels
2- Systems
3- Source
4- Component
5- Status
6- Mtceteam
7- Mtcetype
8- Grantype
9- Institut
10- Employee
11- Developh
12- Tests
13- Fieldtype
14- Companam
15- Comptype

16- Chart

17- Compdf

18- Struct

19- Level1

20- Level2

21- Level3

22- Actual

23- Config

1.1 "Levels" form

It is vital that the form "Levels" be executed first in order to tell the hierarchy of the new software system under-development; this will help in telling how Large the software is.

Hence, upon running the above mentioned form, any previously defined levels will appear allowing the user to add, maybe, or just query the levels. If none is available then the user is requested to fill in the levels of the system plus the number of each level. The software system might consist of:

- system, subsystem, module, submodule, program, procedure, etc.

or

- system, program, procedure, function, etc.

Depending how large the software is, the user should be able to determine the nesting levels. One very important remark to be made at this step is that the level numbers should be in the range of (1..3) including. The tool just would not work if those level numbers are not defined and here is why:

Level number one is given only to the software system's names; i.e., there should be one level number 1 in any project or software development. Next, comes all the functional modules which are of level two. Only the source code modules (procedures, functions, etc.) that are exact implementation of the functional modules are of level three. For levels two and three we may have more than one item; e.g., subsystem, module, submodule, etc. are of level two, and procedure, function are of level three. If the software is not large, it is enough to have level one and level three, i.e., skipping level two is no problem on one condition that the associated forms with level two are not run. And if the software is so small that it consists of one module, then the user has to define a name for the software and another for the module assigning 1 as the software level number and 3 for the module.

The job of this form, hence, is to accept the nesting levels of a software system plus their associated level numbers. This form does some checking on the level numbers insuring that there exists exactly one level number 1, and at least one level number 3 and that the range of the level numbers does not go beyond (1..3).

1.2 "Systems" form

The tool will accept the definition of the names of the software systems under development. Upon the entry of each name a new record is created in the table Systems as well as in the table Source, that is because in the "Chart" form there is field called Parent that both source code names and software names share. Whenever a piece of source code is the parent of another then the value of the field is selected from table Source. But when the parent of a source code is the software itself then the value of the field is selected from the Systems table. This can not possibly happen, to select values from two tables for one field and hence the decision was to insert the name of each software in the table Source and to give it a new code.

This form will also create a new record for the software in the Versions table giving it a version number of ZERO.

1.3 "Chart" form

This form relates a source code module to a software system where it belongs as well as to a parent source code thus drawing the hierarchy of the software. This form gives the newly defined source code module a revision number ZERO, a record is created in the Source_Version table for this purpose. Of course, a record is created in the Systemchart table for the mentioned relation (between software and source code module).

1.4 "Compdf" form

This form relates a component to a software system where it belongs, giving it a revision number of ZERO, and a record is created in the Component_Version table for this purpose. Of course, a record is created in the Comp_Def table for the mentioned relation (between software and component).

1.5 "Config" form

This form releases a configuration of a software upon finishing its development and before being distributed to a client institution. The key to this form is the Configuration Release Identification which should be unique to each form. The form checks the Identification for duplicates, and in case none is found then all other data is entered, including the Systems' tests and Integration tests and the total time resources taken to complete the whole project. Any unfinished work can be saved in the specified block on page two of the form.

This form creates a record for each release, specifying the software name, status, date of release and type of release - Development or Maintenance.

2. Form's Technical Description of the Maintenance Component

The Maintenance component includes the following forms which are in nature more complicated - than the Development component's. Some forms include important algorithms that need be explained in this chapter, while the simple ones can be referred to in the User Manual.

1- Bug
2- Proposal
3- Approval
4- Specs
5- Design
6- Implement
7- Mtactual
8- Totactual
9- Reject
10- Mtconfig
11- Postjob
12- Addsrc

13- Rmsrc

2.1 "Bug" form

This is the first form to be filled-in when a change request comes from a client institution. The change request should look like this "Bug" form. First an identification is given to this report, then the name of the software that the report is about and its version number. Then the user has to specify if the report is a Bug report or an Enhancement report; there is a field where either a "B" or an "E" is required to identify the type of the report.

The name of the client institution and the name of the employee who requested the change are also needed for future records. At this stage, a procedure is invoked called "Get_Confignumber" to check if the software name (and that version number) is released or not. If not then a message is prompted to the user saying that the software is not released yet; this might suggest an error in the version number of the software.

If everything is okay, then a detailed description - can be brief - is required to describe the type of change.

2.2 "Proposal" form

This form is the result of bug report. There is a one-to-one relationship between a bug report and a proposal form. It is a more formal representation of the information found in a Bug report. In this form an Authorized staff member has to "propose" the changes requested in the corresponding bug report as well as stating the reason for this change - this reason has to be the Authorized staff member's opinion.

