To the Graduate Council:

I am submitting herewith a thesis written by John Philip Gibson entitled "Design and Implementation of the Cleanroom Process Database." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Jesse Poore, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor and
Dean of The Graduate School

DESIGN AND IMPLEMENTATION OF THE CLEANROOM PROCESS DATABASE

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

John Philip Gibson

May 1995

Copyright © 1995 by John Philip Gibson

All rights reserved

To Amy

ACKNOWLEDGMENTS

I want to thank Dr. Jesse Poore for providing me with the opportunity to work on this project and to be a part of his research team.

Sincere thanks also go to Dr. Carmen Trammell for the countless hours she spent reviewing my work. Her suggestions, positive attitude, and endless supply of snacks made the work so much more interesting and enjoyable.

Gratitude is also extended to Dr. David Straight for his service on the thesis committee.

I would like to wholeheartedly thank the entire research team at the Software Quality Research Laboratory: Gwen Walton, Stacy Prowell, Ingrid Rothenberg, Jenny Morales, and Sara Biery. Your suggestions and comments were a marvelous help throughout course of this project.

I would like to thank my parents and family back home in Australia. They continue to be a constant source of encouragement.

Lastly, I want to thank my wife, best friend, love, and psychiatrist Amy. Without her endless support and patience I would never had made it through.

ABSTRACT

This thesis presents the design and implementation for the Cleanroom Process Database (CPD), a database to support a Cleanroom approach to software engineering. In particular, the database supports the Cleanroom methodology as followed by the Software Quality Research Laboratory at the University of Tennessee and described in the "Software Quality Research Laboratory Cleanroom Process Description."

This work provides the data structures and data access methodology which allow for a complete development history of each project to be tracked from initiation to completion. The database has been designed to maintain process management metrics, development metrics, and graphical or textual documents used for reference or created during the course of development. By analyzing this collected data both within and across projects through appropriately designed reports, the overall software development process can be improved. Examples of such reports have been developed and are included in the thesis.

The specification for the user interface is provided in which all possible stimuli generated from the user interface are listed. A natural language specification is outlined in which the responses to these stimuli are described. The natural language specification is augmented with illustrations of the data entry screens and sample reports. A usage model for the system is included. Suggestions for future enhancements to the CPD are also listed.

The CPD was implemented as an Oracle Version 7.0 client/server relational database application on the SunOS 4.1.3 platform. A forms interface to the CPD was designed using Oracle Forms Version 2.0. Reports were designed using Oracle Reports Version 2.0.

Contents

Chapter 1 Introduction **1**
1.1 Cleanroom Software Engineering 1
1.2 SQRL Cleanroom Process Description 2
1.3 Objectives and Motivation 3
1.4 Overview 4

Chapter 2 Data Collection **5**
2.1 Process Description Data Structures 5
2.1.1 Process Data and Classification 6
2.1.2 Artifact Data and Classification 7
2.1.3 Instances of Artifact Categories 9
2.2 Artifact Categories 10

Chapter 3 Data Access **12**
3.1 Core Database Functions 12
3.1.1 Project Information 13
3.1.2 Data by Process 14
3.1.3 Data by Artifact 15
3.2 Classes of Database Users 15
3.2.1 Administrative Functions 16
3.2.2 SQRL User Functions 16
3.2.3 Joe User Functions 17

Chapter 4 Data Structures **18**
4.1 UTK Oracle Database Structure 18
4.2 Oracle Table Conventions 20
4.3 Relationships Among Tables 23
4.4 Database Table Definitions 24

4.5 Database Table Views. 48

Chapter 5 User Stimuli **51**
5.1 Stimuli Structure. 51
5.2 List of Stimuli. 52
5.2.1 Operating System/Application Interface 52
5.2.2 Logon Screen. 53
5.2.3 Logon Alert 53
5.2.4 Main Window 53
5.2.5 Browse Processes Window 54
5.2.6 Show Schema Window 55
5.2.7 Browse Artifacts Window 55
5.2.8 Projects Window 56
5.2.9 Modify Project Window 56
5.2.10 New Project Window. 56
5.2.11 Increments Window. 57
5.2.12 Modify Increments Window 57
5.2.13 New Increments Window 57
5.2.14 About Window. 57
5.2.15 Processes Window. 57
5.2.16 Artifact Categories Window 58
5.2.17 Artifact File Types Window. 58
5.2.18 File Summary Window 58
5.2.19 Class List Window. 59
5.2.20 Process List Window 59
5.2.21 Project List Window 59
5.2.22 In/Out List Window. 60
5.2.23 Increment List Window 60
5.2.24 File Type List Window 60
5.2.25 Run Report. 60
5.2.26 Run Form. 61
5.2.27 New File Window 61
5.2.28 Update File Window 61
5.2.29 Extract File Browser 62
5.2.30 Insert File Browser 62

Chapter 6 Natural Language Specification **63**
6.1 Menus. 63
6.1.1 Action Menu 64
6.1.2 Help Menu. 65
6.1.3 Oracle Default Menu 66
6.2 Generic Buttons 66
6.3 Windows of the CPD Interface. 66

6.3.1 Invocation and Logon 66
6.3.2 Main Window 68
6.3.3 Projects Window 68
6.3.4 Modify Project Window 70
6.3.5 New Project Window 70
6.3.6 Increments Window 71
6.3.7 Modify Increment Window 72
6.3.8 New Increment Window 73
6.3.9 About the CPD Window 73
6.3.10 Browse Processes Window 74
6.3.11 Browse Artifacts Window 78
6.3.12 Processes Window 81
6.3.13 File Type Window 82
6.3.14 Artifact Category Window 83
6.3.15 File Summary Window 85
6.3.16 New File Window 87
6.3.17 Update File Window 89
6.3.18 Extract File Browser 89
6.3.19 Insert File Browser 90
6.4 Forms Artifacts 90
6.4.1 Tasking Form 91
6.4.2 Verification Meeting Form 93
6.4.3 Verification Fault Form 96
6.4.4 Compilation Session Form 99
6.4.5 Compilation Fault Form 100
6.4.6 Testing Failure Form 103
6.4.7 Incremental Development Plan Form 105
6.4.8 Customer Interaction Meeting Form 106
6.4.9 Process Control Meeting Form 106
6.5 Reports Artifacts 107
6.5.1 Fault and Failure Distribution 108
6.5.2 Project Summary 109
6.5.3 Verification Summary 109
6.5.4 Compilation Summary 110
6.5.5 Testing Summary 110
6.5.6 Incremental Development Plan 112
6.6 Shell Utilities 113
6.7 Usage Model for the CPD 114
6.8 Mapping of Forms and Windows to Database Tables 127
6.9 Mapping of Reports to Database Tables 130

Chapter 7 Future Work **132**
7.1 Mechanism to Show Process and Artifact Structure 132

7.2 Dynamic Process Schema/Structure Generation 133
7.3 Enhanced Process-Driven Interface 133
7.4 Default Project and Increment Selection 134
7.5 Forms for Future Addition 134
7.6 Expanded Artifact Query Criteria 135
7.7 Clustering CPD Tables 135
7.8 Restricting Access to Artifacts Based on Type of Use 135
7.9 Updating Tables Directly Through Other Tools 136
7.10 Automated RCS Log Insertion 136

BIBLIOGRAPHY **137**

APPENDICES **139**

Appendix A Database Functions and User Privileges **140**
A.1 CPD Functions by Control Flow and User 141

Appendix B Usage Models **142**
B.1 Main Window Usage Model 143
B.2 Browse Processes Window [Without Structure] Usage Model 144
B.3 Browse Processes Window [With Structure] Usage Model 145
B.4 Browse Artifacts Window Usage Model 146
B.5 Browse Artifacts/Browse Processes Artifact Selection Usage Model . 147
B.6 Artifact Categories Usage Model 148
B.7 File Summary, New/Update File Artifact Usage Model 149

VITA **150**

List of Figures

6.1 Action Menu 64
6.2 Help Menu 65
6.1 Logon Screen 67
6.2 Logon Alert 67
6.3 Main Window 69
6.4 Projects Window 70
6.5 New Project Window 71
6.6 Increments Window 72
6.7 New Increment Window 73
6.8 About the CPD Window 74
6.9 Browse Processes Window 75
6.10 Process Structure Window 77
6.11 Browse Artifacts Window 79
6.12 Browse Artifacts Class Component List 81
6.13 Processes Window 82
6.14 Process Selection List 82
6.15 File Type Window 83
6.16 Artifact Category Window 84
6.17 Artifact Class Selection 85
6.18 In/Out List 85
6.19 File Summary Window 86
6.20 Artifact File Type List 87
6.21 New File Window 88
6.22 Extract File Browser 90
6.23 Tasking Form 91
6.24 List of Valid Projects 92
6.25 List of Valid Increments 92
6.26 List of Valid Tasking Options 92

6.27 Verification Meeting Form 93
6.28 List of Valid Compiled Versions of Code 94
6.29 List Containing Next Verification Meeting Number 94
6.30 List of Developers 95
6.31 Verification Fault Form 97
6.32 List Containing Next Verification Fault Number 98
6.33 Compilation Session Form 99
6.34 Compilation Fault Form 101
6.35 Testing Failure Form 104
6.36 Incremental Development Plan Form 105
6.37 Customer Interaction Meeting Form 107
6.38 Process Control Meeting Form 108
6.39 Fault and Failure Distribution 109
6.40 Project Summary 110
6.41 Verification Summary 111
6.42 Compilation Summary 111
6.43 Incremental Development Plan Parameter Form 112
6.44 Incremental Development Plan 113

List of Tables

4.1 PROJECT Table 25
4.2 INCREMENT Table 25
4.3 PROCESS Table 26
4.4 PROCESS_ARTIFACT Table 26
4.5 ARTIFACT_CATEGORY Table 27
4.6 PROJ_SPEC_ART_CAT Table 27
4.7 INC_SPEC_ART_CAT Table 28
4.8 ARTIFACT_CLASS Table 28
4.9 ARTIFACT_FILE_INSTANCE Table 29
4.10 PROJ_ART_FILE_INST Table 29
4.11 INC_ART_FILE_INST Table 30
4.12 FILE_TYPE Table Data 30
4.13 RCS_SUMMARY Table 31
4.14 RCS_VER_INFO Table 31
4.15 TASKING_SQRL Table 32
4.16 TASKING_PROJ Table 32
4.17 TASKING_INC Table 32
4.18 TASKING Table 33
4.19 INC_PLAN Table 33
4.20 INC_PLAN_FUNC Table 34
4.21 COMPILATION Table 34
4.22 COMP_MEET Table 35
4.23 TESTING Table 36
4.24 ENGINEERING_CHANGES Table 36
4.25 ENG_CH_FILENAMES Table 37

4.26 ENG_CHG_FUNCTIONS Table 37
4.27 ENG_CH_VERIFIERS Table 38
4.28 VERIFICATION Table 39
4.29 VERF_VERIFIERS Table 39
4.30 VERF_FILENAMES Table 40
4.31 VERF_FUNCTIONS Table 40
4.32 VER_MEET Table 41
4.33 VER_MEET_VERIFIERS Table 41
4.34 VER_MEET_FILENAMES Table 42
4.35 PROJECT_TOTAL Table 43
4.36 INCREMENT_TOTAL Table 44
4.37 CUST_INT Table 45
4.38 CUST_SQRL_ATTEND Table 45
4.39 CUSTOMER_ATTEND Table 46
4.40 CONTROL_CHECK Table 46
4.41 CONTROL_SQRL_ATTEND Table 47
4.42 CONTROL_STDS Table 47
4.43 PROC_ART_TYPE View 48
4.44 BROWSE_ART View 49
4.45 BROWSE_ART_INST View 49
4.46 SQRL_TASKS View 50
6.1 List of Usage Model States and Arcs 114
6.2 Mapping Forms and Windows to Tables 127
6.3 Mapping Reports to Tables 130

Chapter 1

Introduction

This chapter begins with a brief overview of the main ideas embodied in Cleanroom Software Engineering. Next, the Software Quality Research Laboratory Cleanroom Process Description is outlined followed by the identification of the objectives and scope of the thesis. The chapter concludes with an outline of the structure of the thesis.

1.1 Cleanroom Software Engineering

The Cleanroom method of software engineering, hereafter referred to as Cleanroom, is a process for developing high quality software with certified reliability. Cleanroom covers the complete software development life cycle including planning, specification, verification, coding, testing, and certification (Hausler et al., 1994). The focus of Cleanroom is on error prevention. The operational goal of Cleanroom is the production of zero defect software prior to the first execution through rigorous design and verification. Cleanroom relies on formal methods and correctness verification in order to create a correct design and uses statistical methods to certify field performance (Poore,

1990). Cleanroom recommends a top-down, incremental approach with correctness verification required at each step (Dyer, 1993).

Specific practices that implement the principles of Cleanroom must be tailored by each team for the specific environment. Such a process description has been developed and documented by the Software Quality Research Laboratory (SQRL) at the University of Tennessee.

1.2 SQRL Cleanroom Process Description

The SQRL Cleanroom Process Description, hereafter referred to as the Process Description, defines a set of processes and artifacts through which (1) a software product of certified reliability is produced, (2) the practice of the Cleanroom method is improved, and (3) a design record and development history is preserved. A hierarchy of high-level and low-level Cleanroom processes is described. High-level processes are those which contain subprocesses whereas low-level processes are those which cannot be further decomposed. Low-level processes are termed "leaf" processes. High-level processes are represented by a process schema for concurrent, cooperating processes. Control flow among sub-processes in the schema is specified as either concurrent, sequential, alternative, or iterative.

Process artifacts are described in terms of their content, origin, use, and maintenance. Process artifacts are divided into three categories: input, inout, and output. Input artifacts predate the process, are used by the process and are not changed by it. Inout artifacts predate or are created by the process, and are returned for future use outside the process. Output artifacts are created by the process and returned for use outside it.

1.3 Objectives and Motivation

Although many major software systems have been designed using the Cleanroom method, tools to assist in its implementation have only recently begun to emerge. The Process Description is providing the much needed structure on which tools may be built. This thesis describes an effort to design and implement the Cleanroom Process database (CPD). The database is to be used as a support tool to the Cleanroom methodology followed by the Software Quality Research Laboratory at the University of Tennessee as described in the Process Description.

The CPD data structure provides the means by which a complete development history can be maintained for each Cleanroom project undertaken by the SQRL. The data to be stored includes process management metrics, project development metrics, and graphical or textual documents used for reference or created during the course of development. Code may be archived in the database at the completion of an increment of development.

The CPD graphical interface was designed to accommodate access to the development data in an efficient and logical manner. A comprehensive set of data entry screens was designed through which data on all aspects of development can be entered or queried from the database. Various reports have been designed as part of the CPD. By analyzing this collected data both within and across projects through appropriately designed reports, the overall software development process can be improved.

The database should speed the software development process in three ways. First, wherever possible, database interaction has been designed to be "process-driven" which assists the user in adhering to the Cleanroom practices of the Process Description. Second, the database facilitates an organized method of access to the process and project data. Third, it provides automation for many of the record keeping and metrics collection tasks that are currently performed by hand.

1.4 Overview

Chapters 2 and 3 provide an overview of the data collected in the CPD and of the data access scheme. Chapter 4 provides a concrete illustration of the data maintained by the database by describing the structure and content of each of the tables and views of the CPD. The user stimuli to the system are listed in Chapter 5. Chapter 6 provides the natural language specification for the system and also includes the CPD usage model and the mapping of forms and reports to the data structures of Chapter 4. Suggested future modifications and enhancements to the system are discussed in the final chapter.

Chapter 2

Data Collection

This chapter describes the data collection requirements for the CPD. First, the data structures of the Process Description are presented and classified. Second, artifact categories which have been designed as Oracle Forms or Oracle Reports applications are listed.

2.1 Process Description Data Structures

The Process Description defines two major structures in describing Cleanroom practiced by the SQRL at the University of Tennessee: a set of Cleanroom processes, and a set of Cleanroom artifacts. The following sections describe the data corresponding to these structures which is collected and maintained in the CPD. Considerations and extensions to the Process Description necessary to map the structures of the Process Description to database form are discussed.

2.1.1 Process Data and Classification

The Cleanroom processes are presented in the Process Description as high-level processes which are continually decomposed in a stepwise fashion into their constituent lower-level processes to form a process hierarchy. The ability to enhance the process is built into the structure of the Process Description (a leaf process called "Process Review and Improvement" has been defined for this very reason). In order to tailor Cleanroom to other process environments, the database has the flexibility of being able to redefine relationships between processes, process definitions, and the addition and deletion of processes. Thus, the basic process structure is maintained within tables of the CPD. A forms interface to the tables is provided which is flexible enough to accommodate changes to the process hierarchy. Process properties include:

- the process *name,*
- a *"leaf"* property to indicate a lower-level process,
- an entry indicating the *parent* of the process,
- the process *number* at a level in the hierarchy, and
- a brief *description* of the process.

From this information, the database interface builds the structure of the process hierarchy and allows for navigation between high-level and low-level processes.

The process schema described in the Process Description are statically built into the interface for the CPD. Additional process information concerning process interrelationships needs to be maintained in the database in order to dynamically create the process schema. These more complex process interrelationships are presented in Chapter 7 as items for future implementation.

2.1.2 Artifact Data and Classification

The Process Description defines its artifacts in terms of their content, origin, use, and maintenance. The database defines its artifacts in a similar manner, yet, some slight extensions to the Process Description artifact classification were required. The artifact characteristics which are now defined include:

- the *name* for the artifact category,
- the list of the *Cleanroom processes* that use the artifact category,
- the *in/out* use of an artifact by a process,
- the artifact *type*,
- the *parent class* to which an artifact category belongs,
- the textual description of the *content* of the artifact, and
- the *scope* in which an artifact is defined.

The first three artifact characteristics are defined in the Process Description. Note that instances of an artifact category can be used by multiple processes. The in/out use by a given process is one of input, inout, or output. Restrictions on the in/out use are not enforced in the CPD and are used only for the informative purposes.

Different types of artifacts are required to be stored and retrieved from the database in different ways. Consider the existing artifacts *Verification Fault Record Form*, *CDL Documentation*, and *Incremental Development Plan* for example. The *Verification Fault Form* is an Oracle Forms application. Whenever the user wishes to reference the *Verification Fault Form*, the CPD must know to execute the appropriate Oracle Forms application. *CDL Documentation* exists as a FrameMaker file used for reference when using CDL. If the user wishes to refer to this document then the CPD needs to know that the artifact is a file that must be extracted from the database and output to a printer, to the screen, or to the filesystem. Similarly, the CPD needs to know special information in order to view an *Incremental Development Plan* artifact which exists as an Oracle Reports appli-

cation. In addition, the content of some artifacts described in the Process Description is merely a collection of other artifacts. Consider *Reference Material* for example. *Reference Material* is an artifact category which identifies a group of component artifact categories which are used for reference over all projects. The *CDL Documentation* artifact is one of the components of the *Reference Material.* Thus, the characteristic of artifact *type* is defined.

Broadly, artifact categories are one of four types: report, form, file, or class. A *report* artifact type is a report which has been designed as an Oracle Reports application. Similarly, a *forms* artifact is a pre-defined Oracle Forms application. Other artifacts which are not of type form or report may be a certain type of *file*. For example, many documents entered into the database will be FrameMaker files and most code entered into the database will be C or CDL files. Additional file types can easily be specified. Finally, an artifact category may be of type *class*. A *class* of artifacts is a named group of artifact categories which can be used collectively as either input, inout or output to a given process step. Only a loose relationship existed in the Process Description between the type of use of artifact classes at a given process step, and the components of the artifact class itself. A more complete artifact hierarchy similar to the process hierarchy was developed in which artifact categories belong directly to one and only one *parent* class.

A core group of artifacts has been defined in the Process Description for collection or reference during the course of a Cleanroom Project. Although this list of artifacts is comprehensive, not all data to be collected for every project can easily fit into any one of the defined artifact categories. Certain data may be specific to a particular project or increment of a project. Hence it was useful to specify a sixth artifact characteristic in order to classify artifacts in terms of their *scope* of use. The *scope* of an artifact determines the maximum level to which access to an instance of artifact is restricted. The valid scope values are all-projects, project-specific, and increment-specific. *All-projects* artifacts belong

to the core group of artifacts defined in the process description. *Project-specific* artifacts are those which do not match any of the artifact category definitions of the Process Description and access to these artifacts is restricted to all increments of a specific project. *Increment-specific* artifacts are those that do not match any of the artifact category definitions from the process description and for which access is restricted to a specific increment of a particular project. The scope property is useful in that the user of the CPD can, in addition to the core artifact group, list only artifacts specific to the particular project or increment of interest. Thus unnecessary information can be pruned from an artifact query.

2.1.3 Instances of Artifact Categories

The artifacts defined in the Process Description represent certain categories of data that are to be collected, referenced, or updated at various steps in the Cleanroom Process. Multiple instances of an artifact category can be added to the database and these instances can be mapped to specific projects or specific increments of projects.

Artifact categories classified as *all-projects* have the greatest flexibility in that each instance can be either defined to be referenced by all projects, a specific project, a set of projects, or a specific increment or set of increments for a selected project. For example, consider the *Usage Model* artifact category. This artifact category is classified as all-projects. For a given project, however, it might be useful to have a usage model representation of each increment of each project in addition to a usage model for the entire project. Furthermore, the user may wish to develop a "stimulus-as-state" usage model and a "stimulus-as-arc" usage model. Each of these usage models would be an instantiation of the *Usage Model* artifact category and each could be tied to a specific project or specific increment of a project.

Instances of *project-specific* artifact categories can be defined for reference by either all increments of the project specified in the artifact category definition, by a specific increment or by a collection of increments of work on the project. For example, a signal chart might be required for reference in developing controller software for company X. This signal chart would be specific to this project and of no use in any other project that is undertaken. A new project-specific artifact category could be developed and tied directly to project X. Different signal charts may be required for different increments of work for this project, so instances of the artifact category could be tied to specific increments of the artifact specified in the artifact category definition.

Increment-specific artifact categories could have one or more instantiations, access to which is restricted to the increment and project pair specified in the artifact category definition.

2.2 Artifact Categories

Several existing artifacts naturally lend themselves to forms or reports which could be generated from tables of data. For example, the *Verification Fault Form* which is typically filled out at a verification session can be mapped to database tables and ultimately to an Oracle Forms application. A *Verification Fault Summary* can be generated as an Oracle Reports application from the data contained within these tables. Other artifacts do not lend themselves as easily to tabular form. An example is *CDL Preprocessor Documentation* which exists as a FrameMaker document used for reference at various stages in the Cleanroom Process. Artifacts such as this are stored in the database as instances of an artifact category of *file* type. The appropriate file type is specified in the instance definition. Artifact instances such as these may be retrieved for reference on command.

The following artifact categories have been designed as Oracle Forms applications.

- Compilation Fault Form
- Compilation Fault Records
- Compilation Session Form
- Compilation Session Records
- Customer Interaction Meeting Form
- Customer Interaction Meeting Records
- Increment Development Plan Form
- Verification Fault Form
- Verification Fault Records
- Verification Meeting Form
- Verification Meeting Records
- Tasking Form
- Tasking Records
- Testing Failure Form
- Testing Failure Records

The following artifact categories have been designed as Oracle Reports applications.

- Compilation Fault Summary
- Incremental Development Plan
- Fault and Failure Distribution
- Project Summary
- Testing Failure Summary
- Verification Fault Summary

A textual and graphical description of these new artifact categories is provided in Chapter 6.

Chapter 3

Data Access

This chapter describes the method of access to the CPD data and the impetus for the method. The major functions of the database are identified and mapped to the flow of control through the CPD interface. Also, the partitioning of the system functionality among the anticipated kinds of users of the system is discussed.

3.1 Core Database Functions

A primary concern of the CPD is to facilitate organized access to the Cleanroom artifacts. This includes access to artifacts by the classification criteria outlined in Chapter 2. Additionally, instances of artifact categories are available by project for project-specific artifacts, or by a combination of project and increment for increment-specific artifacts. The database also stores the structure of the Cleanroom process hierarchy and relationships between Cleanroom artifact categories and the Cleanroom processes. The following list represents the basic functionality of the database.