The corresponding record in the table Change_Proposal will have two other fields left blank, namely the Approved and the Crid. The first is flagged with a 'Y' if the change has been approved - from the Approval form - and with an 'N' if the change has been rejected - from the Reject form. The second field, the Configuration Release Identification, is filled-in when a configuration is released after the whole maintenance action is complete.

2.3 "Approval" form

This form is initiated by an Authorized staff member after a change request is studied and turned out to be essential to implement it. The user will have to assign a unique Change Approval Identification for each approved change, as well as the Identification of the Change Proposal for which this approval is done.

If the change proposal is assigned another change approval then the form is able to detect it and reject the given information, because as mentioned earlier there is a one-to-one relationship between each change proposal and change approval.

If the change proposal is not assigned to another change approval then the Approved field in the Change_Proposal table of that specific change proposal - which name is in the change approval form - is flagged with a 'Y'.

The form also includes a text paragraph requesting the Identification of Change, of course prepared by the Authorized staff member, as well as the priority of implementation - whether it is urgent to implement the change or it can wait for some time. Another text is there to enter the consequences of these changes, on the other source codes, if not implemented. Finally, the staff member has to determine the names of the software source codes involved in the change.

2.4 "Specs" form

This form is initiated for one change approval or for many; i.e., if some change approvals were found to be not that urgent then the maintainers can wait until they have a "bunch" of change requests already approved - could be related some how - and that it is time to determine their specifications. Then some Authorized staff member would formulate the specifications of the whole changes and this is where they enter these specifications along with the change approvals involved.

In this form a checking is done for each change approval involved to see if it has been rejected lately; i.e., after it has been approved. If everything is okay, then the consequences of the changes on the other source codes are determined.

Also the software affected by the changes are determined and entered, the same for the Tests needed to verify the correctness of the changes as well as the time estimates for each phase involved in the maintenance.

2.5 "Design" form

A Module Design form is initiated for each of the Software Affected declared in the previous form "Specs". The main aim of this form is to first identify the Level of the module, relate it to a Maintenance Specification, and then move to another screen - by only pressing [Next Field] or [Next Record] key. This new screen is an exact copy of the original one for each level type, but the only difference is that this one is meant for maintenance and changing designs not for the originals.

For instance, if the Software Affected, for which this - "Design" - form is initiated, is of Level One then the new screen would be a copy of the Level One form. The only difference - technically speaking - is that in the Level One form a flag is set to 'L' and here the same flag is set to 'M', meaning Development and Maintenance, respectively.

Hence, from this form four screens can be popped up, three of which are one for each level and the fourth is for the other components that need modifications - also should be declared as Affected in the previous form.

A note is worth mentioning here, the word "exact" might not be so for all of the four new screens, because some information are not needed at Maintenance stage. It is obvious that these new changes - entered from the above mentioned four new forms - use the same tables in storing their information as the original Level entry forms, except for the flag, of course, which distinguishes between multiple information for the same source code module/component each for either Development or Maintenance purposes.

The validations are many in this form. It starts with the name of the source code/component affected. The system checks if the entered name is really declared as Affected or not. If it is not declared as such, then no further work is done in the form, otherwise, the process is permitted to continue and next the Module Design Identification is validated against being given to another module design.

The normal Status, Baseline established by, Date, Designed by, etc. are filled-in and the last item on the form is the related Maintenance Specification Identification. The system here checks first whether the entered Maintenance Specification Identification is the one in which the above mentioned source code/component was declared as Affected; i.e., checking if the correct chain of forms and stages are being followed. Next, it checks if this chain of change proposal, approval, up to this maintenance specification is already released. If so then the user has no right to modify a source code/component, even if declared some where as affected, without a new change proposal, approval, etc. chain.

If the validation and verification process at this field turns out to be okay, then [Next Field] or [Next Record] keys would lead to one of the four new screens discussed above.

2.6 "Mtactual" form

This form is used to enter the Actual time resources spent in any maintenance action on a piece of source code, as well as the Finishing Date of the process for each maintainer working on the module. It is initiated after the Implementation phase is successfully done.

The form requests from the user the software name - Level1 Name - and its version number, in addition to the source code module's name and its revision number. The Module Design Identification is needed to tell what is the Design Document of the source code in question the user needs to enter the Actual data. Note that in the Mentioned Module Design Identification form the Estimates were entered as well as the name of each staff member working on it and the Date each started implementing the changes.

The form does validate the Module Design Identification if it really is the Design Document for the source code in question. If not then the process is not allowed to continue. Otherwise, the "Compare_Resources" table where the Estimates were entered is updated by filling-in the "Actual" field, and the "Developedby" table is updated by filling-in the "Enddate" field both just entered in the form.

2.7 "Mtconfig" form

This form is the Configuration release of a software product after performing some maintenance actions and modifications on it. The first input for this form is the Configuration Release Identification which must not be one of the Development Configuration Release Identification. Stating this means that the system does the checking on the validity of the Identification entered.

Next, the system prompts the user for the Change Proposal Identifications that lead to this release, without the Identifications of the other forms because it will get them internally. Also the system checks if these Proposals have already been assigned to another configuration release.

If not then the popped up - generated new version number of the software system - if approved by the user - is then used to update the table “Versions”. It assigns this generated version number to the field “newver”, sets the field “crid” equal to the Configuration Release Identification in question, and creates a new record assigning the field “oldver” to the new version number leading the two other fields (“newver” and “crid”) blank for the next release of this software system.

2.8 “Postjob” form

This form is initiated after the maintenance configuration release of a certain software product is performed. It first updates the two tables “Source_Version” and “Component_Version” by creating new records for each source code module or component related to the old version, now related to the new version.

By doing this I will be relating all the source code modules and components - of the old version of the software product - to the new released version.

The above is true except for few cases where the user does not want to include in the new version certain source code modules or components. The next block in this form is for the deletion of Source Code Modules from the new release, and the third block is for the deletion of Components from the new release.

The corresponding data in the other tables will be updated as well; i.e., the modified source code modules' or components' descriptions will belong now to the new version of the software product. The unmodified ones' also have to be related to the new version to ensure consistency in all of the configurations of a certain software system because after all a software system is composed of all of the originally defined source code modules and components - whether modified or not. Hence, all have to be related to any subsequent version numbers of the same software.

The last block initiates the process of updating all of the database tables incrementing the "newverno" and "newrevno" fields each by One. Thus the new change documents of any modified source code module or component will be related to the new version number of the software system through the field "newverno", and will be announced as belonging to the new Revision through the field "newrevno".

For the unmodified source code modules and/or components, they are declared as belonging to the new version number of the software system - through the field "newverno" - without incrementing their Revision numbers.

This form also contains additional procedures to check if this form has been successfully initiated before. This can be done by checking the two tables "Source_Version" and "Component_Version" for records belonging to the New Version Number of the software system. If there are such records, then this means that the "Job" has been performed successfully earlier, hence the user is notified that the records are already updated and is requested to leave the form.

If no such records exist, then the process goes on normally. The Creation process proceeds and any deletion requested from the user - blocks two and three - are performed, and if in the last block the user's answer is 'Y' then the tables are updated, else if 'N' then no tables are updated and the already created records in the "Source_Version" and "Component_Version" tables are deleted.

3. Experimental Results

This tool was testes on a real software system similar to what the software company LOGOS is developing to the Lebanese American University (L.A.U.), but using less modules and data. The tool behaved as specified: original documentations were entered - brief and detailed - maintenance actions were only allowed to be implemented through the proper channel and by authorized users, configuration release for each version of a software system solved the version control problem and no one software (or module within a software) was modified without increasing its version (revision) number - at configuration release time. The tool was also tested on a small program consisting of only three procedures and it behaved equally the same.

4. Hardware/Software Requirements

To run the tool, the user needs as a hardware requirements a 486 Personal Computer (PC), with minimum 8 mega bytes of RAM, and as a software requirements the installation of the ORACLE environment on the PC which includes the sqlforms, sqlmenus, sqlreportwriter, etc.

Chapter 6

Conclusions

This work is a direct implementation of a Software Maintenance Model (SMM) that creates a proper procedure for any maintenance action to be implemented. The Software Configuration Management (SCM) discipline is also applied to the tool ensuring that no information is omitted from any phase, that phases or stages are properly related to one another, and that any status accounting or report can be generated at any time.

The version control problem is dealt with in this tool, generating internally a new version number for each software system - or even source code part - after being modified. Hence, at any one time the programmer or maintainer can tell which revision of a source code module of what version of a software system he/she is working on/modifying.

Since this tool is directed towards procedural languages, it can be enhanced even more by allowing it to accept the maintenance of softwares written in an unprocedural languages. For example a system or project developed under ORACLE would need to concentrate on the “Forms”, “Reports”, and “Menus” since this environment is manifested mainly by these three items.

Appendix A

User Manual

This user manual is a brief description of all the forms in the tool. Each form's description contains a brief summary of the form's function and purpose as well as a list of the function keys, to be used in the form, and their meaning. The forms are listed in the order that the user has to use the tool at least at the very start in the development and maintenance phases. The function keys are almost the same in all the forms except in a few cases.