- Define/Modify a Cleanroom process and the process hierarchy
- View the Cleanroom process hierarchy
- Define/Modify a Cleanroom artifact category

- View Cleanroom artifact category properties
- Define a new project / Modify an existing project definition
- View the project descriptions / Select a project with which to work
- Define a new increment of an existing project / Modify an existing Increment definition
- View the increment descriptions / Select an increment with which to work
- Define/Modify a valid Cleanroom artifact file type
- View artifact type properties
- Insert/Modify/Delete an instance of a defined Cleanroom artifact category
- Extract/view/print any existing instance of a Cleanroom artifact category

In accordance with the two major structures defined in the Process Description, data access in the CPD follows two major threads of control, data manipulation by Cleanroom process, and data manipulation by Cleanroom artifact. Before forking along either of these paths, project and increment information are available to the user.

3.1.1 Project Information

No matter if the user is beginning a new project, a new increment of a project, or accessing existing data, the first operation performed is to access project or increment information. Defining a new project is done upon initiation of a new CPD session. The user can also select a project or a project and increment with which to work. Thus, the following list represents the functionality that is available from the *Main Window* before the flow of control forks to follow either the artifact or the process path.

- Define a new project / Modify an existing project definition
- View the project descriptions / Select a project with which to work
- Define a new increment of an existing project / Modify an existing Increment definition
- View the increment descriptions / Select an increment with which to work

3.1.2 Data by Process

Wherever possible, it is desired that the CPD assists the engineer in the development of software within the Cleanroom framework. Hence, one thread of control is driven by the Cleanroom processes as specified in the Process Description. Regardless of the stage of development, the CPD prompts the user with the relevant artifacts which need to be accessed. The CPD lists the artifacts categorized by the rules governing the type of use of a given artifact at a certain process step, i.e., input, inout, and output. An example is evident in the *Testing* process. The user informs the CPD that the stage of development of project "X" has reached *Testing* and is prompted with a list of all input, inout, and output artifacts. Listed input artifact categories include *Software and Documentation*, *Testing Failure Form*, *Testing Policies*, and the *Top-Level Box Structure Specification*. Listed inout artifact categories include *Testing Failure Records*. No output artifact categories exist for this process.

The following list represents functionality available to the user along the process thread of control either through menu options or as responses to stimuli associated with other interface elements.

- Define/Modify a Cleanroom process and the process hierarchy
- View the Cleanroom process hierarchy
- Define/Modify a Cleanroom artifact category
- View Cleanroom artifact category properties
- View the project descriptions / Select a project with which to work
- View the increment descriptions / Select an increment with which to work
- Define/modify a valid Cleanroom artifact file type
- View artifact type properties
- Insert/Modify/Delete an instance of a defined Cleanroom artifact category
- Extract/view/print any existing instance of a Cleanroom artifact category

3.1.3 Data by Artifact

Access to artifacts through the process thread of control is restricted to those artifacts pertaining to a particular project or project and increment combination. This presents a problem regarding speed of database access. An engineer wishing to update an instance of a known artifact category must navigate up and down the Process hierarchy in order to access the artifact. It is clear that a second thread of control is necessary, that is to access data by artifact characteristic.

For ease of access to the artifacts, the CPD has the ability to access artifacts across projects and increments and also according to the classification criteria listed in Chapter 2. So, for example, if a developer wishes to view all reports over multiple projects, then he/she may query the database for all artifacts of report type for all projects and proceed to select those of interest. The following list represents the functionality available through the artifact thread of control either through menu options or as responses to stimuli associated with other interface elements.

- Define/Modify a Cleanroom artifact category
- View Cleanroom artifact category properties
- View the project descriptions / Select a project with which to work
- View the increment descriptions / Select an increment with which to work
- Define/modify a valid Cleanroom artifact file type
- View artifact type properties
- Insert/Modify/Delete an instance of a defined Cleanroom artifact category
- Extract/view/print any existing instance of a Cleanroom artifact category

3.2 Classes of Database Users

Three different types of users are anticipated for the CPD, the administrative user, the SQRL user, and Joe user. The differences in the functional profiles of each of these

users is outlined in the following sections. A summary of the functionality partitioning may be found in Table A.1.

3.2.1 Administrative Functions

Certain sensitive database functions go beyond the typical actions anticipated for normal interaction with the database during project development. The Cleanroom process structure should be stable thus it should be infrequently modified. A set of core Cleanroom artifacts are defined by the Process Description. Any project development data to be collected should, most often, match one of these core artifacts making the definition of new categories of artifacts a special occurrence.

The administrative user has unlimited access to all database functions. Any function that pertains to the modification of the Cleanroom process structure including artifact category definition is restricted to a user with administrative privileges. The following list compiles those functions that are peculiar to the administrative user.

- Define/Modify a Cleanroom process and the process hierarchy
- Define/Modify a Cleanroom artifact category
- Definition/Modification of valid artifact file types

3.2.2 SQRL User Functions

If an engineer uses the database in conjunction with a development project, normal database operations are conducted with the privileges of a SQRL User. The following list compiles those functions that are peculiar to a SQRL User.

- Insert/Modify/Delete an instance of a defined Cleanroom artifact category
- Define a new project / Modify an existing project definition
- Define a new increment of an existing project / Modify an existing Increment definition

3.2.3 Joe User Functions

There may be other users of the system to which privileges may be further restricted. Joe User is afforded "read-only" access to all information in the database. This may be useful if, for example, the CPD was used in conjunction with a software engineering class for obtaining data on error distribution across stages of development of certain projects. Thus, Joe User can generate reports, query data, and view artifacts but he/she is not permitted to alter or add any data to the database.

Chapter 4

Data Structures

An overview of the implementational details of the database structure is provided in Chapter 4. A brief description of the architecture of the UTKCS_DB Oracle database installation is initially provided. Conventions used in table definitions are then discussed, followed by a description of each of the CPD tables and the relationships among them. Database views and triggers are also listed.

4.1 UTK Oracle Database Structure

The CPD is designed as an Oracle7 client-server relational database application. However, both the Oracle server and application clients are stationed on a single machine running SunOS 4.1.3. The CPD database tables are created with the use of an SQL*Plus batch file. The interface to the application was designed using Oracle Forms Version 4.0 and Oracle Reports Version 2.0. Manipulation of data in the CPD is achieved by way of Oracle SQL. The SQL code is laid out in the Oracle Forms designer, as well as within pre-compiled blocks of PL/SQL and Pro*C code. PL/SQL is Oracle's procedural language extension to SQL. Pro*C contains extensions to the C language which allow for embedded SQL code.

Environment Variables. In order to use the CPD, the following list of environment variables must be modified or initialized.

```
Add to your PATH:     /cedar/homes/oracle/product/7.0.16/bin
Add to your LD_LIBRARY_PATH: /cedar/homes/oracle/product/
      7.0.16/lib
setenv ORACLE_TERM ncd
setenv ORACLE_HOME /cedar/homes/oracle/product/7.0.16
setenv ORACLE_SID UTKCS_DB
setenv REP_PATH /cedar/homes/sqrl/cpd/reports
TK2_ICON /cedar/homes/sqrl/cpd/icons
setenv ORACLE_PATH /cedar/homes/sqrl/cpd:/cedar/homes/sqrl/
      cpd/forms:/cedar/homes/sqrl/cpd/bitmaps
```

ORACLE_HOME specifies the location of the Oracle installation on the system. ORACLE_TERM sets up correct terminal emulation. Refer to the ORACLE for UNIX Documentation Set for the correct value for this environment variable for the terminal from which the CPD is invoked. ORACLE_SID specifies the name of the database with which to connect. REP_PATH specifies the search path that Oracle should use when attempting to locate a report to run from within a form application. It also specifies the path to search for other external objects used in reports such as PL/SQL libraries, external queries, and external boilerplate. TK2_ICON specifies the search path that Oracle should use when attempting to load any icons into the forms of the interface to the application. ORACLE_PATH specifies the path Oracle should search when attempting to run a form or report from the command line.

Parameter Files. The initialization parameter file for the UTKCS_DB database is located in

```
$ORACLE_HOME/dbs/initUTKCS_DB.ora.
```

Log Files. The log files for the UTKCS_DB database are

```
/cedar/homes/oracle/data/UTKCS_DB/UTKCS_DB01.log, and
/cedar/homes/oracle/data/UTKCS_DB/UTKCS_DB02.log.
```

Tablespaces and Data Files. A temporary tablespace named TEMP was created for the temporary segments. A tablespace named SQRL_CPD holds the CPD data. A tablespace named RB_SEGS was created to hold the rollback segments for the database. The data files for the SYSTEM, TEMP, RB_SEGS, and SQRL_CPD tablespaces of the UTKCS_DB database are

```
/cedar/homes/oracle/data/UTKCS_DB/UTKCS_DB_system01.dbf,
/cedar/homes/oracle/data/UTKCS_DB/temp01.dbf,
/cedar/homes/oracle/data/UTKCS_DB/rbs01.dbf, and
/cedar/homes/oracle/data/UTKCS_DB/SQRL1.dbf.
```

Security Roles. Several security roles were created. The BASIC_USER role is granted to all users. The CPD_ADMIN role allows a user full query, insert, update, and delete access to all tables of the CPD. The CPD_SQRL role allows query, update, and insert privileges to all tables other than those holding process and artifact category information, to which query-only access is provided. Such tables include PROCESS, ARTIFACT_CATEGORY, PROJ_SPEC_ART_CAT, INC_SPEC_ART_CAT, PROCESS_ART_CAT, ART_CLASS, and FILE_TYPE. Query-only access on these tables is granted to the CPD_SQRL role. The CPD_JOE role affords query-only access to all tables of the CPD.

4.2 Oracle Table Conventions

All table names are presented in upper case with underscores inserted for readability. Table field names are presented with the first letter capitalized and the remainder in lower case. If, for descriptive purposes, it is desired that a field name be composed of two words, the two words are compressed into one and the first letter of each is capitalized. For example, the increment number field appears as *IncNo*.

The native Oracle7 data types that are used are VARCHAR2(*size*), NUMBER(*size,d*), DATE, and LONG RAW. The VARCHAR2 datatype stores alphanumeric

data with a variable length of up to *size* bytes (1 <= *size* <= 2000). NUMBER stores fixed and floating point numbers with up to 38 digits of precision. The precision specifies the total number of digits. The scale specifies the number of digits to the right of the decimal point. The specification of precision and scale is optional. DATE stores valid dates ranging from January 1, 4712 B.C. to December 31, 4712 A.D. Oracle7 allows for several date formats to be used. The standard Oracle7 default date format is DD-MON-YY. The TO_DATE(*string, "format"*) and TO_CHAR(*string, "format"*) functions are provided to convert dates to and from many other formats. For consistency, the MM/DD/YY format is used for dates throughout the application. The HH:MI AM format is used for all fields which require the time to be inserted. LONG RAW data is for binary data that is not to be interpreted such as graphics, sound, and documents. LONG RAW data cannot be indexed.

As required by Oracle7 *entity integrity*, each table contains a number of *primary keys*, the minimum keys necessary to uniquely identify a record in the table (Bobrowski, 1994). In the data definition file, any *primary key* constraints are listed at the conclusion of the table definition, for example,

```
create table INCREMENT (
ProjName     varchar2(50),
IncNum       number,
StartDate    date,
FinishDate   date,
Description varchar2(1000),
PRIMARY KEY(ProjName, IncNum)
 );
```

Oracle *indexes* are created on many of the tables for fast, direct access to a specific record. Multiple tables are often joined together to create the specific combinations of data that are needed for the various Process steps and Artifacts. Indexes are placed on the fields of the tables that are present in the *where* clause of the join provided the fields are not primary or unique keys. Oracle7 automatically places Indexes on the primary keys and on fields which are part of a *unique* constraint (Bobrowski, 1994). Indexes are placed on for-

eign keys provided a list of values is not used to validate data in the Forms interface. An example of the creation of an index on the ProjName field of the PROJECT table is provided below:

```
create index PROJECT_ProjName on PROJECT(ProjName);
```

A list of these indexed fields is provided for each table. Note that each index must have a unique name. Where possible, the convention for naming indexes is to combine the table name and field names separated by an underscore. Since the maximum length possible for an index name is 30 characters, only the letters in the field names before the second capital letter are included in the index definition for indexes on many fields, e.g., if an index were to be placed on the ProjName and IncNum fields of the INCREMENT table would be named INCREMENT_ProjInc.

Validation of data may be performed at the server, commit time, the client application when inserted into a field, or at both the client and the server. Constraints may be placed on fields in table entries to validate entries. For example, in the creation of the VERIFICATION table the following constraint was included in the definition of the *Stage* field in the table to be sure that they only take on one of a specified list of values.

```
Stage       varchar2(17)
            check          (stage in('Original Design',
      'Design Change', 'Compilation Error', 'Testing Fail-
      ure'  )
                           ),
```

Thus, attempted insertion of a value not matching those listed would result in failure. However, this arrangement presents certain problems to the client-server nature of Oracle. That is, a by-product of this failure would be an increase in network traffic as well as more work for the server in determining whether the insertion should be allowed. A message would be sent across the network to the server containing the data to be inserted and a message sent back to the client containing the result of the transaction. This list of values may be defined at the client instead of, or as well as, at the server. The list of values may

then be displayed in a pop-up listbox in the Form application, allowing the user to select from valid options. Most validation has been designed to occur at the client application rather than at the server. This reduces both the load on the server and network overhead (Bobrowski, 1994). The *unique* constraint is placed on fields or sets of fields in certain tables to assure that a unique value or set of values is maintained for all records in the table.

In the context of the Oracle7 server, a trigger defines an action, which is defined by a PL/SQL code block, that occurs in response to some pre-defined database event such as insert, update, and delete events on a table. An example of a trigger might be the performance of some complex security or data integrity rules before inserting data into a table. In the context of Oracle Forms Version 4.0, the range of values that defines a database event is extended to include interface events, for example, the pressing of a button on the interface. Once again to reduce the strain placed on the server, most triggers are defined in the form application rather than at the server.

4.3 Relationships Among Tables

Primary key to *foreign key* relationships are defined between the fields of many of the tables of the CPD. Defining foreign keys is a method used to make sure that data in related columns of different tables remain consistent. The values of a primary key field of a table, the master table, must match the value in a field of another table, the detail table. The field in the detail table, which references the master table primary key, is called the foreign key. For example, the ProjName field in the INCREMENT table references the ProjName field of the PROJECT table. Thus, a user must signal the start of a new project by inserting an entry for the project into the PROJECT table before entering data for any increment of that project.

Other constraints may be placed on tables. The *on delete cascade* constraint is placed on records in a detail table which has a foreign key constraint. This constraint assures that records in the detail table are deleted when records in the master table with fields corresponding to the foreign key constraint of the detail table are deleted.

Clustering is a method of storing closely related tables on the same area of the disk. An example of tables that are closely related is the case of data being queried from a form in which a Master-Detail relationship is defined between the blocks corresponding to two separate tables. Clustering allows the rows of two tables to be interleaved together in a single area called the cluster. A unique cluster key identifies the field or fields by which the tables are usually joined in a query. Clustering tables in Oracle7 reportedly has minimal benefits (Bobrowski, 1994). Thus clustering has not been performed on the CPD tables and is suggested in Chapter 7 as an option for possible future implementation.

4.4 Database Table Definitions

This section provides a brief overview of the composition and use of each of the tables. Most of the tables are updated by manual insertion of records through the data entry screens of Oracle Forms Version 4.0. Some tables are be updated through user exits attached to objects in the forms which load files from the filesystem. Data compression is performed before loading the binary files into the database. A few tables are loaded from pre-formatted data contained in ascii files using Oracle's *sqlload* utility. A description of the use of each Oracle table of the CPD is provided followed by a presentation of the data structure. The data structure is presented in the following three field format: the *field name*, the *data type* of the field, and a brief *description* of the field. Primary key fields are displayed in bold and foreign key fields are italicized.

Table 4.1 lists the data contained in the PROJECT table. It maintains one entry for each project for which there is data in the CPD.

Table 4.1: PROJECT Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
StartDate	date	Start date of project
FinishDate	date	Completion date of project
Description	varchar2(1000)	Brief description of the project

Table 4.2 lists the data contained in the INCREMENT table. It maintains one record for each increment of a project for which there is an entry in the PROJECT table. The ProjName field of this table references the ProjName field of the PROJECT table.

Table 4.2: INCREMENT Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
StartDate	date	Start date of increment
FinishDate	date	Completion date of increment
Description	varchar2(1000)	Brief description of the increment

Table 4.3 lists the data contained in the PROCESS table. It maintains one entry for each Cleanroom process. The structure of the Cleanroom process hierarchy is built from the entries in this table. An Oracle radio group is used in the forms interface to restrict the value of the Leaf field to the appropriate value. The value of the Leaf field is NULL for any high-level process, and "(leaf)" for a low-level process. Triggers in the forms interface are used to maintain the validity of the ProcNum field upon insertion of a new process or deletion of an existing process. The value of ParentProc is NULL for the process at the

root of the Cleanroom process tree. A list of values is generated in the forms interface to ensure that ParentProc entries are restricted to defined processes.

Table 4.3: PROCESS Table

Field Name	Data Type	Description
ProcName	varchar2(50)	Name of the Cleanroom process step
ParentProc	varchar2(50)	Parent Process (higher in the process hierarchy)
ProcNum	number(3)	Number of the process at this level
Leaf	varchar2(6)	Leaf property of a process
Description	varchar2(1000)	Brief description of the process

Table 4.4 lists the data contained in the PROCESS_ARTIFACT table. It maintains one entry for each category of Cleanroom artifacts used by a given Cleanroom process. It also contains an entry for the type of use. The ProcName field of this table references the ProcName field of the PROCESS Table. A constraint is placed on the UseType field so that the type of use is one of input, inout, or output. A foreign key constraint is placed on the ArtCatName field so that an entry in this field must match an entry in the ArtCatName field of the ARTIFACT_CATEGORY table. The PROCESS_ARTIFACT_UseType index exists on the UseType field since this field is present in the *where* clause of a query used in the *Browse Processes Window.*

Table 4.4: PROCESS_ARTIFACT Table

Field Name	Data Type	Description
ProcName	varchar2(50)	Name of the Cleanroom process step
ArtName	varchar2(50)	Name of artifact required by the process step
UseType	varchar2(6)	The type of use of this artifact at this step (One of input, inout, output)

Table 4.5 lists the data contained in the ARTIFACT_CATEGORY table. It maintains one record for each artifact category defined in the Process Description. Special con-

straints are placed on the Scope and ArtType fields. Valid values for Scope are "Top-Level", "Project-Specific", and "Increment-Specific". Valid values for ArtType are "Form", "Report", "File", and "Class". An Oracle radio group is used in the forms interface to restrict the values entered into these fields to those of a valid type. The ARTIFACT_CATEGORY_Scope and ARTIFACT_CATEGORY_ArtType indexes exist on the Scope and ArtType fields of this table since these fields are present in the where clause of queries made from the *Browse Artifacts Window.*

Table 4.5: ARTIFACT_CATEGORY Table

Field Name	Data Type	Description
ArtCatName	varchar2(50)	Name of the Artifact Category
Scope	varchar2(18)	Scope of the artifact category
ArtType	varchar2(6)	Type of the artifact
ExecName	varchar2(30)	Executable for the artifact (if a form or report type)
Description	varchar2(1000)	Description of the artifact category

Table 4.6 lists the data contained in the PROJ_SPEC_ART_CAT table. It maintains one record for each project to which an artifact category definition is constrained. The ArtCatName field of this table references the ArtCatName field of the ARTIFACT_CATEGORY table with a cascading delete constraint. The ProjName field of this table references the ProjName field of the PROJECT table with a cascading delete constraint.

Table 4.6: PROJ_SPEC_ART_CAT Table

Field Name	Data Type	Description
ArtCatName	varchar2(50)	Name of the Artifact Category
ProjName	varchar2(50)	Name of the project to which the artifact is restricted

Table 4.7 lists the data contained in the INC_SPEC_ART_CAT table. It maintains one record for each increment of a project to which an artifact category definition is con-

strained. The ArtCatName field of this table references the ArtCatName field of the PROJ_SPEC_ART_CAT table with a cascading delete constraint. The ProjName and IncNum fields of this table reference the ProjName and IncNum fields of the INCREMENTS table with a cascading delete constraint.

Table 4.7: INC_SPEC_ART_CAT Table

Field Name	Data Type	Description
ArtCatName	varchar2(50)	Name of the Artifact Category
ProjName	varchar2(50)	Name of the project to which the artifact is restricted
IncNum	number(3)	Name of the increment of the project to which the artifact is restricted

Table 4.8 lists the data contained in the ARTIFACT_CLASS table. It maintains one record for each member of a particular class of artifacts. This corresponds to one record for each record in the ARTIFACT_CATEGORY table which has an ArtType of "Class". The ArtCatName field of this table references the ArtCatName field of the ARTIFACT_CATEGORY table with a cascading delete constraint. When inserted into a form, the MemberArtName is validated against entries of the ArtCatName field of the ARTIFACT_CATEGORY table.

Table 4.8: ARTIFACT_CLASS Table

Field Name	Data Type	Description
ArtClassName	varchar2(50)	Name of the Artifact Class
MemberArtName	varchar2(50)	Name of artifact that is a member of the class

Table 4.9 lists the data contained in the ARTIFACT_FILE_INSTANCE table. It is responsible for storage of instances of all artifacts of file type. The ArtCatName field of this table references the ArtCatName field of the ARTIFACT_CATEGORY table. The Type field of this table references the Type field of the FILE_TYPE table. A unique con-

straint is placed on the Scope, Type, FileName, and RcsNum fields of this table. The ARTIFACT_FILE_INSTANCE_Scope and ARTIFACT_FILE_INSTANCE_Type indexes exist on the Scope and Type fields since these fields are present in the "where" clause of queries made from the *File Summary Window.*

Table 4.9: ARTIFACT_FILE_INSTANCE Table

Field Name	Data Type	Description
ArtCatName	varchar2(50)	Name of the artifact
ArtFileId	number(3)	Artifact number
Scope	varchar2(50)	Scope of the artifact(Top-Level...)
Type	varchar2(5)	Artifact File Type (e.g. FMN, WPF, MIF, C, CDL)
FileName	varchar2(50)	Filename associated with the artifact
RcsNum	number(5)	RCS number of the currently stored document
RcsDate	date	Date submitted to RCS
Document	long raw	Actual document stored in database
Description	varchar2(1000)	Brief description of the document

Table 4.10 lists the data contained in the PROJ_ART_FILE_INST table. It maintains one record for each project to which an artifact instance is constrained. The ArtCatName and ArtFileId fields of this table reference the ArtCatName and ArtFileId fields of the ARTIFACT_FILE_INSTANCE table with a cascading delete constraint. The ProjName field of this table references the ProjName field of the PROJECT table with a cascading delete constraint.

Table 4.10: PROJ_ART_FILE_INST Table

Field Name	Data Type	Description
ArtCatName	varchar2(50)	Name of the Artifact Category
ArtFileId	number(6)	Artifact File Id Number
ProjName	varchar2(50)	Name of the project to which the artifact is restricted

Table 4.11 lists the data contained in the INC_ART_FILE_INST table. It maintains one record for each increment of a project to which an artifact category definition is constrained. The ArtCatName, ArtFileId and ProjName fields reference the ArtCatName, ArtFileId, and ProjName fields of the PROJ_ART_FILE_INST table with a cascading delete constraint. The IncNum field references the IncNum field of the INCREMENT table with a cascading delete constraint.

Table 4.11: INC_ART_FILE_INST Table

Field Name	Data Type	Description
ArtCatName	varchar2(50)	Name of the Artifact Category
ArtFileId	number(6)	Artifact File Id Number
ProjName	varchar2(50)	Name of the project to which the artifact is restricted
IncNum	number(3)	Name of the increment of the project to which the artifact is restricted

Table 4.12 lists the data contained in the FILE_TYPE table. It maintains one entry for each type of artifact that is stored in the database.