Index

Totactual	20
Mtconfig	20
Postjob	21
Reject	22
Addsrc	22
Rmsrc	23

"Levels" Form

This form is used to enter the nesting levels of the system or project in question, i.e., the naming of the system's hierarchy. The company can define the complete hierarchy for general purposes, and for each software in particular itcan assign any of the defined levels (not necessarily all of them.)

Example of the nesting levels:

- System or Project,
- Subsystem,
- Module,
- Submodule,
- Program,
- Procedure and
- Function, etc.

The user has the option of defining some or all of these. But he/she has to be careful as when it comes to the LevelNumbers. This form includes only one block, but with two fields. The Level Name and the Level Number. As mentioned in the "System" form, the first level (system or project) is of level One. All subsequent levels that are functional levels - not actual source code - belong to Level Two. Only those that are the real codes belong to Level Three. Hence, this tool uses the convention of having Three Levels, only the names defined as "systems" or "projects" belong to Level One, those who represent the actual pieces of source code belong to Level Three, and the rest - Functional Levels - belong to Level Two.

In order to navigate among

- Fields, use [Next Field], [Previous Field]
- Records, use [Next Field], [Up Arrow]

In order to save the changes:

- Press [Commit]

In order to exit thr form:

- Press [Exit Form]

As applied in all the tool's forms, the user can not delete after Commit.

"Systems" Form

This form is used to define the names of the software systems or projects being developed by the software company owning this tool. Any name defined in this form will belong to Level One (described later.)

This form is composed of only one block. Entering this form will cause any previously defined names to be displayed, allowing the user to query any name. So, in this form only query, definition, and update of names is allowed; deleting is totally prohibited.

In order to go from one field to another:

- Press [Next Field], [Previous Field]

In order to save the changes:

- Press [Commit]

In order to exit the form:

-Press [Exit Form].

"Source" Form

This form is used to define all the source codes' names - belonging to both Level Two and Level Three. Some of the names might already exist but belonging to another aoftware, so no need to define them twice. The user will later relate the source code name to the new software.

On entering this one-block form the user will see the names of the softwares already defined in the "System" form. This is just for internal purposes. In this form the user only defines the names without relating them, yet, to a software.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

Deletion after Commit is not allowed.

"Component" Form

This form is used to define the names of the components - Files, Tables, etc. - used in the software being developed. Some of the names might already exist but belonging to another aoftware, so no need to define the name twice. The user will later relate the component name to the new software.

Entering the form the previously defined component names will be displayed, and the user has the right to add more names, modify any, but not to delete any name.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Status" Form

This form is used to enter the Status Conditions that might be needed in different situations; e.g. Under-Development, Complete, etc.

This form contains only one block. On entering it any previously defined status conditions will be displayed and the user has the right of only to add some more, modify any, but not delete.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Mtceteam" Form

This form is used to enter the Names of the personnel working for the software company - owning this tool. They may be working as development or maintenance team, no difference.

This form contains only one block with three fields. The Staff Member Name, Authorized (Y/N) in order to control later the maintenance actions, and Position kept just for the record. On entering this form any previously defined records will be displayed allowing the user of only to add some more, modify any, but not delete.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field] on the last field

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Mtcetype" Form

This form is used to enter the Types of the maintenance activities which are mainly Corrective, Adaptive, and Perfective.

This form contains only one block with one field the Name. On entering it any previously defined names will be displayed allowing the user of only to add some more, modify any, but not delete.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Grantype" Form

This form is used to enter the Types of the Grants given on a component for any staff member working on the same software.

This form contains only one block with one field the Name or Type. On entering it any previously defined types will be displayed allowing the user of only to add some more, modify any, but not delete.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Institut" Form

This form is used to enter the Names of the Client Institutions that the company is developing software to.

This form contains only one block with one field the Name. On entering it any previously defined names will be displayed allowing the user of only to add some more, modify any, but not delete.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Employee" Form

This form is used to enter the Names of the Employees working at the Client Institutions that the company is developing software to.

This form contains two blocks. The first includes the name of the client institution where the employees to be defined work to. The second includes the Name of the Employee and its Position. On entering this form and assigning a client institution any previously defined names - as working for it - will be displayed allowing the user of only to add some more, modify any, but not delete.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field] at the last Field
- Block, use [Next Record] and [Previous Record]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Developh" Form

This form is used to enter the Names of the Development Phases used in developing any project at the software company.

This form contains only one block with one field the Name. On entering it any previously defined names will be displayed allowing the user of only to add some more, modify any, but not delete.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Tests" Form

This form is used to enter the Tests' Names used in all the phases of the life cycle of a project.

This form contains only one block with one field the Name. On entering it any previously defined names will be displayed allowing the user of only to add some more, modify any, but not delete.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Fldtype" Form

This form is used to enter the Types that any field in a table or file might belong to. E.g., Character, Number, Alpha-Numeric, etc.