Table 4.12: FILE_TYPE Table Data

Field Name	Data Type	Description
Type	varchar2(5)	Artifact Type e.g. FMN, WPF, MIF, C, CDL)
Description	varchar2(200)	Description of the type of document

Table 4.13 lists the data contained in the RCS_SUMMARY table. It maintains a unique record for each file in a project that is maintained by RCS. A unique constraint is placed on the ProjName and FileName fields. A foreign key constraint with the cascading delete option is placed on the ProjName field referencing the ProjName field of the

PROJECT table. The *RcsInfo_rcs_sum_insert* trigger fires when an entry is inserted into this table to ensure a unique FileId field.

Table 4.13: RCS_SUMMARY Table

Field Name	Data Type	Description
FileId	number(6)	Unique File identification number
ProjName	varchar2(50)	Project Name
FileName	varchar2(30)	Name of the file
Description	varchar2(2000)	Description of the file

Table 4.14 lists the data contained in the RCS_VER_INFO table. This information is extracted from the RCS log. The FileId field references the RCS_SUMMARY table with a cascading delete constraint. The *RcsInfo_rcs_ver_insert* trigger fires when an entry is inserted into this table to ensure that the FileId in this table corresponds to the FileId most recently inserted into the RCS_SUMMARY table. The FileId field references the FileId field in the RCS_SUMMARY table. The FILESEQ sequence assigns unique identification numbers to the files submitted into the RCS_SUMMARY table.

Table 4.14: RCS_VER_INFO Table

Field Name	Data Type	Description
FileId	number(6)	Unique File identification number
RcsNum	varchar2(10)	Revision number of the file
RcsDate	date	Date this revision checked-in to RCS
Author	varchar2(20)	Author checking-in this revision
Description	varchar2(2000)	Description of changes since last revision

Table 4.15 lists the data contained in the TASKING_SQRL table. It contains information on the tasks assigned to each SQRL member. A unique identification number is assigned to each record. A "unique" constraint is placed on the SqrlLast and SqrlFirst

fields so that a unique combination of a given name and surname exists for each identification number.

Table 4.15: TASKING_SQRL Table

Field Name	Data Type	Description
SqrlId	number(3)	Sqrl member ID number
SqrlLast	varchar2(20)	Last name of team member
SqrlFirst	varchar2(20)	First name of team member

Table 4.16 lists the data contained in the TASKING_PROJ table. It contains one entry for each project in which a SQRL member with the associated identification number participated. The SqrlId field references the TASKING_SQRL table with a cascading delete constraint. The ProjName field references the ProjName field of the PROJECTS table with a cascading delete constraint.

Table 4.16: TASKING_PROJ Table

Field Name	Data Type	Description
SqrlId	number(3)	Sqrl member ID number
ProjName	varchar2(50)	Project Name

Table 4.17 lists the data contained in the TASKING_INC table. It contains one record for each increment of a project in which a SQRL member with the associated identification number participates. The SqrlId and ProjName fields reference the TASKING_PROJ table with a cascading delete constraint. The ProjName and IncNum fields reference the ProjName and IncNum fields of the INCREMENT table with a cascading delete constraint.

Table 4.17: TASKING_INC Table

Field Name	Data Type	Description
SqrlId	number(3)	Sqrl member ID number

Table 4.17: TASKING_INC Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
Comments	varchar2(1000)	Responsibility description / comments

Table 4.18 lists the data contained in the TASKING table. It contains one entry for each task assigned to the SQRL member with the associated identification number. The SqrlId, ProjName and IncNum fields references the TASKING_INC table with a cascading delete constraint. The TASKING_PosnAssign index exists on the PosnAssign field since this field is used in the *where* clause of queries in many forms to generate lists of values to validate the inserted data.

Table 4.18: TASKING Table

Field Name	Data Type	Description
SqrlId	number(3)	Sqrl member ID number
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
PosnAssign	varchar2(20)	Position assigned to member

Table 4.19 lists the information contained in the INC_PLAN table. It maintains information for each version of the increment plan. The ProjName and IncNum fields reference the INCREMENT table with the cascading delete constraint.

Table 4.19: INC_PLAN Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(5)	Version number of the corresponding increment plan artifact
EstStDate	date	Estimated start date of work on increment

Table 4.19: INC_PLAN Table

Field Name	Data Type	Description
EstFnDate	date	Estimated completion date on increment
Description	varchar2(1000)	Description of goals for this increment

Table 4.20 lists the information contained in the INC_PLAN_FUNC table. It maintains one entry for each function to be implemented for a given version of the increment plan for a specific project. The ProjName, IncNum, and Version fields reference the INC_PLAN table with the cascading delete constraint. The INC_PLAN_FUNC_FuncNum index exists on the FuncNum field of this table since this field is a foreign key value.

Table 4.20: INC_PLAN_FUNC Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(5)	Version number of the corresponding increment plan artifact
FuncNum	number(5)	Function ID Number
FuncName	varchar2(25)	Function to be developed at this increment
Description	varchar2(1000)	Description of the function

Table 4.21 lists the data contained in the COMPILATION table. It contains one entry for each compilation fault found in a given version of code from a given increment within a given project. The ProjName, IncNum, VerNum, and CompNum fields reference the COMP_MEET table with the cascading delete constraint.

Table 4.21: COMPILATION Table

Field Name	Data Type	Description
ProjName	varchar2(25)	Project Name
IncNum	number(3)	Increment Number

Table 4.21: COMPILATION Table

Field Name	Data Type	Description
VerNum	number(3)	Code version number being verified
CompNum	number(3)	Compilation Session Number
CompErrNum	number(3)	Compilation Error Number
EngChgNum	number(3)	Engineering Change Number
Classification	varchar2(50)	Error Classification
VerfResult	varchar2(9)	Result of verification of change(NYV/correct/incorrect)

Table 4.22 lists the data contained in the COMP_MEET table. It contains one record for each compilation of a particular version of the code in an increment of a given project. The ProjName and IncNum fields reference the INCREMENT table with a cascading delete constraint. The SqrlId field references the TASKING_SQRL table with a cascading delete constraint.

Table 4.22: COMP_MEET Table

Field Name	Data Type	Description
ProjName	varchar2(25)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Document version number being compiled
CompNum	number	Compilation Number
CompDate	date	Date of compilation
SqrlId	number(3)	Sqrl member ID number
SqrlLast	varchar2(20)	Last name of team member compiling this version
SqrlFirst	varchar2(20)	First name of team member compiling this version
CompResult	varchar2(9)	Result of compilation of code (one of correct/incorrect)

Table 4.23 lists the information contained in the TESTING table. It contains one entry for each testing failure found in a given version of code from a given increment within a given project. The ProjName and IncNum fields reference the INCREMENT table with a cascading delete constraint. The TESTING_VerfResult index exists on the

VerfResult field since this field is often present in the *where* clause of a query in the *Testing Failure Form.*

Table 4.23: TESTING Table

Field Name	Data Type	Description
ProjName	varchar2(25)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Document version number being tested
Classification	varchar2(50)	Error Classification
SqrlId	number(3)	Sqrl member ID number
SqrlLast	varchar2(20)	Last name of team member testing this version
SqrlFirst	varchar2(20)	First name of team member testing this version
TestCaseNum	number	Test Case when failure discovered
TestFailNum	number	Testing failure number
FailType	varchar2(8)	Type of failure(stop/continue)
TransitionNum	number(3)	Transition Number
EngChgStatus	verchar2(12)	Engineering Change Status (designed/not designed)
EngChgNum	number(3)	Engineering Change Number
VerfResult	varchar2(9)	Result of verification of change (NYV/ correct/incorrect)

Table 4.24 lists the information contained in the ENGINEERING_CHANGES table. It contains one entry for each engineering change made to a given version of code from a given increment within a given project. The ProjName and IncNum fields reference the INCREMENT table with a cascading delete constraint.

Table 4.24: ENGINEERING_CHANGES Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Code version number being verified
EngChgNum	number(3)	Engineering Change Number

Table 4.24: ENGINEERING_CHANGES Table

Field Name	Data Type	Description
IncOrig	number(3)	Increment Number of Origin of the Problem
Comments	varchar2(1000)	Description of the error and the fix

Table 4.25 lists the information contained in the ENG_CH_FILENAMES table. It contains one entry for each RCS version of a file to which an engineering change has been made. The ProjName, IncNum, VerNum, and EngChgNum fields reference the ENGINEERING_CHANGES table with a cascading delete constraint.

Table 4.25: ENG_CH_FILENAMES Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Code version number being verified
EngChgNum	number(3)	Engineering Change Number
FileNum	number(3)	File ID number
FileName	varchar2(20)	Name of File Changed
RcsNum	number(3)	RCS version number of file

Table 4.26 lists the information contained in the ENG_CHG_FUNCTIONS table. It contains one entry for each function of each RCS version of a file to which an engineering change has been made. The ProjName, IncNum, VerNum, and EngChgNum fields reference the ENGINEERING_CHANGES table with a cascading delete constraint.

Table 4.26: ENG_CHG_FUNCTIONS Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Code version number being verified
EngChgNum	number(3)	Engineering Change Number

Table 4.26: ENG_CHG_FUNCTIONS Table

Field Name	Data Type	Description
FileNum	number(3)	File ID number
FuncNum	number(3)	Function ID number
FuncName	varchar2(20)	Name of Function Changed

Table 4.27 lists the information contained in the ENG_CH_VERIFIERS table. It contains one record for each designer responsible for an engineering change to a partcular version of the code in an increment of a given project. The ProjName, IncNum, VerNum, and EngChgNum fields reference the ENGINEERING_CHANGES table with a cascading delete constraint.

Table 4.27: ENG_CH_VERIFIERS Table

Field Name	Data Type	Description
ProjName	varchar2(25)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Code version number being verified
EngChgNum	number(3)	Engineering Change Number
SqrlId	number(3)	Sqrl ID number
SqrlLast	varchar2(20)	Last name of team member verifying this version
SqrlFirst	varchar2(20)	First name of team member verifying this version

Table 4.28 lists the information contained in the VERIFICATION table. It contains one entry for each verification fault that was found in a given version of code from a given increment within a given project. The ProjName and IncNum fields reference the INCREMENT table with a cascading delete constraint. The VERIFICATION_Result index exists

on the Result field since this field is often present in the *where* clause of a query in the *Verification Fault Form.*

Table 4.28: VERIFICATION Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Code version number being verified
VerfErrNum	number(3)	Verification Error Number
VerfMeetNum	number(3)	Verification Meeting Number
VerfTime	date	Time of Review
EngChgNum	number(3)	Engineering Change Number
Classification	varchar2(50)	Error Classification
Result	varchar2(9)	Result of verification of engineering change (one of NYV/correct/incorrect)
Stage	varchar2(45)	Stage Error Found (One of Original Design, Design Change, Compilation Error, Testing Failure)
Comments	varchar2(1000)	Description of the error and the fix

Table 4.29 lists the information contained in the VERF_VERIFIERS table. It contains one record for each SQRL member responsible for verifying a partcular version of the code in an increment of a given project. The ProjName, IncNum, VerNum, and VerfErrNum fields reference the VERIFIERS table with a cascading delete constraint.

Table 4.29: VERF_VERIFIERS Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Code version number being verified
VerfErrNum	number(3)	Verification Error Number
VerifierNum	number(3)	Verifier ID number
SqrlLast	varchar2(20)	Last name of team member verifying this version

Table 4.29: VERF_VERIFIERS Table

Field Name	Data Type	Description
SqrlFirst	varchar2(20)	First name of team member verifying this version

Table 4.30 lists the information contained in the VERF_FILENAMES table. It contains one entry for each RCS version of a file to which a verification fault has been recorded. The ProjName, IncNum, VerNum, and VerfErrNum fields reference the VERIFICATION table with a cascading delete constraint.

Table 4.30: VERF_FILENAMES Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Code version number being verified
VerfErrNum	number(3)	Verification Error Number
FileNum	number(3)	File ID number
FileName	varchar2(20)	Name of File Changed
RcsNum	number(3)	RCS version number of file

Table 4.31 lists the information contained in the VERF_FUNCTIONS table. It contains one entry for each function of each RCS version of a file to which a verification fault was found. The ProjName, IncNum, VerNum, VerfErrNum, and FileNum fields reference the VERF_FILENAMES table with a cascading delete constraint.

Table 4.31: VERF_FUNCTIONS Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Code version number being verified
VerfErrNum	number(3)	Verification Error Number

Table 4.31: VERF_FUNCTIONS Table

Field Name	Data Type	Description
FileNum	number(3)	File ID number
FuncNum	number(3)	Function ID number
FuncName	varchar2(20)	Name of Function Changed

Table 4.32 lists the information contained in the VER_MEET table. It contains one record for each verification meeting of a particular version of the code in an increment of a given project. The ProjName and IncNum fields reference the INCREMENT table with a cascading delete constraint.

Table 4.32: VER_MEET Table

Field Name	Data Type	Description
ProjName	varchar2(25)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Document version number being compiled
VerMeetNum	number	Verification Session Number
VerfDate	date	Date of Verification Session
VerfStTime	date	Start Time of Verification Session
VerfFnTime	date	Finish Time of Verification Session
Comments	varchar2(1000)	Description of the error and the fix

Table 4.33 lists the information contained in the VER_MEET_VERIFIERS table. It contains one record for each SQRL member responsible for verifying a particular version of the code in an increment of a given project. The ProjName, IncNum, VerNum, and VerfErrNum fields reference the VER_MEET table with a cascading delete constraint.

Table 4.33: VER_MEET_VERIFIERS Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number

Table 4.33: VER_MEET_VERIFIERS Table

Field Name	Data Type	Description
VerNum	number(3)	Code version number being verified
VerMeetNum	number(3)	Verification Error Number
SqrlId	number(3)	Sqrl ID number
SqrlLast	varchar2(20)	Last name of team member verifying this version
SqrlFirst	varchar2(20)	First name of team member verifying this version

Table 4.34 lists the information contained in the VER_MEET_FILENAMES table. It contains one entry for each RCS version of a file which has been verified, whether or not a verification fault was recorded. The ProjName, IncNum, VerNum, VerfErrNum, and FileNum fields reference the VER_MEET table with a cascading delete constraint. The SqrlId field references the TASKING_SQRL table with a cascading delete constraint.

Table 4.34: VER_MEET_FILENAMES Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Code version number being verified
VerMeetNum	number(3)	Verification Meeting Number
FileNum	number(3)	File ID number
FileName	varchar2(20)	Name of File Changed
RcsNum	number(3)	RCS version number of file

Table 4.35 lists the information contained in the PROJECT_TOTAL table. It contains one record for each project in the database. It contains metrics totals for the specified project. A trigger exists such that upon insertion of a new project into the PROJECT table, the PROJECT_TOTAL table is also be updated with a corresponding entry. The ProjName

references the ProjName field of the PROJECT_TOTAL table with a cascading delete constraint.

Table 4.35: PROJECT_TOTAL Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
NumFilesC	number(3)	Number of C Files in this project
NumFilesHdr	number(3)	Number of header files in this project
NumNewFilesC	number(3)	Number of new C files in this project
NumNew-FilesHdr	number(3)	Number of new header files in this project
LOCode	number(6)	Number of total lines of code
LOC	number(6)	Number of total lines of C code
LOSemi	number(6)	Number of semicolons
LOComments	number(6)	Number of lines of comments
NumBlanks	number(6)	Number of blank lines
NumDefines	number(6)	Number of *#define* statements
NumIncludes	number(6)	Number of *#include* statements
NumFors	number(6)	Number of *for* statements
NumWhiles	number(6)	Number of *while* statements
NumSwitch	number(6)	Number of *switch* statements
NumIf	number(6)	Number of *if* statements
VerErrors	number(6)	Number of verification errors
CompErrors	number(6)	Number of compilation errors
TestFailures	number(6)	Number of testing failures

Table 4.36 lists the information contained in the INCREMENT_TOTAL table which contains one record for each increment of each project in the database. It contains

metrics totals for the given increment. The ProjName and Incnum fields reference the INCREMENT table with a cascading delete constraint.

Table 4.36: INCREMENT_TOTAL Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
NumFilesC	number(3)	Number of C Files in this increment
NumFilesHdr	number(3)	Number of header files in this increment
NumNewFilesC	number(3)	Number of new C files in this increment
NumNew-FilesHdr	number(3)	Number of new header files in this increment
LOCode	number(6)	Number of total lines of code
LOC	number(6)	Number of total lines of C code
LOSemi	number(6)	Number of semicolons
LOComments	number(6)	Number of lines of comments
NumBlanks	number(6)	Number of blank lines
NumDefines	number(6)	Number of *#define* statements
NumIncludes	number(6)	Number of *#include* statements
NumFors	number(6)	Number of *for* statements
NumWhiles	number(6)	Number of *while* statements
NumSwitch	number(6)	Number of *switch* statements
NumIf	number(6)	Number of *if* statements
VerErrors	number(6)	Number of verification errors
CompErrors	number(6)	Number of compilation errors
TestFailures	number(6)	Number of testing failures

Table 4.37 lists the information contained in the CUST_INT table. It maintains a collection of information on customer interaction meetings for a particular project. The

ProjName and IncNum fields reference the INCREMENT table with a cascading delete constraint.

Table 4.37: CUST_INT Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
MeetNum	number(5)	Meeting Number of this type for the given project
StartTime	date	Start date of increment
FinishTime	date	Duration of Meeting (hrs)
Comments	varchar2(1000)	Comments about the meeting contents

Table 4.38 lists the information contained in the CUST_SQRL_ATTEND table. It contains one entry for each SQRL member present at a Customer Interaction meeting. The ProjName, IncNum, and MeetNum fields reference the CUST_INT table with a cascading delete constraint. The SqrlId field references the TASKING_SQRL table with a cascading delete constraint.

Table 4.38: CUST_SQRL_ATTEND Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
MeetNum	number(5)	Meeting Number of this type for the given project
SqrlNum	number(3)	Sqrl ID number
SqrlLast	varchar2(20)	Last name of team member in attendance
SqrlFirst	varchar2(20)	First name of team member in attendance
Comments	varchar2(200)	Any comments about the member's attendance

Table 4.39 lists the information contained in the CUSTOMER_ATTEND table. It contains one entry for each Customer member present at a specified customer interaction

meeting. The ProjName, IncNum, and MeetNum fields reference the CUST_INT table with a cascading delete constraint.

Table 4.39: CUSTOMER_ATTEND Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
MeetNum	number(5)	Meeting Number of this type for the given project
CustNum	number(3)	Customer ID number
CustLast	varchar2(20)	Last name of customer member in attendance
CustFirst	varchar2(20)	First name of customer member in attendance
Organization	varchar2(25)	Company/Organization represented by the customer
Comments	varchar2(200)	Any comments about the member's attendance

Table 4.40 lists the information contained in the CONTROL_CHECK table. It contains one entry for each process control meeting that was held for a given project. The ProjName and IncNum fields reference the INCREMENT table with a cascading delete constraint.

Table 4.40: CONTROL_CHECK Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
VerNum	number(3)	Compiled Version
MeetNum	number	Process control meeting number for this project
StartTime	date	Meeting date/start time of check on process control
FinishTime	date	Meeting finish time
ProcName	varchar2(50)	Name of process being monitored(one of verification, compilation, testing)
Result	varchar2(10)	Result of check(under control / out of control)
Continue	varchar2(1)	Do we want to continue?(Y/N)
RevDate	number(3)	Date of next Process Control review

Table 4.40: CONTROL_CHECK Table

Field Name	Data Type	Description
Comments	varchar2(1000)	Comments on why process was out of control

Table 4.38 lists the information contained in the CONTROL_SQRL_ATTEND table. It contains one entry for each SQRL member present at a Process Control meeting. The ProjName, IncNum, and MeetNum fields reference the CONTROL_CHECK table with a cascading delete constraint. The SqrlId field references the TASKING_SQRL table with a cascading delete constraint.

Table 4.41: CONTROL_SQRL_ATTEND Table

Field Name	Data Type	Description
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
MeetNum	number(5)	Meeting Number of this type for the given project
SqrlNum	number(3)	Sqrl ID number
SqrlLast	varchar2(20)	Last name of team member in attendance
SqrlFirst	varchar2(20)	First name of team member in attendance
Comments	varchar2(200)	Any comments about the member's attendance

Table 4.42 lists the information contained in the CONTROL_STDS table. It contains the currently agreed upon process control measures.

Table 4.42: CONTROL_STDS Table

Field Name	Data Type	Description
VerNum	number(5)	Version number of control standards
VerDate	date	Date this version of control standards submitted
VerRate	number(6,2)	Verification errors per KLOCDL
CompRate	number(6,2)	Compilation errors per KLOC
ExeRate	number(6,2)	Execution errors per KLOC

4.5 Database Table Views

This section lists the database views which are used in the CPD. These views are queried by forms in the interface and are not used for insertion of new information or for updating existing information.

Table 4.43 lists the data contained in the PROC_ART_TYPE View. This view is used in updating the artifact lists of the INPUT, INOUT, and OUTPUT blocks of the *Browse Processes* window.

Table 4.43: PROC_ART_TYPE View

Field Name	Data Type	Description
ArtName	varchar2(50)	Name of artifact required by the process step
Scope	varchar2(18)	Scope of the artifact category
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
ProcName	varchar2(50)	Name of the Cleanroom process step
UseType	varchar2(6)	The type of use of this artifact at this step (One of input, inout, output)
ArtType	varchar2(6)	Type of the artifact
ExecName	varchar2(30)	Executable for the artifact (if a form or report type)

The definition of the view is given below:

```
CREATE VIEW PROC_ART_TYPE as(
SELECT DISTINCT PROCESS_ARTIFACT.ArtName, Scope,
      PROJ_SPEC_ART_CAT.ProjName, incnum, ProcName, UseType,
      ArtType, ExecName
FROM PROCESS_ARTIFACT, PROJ_SPEC_ART_CAT, INC_SPEC_ART_CAT,
      ARTIFACT_CATEGORY
WHERE PROCESS_ARTIFACT.ArtName = PROJ_SPEC_ART_CAT.ArtCat-
      Name(+)
AND PROJ_SPEC_ART_CAT.projname = INC_SPEC_ART_CAT.pro-
      jname(+)
AND PROCESS_ARTIFACT.ArtName = ARTIFACT_CATEGORY.ArtCatName
 );
```

Table 4.44 lists the data contained in the BROWSE_ART View. This view is used in the *Browse Artifacts* window to query over artifact attributes from multiple tables.

Table 4.44: BROWSE_ART View

Field Name	Data Type	Description
ArtCatName	varchar2(50)	Name of the Artifact Category
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
Scope	varchar2(18)	Scope of the artifact category
ArtType	varchar2(6)	Type of the artifact
ExecName	varchar2(30)	Executable for the artifact (if a form or report type)
Description	varchar2(1000)	Description of the artifact category

The definition of the view is given below:

```
CREATE VIEW BROWSE_ART as(
SELECT DISTINCT ARTIFACT_CATEGORY.ArtCatName,
      PROJ_SPEC_ART_CAT.ProjName,incnum, scope, ArtType,
      ARTIFACT_CATEGORY.Description, EXECNAME
FROM PROJ_SPEC_ART_CAT, INC_SPEC_ART_CAT, ARTIFACT_CATEGORY
WHERE ARTIFACT_CATEGORY.ArtCatName = PROJ_SPEC_ART_CAT.Art-
      CatName(+)
AND PROJ_SPEC_ART_CAT.projname = INC_SPEC_ART_CAT.pro-
      jname(+)
 );
```

Table 4.45 lists the data contained in the BROWSE_ART_INST View. This view is used for querying data on artifacts of file type from the *File Summary* Window.

Table 4.45: BROWSE_ART_INST View

Field Name	Data Type	Description
ArtCatName	varchar2(50)	Name of the Artifact Category
ArtFileId	number(3)	Artifact number
FileName	varchar2(50)	Filename associated with the artifact
RcsNum	number(5)	RCS number of the currently stored document
FileType	varchar2(5)	Artifact File Type (e.g. FMN, WPF, MIF, C, CDL)
RcsDate	date	Date submitted to RCS

Table 4.45: BROWSE_ART_INST View

Field Name	Data Type	Description
Description	varchar2(1000)	Brief description of the document

The definition of the view is given below:

```
CREATE VIEW BROWSE_ART_INST as(
SELECT DISTINCT ARTIFACT_FILE_INSTANCE.ArtCatName,
      ARTIFACT_FILE_INSTANCE.ArtFileId, FileName, RcsNum,
      FileType, EcsDate, ARTIFACT_FILE_INSTANCE.Description,
      PROJ_ART_FILE_INST.ProjName,incnum
FROM PROJ_ART_FILE_INST, INC_ART_FILE_INST,
      ARTIFACT_CATEGORY, ARTIFACT_FILE_INSTANCE
WHERE ARTIFACT_FILE_INSTANCE.ArtCatName =
      PROJ_ART_FILE_INST.ArtCatName(+)
AND PROJ_ART_FILE_INST.projname = INC_ART_FILE_INST.pro-
      jname(+)
 );
```

Table 4.46 lists the data contained in the SQRL_TASKS View. This view is queried in multiple windows in order to obtain a list of SQRL members registered with the system associated with a desired tasking category.

Table 4.46: SQRL_TASKS View

Field Name	Data Type	Description
SqrlId	number(3)	Sqrl member ID number
SqrlLast	varchar2(20)	Last name of team member
SqrlFirst	varchar2(20)	First name of team member
ProjName	varchar2(50)	Project Name
IncNum	number(3)	Increment Number
PosnAssign	varchar2(20)	Position assigned to member

The definition of the view is given below:

```
CREATE VIEW SQRL_TASKS as
SELECT TASKING.SqrlId, SqrlLast, SqrlFirst, ProjName,
      IncNum, PosnAssign
FROM TASKING, TASKING_SQRL
WHERE TASKING.SqrlId = TASKING_SQRL.SqrlId;
```

Chapter 5

User Stimuli

This chapter presents a list of all possible stimuli which could be generated during interaction with each component of the CPD interface.

5.1 Stimuli Structure

In the following sections, a list is compiled of all possible stimuli for each interface component of the windows comprising the core functionality of the CPD. Stimuli are not provided for the individual forms artifacts or reports artifacts. Rather, generic stimuli are provided to represent the generation of a report or execution of a form. All stimuli are not provided for default Oracle features such as the default Oracle menus of certain forms and reports. Only stimuli for default functionality which are responsible for state transition are included in this stimulus listing. The user is referred to the Oracle Reference Manual Set for information on the functionality of these features. Stimuli are presented as ordered 6-tuples of the form:

$$S = C \times Val\ Env \times Inv\ Env \times Inv\ Env(3) \times CallWin \times ArtType \times Update\ File.$$

C is the set of stimulus classes which are composed *action*: *interface element*. *Action* may take on the values of *select*, *signal*, or *open*. *Select* refers to any interaction with the system that is equivalent to "single-clicking" on the interface element. *Signal* refers to any stimulus from the operating system. *Open* refers to any interaction with the system that is equivalent to "double-clicking" on the interface element. *Interface element* is represented as a hierarchy from the window name down to the specific screen element. Window names appear in the class list in abbreviated form.

Val Env is the set containing valid environment variables and valid logon parameters. *Inv Env* is the set containing invalid environment variables and invalid logon parameters. *Inv Env(3)* is the set containing the specific invalid environment variables and invalid logon parameters which have been entered into the logon screen three times running. *CallWin* is a one member set which contains the name of the previous window. *ArtType* is a set containing all properties of a selected artifact. *Update File* corresponds to the set containing the current value of the *Update File* Radio Box on the *Update File Form*.

Only the significant 6-tuple elements are included in the stimulus. For example, <Select:CPD.LA.OK> is the stimulus produced upon selecting the OK button on the *Logon Alert*, and no extra information is required.

5.2 List of Stimuli

5.2.1 Operating System/Application Interface

- <Signal:CPD> is produced by executing the command *cpd* from the prompt.

5.2.2 Logon Screen

- <Select:CPD.LS.Connect, *Val Env*> is produced upon selecting the *Connect* button on the *Logon Screen* with valid parameters in the logon fields and valid environment variables.
- <Select:CPD.LS.Connect, *Inv Env*> is produced upon selecting the *Connect* button on the *Logon Screen* with invalid parameters in the logon fields and/or invalid environment variables.
- <Select:CPD.LS.Connect, *Inv Env(3)*> is produced upon selecting the *Connect* button on the *Logon Screen* with invalid parameters in the logon fields and/or invalid environment variables three time running.
- <Select:CPD.LS.Cancel, *CallWin*> is produced upon selecting the *Cancel* button on the *Logon Screen.*

5.2.3 Logon Alert

- <Select:CPD.LA.OK> is produced upon selecting the OK button on the *Logon Alert.*

5.2.4 Main Window

- <Select:CPD.MW.Action.Connect> is produced upon selecting the item *Connect* from the Action Menu.
- <Select:CPD.MW.Action.Projects...> is produced upon selecting the item *Projects...* from the *Action Menu.*
- <Select:CPD.MW.Action.Increments...> is produced upon selecting the item *Increments...* from the *Action Menu.*
- <Select:CPD.MW.Action.Exit> is produced upon selecting the item *Exit* from the *Action Menu.*

- <Select:CPD.MW.Help.About> is produced upon selecting the item *About* from the *Action Menu.*
- <Select:CPD.MW.Artifacts> is produced upon selecting the *Artifacts* image.
- <Select:CPD.MW.Processes> is produced upon selecting the item *Processes* image.

5.2.5 Browse Processes Window

- <Select:CPD.BP.Action.Connect> is produced upon selecting the item *Connect* from the *Action Menu.*
- <Select:CPD.BP.Action.Projects...> is produced upon selecting the item *Projects...* from the *Action Menu.*
- <Select:CPD.BP.Action.Increments...> is produced upon selecting the item *Increments...* from the *Action Menu.*
- <Select:CPD.BP.Action.Process Definitions> is produced upon selecting the item *Process Definitions* from the *Action Menu.*
- <Select:CPD.BP.Action.Artifact Categories> is produced upon selecting the item *Artifact Categories* from the *Action Menu.*
- <Select:CPD.BP.Action.File Types> is produced upon selecting the item File Types from the *Action Menu.*
- <Select:CPD.BP.Action.Exit> is produced upon selecting the item *Exit* from the *Action Menu.*
- <Select:CPD.BP.Help.About> is produced upon selecting the item *About* from the *Action Menu.*
- <Select:CPD.BP.Show Structure> is produced upon selecting the *Show Structure* button.
- <Select:CPD.BP.Select, *Artifact Type*> is produced upon selecting the *Select* button.
- <Select:CPD.BP.Cancel> is produced upon selecting the *Cancel* button.

- <Select:CPD.BP.ProcList> is produced by selecting an item from the *Process List.*
- <Select:CPD.BP.ArtList> is produced by selecting an item from either of the *Input, Inout, or Output Artifact Lists.*
- <Select:CPD.BP.ArrowRt> is produced by selecting the *Right Arrow* icon.
- <Select:CPD.BP.ArrowLt> is produced by selecting the *Left Arrow* icon.

5.2.6 Show Schema Window

- <Select:CPD.SSCH.Hide Schema> is produced upon selecting the *Hide Schema* button.

5.2.7 Browse Artifacts Window

- <Select:CPD.BA.Action.Connect> is produced upon selecting the item *Connect* from the *Action Menu.*
- <Select:CPD.BA.Action.Projects...> is produced upon selecting the item *Projects...* from the *Action Menu.*
- <Select:CPD.BA.Action.Increments...> is produced upon selecting the item *Increments...* from the *Action Menu.*
- <Select:CPD.BA.Action.Artifact Categories> is produced upon selecting the item *Artifact Categories* from the *Action Menu.*
- <Select:CPD.BA.Action.File Types> is produced upon selecting the item File Types from the *Action Menu.*
- <Select:CPD.BA.Action.Exit> is produced upon selecting the item *Exit* from the *Action Menu.*
- <Select:CPD.BA.Help.About> is produced upon selecting the item *About* from the *Action Menu.*
- <Select:CPD.BA.Query> is produced upon selecting the *Query* button.

- <Select:CPD.BA.Select, *Artifact Type*> is produced upon selecting the *Select* button.
- <Select:CPD.BA.Cancel> is produced upon selecting the *Cancel* button.
- <Select:CPD.BA.Show Structure> is produced upon selecting the *Show Structure* button.
- <Select:CPD.BA.Scope> is produced upon selecting an item from the *Scope* radio group.
- <Select:CPD.BA.Type> is produced upon selecting an item from the *Type* radio group.
- <Select:CPD.BA.ArtList> is produced upon selecting an item from the *Artifact List*.

5.2.8 Projects Window

- <Select:CPD.PRJ.Select, *CallWin*> is produced upon selecting the *Select* button.
- <Select:CPD.PRJ.Modify> is produced upon selecting the *Modify* button.
- <Select:CPD.PRJ.Cancel, *CallWin*> is produced upon selecting the *Cancel* button.
- <Select:CPD.PRJ.New> is produced upon selecting the *New* button.
- <Select:CPD.PRJ.ProjectList> is produced upon selecting an item from the *Project List*.

5.2.9 Modify Project Window

- <Select:CPD.MODP.OK> is produced upon selecting the *OK* button.
- <Select:CPD.MODP.Cancel> is produced upon selecting the *Cancel* button.

5.2.10 New Project Window

- <Select:CPD.NEWP.OK> is produced upon selecting the *OK* button.

- <Select:CPD.NEWP.Cancel> is produced upon selecting the *Cancel* button.

5.2.11 Increments Window

- <Select:CPD.INC.Select, *CallWin*> is produced upon selecting the *Select* button.
- <Select:CPD.INC.Modify> is produced upon selecting the *Modify* button.
- <Select:CPD.INC.New> is produced upon selecting the *New* button.
- <Select:CPD.INC.Cancel, *CallWin*> is produced upon selecting the *Cancel* button.
- <Select:CPD.INC.IncList> is produced upon selecting an item from the *Increments List.*

5.2.12 Modify Increments Window

- <Select:CPD.MODI.OK> is produced upon selecting the *OK* button.
- <Select:CPD.MODI.Cancel> is produced upon selecting the *Cancel* button.

5.2.13 New Increments Window

- <Select:CPD.NEWI.OK> is produced upon selecting the *OK* button.
- <Select:CPD.NEWI.Cancel> is produced upon selecting the *Cancel* button.

5.2.14 About Window

- <Select:CPD.ABT, *CallWin*> is produced upon selecting the image displayed in the *About Window.*

5.2.15 Processes Window

- <Select:CPD.PROC.OK> is produced upon selecting the *OK* button.

- <Select:CPD.PROC.Cancel> is produced upon selecting the *Cancel* button.
- <Select:CPD.PROC.Process> is produced by selecting in the *Process* text box.
- <Select:CPD.PROC.ParProcess> is produced by selecting in the *Parent Process* text box.

5.2.16 Artifact Categories Window

- <Select:CPD.BA.OK, *CallWin*> is produced upon selecting the *OK* button.
- <Select:CPD.BA.Cancel, *CallWin*> is produced upon selecting the *Cancel* button.
- <Select:CPD.BA.Scope> is produced upon selecting an item from the *Scope* radio group.
- <Select:CPD.BA.Type> is produced upon selecting an item from the *Type* radio group.
- <Select:CPD.BA.ProjList> is produced upon selecting in the *Project List.*
- <Select:CPD.BA.IncList> is produced upon selecting in the *Increment List.*
- <Select:CPD.BA.ProcList> is produced upon selecting in the *Process List.*
- <Select:CPD.BA.ClassList> is produced upon selecting in the *Class List.*
- <Select:CPD.BA.InOutList> is produced upon selecting in the *In/Out List.*

5.2.17 Artifact File Types Window

- <Select:CPD.FT.OK, *CallWin*> is produced upon selecting the *OK* button.
- <Select:CPD.FT.Cancel, *CallWin*> is produced upon selecting the *Cancel* button.

5.2.18 File Summary Window

- <Select:CPD.FSUM.Cancel, *CallWin*> is produced upon selecting the item *Exit* from the *Action Menu.*
- <Select:CPD.FSUM.Delete> is produced upon selecting the *Delete* button.

- <Select:CPD.FSUM.Update> is produced upon selecting the *Update* button.
- <Select:CPD.FSUM.New> is produced upon selecting the *New* button.
- <Select:CPD.FSUM.Extract> is produced upon selecting the *Extract* button.
- <Select:CPD.FSUM.Query> is produced upon selecting the *Query* button.
- <Select:CPD.FSUM.FileList> is produced upon selecting an item from the *File List.*
- <Select:CPD.FSUM.ProjName> is produced upon selecting in the *Project Name* text field.
- <Select:CPD.FSUM.IncNum> is produced upon selecting in the *Increment Number* text field.
- <Select:CPD.FSUM.FileType> is produced upon selecting in the *File Type* text field.

5.2.19 Class List Window

- <Select:CPD.CLW.Find> is produced upon selecting the *Find* button.
- <Select:CPD.CLW.OK, *CallWin*> is produced upon selecting the *OK* button.
- <Select:CPD.CLW.Cancel, *CallWin*> is produced upon selecting the *Cancel* button.

5.2.20 Process List Window

- <Select:CPD.PROCLW.Find> is produced upon selecting the *Find* button.
- <Select:CPD.PROCLW.OK, *CallWin*> is produced upon selecting the *OK* button.
- <Select:CPD.PROCLW.Cancel, *CallWin*> is produced upon selecting the *Cancel* button.

5.2.21 Project List Window

- <Select:CPD.PROJLW.Find> is produced upon selecting the *Find* button.

- <Select:CPD.PROJLW.OK, *CallWin*> is produced upon selecting the *OK* button.
- <Select:CPD.PROJLW.Cancel, *CallWin*> is produced upon selecting the *Cancel* button.

5.2.22 In/Out List Window

- <Select:CPD.IOLW.Find> is produced upon selecting the *Find* button.
- <Select:CPD.IOLW.OK, *CallWin*> is produced upon selecting the *OK* button.
- <Select:CPD.IOLW.Cancel, *CallWin*> is produced upon selecting the *Cancel* button.

5.2.23 Increment List Window

- <Select:CPD.INCLW.Find> is produced upon selecting the *Find* button.
- <Select:CPD.INCLW.OK, *CallWin*> is produced upon selecting the *OK* button.
- <Select:CPD.INCLW.Cancel, *CallWin*> is produced upon selecting the *Cancel* button.

5.2.24 File Type List Window

- <Select:CPD.FTLW.Find> is produced upon selecting the *Find* button.
- <Select:CPD.FTLW.OK, *CallWin*> is produced upon selecting the *OK* button.
- <Select:CPD.FTLW.Cancel, *CallWin*> is produced upon selecting the *Cancel* button.

5.2.25 Run Report

- <Select:CPD.RPT.PARAM.File.Exit, *CallWin*> is produced upon selecting the item *Exit* from the *File Menu* on the *Report Parameter Form.*

- <Select:CPD.RPT.PARAM.File.Cancel, *CallWin*> is produced upon selecting the item *Cancel* from the *File Menu* on the *Report Parameter Form.*
- <Select:CPD.RPT.PRV.File.Exit, *CallWin*> is produced upon selecting the item *Exit* from the *File Menu* on the report previewer.
- <Select:CPD.PARAM.Run> is produced upon selecting the *Run Report* button on the *Report Parameter Form.*

5.2.26 Run Form

- <Select:CPD.FRM.Action.Exit, *CallWin*> is produced upon selecting the item *Exit* from the *Action Menu.*

5.2.27 New File Window

- <Select:CPD.NF.OK> is produced upon selecting the *OK* button.
- <Select:CPD.NF.Cancel> is produced upon selecting the *Cancel* button.
- <Select:CPD.NF.ProjName> is produced upon selecting in the *Project Name* text field.
- <Select:CPD.NF.IncNum> is produced upon selecting in the *Increment Number* text field.
- <Select:CPD.NF.FileType> is produced upon selecting in the *File Type* text field.

5.2.28 Update File Window

- <Select:CPD.UF.OK, *Update File*> is produced upon selecting the *OK* button.
- <Select:CPD.UF.Cancel> is produced upon selecting the *Cancel* button.
- <Select:CPD.UF.ProjName> is produced upon selecting in the *Project Name* text field.
- <Select:CPD.UF.IncNum> is produced upon selecting in the *Increment Number* text field.

- <Select:CPD.UF.FileType> is produced upon selecting in the *File Type* text field.
- <Select:CPD.UF.Yes> is produced upon selecting *Yes* from the *Replace Existing File* Radio Group.
- <Select:CPD.UF.No> is produced upon selecting *No* from the *Replace Existing File* Radio Group.

5.2.29 Extract File Browser

- <Select:CPD.ETL.OK> is produced upon selecting the *OK* button.
- <Select:CPD.ETL.Cancel> is produced upon selecting the *Cancel* button

5.2.30 Insert File Browser

- <Select:CPD.IFL.OK> is produced upon selecting the *OK* button.
- <Select:CPD.IFL.Cancel> is produced upon selecting the *Cancel* button

Chapter 6

Natural Language Specification

This chapter provides the natural language description of the CPD interface. Sections 6.1 describes the menu options on the windows of the CPD. Section 6.2 presents button options whose functionality is consistent over many windows of the interface. Sections 6.3.1 through 6.3.18 provide textual description of responses to the stimuli presented in Chapter 5. Screen dumps of windows, forms, and reports are provided for reference. Section 6.4 lists the forms artifacts which have been created. Section 6.5 lists the reports artifacts that have been created. Section 6.6 describes the utility program to strip information from the RCS log. Section 6.7 provides a usage model for the system in tabular form. Section 6.8 provides a mapping from the forms to Oracle tables of the CPD. Section 6.9 provides a mapping from the reports to the Oracle tables of the CPD.

6.1 Menus

This section presents the menus used in the CPD. The response resulting from the selection of each of the menu options is discussed.

6.1.1 Action Menu

The *Action Menu* (Figure 6.1) provides a maximum of seven options to the user: *Connect, Projects..., Increments..., Process Definitions, Artifact Categories, File Types* and *Exit.*

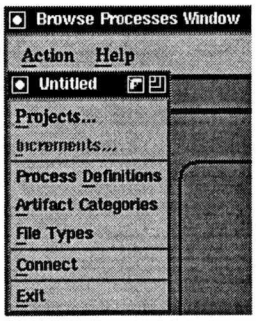

Figure 6.1: Action Menu

Connect Option. Selecting *Connect* from the *Action Menu* invokes the *Logon Screen* to allow a user to connect to the Oracle server as another username without having to exit and invoke the software a second time. This option is useful for performing restricted database administration functions. For example, a SQRL user might find it necessary to enter some data for a particular project that does not match an existing artifact category. A regular user would not have the privilege to create a new artifact category and could connect as a specified administrative user to complete the task.

Projects... Option. The *Projects...* selection invokes the *Projects Window.* From this window the user may select a project, add new projects, modify existing projects, or delete data associated with specific projects.

Increments... Option. The *Increments...* menu option is disabled until a project has been selected as the current project. Once enabled, the *Increments...* menu option invokes the *Increments Window.* This window provides the user with the ability to select an increment

of a project, to add data on new increments, modify data on existing increments, or delete data on existing increments.

Process Definitions Option. The *Process Definitions* menu option invokes the *Processes Window.* The *Processes Window* allows for the definition of new Cleanroom processes, the modification of existing Cleanroom process definitions, or the deletion of existing Cleanroom process definitions.

Artifact Categories Option. The *Artifact Categories* selection invokes the *Artifact Category Window.* This window allows options for modification of the definition of the Cleanroom artifact categories.

Exit Option. Selecting *Exit* from the *Action Menu* terminates the database session.

6.1.2 Help Menu

The *Help Menu* (Figure 6.2) provides the user with two help options: *About*, and *Help System.*

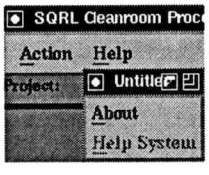

Figure 6.2: Help Menu

About Option. Selecting *About* from the menu displays the *About the CPD Window.*

Help System Option. The *Help System* option is disabled. In the future, selecting the *Help System* option will invoke online help for the CPD.

6.1.3 Oracle Default Menu

Some windows of the CPD display the default menu provided with the Oracle Forms Designer. Refer to the Oracle Forms Documentation Set for a complete description of the options and functionality of this menu.

6.2 Generic Buttons

The *OK* and *Cancel* buttons appear on many of the windows the CPD. Unless otherwise specified, the functionality of these buttons is as outlined below.

OK Button. Selecting the *OK* button returns the user to the calling window and any pending transactions are committed to the database.

Cancel Button. Selecting the *Cancel* button returns the user to the calling window and any pending transactions are rolled back.

6.3 Windows of the CPD Interface

This section describes the set of windows which compose the interface to the CPD. Individual artifact categories of form or report type are not considered in this section.

6.3.1 Invocation and Logon

The CPD is invoked by typing *cpd* on the command line. Upon invocation of the CPD, the user is confronted with the *Logon Screen* (Figure 6.1). The user may enter a username and the corresponding password with which to initiate an Oracle session. Assuming that the user has set their default database before initiating the session, the *Database* option may be left blank and a connection is made to the appropriate database. The user may specify the database with which to connect before beginning the session by

assigning the appropriate database name to the ORACLE_SID environment variable. Alternatively, the user can enter the database name into the *Database* field.

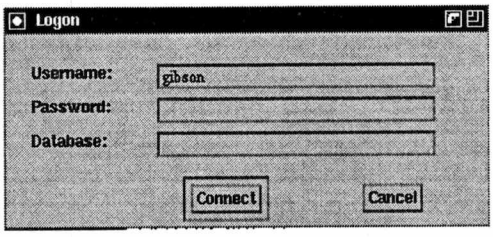

Figure 6.1: Logon Screen

Connect Button. Selecting the *Connect* button signals the system to attempt to initiate a session using the parameters currently in the *Logon Screen*. Upon successful logon, the *Main Window* is displayed.

Cancel Button. Upon three failed logon attempts or by selecting the *Cancel* button from the *Logon screen*, the application terminates. If a logon is attempted with either invalid parameters or invalid environment variables, an Oracle *Logon Alert* (Figure 6.2) is displayed.

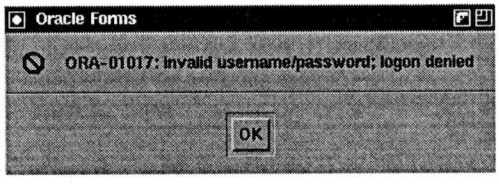

Figure 6.2: Logon Alert

OK Button. Selecting the *OK* button dismisses the *Logon Alert* and returns the user to the *Logon Screen*.

6.3.2 Main Window

The *Main Window* (Figure 6.3) is displayed upon successful logon. After invoking the *Main Window*, the user can take one of three major paths through the database system: manipulate project and/or increment data, manipulate data by Cleanroom artifact, or manipulate data by Cleanroom process. If a project has been selected as the current project, the project name is displayed in the upper left corner of the window. If an increment of a selected project has been selected as the current increment, the increment number is displayed in the upper right corner of the window.

Menu. The *Action* menu on the *Main Window* provides a maximum of four options to the user: *Connect, Projects..., Increments...,* and *Exit.*

Artifacts Icon. Selecting the *Artifacts* icon from the *Main Window* invokes the *Browse Artifacts Window.*

Processes Icon. Selecting the *Processes* icon from the *Main Window* invokes the *Browse Processes Window.*

6.3.3 Projects Window

The *Projects Window* (Figure 6.4) provides the user with a list of all projects which have been registered with the CPD. Upon initiation, the *Projects* list is populated with records queried from the PROJECT table. If the user selects a project from the project list, the selected project and its description are displayed beneath the project list. Four buttons are available on this window: *Select, Modify, New,* and *Cancel.*

Select Button. Selecting the *Select* button makes the highlighted project the current project and returns the user to the calling window. The current project is displayed at the top left of the calling window.