This form contains only one block with one field the Name. On entering it any previously defined names will be displayed allowing the user of only to add some more, modify any, but not delete.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Companam" Form

This form is used to enter the Name of the Development Company owning this tool.

This form contains only one block with two fields the Name and Effective. The Name is used, of course, for the company's name and the Effective is a flag (Y/N) to tell the user if the name is no longer effective. Hence, only one effective name should function at any one time.

On entering this any previously defined names will be displayed allowing the user of only to add some more, modify any, but not delete.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Comptype" Form

This form is used to enter the Types of a Component. E.g., Table, File, Index, etc.
This form contains only one block with one field the Name or Type. On entering it any previously defined names will be displayed allowing the user of only to add some more, modify any, but not delete.

In order to navigate among

- Records, use [Next Field] and [Previous Field]

In order to commit:

- Press [Commit]

In order to exit the form:

- Press [Exit Form].

"Chart" Form

This form is used to relate a source code name to a software name. Definitely, both should be previously defined in the above discussed forms.
This form is composed of three blocks. The first one contains the Level One Name and the other levels' Name as well the software Version Number and the level two or three Revision Number. The second block contains the Parent Name and the Level Name. This

block will form the hierarchy of the software - its name, its subsequent children's names and their levels. The third block is used only when the user is relating a source to a software as a Perfective Maintenance activity.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for both Software and Other Level

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Compdf" Form

This form is used to relate a component name to a software name. Definitely, both should be previously defined in the above discussed forms.

This form contains four blocks. The first one includes the Component Type, the Software and the Component Names as well as the Software Version Number and the Component Revision Number. The second block includes the name of the person who created the component. The third block includes a detailed description of the component - what it does, what for used, etc. And the fourth block is used only when the Component is being related to the Software after a Perfective Maintenance activity.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]

- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, Component, and the Creator

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Struct" Form

This form is used to grant a component to a user with a special grant type. Also this form defines the Structure of each component; e.g., if the component is a table then this form allows the user to identify the fields of the table, their names, length, and a detailed description of each field.

Thus this form contains three blocks. The first includes the software name - that this component is related to - then the component name, as well as the version and revision numbers for the software and component, respectively. The second block includes the authorized users' names along with the grant type. And the last block includes the Structure of the tables and a definition and description of all the fields related to the component; Upon finishing the definition of one field, use [Previous Record] to move to the 'Field Name' field and press [Down Arrow] to define another field.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Down Arrow]
- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, Component, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Level1" Form

This form is used to describe or document modules of Level One, i.e., complete project.

This form contains five blocks. The first includes the Software Name and its Version Number. The second includes a detailed description of the Nature of the software under-development. The third block includes Estimates for the whole development process, which needs to specify each development Phase Name and its Estimated time resources to be spent on it. The fourth block includes the Hardware Requirements for the software to be safely installed in the user's environment. Where as the fifth block includes any Constraints on the software - this might be the number of users, etc.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field]
- Blocks, use [Next Record] and [Previous Record]
- Pages, use [Next Record] and [Previous Record]

In order to view a list of names for the Software

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Level2" Form

This form is used to describe or document modules of Level Two, i.e., Subsystems, programs, etc.

This form contains three blocks. The first includes the Software Name and its Version Number and the Level two Name and its Revision Number. The second block includes a detailed description of the Purpose of the "module" in-question. The third block includes the Description of this Functional "module",i.e., describing its Function.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field]
- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, "Modules"

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Level3" Form

This form is used to describe or document modules of Level Three, i.e., procedures and functions.

This form contains ten blocks. The first includes the Software Name and its Version Number and the Level three Name and its Revision Number. The second block includes a detailed description of the Purpose of the "module" in-question. The third block includes the Algorithm of this Source Code, of cource it is up to the user to tell how detailed the description of the algorithm should be. The fourth block includes the Names of the Staff Member personnel working on developing this "module" or source code. This block includes also the StartDate - of working - of each personnel on this source code.

The fifth block is for defining the components used by this source code, i.e., what tables or files, etc. The sixth block is used to define the Source Code Names that this source code calls (its Callees). The seventh block includes the Estimates for each development phase needed by this source code. The eighth block is used to define the Input Parameters used by this source code and the ninth block is used to list the Output Parameters produced from this source code. The Tenth block is used to enter any comment on any thing related to the source code in-question.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field]

- Records for the 'Developed by' field, use [Down Arrow]
- Blocks, use [Next Record] and [Previous Record]
- Pages, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, and "Modules"

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Actual" Form

This form is used to enter the Actual time resources taken by a specific source code - of Level three - during its assigned phases in development. Note that the Estimates were entered from the "Level3" form (block seven). This form is used also to define the tests performed on that specific source code.