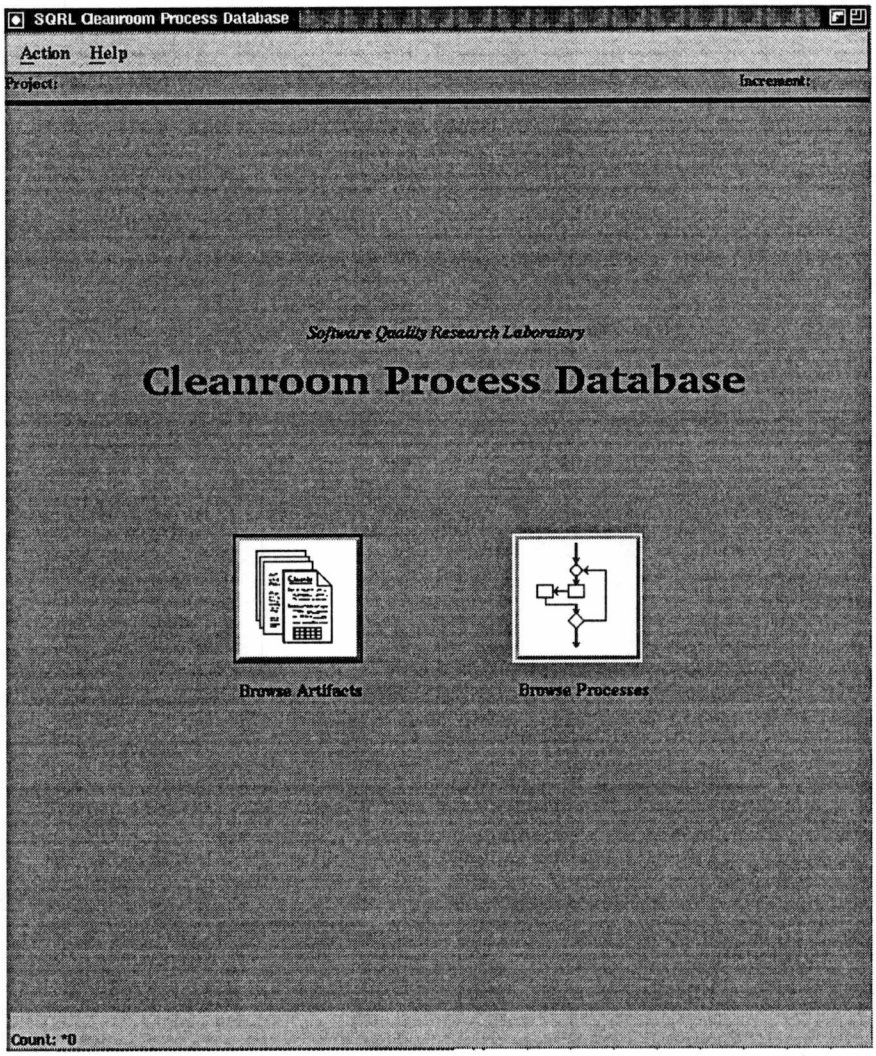

Figure 6.3: Main Window

New Button. Selecting the *New* button displays the *New Project Window.* This window allows the user to enter a name, start date, and description of the new project.

Modify Button. Selecting the *Modify* button displays the *Modify Project Window.* The information contained in this window corresponds to the project name which was highlighted in the project list of the *Projects Window* and can be updated or deleted as desired.

Cancel Button. Selecting the *Cancel* button returns the user to the calling window.

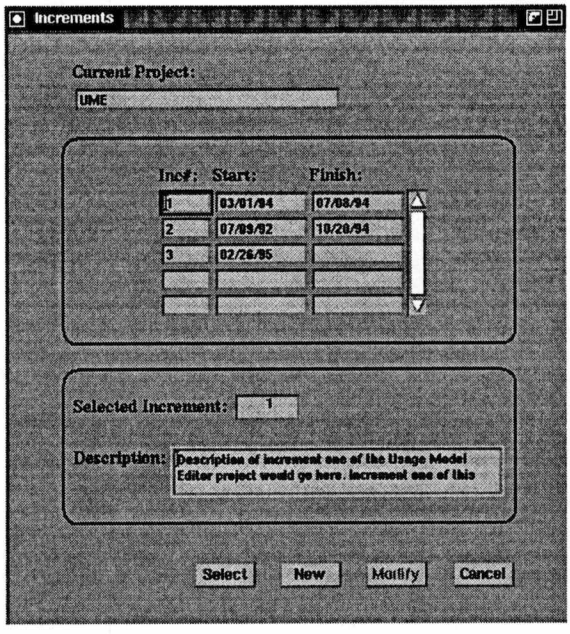

Figure 6.4: Projects Window

6.3.4 Modify Project Window

The *Modify Project Window* allows the descriptive information about a project held in the PROJECT table of the database to be updated. The project name, start date, completion date, and description can be altered. The information contained in the window corresponds to the project name which was highlighted in the project list of the *Projects Window.* Two buttons are available on this window: *OK* and *Cancel.*

Menu. The Oracle Forms default menu is displayed on this window.

6.3.5 New Project Window

The *New Project Window* (Figure 6.5) allows descriptive information about a new project to be inserted into the database. Through this window, records can be inserted, updated, queried, and deleted from the PROJECT table. The project name, start date, com-

pletion date, and description can be entered. Two buttons are available on this window: *OK* and *Cancel.*

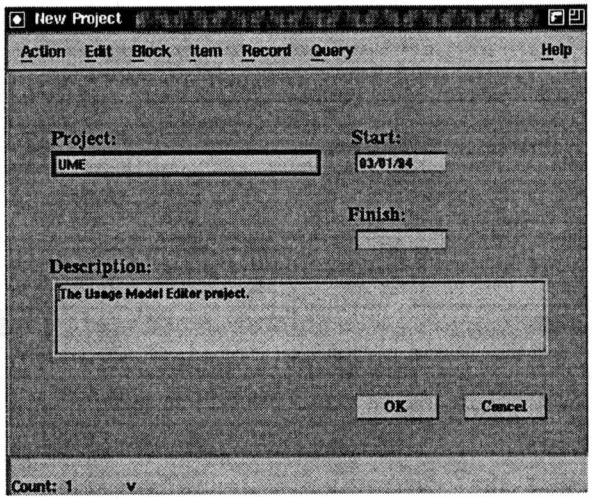

Figure 6.5: New Project Window

Menu. The Oracle Forms default menu is displayed on this window.

6.3.6 Increments Window

The *Increments Window* (Figure 6.6) provides the user with a list of all increments for the currently selected project which have been registered with the CPD. Upon initiation, the *Increments* list is populated with records queried from the INCREMENT table. If the user selects an increment from the list the selection and its description are displayed beneath the increment list. Four buttons are available on this window: *Select, Modify, New,* and *Cancel.*

Select Button. Selecting the *Select* button makes the highlighted increment the current increment and returns the user to the calling window. The current increment is displayed at the top right of the calling window.

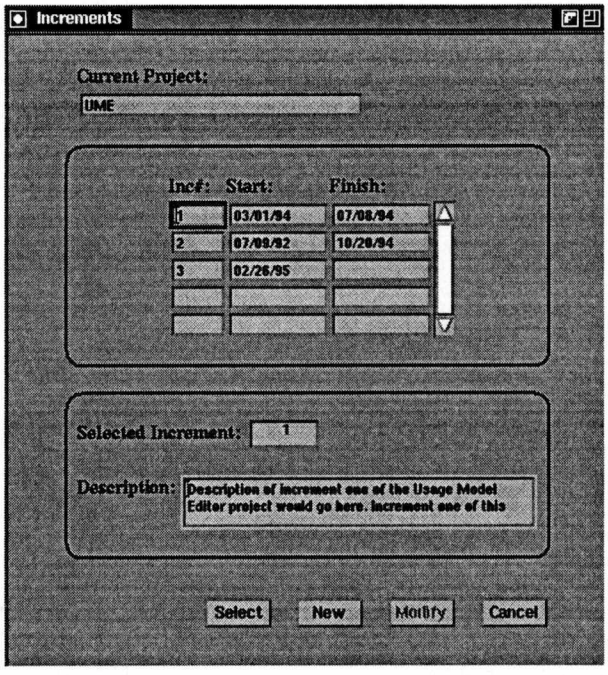

Figure 6.6: Increments Window

New Button. Selecting the *New* button displays the *New Increment Window.* This window allows the user to enter a number, start date, and description of the new increment.

Modify Button. Selecting the *Modify* button displays the *Modify Increment Window.* The information contained in this window corresponds to the increment number which was highlighted in the increment list of the *Increments Window* and can be updated or deleted as desired.

Cancel Button. Selecting the *Cancel* button returns the user to the calling window.

6.3.7 Modify Increment Window

The *Modify Increment Window* allows the descriptive information about an increment to be edited. The increment name, start date, completion date, and description of a valid increment of a valid project can be altered. The information contained in the window corresponds to the increment number which was highlighted in the increment list of the

Increments Window and corresponds directly to a record of the INCREMENT table. Two buttons are available on this window: *OK* and *Cancel.*

Menu. The Oracle Forms default menu is displayed on this window.

6.3.8 New Increment Window

The *New Increment Window* (Figure 6.7) allows descriptive information about a new increment of a selected project to be inserted into the database. Through this window, a new increment number, start date, completion date, and description can be entered into the INCREMENT table. Two buttons are available on this window: *OK* and *Cancel.*

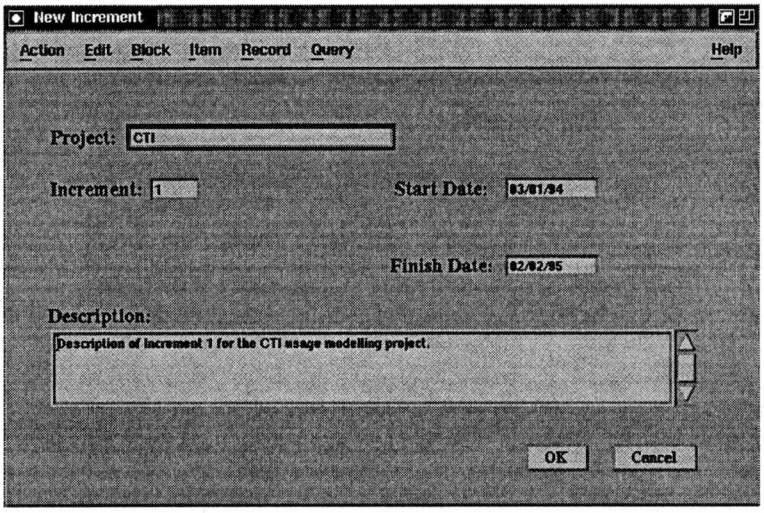

Figure 6.7: New Increment Window

Menu. The Oracle Forms default menu is displayed on this window.

6.3.9 About the CPD Window

The *About the CPD Window* (Figure 6.8) contains the text "The Cleanroom Process Database is being developed by the Software Quality Research Laboratory at the University of Tennessee". A bitmap of the CPD team members is displayed.



Figure 6.8: About the CPD Window

SQRL Image. Selecting the *SQRL Image* dismisses the *About the CPD Window* and returns the user to the calling window.

6.3.10 Browse Processes Window

The *Browse Processes Window* (Figure 6.9) provides the user with a hierarchical list of the Cleanroom processes along with a list of the artifacts associated with a particular process. Upon initiation, the *Process* list is populated with the records returned by a query on the PROCESS table which selects all processes which have the "Cleanroom Project" process as a parent. The user can navigate through the process hierarchy by selecting a process in the list followed by a selecting either the *Arrow Left* icon or the *Arrow Right* icon. Three buttons are displayed on this window: *Show Structure*, *OK*, and *Cancel*.

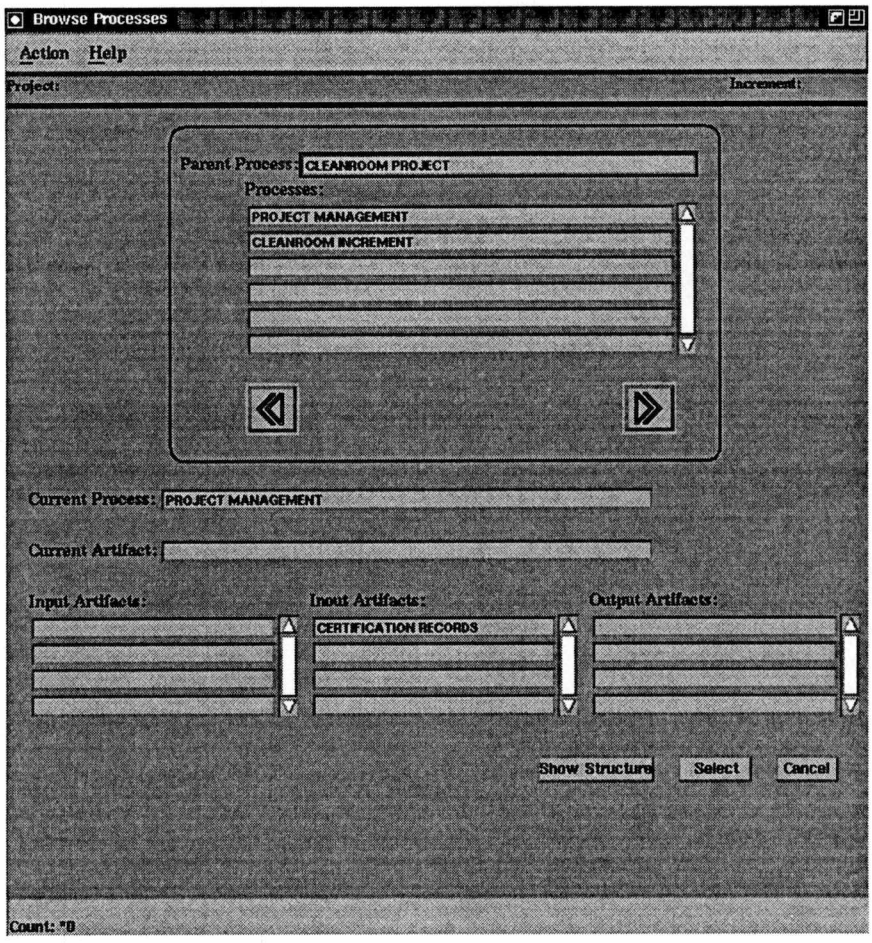

Figure 6.9: Browse Processes Window

Menus. The *Action* menu (Figure 6.1) provides a maximum of seven options to the user: *Connect, Projects..., Increments..., Process Definitions, Artifact Categories, File Types* and *Exit.*

Process Selection. If the user selects a process from the list the selected process is displayed in the *Current Process* text box beneath the process list. The parent process in the process hierarchy associated with the selected process is displayed above the process list in the *Parent Process* text box. If the process is a leaf in the process hierarchy, the text "(leaf)" is displayed to the left of the process in the list.

Artifacts associated with the selected process in the process list are displayed in the three artifact windows at the bottom of the screen according to the classification of the artifact as either input, inout, or output. Records are selected from the PROC_ART_TYPE view into the appropriate artifact listbox. Cleanroom artifact categories specified in the Process Description to be used at the selected process step are displayed in the appropriate artifact list according to the *in/out* use. Additional artifact categories may appear if the user selects a project, or a project and increment pair. All artifact categories specific to a selected project that are also associated with the selected process are displayed. All artifact categories specific to a selected project and increment pair that are also associated with the selected process are also displayed.

Arrow Left Icon. Selecting the *Arrow Left Icon* allows the user to navigate higher in the process hierarchy. The process list is populated with all processes one step higher in the hierarchy. If the *Process Structure Window* is displayed, then it will be refreshed to display the schema associated with the newly listed processes.

Arrow Right Icon. Selecting the *Arrow Right Icon* allows the user to navigate lower in the process hierarchy. The process list is populated with all sub-processes contained within the selected process. If the *Process Structure Window* is displayed, then it will be refreshed to display the schema associated with the newly listed processes.

Artifact Selection. If the user selects an artifact listed in the *Input*, *Inout*, or *Output* lists the selected artifact is displayed in the *Current Artifact* text box beneath the *Current Process* text box. Upon selecting the *Select* button, the user is presented with the chosen artifact. The *Input*, *Inout*, and *Output* lists are intended to be informative in nature and the restrictions on use implied by these properties are not enforced by the CPD. The current project and increment become the default project and increment values in a *form* or *report*

artifact. Similarly, the current project and increment values are passed in to the *File Summary Window* if the selected artifact is an artifact category of *file* type.

Show Structure Button. The high-level Cleanroom processes are represented in the Process Description by a "process schema" for concurrent, cooperating processes. Selecting the *Show Structure* button displays the *Process Structure Window* (Figure 6.10) which contains this "process schema." This window is asynchronous in nature, that is, the user can continue to work in the *Browse Processes Window* with the *Process Structure Window* displayed. Once displayed, the schema is dynamically updated to reflect the schema associated with the processes displayed in the process list. A highlighted process in the process list is also highlighted in the process schema.

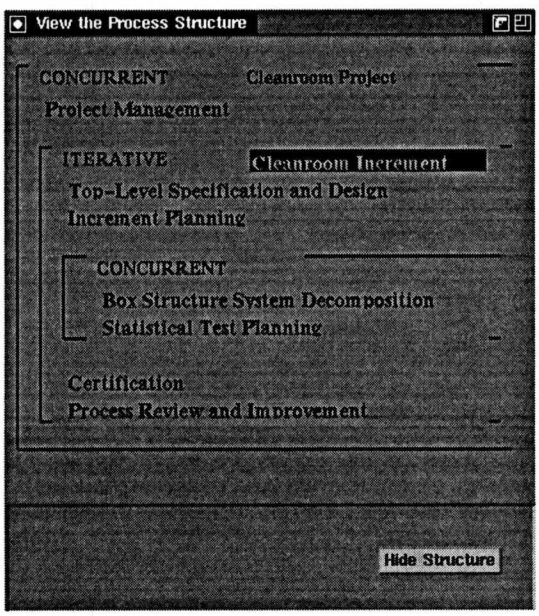

Figure 6.10: Process Structure Window

Select Button. Selecting the *Select* button navigates the user to the selected artifact. The particular response depends on the properties of the artifact category that was highlighted when the *Select* button was selected. For an artifact of type *form,* the specific form executable is run. The user can then insert, delete or modify records through the form. Similarly,

if an artifact of *report* type is selected, the report executable is run. The user can then generate reports according to the parameters of the report parameter form. If an artifact of *file* type is selected, the *File Summary Window* is displayed. This window lists all of the instantiations of the selected Cleanroom artifact category. From this window the user can elect to insert a new file, update an existing file, or delete an existing file. If an artifact of *class* type is selected, the *Class Window* is displayed. This window contains a list of all of the Cleanroom artifacts which are members of the class. The user can select an artifact from this list by highlighting an artifact and selecting the *OK* button. If the selected artifact category is a class, then the *Class Window* is redisplayed with the components of that class. Otherwise, the response is the same as that of selecting the *Select* button from the *Browse Processes Window.* Selecting *Cancel* from the *Class Window* returns the user to the *Browse Processes Window.* Selecting the *Find* button from the *Class Window* returns those artifact categories listed which match the regular expression displayed in the *Find* text box at the top of the window.

Cancel Button. Selecting the *Cancel* button dismisses the *Process Structure Window*, if it is displayed, and returns the user to the *Main Window.*

6.3.11 Browse Artifacts Window

The *Browse Artifacts Window* (Figure 6.11) allows the user to query the artifacts by several criteria. Three buttons appear on this window: *Query*, *Select*, and *Cancel.*

Menu. The *Action Menu* provides six options to the user: *Connect, Projects..., Increments..., Artifact Categories, File Types* and *Exit.*

Artifact Query and Selection. The user can select any of the artifact properties including scope, type, and a substring of a category name. Upon issuing a query by selecting the *Query* button, a list of all artifact categories matching the criteria is displayed in the *Arti-*

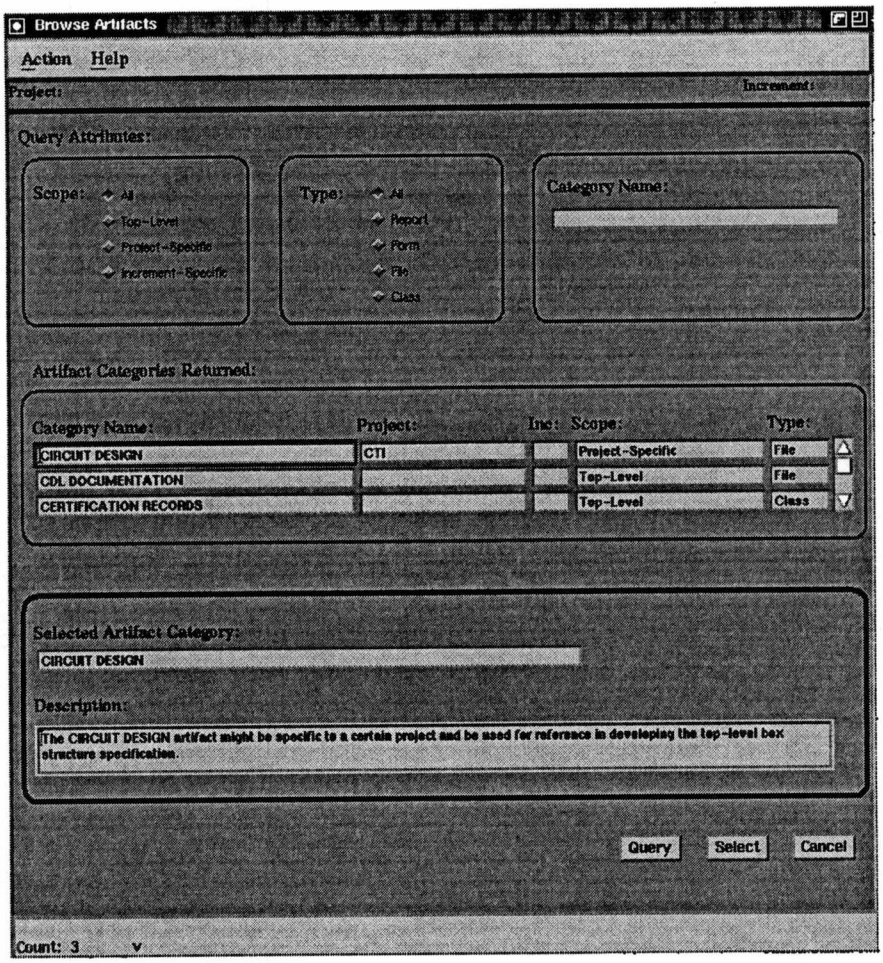

Figure 6.11: Browse Artifacts Window

fact Categories Returned listbox. If the user selects an artifact category from the list the selected artifact category name is displayed in the *Selected Artifact Category* text box with the associated description from the artifact category definition. Once a category has been selected, the user can navigate to the category upon selection of the *Select* button. The current project and increment become the default project and increment values in a *form* or *report* artifact. Similarly, the current project and increment values are passed in to the *File Summary Window* if the selected artifact is an artifact category of *file* type.

Scope Selection. The user can specify the scope of the artifacts to be queried by "checking" one of the items in the radio group associated with the scope values. Note that only

one scope value may be selected. For example, the user might wish to query the database for all data specific to the UME project in which case he would make UME the current project and "check" the "Project-Specific" option in the *Scope* radio group.

Type Selection. The user can specify the type of artifacts to be queried by "checking" one of the items in the radio group associated with the type values. Note that only one type value may be selected. For example, the user might wish to query the database for all artifact categories of file type that are specific to the UME project in which case he would make UME the current project, "check" the "Project Specific" option in the *Scope* radio group, and select "File" from the *Type* radio group.

Category Name Substring. The user can specify a substring of the name of the artifact category in the *Category Name* text box. The '%' symbol defines a wildcard and the '_' symbol defines a position marker.

Query Button. Selecting the *Query* button executes the query with the selected artifact properties and returns a list of matching artifact categories into the *Artifact Category list-box* of the *Artifact Category Window.*

Select Button. Selecting the *Select* button allows the user to navigate to the selected artifact on the artifact list. The particular response that occurs depends on the artifact category that was selected. For an artifact of type *form,* the specific form executable is run. The user can then insert, delete or modify records through the form. Similarly, if an artifact of *report* type is selected, then the report executable is run. The user can then generate reports according to the parameters of the report parameter form. If an artifact of *file* type is selected, the *File Summary Window* is displayed. This form lists all of the instantiations of the selected Cleanroom artifact category. From this form the user may elect to insert a new file, update an existing file, or delete an existing file. If an artifact of *class* type is selected, a window is displayed which contains a list of all of the Cleanroom artifacts

which are members of the class (Figure 6.12). This list is generated from a query on a join on the BROWSE_ART view and the ARTIFACT_CLASS table. The user can select an artifact from this list by highlighting an artifact and selecting the *OK* button. If the selected artifact category is a class, then the *Class Component List* is redisplayed with the components of that class. Otherwise, the response is the same as that of selecting the *Select* button from the *Browse Artifacts* Window. Selecting cancel from the *Class Component List* returns the user to the *Browse Artifacts Window.* Selecting the *Find* button from the *Class Component List* returns those artifact categories listed which match the regular expression displayed in the *Find* text box at the top of the window.

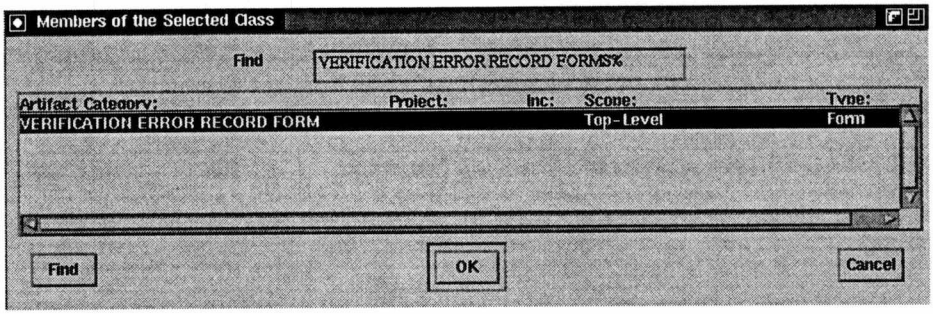

Figure 6.12: Browse Artifacts Class Component List

6.3.12 Processes Window

The *Processes Window* (Figure 6.13) allows for the definition of processes and the relationships between them. The process hierarchy is built from the process interrelationships defined though this window. Existing process definitions can be modified though this window. This functionality is useful since the Process Description might evolve over time. Records in this window correspond to the PROCESS table. Only a user who has been granted the CPD_ADMIN role is permitted to modify the data accessed through this form. Two buttons are displayed on this window: *OK* and *Cancel.*

Menu. The Oracle Forms default menu is displayed on this window.

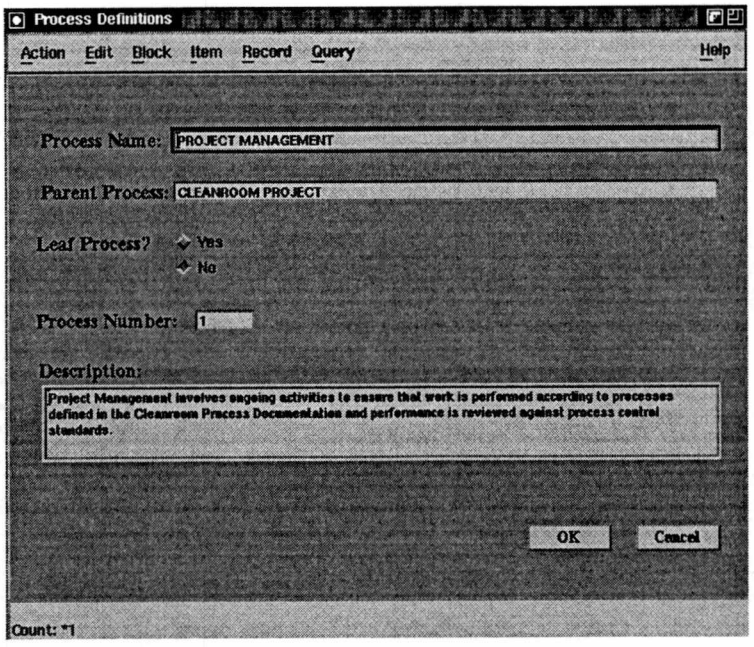

Figure 6.13: Processes Window

Parent Process Selection. The PROCESS table is queried and the resulting list of valid Cleanroom processes (Figure 6.14) is displayed when the user enters the *Parent Process* text field. The process selected from the list is displayed in the *Parent Process* text field.

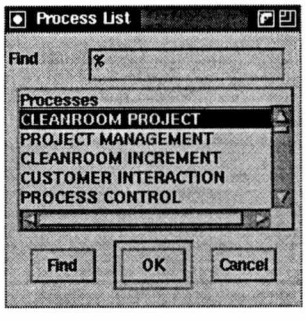

Figure 6.14: Process Selection List

6.3.13 File Type Window

The *File Type Window* (Figure 6.15) allows for the definition of valid Cleanroom artifact file types and for the modification of existing file type definitions. The information

in this window corresponds to the FILE_TYPE table. Two buttons are displayed on this window: *OK* and *Cancel.*

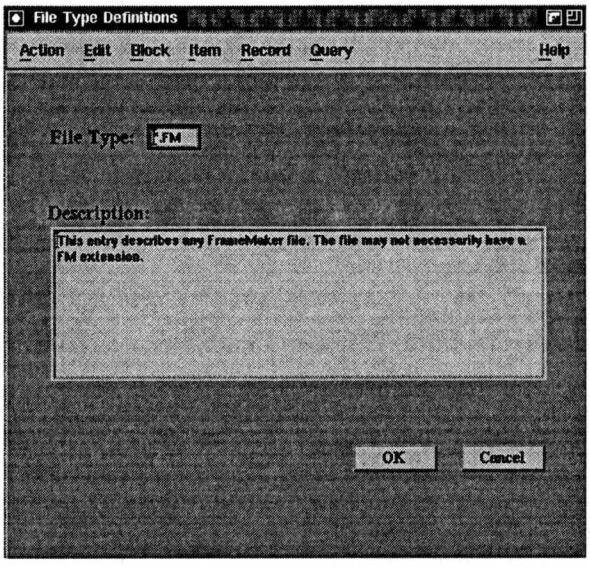

Figure 6.15: File Type Window

Menu. The Oracle Forms default menu is displayed on this window.

6.3.14 Artifact Category Window

The *Artifact Category Window* (Figure 6.16) allows options for the modification of the definition of the Cleanroom artifact categories. The information in the ARTIFACT_CATEGORY, PROCESS_ARTIFACT, PROJ_SPEC_ART_CAT, and INC_SPEC_ART_CAT tables may be inserted, deleted, or updated from this window. The attributes of each artifact category such as scope, classes to which it belongs, processes that use the artifact category as defined by the Process Description, and a textual description of the artifact category may be edited in this form. Only a user who has been granted the CPD_ADMIN role shall be permitted to modify the data accessed through this window. Two buttons appear on this window: *OK* and *Cancel.*

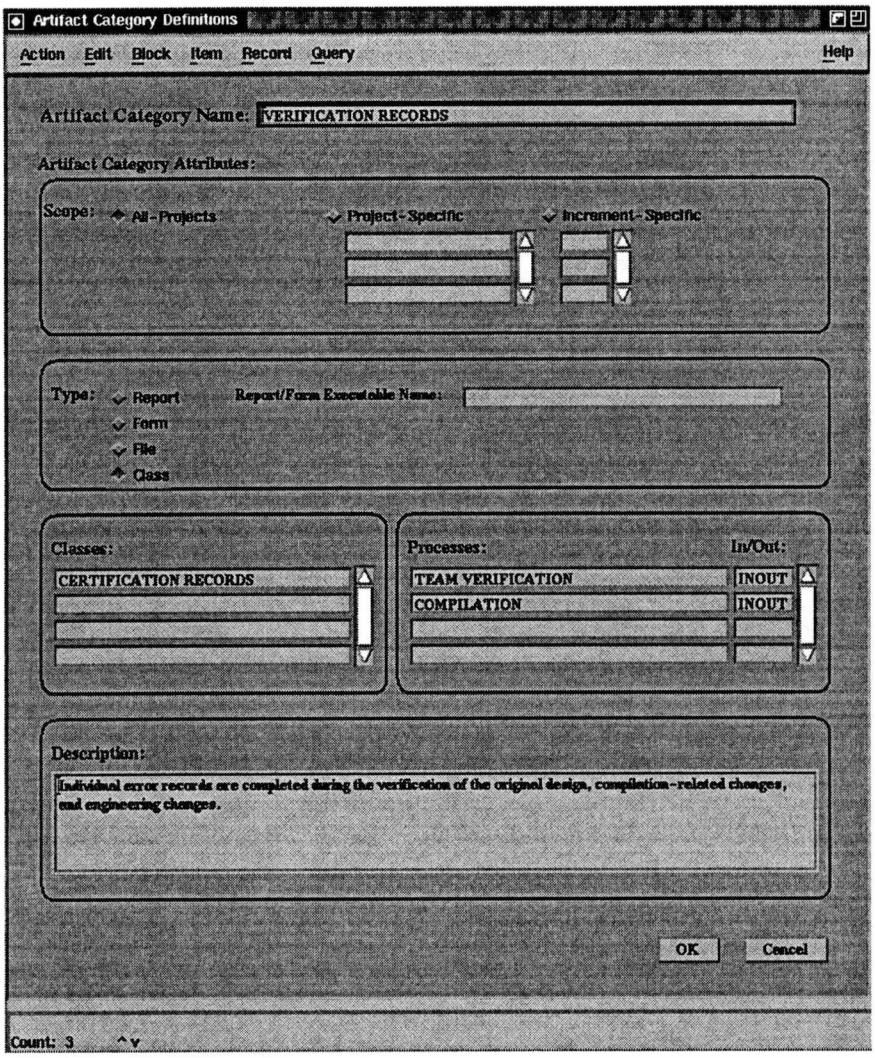

Figure 6.16: Artifact Category Window

Menu. The Oracle Forms default menu is displayed on this window.

Class Selection. Upon entering the *Classes* field of the *Artifact Category Window*, the ARTIFACT_CLASS table (Figure 6.17) is queried and the user is prompted with a list of the artifacts of *class* type. The user can select a class to which the new artifact category shall belong from this list.

Process Selection. Similarly, upon entering the *Processes* list the user is prompted with a list of available processes (Figure 6.14) queried from the PROCESS table.

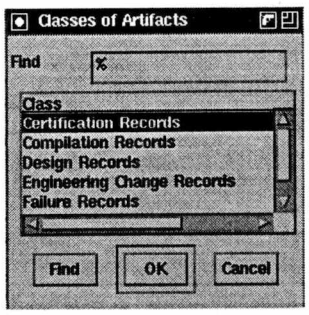

Figure 6.17: Artifact Class Selection

In/Out Selection. Upon entering the In/Out text list a list of the valid types of use (Figure 6.18) are presented to the user. This list is statically created and is not the result of a query on any table.

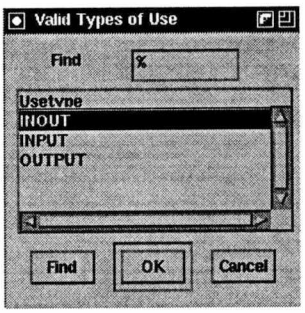

Figure 6.18: In/Out List

6.3.15 File Summary Window

The *File Summary Window* (Figure 6.19) provides the user with a list of all the files in the database for a specific Cleanroom artifact category of *file* type. The user initiates this form from by selecting an artifact category of *file* type from either the *Browse Artifacts* or *Browse Processes Windows*. The artifact category selected in either of the previous windows becomes the artifact category displayed in the *Artifact Category* text field of the *File Summary* form. Five buttons are provided on this window: *Query, Delete, Update, New,* and *Extract.*

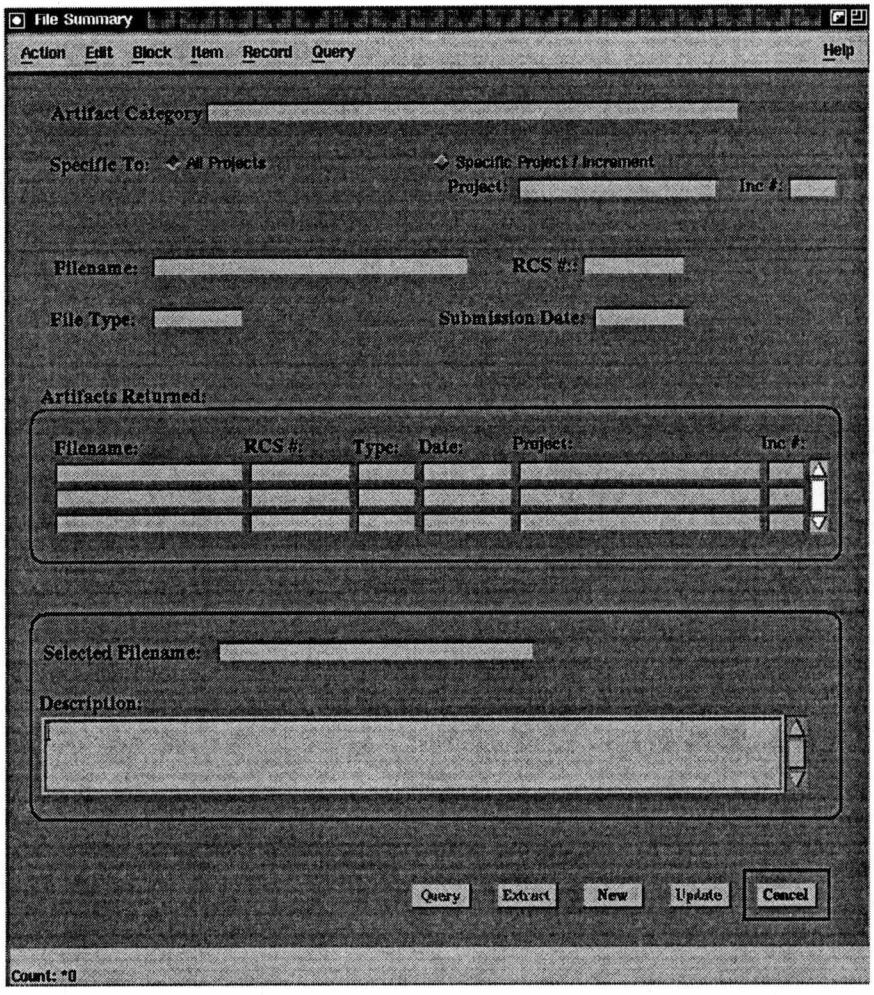

Figure 6.19: File Summary Window

Menu. The Oracle Forms default menu is displayed on this window.

Query Construction. The user may construct a query according to the properties of the selected artifact category contained in the BROWSE_ART_INST view.

File Type selection. Upon entering the *File Type* text field the FILE_TYPE table is queried and the user is prompted with a list of valid file types (Figure 6.20). Upon selecting one of these file types, the selection is displayed in the *File Type* text field.

Query Button. The query is executed and by selecting the *Query* button. All files matching the query criteria shall be displayed in the *Artifact Instances Returned* list.

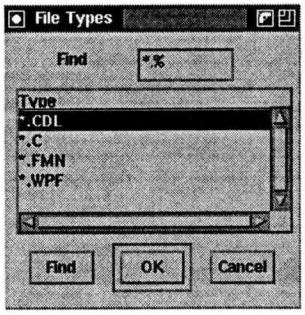

Figure 6.20: Artifact File Type List

Delete Button. Selecting the *Delete* button allows the user to delete files from the database. Whatever file is highlighted when the *Delete* button is selected shall be deleted.

Update Button. Selecting the *Update* button allows the user to update existing records for files in the database. The *Update File Window* is initiated when the user selects the *Update* button. The corresponding record for whatever file is highlighted when the *Update* button is selected shall be placed in the *Update File Window.*

New Button. Selecting the *New* button allows the user to insert new records for files in the database. The *New File Window* is initiated when the user selects the *New* button.

Extract Button. Selecting the *Extract* button allows the user to extract existing files from the database to the file system. Selecting the *Extract* button results in the initiation of the *Extract To Location File Browser.* This window prompts the user for the location in the file system to extract the file.

6.3.16 New File Window

The *New File Window* (Figure 6.21) allows the user to insert a new file into the database from the file system. The user may specify the various properties of the artifact instance which is being added to the database. An entry appears in the

ARTIFACT_FILE_INSTANCE table for each file which is an instance of a particular artifact category.

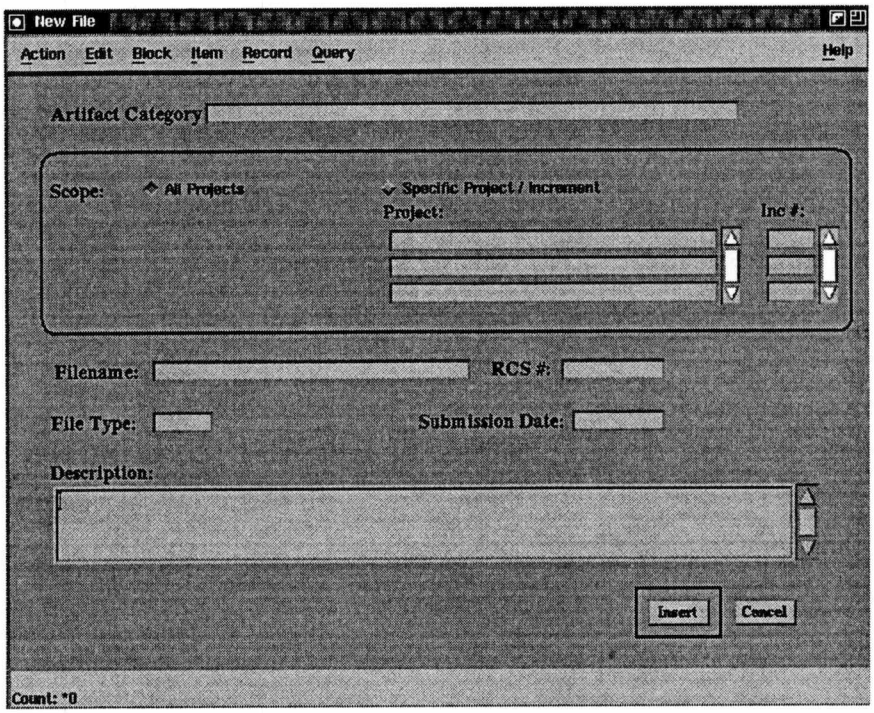

Figure 6.21: New File Window

The file may be tied to multiple projects or to multiple increments of projects. A record is maintained in the PROJ_ART_FILE_INST table for each project to which the file corresponds. A record is maintained in the INC_ART_FILE_INST table for each increment of a project to which a file corresponds. Two buttons appear on this window: *OK* and *Cancel.*

Menu. The Oracle Forms default menu is displayed on this window.

OK Button. Selecting the *OK* button, initiates the *Insert from File Browser* to prompt the user for a location from which to insert a file.

6.3.17 Update File Window

The *Update File Window* allows for the update of an existing Cleanroom artifact of file type which exists in the database. The record corresponding to the selected file in the *File Summary Window* is placed in this window. The user may update the various properties of the artifact instance which exists in the database. This window is similar to the *Artifact Category Window* in that the information being updating corresponds to entries in the ARTIFACT_FILE_INSTANCE, PROJ_ART_FILE_INST, and INC_ART_FILE_INST tables. Two buttons appear on this window: *OK* and *Cancel.*

Menu. The Oracle Forms default menu is displayed on this window.

OK Button. Selecting the *OK* button with the *Replace Existing File* radio group set to Yes initiates the *Insert File Browser.* The user can select a file from the file browser to update the existing file that is currently stored in the database. Selecting the *OK* button with the *Replace Existing File* radio group set to *No* commits any changes made to the file definition to the database without updating the actual file itself.

6.3.18 Extract File Browser

This window (Figure 6.22) prompts the user for the location in the file system to which to extract the file. Three buttons appear on this window: *OK*, *Filter*, and *Cancel.*

OK Button. Selecting the *OK* button results in the acceptance of the file location, extracts the file to the specified location, and returns the user to the *File Summary Window.*

Filter Button. Selecting the Filter button populates the directory and file list with entries matching the specified filter text.

Cancel Button. If this button is selected, the transaction shall be aborted and control returned to the *File Summary Window.*

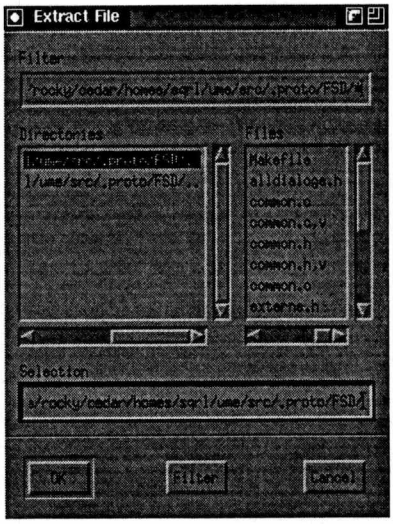

Figure 6.22: Extract File Browser

6.3.19 Insert File Browser

The *Insert File Browser* prompts the user for the location from which to load the file. Three buttons appear on this window: *OK*, *Filter*, and *Cancel.*

OK Button. Selecting the *OK* button results in the acceptance of the file location, inserts the new record including the file from the file system, and returns the user to the *File Summary Window.*

Filter Button. Selecting the Filter button populates the *Directories* and *Files* lists with entries matching the specified filter text.

Cancel Button. If this button is selected, the transaction shall be aborted and control returned to the *File Summary Window.*

6.4 Forms Artifacts

A description is provided in this section of each of the artifact categories which are classified as *form* type.

6.4.1 Tasking Form

Through the *Tasking Form* (Figure 6.23) records in the TASKING, TASKING_SQRL, TASKING_PROJ, and TASKING_INC tables can be inserted, deleted, or updated. The information in this form is used to validate names of SQRL members inserted into other forms. For example, a list of development team members for a specified increment of a project shall be displayed when the user navigates to the verifiers field of the *Verification Meeting Form.* Two buttons appear on this form: *OK* and *Cancel.*

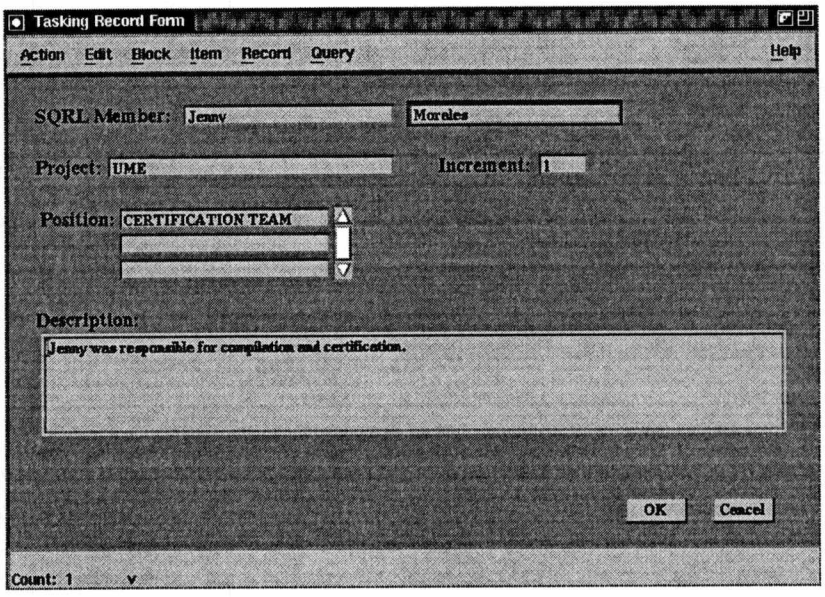

Figure 6.23: Tasking Form

Menu. The Oracle Forms default menu is displayed on this form.