This form contains four blocks. The First - as usual - includes the Software Name and its Version Number, and the Source code Name and its Revision Number. The second block includes the previously entered phases and their estimated time resources, and hence the user here has only to enter the Actual time resources. The third block includes the Tests performed on the Source Code and their Status. While the fourth block includes the Name of each person who has worked on this Source Code along with the StartDate, and here the user has only to enter the EndDate - date of finishing his/her work on that piece of Source Code.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field] on last Field
- Blocks, use [Next Record] and [Previous Record]
- Pages, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, Tests, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Config" Form

This form is used to Release a Configuration for a specific Software System.

It contains six blocks. The first block includes the Software Name that is to be Released. The second block includes the Configuration Release Number, along with its Status and the Name of the person authorized to Release it. The third block includes the entry of the Actual time resources spent in developing this software - during each phase. The Phases' names and the Estimates were already entered in “Level1” form. The fourth block includes the Integration and System Tests’ performed on the Software along with its Status. The fifth block includes the Name of the Client Institution that this software is Distributed to. Finally, the sixth block includes any Unfinished Work in this Release.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field] on last Field
- Blocks, use [Next Record] and [Previous Record]
- Pages, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, Tests, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

“Bug” Form

From its name it is apparent that this form is used to enter a Bug/Enhancement report. This report should be coming from a Client Institution.

This form includes three blocks. Obviously, the first block includes the Software Name that the report is about along with its Version Number. This block also includes the Report Identification - a Name given to each Report to identify it later on. The second block includes the Type of the Report - Bug or Enhancement (B/E) - and the Date reported. It also includes the Name of the Client Institution and the Name of its Employee who found the bug. The third block includes a detailed Description of the fault reported, in what condition, etc.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field]
- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, Change Proposals, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Proposal" Form

This form is used to enter a Change Proposal related to a previously defined Bug Report.

This form is composed of four blocks. The first block includes the Identification of the Change Proposal along with the Date of proposing the change. The second block includes information related to the Bug/Enhancement Report causing this Change Proposal. It also includes the Name of the Maintenance Person who Proposed this change. The third block includes a Description of the change and the fourth block includes a detailed Reason for Proposing this Change.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field]
- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, Change Approvals, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Approval" Form

This form is used to enter a Change Approval On a Change Proposal previously defined.

This form contains six blocks. The first block includes the Identificationof the Change Approval, and a Date of the Approval. The second block includes the Name of the Change Proposal related to this Approval, the Status of the Change Approval, the Name of the person responsible for establishing the baseline and of the person authorized to Approve the Change, and finally the Type of Change (corrective,etc.) The third block includes an Identification of the Change - a detailed one. The fourth block is used to state the Priority of Implementation and the fifth block includes the Consequences of the Changes if Not Implemented. Finally, the sixth block defines the Software Names that are Involved in the Change - they do not have to be affected, but involved.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field]
- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Specs" Form

This form is used to enter the Specifications of the Approved Changes.

This form contains seven blocks. The first includes an Identification of the Maintenance Specification and a Date. The second block includes the Name of the person who Formulated the Specifications, and the Name of the person who prepared the Baseline, the Status of the form, and the Change Approvals Identifications causing this Maintenance Specification. The third block Identifies the Changes required in this Maintenance step. The fourth block tells the Consequences of the Changes - what additional functionality, what errors it might cause, etc. The fifth block defined the Names of the Software Affected with this Change - to be changed later - and the sixth block defines the Names of

the Tests needed to be performed in order to ensure correctness, and the last block includes the Estimates of time resources for each needed phase in the Change.

Need to mention that in the Change Approval block, whenever the user declares a Change Approval Identification as Related - by assigning a 'Y' in the next field - a [Key Commit] is needed. At the beginning, all the Change Approval Identifications - not yet related to a maintenance specification - are displayed to the user allowing him/her to choose the ones related to this maintenance specification. They should be defined only once upon starting this form; i.e., upon defining the maintenance specification, because when querying the related information of this maintenance specification the tool displays in this block only the previously related ones and the block is skipped automatically.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field]
- Blocks, use [Next Record] and [Previous Record]
- Pages, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Design" Form

This form is used to define the actual Module Designs for the Changes requested. Each Source Code Name or Component Name defined previously, in the "Specs" form, as Affected will have to have its own Design screen so that the new changes are entered and hence Documentation is being consistent with the performed Changes.

This form contains three blocks. The first one includes the Name of the Software and its Version Number, and the Source Code/Component Name in-question and its Revision

Number. The Second block includes an Identification for this Design, a name that identifies it, and a Date, a Status, the Name of the person who established the baseline, and the Name of the person who is authorized to Design these changes. Also the Identification of the Maintenance Specification is entered in order to relate the Designs to a Specification. The third block is used only when a Perfective Maintenance is used.