Project Selection. Upon entering the *Project* field, the user is prompted with a list of all valid project names which have been identified to the database (Figure 6.24).

Increment Selection. Upon entering the *Increment* field, the user is prompted with a list of all valid increments for the specified project name which have been identified to the database (Figure 6.25).

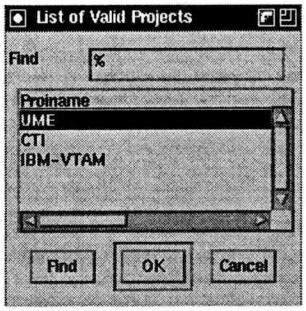

Figure 6.24: List of Valid Projects

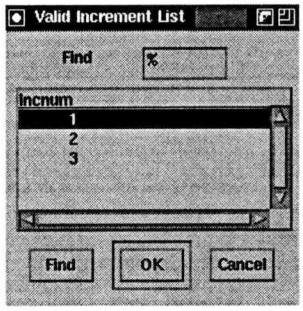

Figure 6.25: List of Valid Increments

Tasking Option Selection. Upon entering the *Position* field, the user is prompted with a list of valid tasking options that the user might occupy (Figure 6.26). This list is static and is not identified through any form.

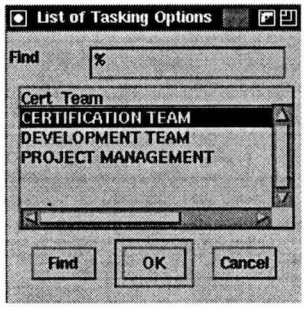

Figure 6.26: List of Valid Tasking Options

6.4.2 Verification Meeting Form

The *Verification Meeting Form* (Figure 6.27) is the first form to be completed in a verification session. This form is completed after the box structure files have been checked into RCS and the *insert_rcs_info* script has been run to load the RCS log information into the database. Through this form, records in the VER_MEET, VER_MEET_VERIFIERS, and VER_MEET_FILENAMES tables may be inserted, deleted, or updated. Six buttons appear on this form: *Unverified VerFF, Unverified CompFF, Unverified TestFF, New VerFF, OK, Cancel.*

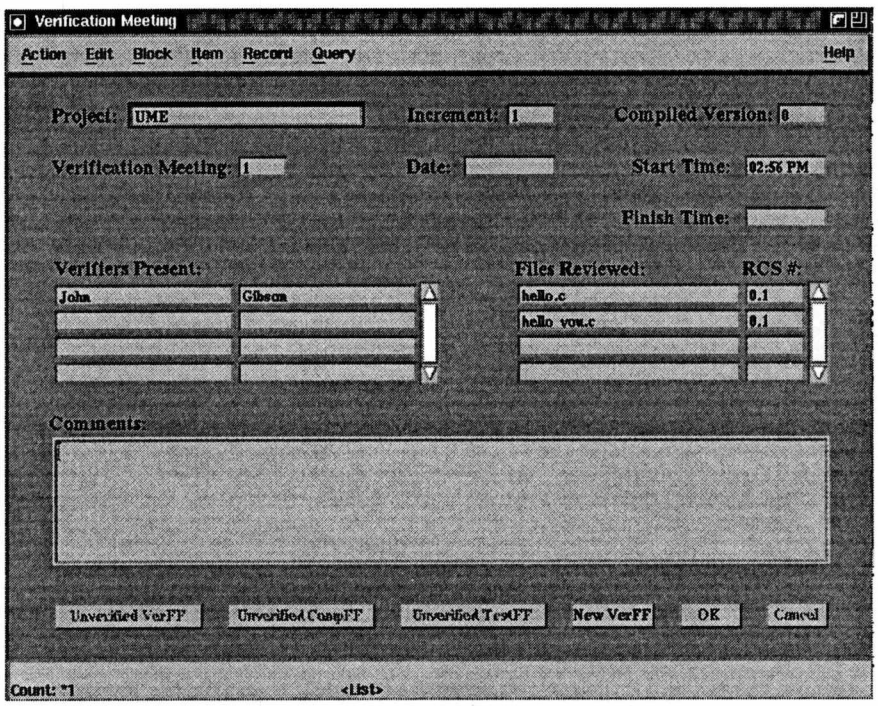

Figure 6.27: Verification Meeting Form

Menu. The Oracle Forms default menu is displayed on this form.

Project Selection. Upon entering the *Project* field, the user is prompted with a list of all valid project names which have been identified to the database (Figure 6.24).

Increment Selection. Upon entering the *Increment* field, the user is prompted with a list of all valid increments for the specified project name (Figure 6.25).

Version Selection. Upon entering the *Version* field, the user is prompted with a list of all valid compiled versions of code for the specified project name and increment (Figure 6.28).

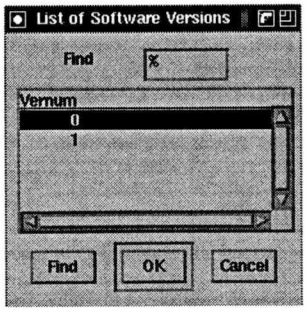

Figure 6.28: List of Valid Compiled Versions of Code

Meeting Number Selection. Upon entering the *Verification Meeting* field, the user is prompted with the next verification meeting number for the specified project name and increment (Figure 6.29).

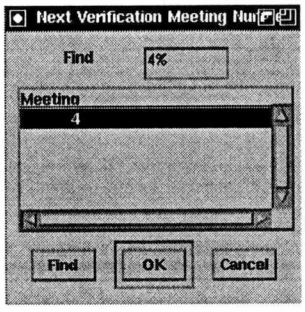

Figure 6.29: List Containing Next Verification Meeting Number

Verifier Selection. The verifiers present at the session may be listed in the *Verifiers Present* listbox. Upon navigation to the *Verifiers Present* list box, the user is prompted with a list of valid developers for the specified project and increment (Figure 6.30).

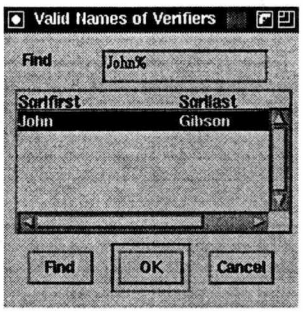

Figure 6.30: List of Developers

Files Reviewed Selection. Upon navigation to the *Files Reviewed* list box, the user is prompted with a list of valid filenames for the specified project. This is generated from information inserted into the database through the *insert_rcs_info* PERL script which strips out information from the RCS log.

Unverified VerFF Button. Upon selection of the *Unverified VerFF* button, the *Verification Fault Form* is displayed. When invoked from this context, the *Verification Fault Form* is passed in the project name, increment number, and verification meeting number. The *Verification Fault Form* queries the database for all unverified faults for the given project and increment. The user can verify the changes one at a time, before returning to the *Verification Meeting* form. The verification meeting number is recorded along with the verified filenames in the space provided at the bottom of the form.

Unverified CompFF Button. Upon selection of the *Unverified CompFF* button, the *Compilation Fault Form* is displayed. When invoked from this context, the *Compilation Fault Form* is passed in the project name, increment number, and verification meeting number. The *Compilation Fault Form* queries the database for all unverified faults for the given project and increment. The user may then verify the changes one at a time, before returning to the *Verification Meeting* form. The verification meeting number shall be placed in the space provided at the bottom of the form.

Unverified TestFF Button. Upon selection of the *Unverified TestFF* button, the *Testing Failure Form* is displayed. When invoked from this context, the *Testing Failure Form* is passed in the project name, increment number, and verification meeting number. The *Testing Failure Form* queries the database for all unverified failures for the given project and increment. The user may then verify the changes one at a time, before returning to the *Verification Meeting* form. The meeting number shall be placed in the space provided at the bottom of the form.

New VerFF Button. Upon selection of the *New VerFF* button, a new *Verification Fault Form* is displayed. Before displaying the VerFF, any pending transactions from the current form are committed. This is done so that when the new VerFF is displayed, the user may be prompted with a list of filenames and RCS numbers that are valid only for the current verification meeting number. This keeps the list of valid filenames from which the user has to choose to as short a length as possible.

When invoked from this context, the *Verification Fault Form* is passed in the project name, increment number, compiled version, and verification meeting number as parameters. The new form uses these as the default values for query or record insertion.

6.4.3 Verification Fault Form

This *Verification Fault Form* (Figure 6.31) is completed for every fault found during verification. Through this form, records in the VERIFICATION, VERF_FILENAMES, VERF_FUNCTIONS, and VER_MEET_FILENAMES tables can be inserted, deleted, or updated. Five buttons appear on this form: *New VerFF*, *Prev*, *Next*, *OK*, and *Cancel*.

Menu. The Oracle Forms default menu is displayed on this form.

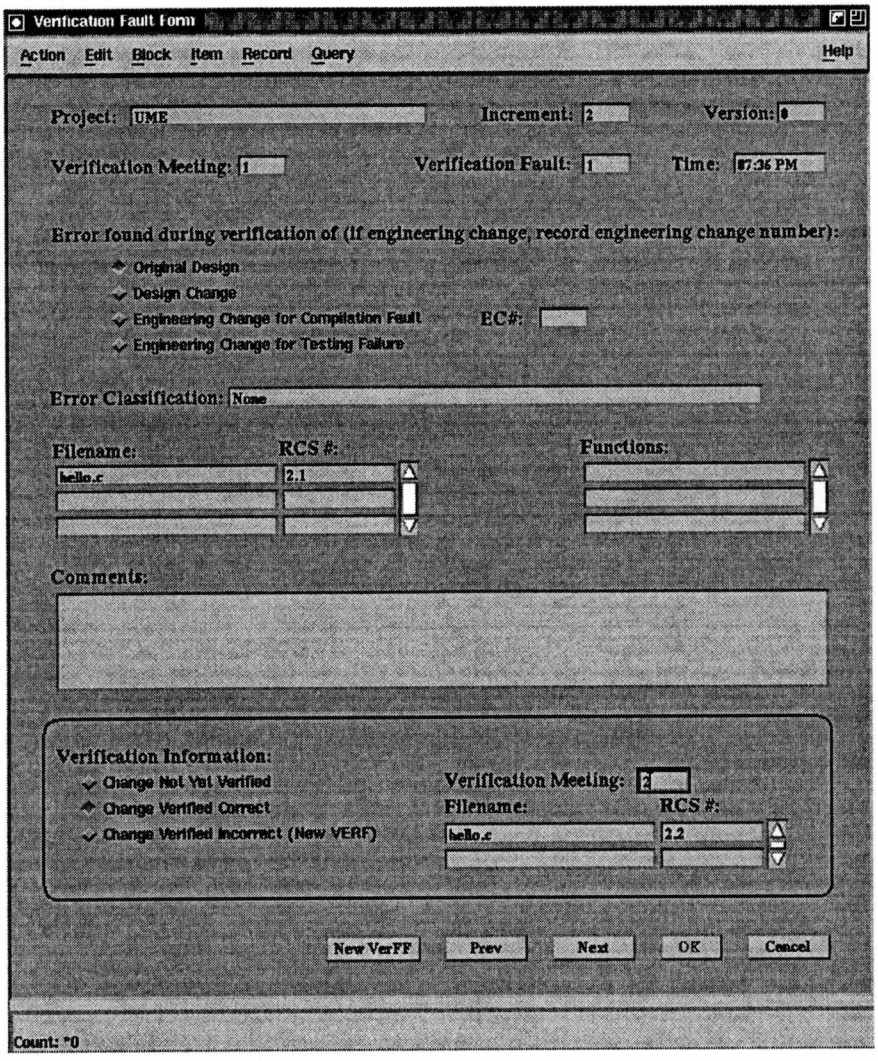

Figure 6.31: Verification Fault Form

Project Selection. Upon entering the *Project* field, the user is prompted with a list of all valid project names (Figure 6.24) which have been identified to the database through the *New Project Form.*

Increment Selection. Upon entering the *Increment* field, the user is prompted with a list of all valid increments for the specified project name (Figure 6.25). These increments would have been identified to the database through the *New Increment Form.*

Version Selection. Upon entering the *Version* field, the user is prompted with a list of all valid compiled versions of code for the specified project name and increment (Figure 6.28). These versions would have been identified to the database through the *Compilation Record Form.*

Meeting Number Selection. Upon entering the *Verification Meeting* field, the user is prompted with the next verification meeting number for the specified project name and increment (Figure 6.29).

Fault Number Selection. Upon entering the *Verification Fault* field, the user is prompted with the next verification fault number for the specified project name and increment.

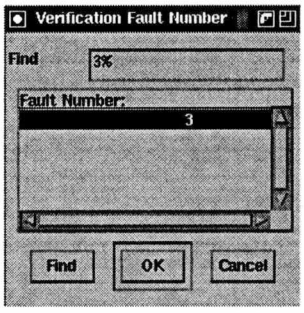

Figure 6.32: List Containing Next Verification Fault Number

Time. The value of the *Time* field defaults to the current time.

Filename Selection. Upon navigation to the *Filename* list box, the user is prompted with a list of valid filenames for the specified project, increment and verification meeting number. This list contains the names of all files which have been verified thus far for the specified project.

New VerFF Button. Upon selection of the *New VerFF* button, all records are cleared from the *Verification Fault Form.* The next verification fault can then be recorded.

Prev Button. Upon selection of the *Prev* button, the previous record is displayed in the form.

Next Button. Upon selection of the *Next* button, the next record is displayed in the form.

6.4.4 Compilation Session Form

The *Compilation Session Form* (Figure 6.33) is used to record information on each compilation performed for a specified increment of a project. Through this form, records in the COMP_MEET table can be inserted, deleted, or updated. Three buttons appear on this form: *New CompFF*, *OK*, and *Cancel.*

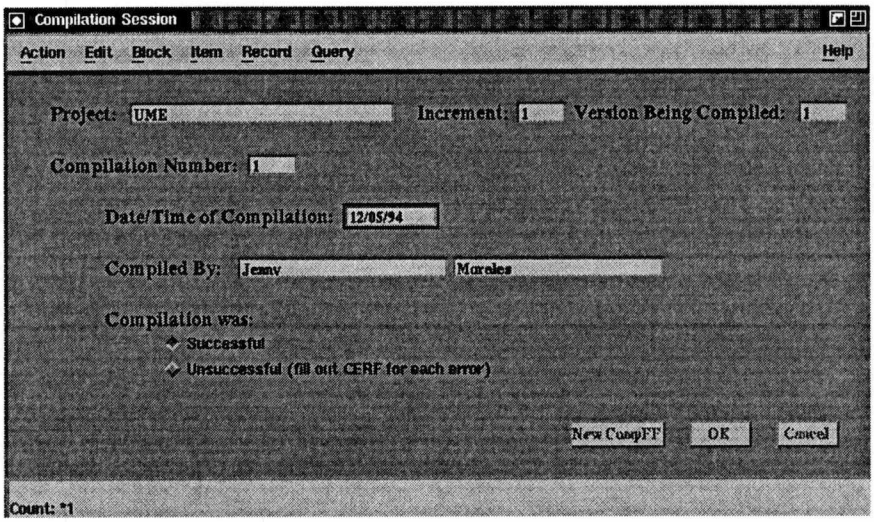

Figure 6.33: Compilation Session Form

Menu. The Oracle Forms default menu is displayed on this form.

Project Selection. Upon entering the *Project* field, the user is prompted with a list of all valid project names (Figure 6.24) which have been identified to the database through the *New Project Form.*

Increment Selection. Upon entering the *Increment* field, the user is prompted with a list of all valid increments for the specified project name *(Figure 6.25)*. These increments would have been identified to the database through the *New Increment Form.*

Version Selection. Upon entering the *Version* field, the user is prompted with a list containing the next compilation version number the specified project name and increment.

Compilation Number Selection. Upon entering the *Compilation Number* field, the user is prompted with the next compilation number for the specified project name and increment (Figure 6.29).

Time. The value of the *Time* field defaults to the current time.

Verifier Selection. Upon navigation to the *Compiled By* text box, the user is prompted with a list of valid certifiers for the specified project and increment.This is generated from information inserted into the *Tasking Form* (Figure 6.30).

New CompFF Button. Upon selection of the *New CompFF* button, a fresh record of the *Compilation Fault Form* is displayed. The next fault may be recorded in this fresh form.

6.4.5 Compilation Fault Form

A *Compilation Fault Form* (Figure 6.34) is completed for every fault found during compilation. Through this form, records in the COMPILATION, ENGINEERING_CHANGES, ENG_CH_FILENAMES, ENG_CH_FUNCTIONS and VER_MEET_FILENAMES may be inserted, deleted, or updated. Six buttons appear on this form: *New VerFF, New CompFF, Prev, Next, OK,* and *Cancel.*

Menu. The Oracle Forms default menu is displayed on this form.

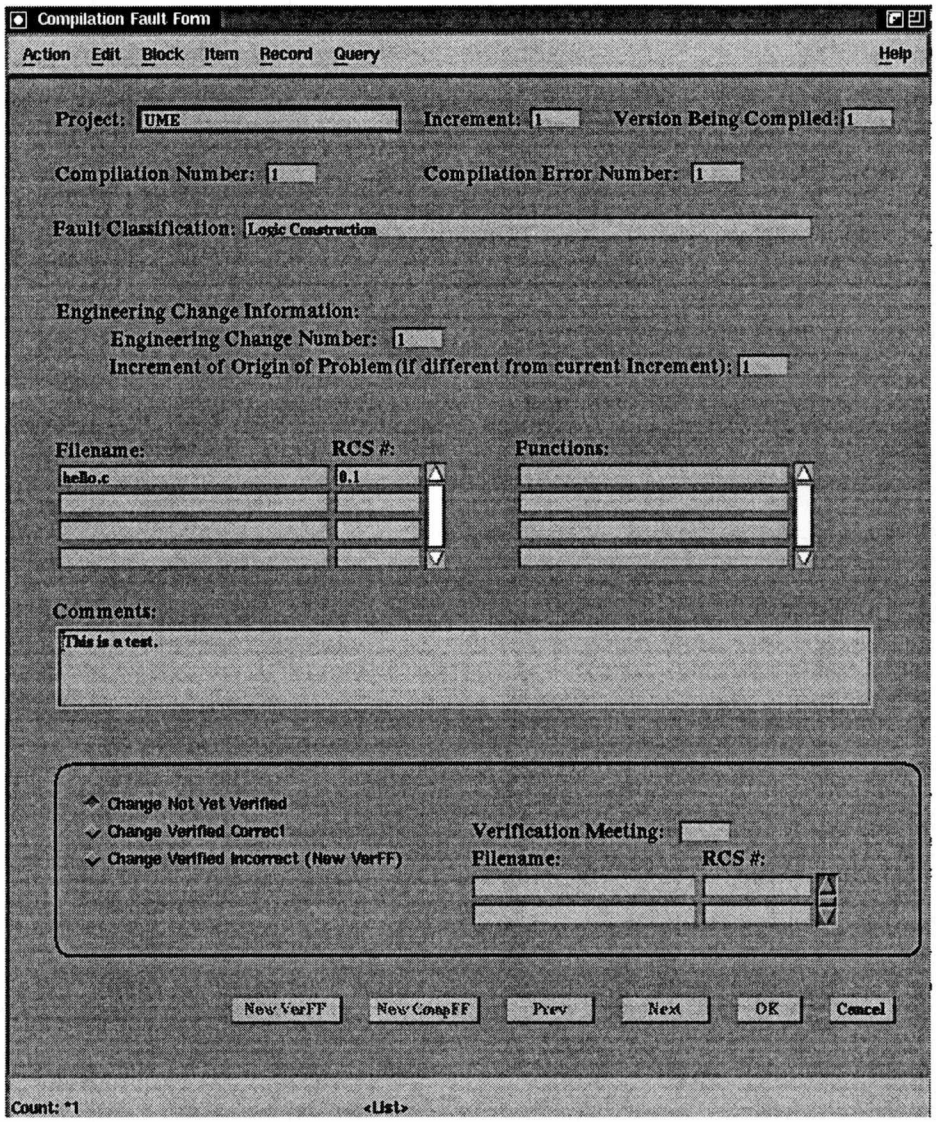

Figure 6.34: Compilation Fault Form

Project Selection. Upon entering the *Project* field, the user is prompted with a list of all valid project names which have been identified to the database through the *New Project Form (Figure 6.24).*

Increment Selection. Upon entering the *Increment* field, the user is prompted with a list of all valid increments for the specified project name. These increments would have been identified to the database through the *New Increment Form (Figure 6.25).*

Version Selection. Upon entering the *Version* field, the user is prompted with a list of all valid compiled versions of code for the specified project name and increment *(Figure 6.28)*. These versions would have been identified to the database through the *Compilation Record Form.*

Time. The value of the *Time* field defaults to the current time.

Compilation Number Selection. Upon entering the *Compilation Number* field, the user is prompted with the highest compilation number for the specified project name and increment. This functionality is similar to the list in the *Verification Fault Form* which prompts the user with the next verification meeting number. (Figure 6.29).

Fault Number Selection. Upon entering the *Compilation Fault* field, the user is prompted with the next compilation fault number for the specified project name and increment. This functionality is similar to the list in the Verification Fault Form which prompts the user for the next verification fault number. (Figure 6.32).

Filename Selection. Upon navigation to the *Filename* list box, the user is prompted with a list of valid filenames for the specified project, increment and verification meeting number. This list shall contain the names of all files which have been verified at the given verification meeting.

New VerFF Button. Upon selection of the *New VerFF* button, the *Verification Fault Form* is displayed. This functionality is useful since a *Verification Fault Form* must be completed for each incorrect engineering change which resulted from a compilation fault.

New CompFF Button. Upon selection of the *New CompFF* button, all records are cleared from the *Compilation Fault Form.* The next compilation fault can then be recorded.

Prev Button. Upon selection of the *Prev* button, the previous record is displayed in the form.

Next Button. Upon selection of the *Next* button, the next record is displayed in the form.

6.4.6 Testing Failure Form

The *Testing Failure Form* (Figure 6.35) is completed for every failure found during testing. Through this form, records in the TESTING,ENGINEERING_CHANGES, ENG_CH_FILENAMES, ENG_CH_FUNCTIONS and VER_MEET_FILENAMES may be inserted, deleted, or updated. Six buttons appear on this form: *New TestFF, New VerFF, Prev, Next, OK,* and *Cancel.*

Menu. The Oracle Forms default menu is displayed on this form.

Project Selection. Upon entering the *Project* field, the user is prompted with a list of all valid project names which have been identified to the database through the *New Project Form* (Figure 6.24).

Increment Selection. Upon entering the *Increment* field, the user is prompted with a list of all valid increments for the specified project name. These increments would have been identified to the database through the *New Increment Form* (Figure 6.25).

Version Selection. Upon entering the *Version* field, the user is prompted with a list of all valid compiled versions of code for the specified project name and increment (Figure 6.28). These versions would have been identified to the database through the *Compilation Session Form.*

Failure Number Selection. Upon entering the *Testing Failure Number* field, the user is prompted with the next testing failure number for the specified project name and increment (Figure 6.29).

Tester Selection. Upon entering the *Tester* field, the user is prompted with a list of the certification team members for the specified increment and project (Figure 6.30).