This form leads to either one of four other forms if and only if the flag for Perfective maitenance is set to 'N' or blank; i.e., corrective maintenance where new design documents are needed, otherwise - if Perfective flag - 'Y' - then this means that already the user has defined the new features whether they be a new module, new source code, etc.

If the new Designs documents are for a "Module" of Level One then another screen/form will pop up namely "Sublev1" that has the same features as the "Level1" form. It serves for the Maintenance purposes, i.e., to enter the new changes on a particular Level1 Module. If the Designs are for a "Module" of Level Two then "Sublev2" form is popped up to allow the user to enter the New Purpose/Description of that "Module". If the Designs are for Level Three "Module" then "Sublev3" form is popped up to allow the user to enter Same Information as found in the "Level3" form, of course new ones. Finally, if the Designs are for a Component then "Subcomp" form is popped up to allow the user to re-describe the Component and Defines for it new Fields. If the purpose is to define new - additional - fields to the component, then all its fields - old and new - have to be re-entered again.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field]
- Blocks, use [Next Record] and [Previous Record]
- Pages, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Implement" Form

This form is a reflection of the actual Implementation of the Designs on the Source Code.

It has three blocks. The first one includes an Identification of this specific Source Code Implementation, a Date of Implementation, a Status, Names of the persons who respectively Implemented the Designs, Understood the Code after Implementation, prepared the Baseline. This block also includes the Identification of the related Module Design. the second block includes the Tests performed on the Source Code after implementing the new designs, and each Tests' Status and Date. The third block is a Comments on the work done.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field]
- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Mtactual" Form

This form is used to enter the Actual time resources taken by a specific source code - of Level three - during its assigned phases in Maintenance. Note that the Estimates were entered from the "Sublevel3" form.

This form contains three blocks. The First - as usual - includes the Software Name and its Version Number, and the Source code Name and its Revision Number and the Module

Design Identification that is related to the new changes of this Source Code. The second block includes the previously entered phases and their estimated time resources, and hence the user here has only to enter the Actual time resources. The third block includes the Name of each person who has worked on implementing the changes to this Source Code along with the StartDate, and here the user has only to enter the EndDate - date of finishing his/her maintenance on that piece of Source Code.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field] on last Field
- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, Level Three Modules, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Totactual" Form

This form is used to enter the Actual time resources taken by a set of maintenance actions grouped together under ***one*** maintenance specification.

This form contains three blocks. The First includes the Maintenance Specification Identification. The second block displays the previously entered phases and their estimated time resources, and hence the user here has only to enter the Actual time resources spent in each phase. The third block includes the Name of each Test that has been performed on the whole set of maintenance actions where the user has only to enter the Outcome of each test and the Date of performing it.

In order to navigate among

- Records, use [Next Field]
- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, Level Three Modules, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Mtconfig" Form

This form is used to Release a Configuration for a specific Software System after maintaining it. It is natural to assign to this Software a new Version Number.

It contains six blocks. The first block includes the Software Name that is to be Released along with its old Version Number. The second block includes the Configuration Release Number, along with its Status and the Name of the person authorized to Release it, and the New Version Number assigned to the software. Here the user is free to accept/reject this new Version Number because somtimes the changes are minor that no need for a new Version Number. This block also includes the Names of the Change Proposals related with this Release or that lead to this release. The third block includes the Name of the Client Institution that this software is Distributed/Re-Distributed to. The fourth block includes the entry of the Actual time resources spent in maintaining this software - during each phase. The Phases' names and the Estimates were already entered in "Specs" form. The fifth block includes the Integration and System Tests' performed on the Software along with the Tests' Status.. Finally, the sixth block includes any Unfinished Work in this Release.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Records, use [Next Field] on last Field
- Blocks, use [Next Record] and [Previous Record]
- Pages, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, Tests, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Postjob" Form

This form is used to update all the DataBase Tables after a Maintenance Configuration Release in case the user accepts the New Version Number of the Software. It relates all affected Source Codes/Components - and their documentations - to the New Version as well as assigning the Unmodified Source Codes/Components - and their documentations - to the New Version.

This form contains four blocks. The first includes the Name of the Maintenance Configuration Release. Once entered, the system displays the corresponding Level1 name and the Old Version Number and Displays the New Version Number. The second block includes the Names of any Source Code not needed - for a reason or another - in the new release. The third block includes the Names of any Components not needed in the New Version. And the last block includes one field (Y/N) telling the system to go ahead and update all the tables.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]

- Records, use [Next Field] on last Field

- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Software, Tests, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Reject" form

This form is used to either Reject a Change Proposal or Cancel it - even after all the phases have been performed.

This form contains three blocks. The first block includes the Identification of the ChangeRejection, and a Date. The second block includes the Name of the Change Proposal related to this Rejection, the Status of the Change Rejection, the Name of the person responsible for establishing the baseline and of the person authorized to Reject the Change Proposal. The third block is the Reason for Rejecting the Change proposal in question - a detailed one.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]
- Blocks, use [Next Record] and [Previous Record]

In order to view a list of names for the Change Proposal, etc.

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Addsrc" form

This form is used ***ONLY*** when the maintainer feels that a certain Source Code Module or Component needs to belong to a certain Version of a software system.

The form is so simple that it is composed of only one block. The name of the software system is required as well as its version number that the item in question is going to belong to. A flag is available to tell if the item is a source code or component. Of course, the source code/component name and revision number is needed.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]

In order to view a list of names for the Software Name,

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

"Rmsrc" form

This form is used when the maintainer feels that a certain Source Code Module or Component needs to belong to a certain Version of a software system.

The form is so simple that it is composed of only one block. The name of the software system is required as well as its version number that the item in question already belongs to and from which it is going to be deleted. A flag is available to tell if the item is a source code or component. Of course, the source code/component name and revision number is needed.

In order to navigate among

- Fields, use [Next Field] and [Previous Field]

In order to view a list of names for the Software Name,

- Press [List Values]

In order to commit

- Press [Commit]

In order to exit the form

- Press [Exit Form].

References

Babich, W. A. (1986) *Software Configuration Management: Coordination for team productivity.* Addison - Wesley, pp. 8, 72.

Bennett, K. H. (1991) The Software Maintenance of Large Software Systems: Management, Methods and Tools. In *Reliability Engineering and System Safety*, England: Elsevier Science Publishers Ltd, pp. 135-154.

Capretz, A. M. and Munro, M. (1992) COMFORM - A Software Maintenance Method Based on the Software Configuration Management Discipline. *In IEEE conf. on Software Engineering,* Durham, England: University of Durham, pp. 183-192.

Longstreet, D. H. (1990) Software Maintenance and Computers. *In IEEE Computer Society Press Tutorial,* pp. 2, 4, 9, 10-11.

Platoff, M., Wagner, M., and Camaratta, J. (1991) An Integrated Representation and Toolkit for the Maintenance of C Programs. *In IEEE conf. on Software Maintenance,* Princeton: Siemens Corporate Research, pp. 129-137.

Rajlich, V., Damaskinos, N., Linos, P., and Khorshid, W. (1990) VIFOR: A Tool for Software Maintenance. *In Software - Practice and Experience,* vol. 20(1), 67-77.

Ryder, B. G. (1989) ISMM - The Incremental Software Maintenance Manager. *In IEEE conf. on Software Maintenance,* New Brunswick, New Jersey: Rutgers University, pp. 142-162.

Sametinger, J. (1990) A Tool for the Maintenance of C++ Programs. *In IEEE conf. on Software Maintenance,* Linz, Austria: University of Linz, pp. 54-59.

Thayer, R. H. (1992) Tutorial: Software Engineering Project Management. *In IEEE Computer Society Press*, pp. 108-117.

Bibliography

General Electric Company, Corporate Information Systems (1986) Software Engineering Handbook, New York, pp. Chp. 1, Chp. 2, Chp. 8, Chp. 9, Chp. 10.

Schach, S. R. (1993) Software Engineering, second edition, Richard D. Irwin Inc., pp. 47, 449.

Thayer, R. and Dorfman, M. (1990) System and Software Requirements Engineering. *In IEEE Computer Society Press Tutorial*, Los Alamitos, pp. 4-16.

Thayer, R. (1992) Tutorial Software Engineering Project Management: Controlling a Software Engineering Project: Elements of Software Configuration Management. In E. H. Bersoff, *IEEE trans. on Software Engineering*, July 1986, pp. 744-751.

Longstreet, D. H. (1990) Software Configuration Management Tools: Change Management vs Change Control. In B. Moquin, *Proceedings of the 1985 IEEE Phoenix Conference on Computers and Communications*, pp. 97-100.

Longstreet, D. H. (1990) Software Configuration Management. R. L. Van Tilburg, *Proceedings of the 1985 IEEE Phoenix Conference on Computers and Communications*, pp. 117-121.

Longstreet, D. H. (1990) Experiences of Developing and Implementing a Configuration Management System for a Large Development Switching Systems. *Proceedings of the 1985 IEEE Phoenix Conference on Computers and Communications*, pp. 126-130.

Longstreet, D. H. (1990) Automated Configuration Management. A. Lobba, *Proceedings of the 1985 IEEE Phoenix Conference on Computers and Communications*, pp. 238-241.