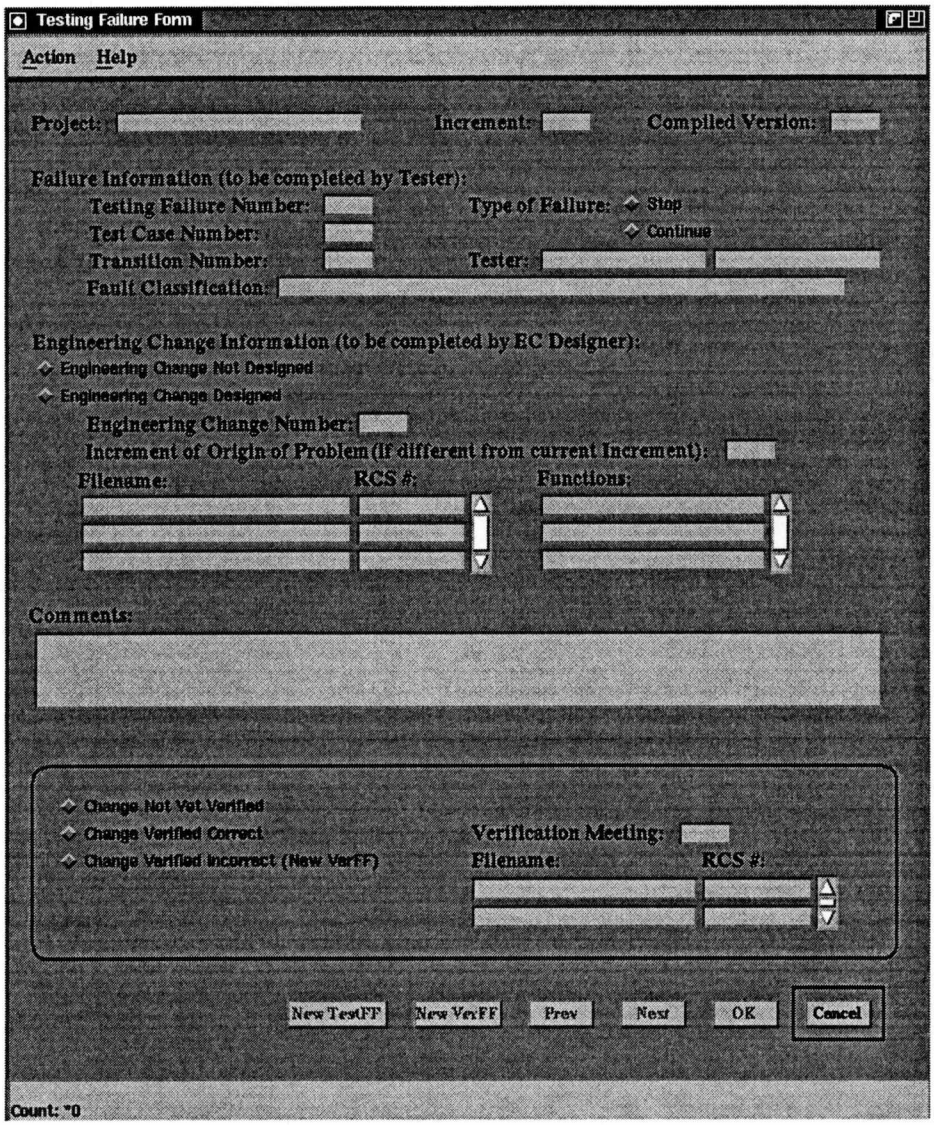

Figure 6.35: Testing Failure Form

Filename Selection. Upon navigation to the *Filename* list box, the user is prompted with a list of valid filenames for the specified project. This list shall contain the names and RCS numbers of all files which have been verified for this project.

New VerFF Button. Upon selection of the *New VerFF* button, the *Verification Fault Form* is displayed. This functionality is useful since a *Verification Fault Form* must be completed for each incorrect engineering change which resulted from a testing failure.

New TestFF. Upon selection of the *New TestFF* button, all records are cleared from the *Testing Failure Form.* The next compilation fault can then be recorded

Prev Button. Upon selection of the *Prev* button, the previous record is displayed in the form.

Next Button. Upon selection of the *Next* button, the next record is displayed in the form.

6.4.7 Incremental Development Plan Form

The *Incremental Development Plan Form* (Figure 6.36) is used to record each version of the increment plan for a given increment of a project. Through this for, records in the INC_PLAN and INC_PLAN_FUNC tables may be inserted, deleted, or updated. Two buttons appear on this form: *OK* and *Cancel.*

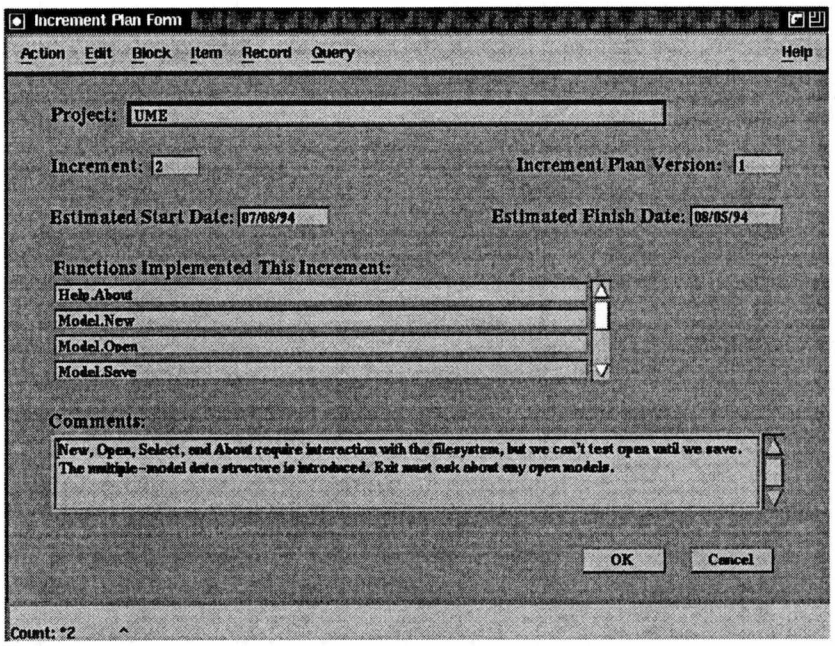

Figure 6.36: Incremental Development Plan Form

Menu. The Oracle Forms default menu is displayed on this form.

Project Selection. Upon entering the *Project* field, the user is prompted with a list of all valid project names which have been identified to the database through the *New Project Form* (Figure 6.24).

Increment Selection. Upon entering the *Increment* field, the user is prompted with a list of all valid increments for the specified project name. These increments would have been identified to the database through the *New Increment Form* (Figure 6.25).

Version Selection. The *Increment Plan Version* defaults to the highest version in the INC_PLAN table.

Date. The *Estimated Start Date* and *Estimated Finish Date* default to the current date.

6.4.8 Customer Interaction Meeting Form

The *Customer Interaction Meeting Form* (Figure 6.37) could be completed for every customer interaction meeting held for a given increment of a project. Through this form, records in the CUST_INT, CUST_SQRL_ATTEND, CUSTOMER_ATTEND tables may be inserted, deleted, or updated. Two buttons appear on this form: *OK* and *Cancel.*

6.4.9 Process Control Meeting Form

The *Process Control Meeting Form* (Figure 6.38) could be completed for every customer interaction meeting held for a given increment of a project. Though this form, records in the CONTROL_CHECK and CONTROL_SQRL_ATTEND tables can be inserted, deleted, or updated. Two buttons appear on this form: *OK* and *Cancel.*

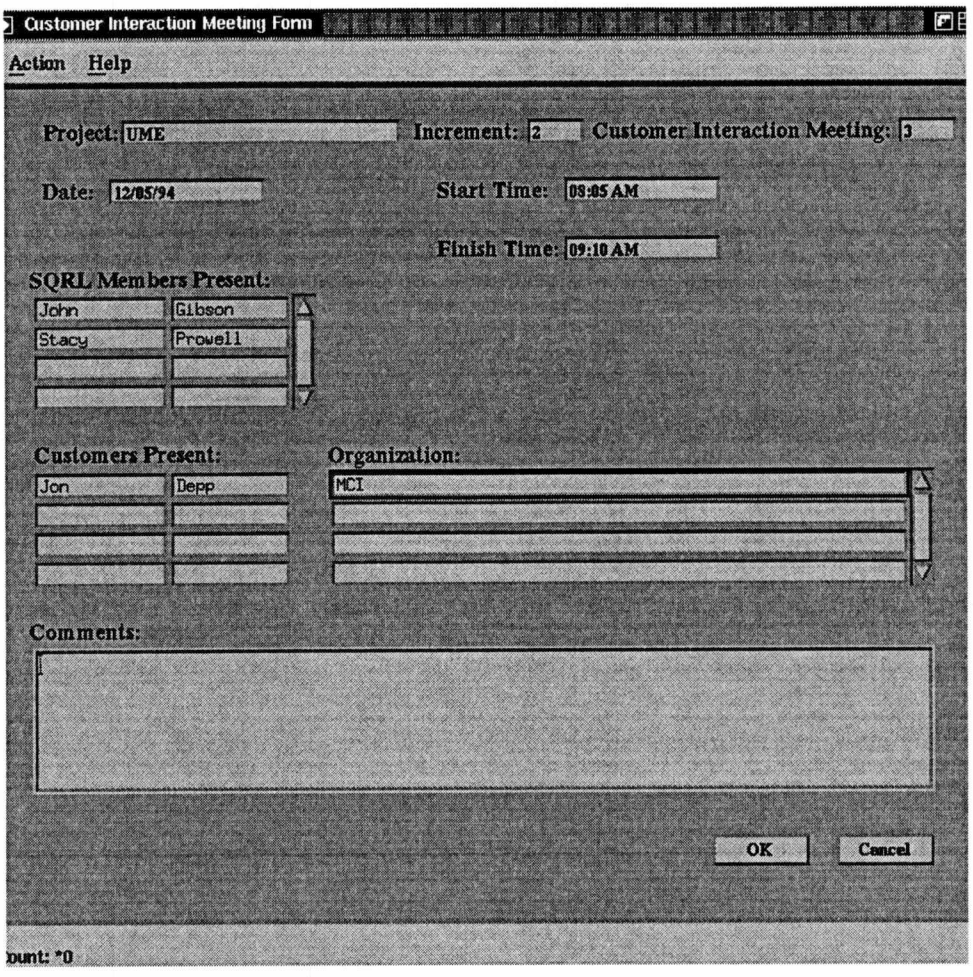

Figure 6.37: Customer Interaction Meeting Form

6.5 Reports Artifacts

A description is provided in this section of each of the artifact categories which are classified as *report* type. Parameter forms were developed for each of these artifact categories, however, only the *Increment Development Parameter Form* is included to illustrate the parameter form format.

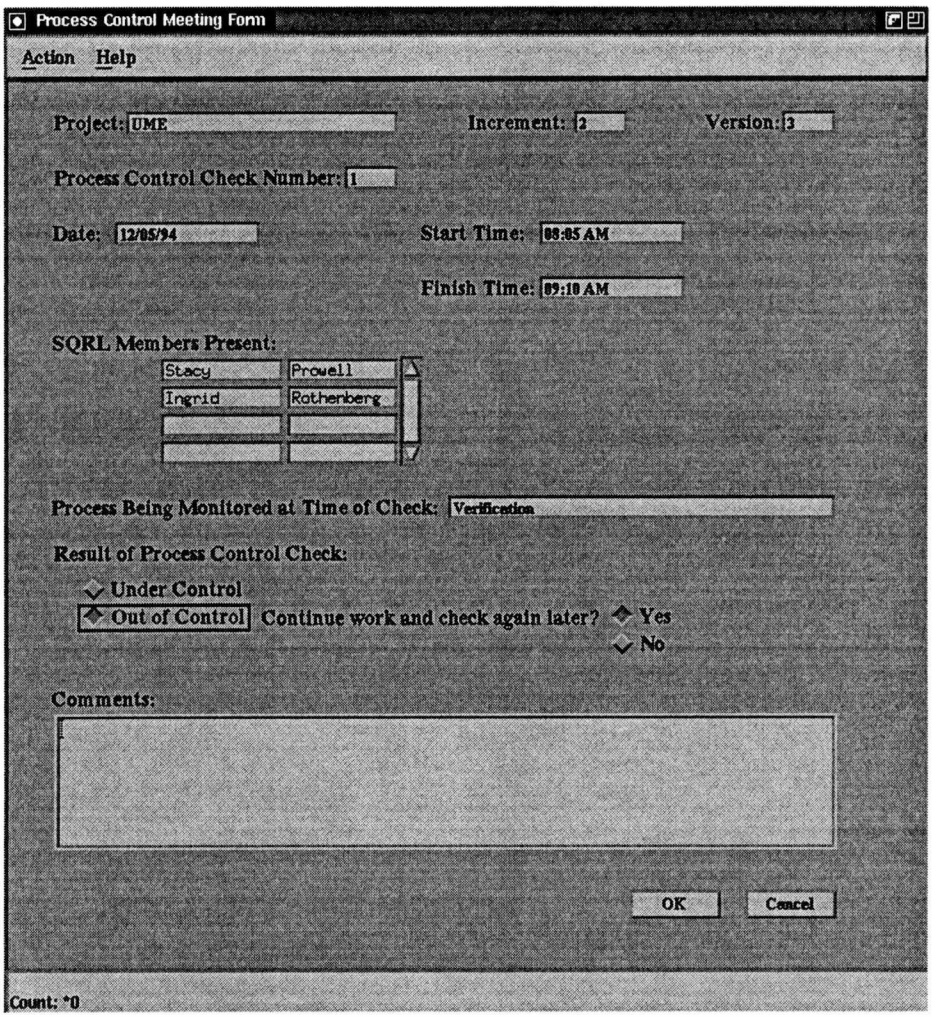

Figure 6.38: Process Control Meeting Form

6.5.1 Fault and Failure Distribution

The *Fault and Failure Distribution* (Figure 6.39) is generated from information contained in the VERIFICATION, COMPILATION, and TESTING tables. This report provides a breakdown of the number and percentage of faults and failures found in verification compilation, and testing respectively for each increment of the project specified in the *Fault and Failure Distribution Parameter Form.*

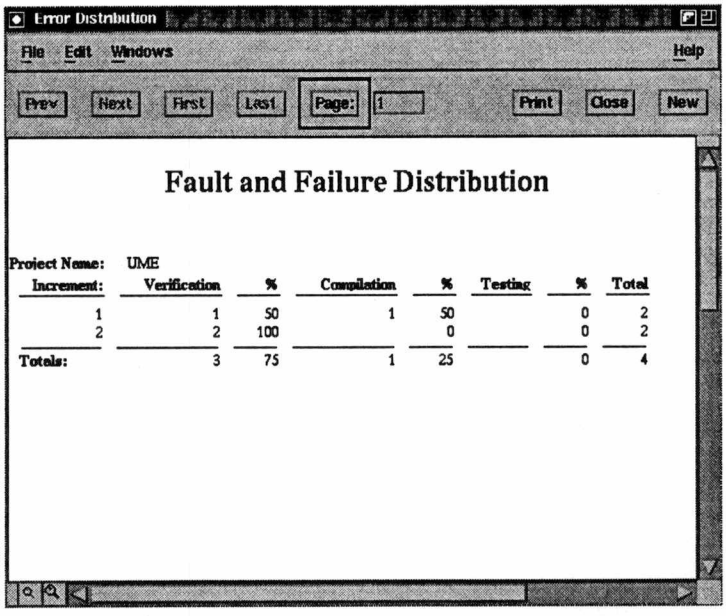

Figure 6.39: Fault and Failure Distribution

6.5.2 Project Summary

The *Project Summary* (Figure 6.40) is generated from information in the PROJECT and INCREMENT tables, and from the SQRL_TASKS view. This report provides a summary for each increment of the project specified in the *Project Summary Parameter Form.*

6.5.3 Verification Summary

The *Verification Summary* (Figure 6.41) is generated from information in the VERIFICATION table. It provides a summary of the verification faults recorded for the project and increment specified in the *Verification Summary Parameter Form.*

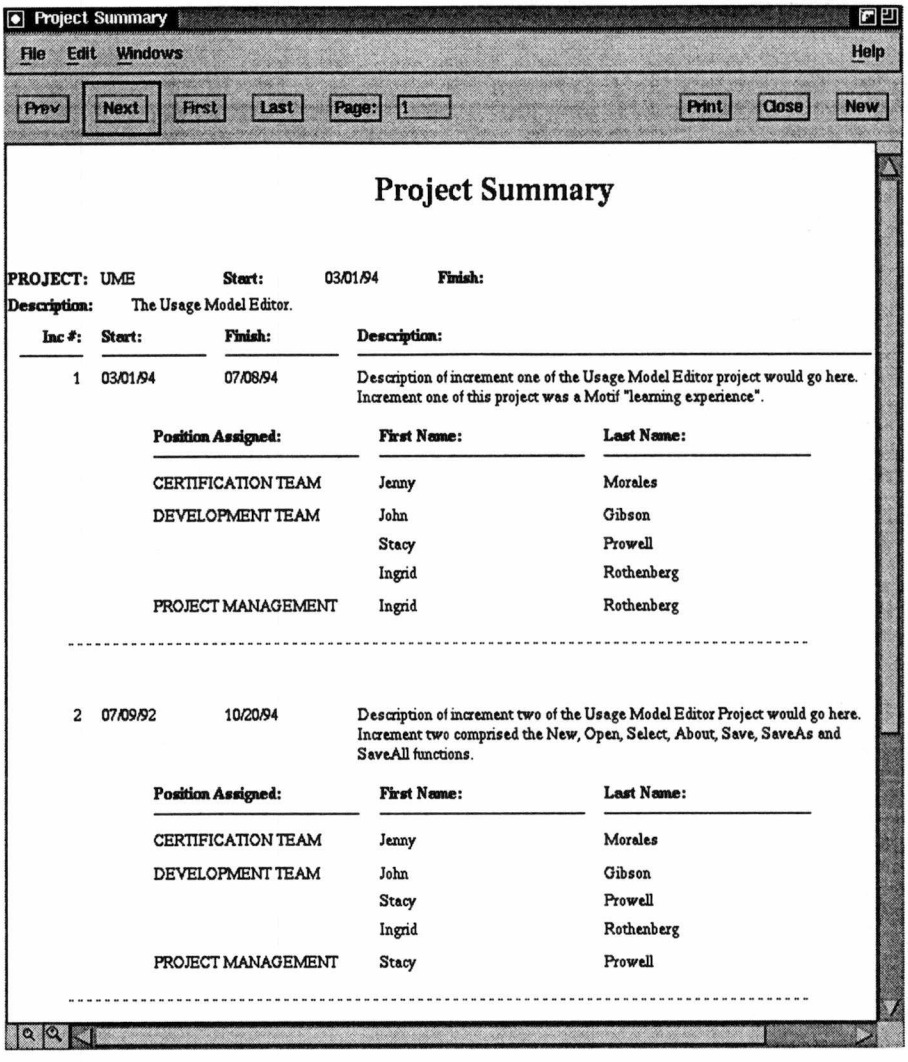

Figure 6.40: Project Summary

6.5.4 Compilation Summary

The *Compilation Summary* (Figure 6.42) is generated from information in the COMPILATION table. This report provides a summary of the compilation faults recorded for the project and increment specified in the *Compilation Summary Parameter Form.*

6.5.5 Testing Summary

The Testing Summary is generated from information in the TESTING table. The

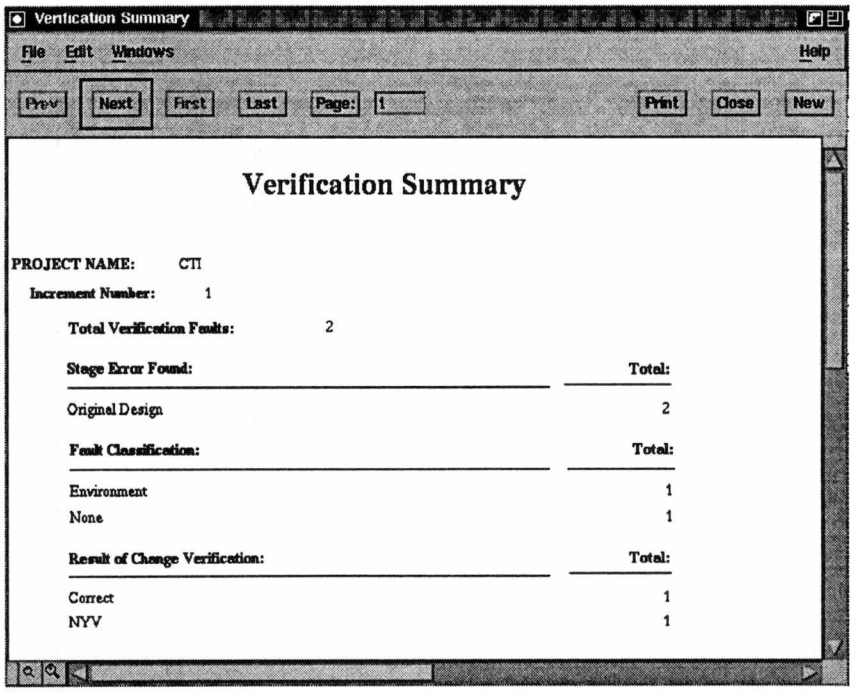

Figure 6.41: Verification Summary

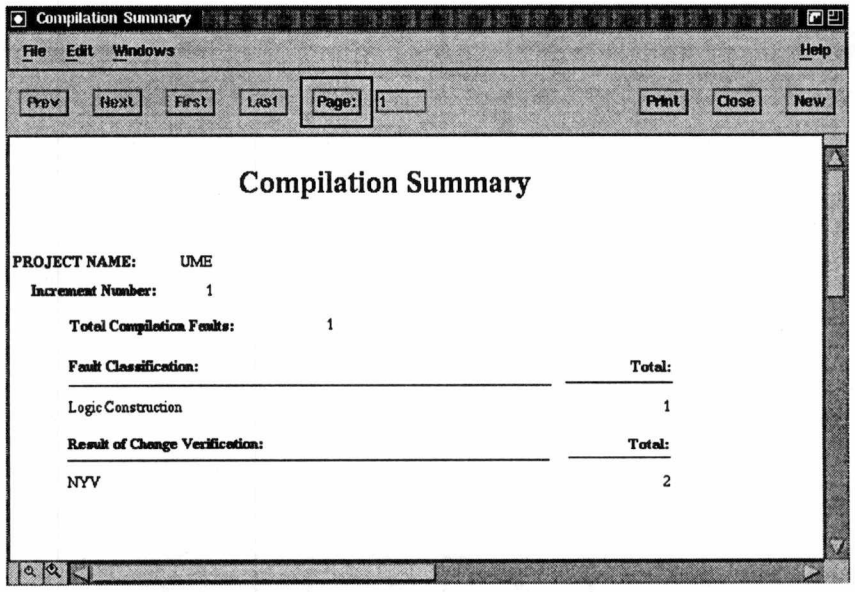

Figure 6.42: Compilation Summary

report provides a summary of the testing failures recorded for the project and increment specified in the Testing Summary Parameter Form.

6.5.6 Incremental Development Plan

The *Incremental Development Plan* is generated from the data input by the *Increment Plan Form*. It provides a summary of a specific version of the increment plan for a given project. The report parameter screen (Figure 6.43) prompts the user for report parameters. The *Project* parameter is validated against values of the ProjName field in the INC_PLAN table. The *Version* parameter is validated against values of the VerNum field in the INC_PLAN table for the specified project name. The version number shall default to the most recent version of the *Incremental Development Plan* should no version number be specified.

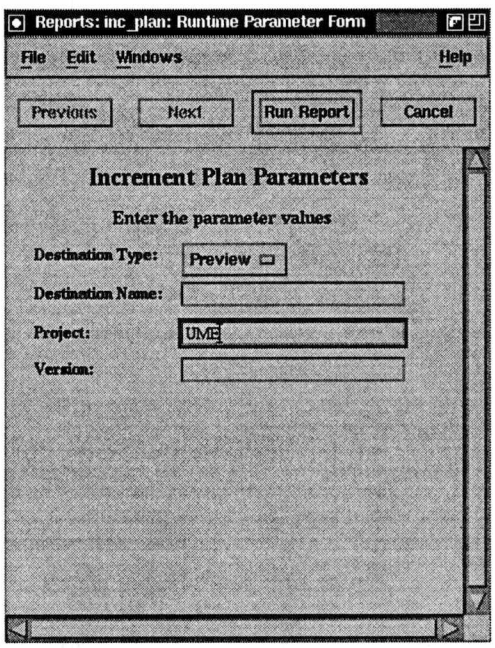

Figure 6.43: Incremental Development Plan Parameter Form

The report previewer (Figure 6.44) is displayed upon the selection of the *Run Report* button with the *Destination Type* parameter option set to "previewer".

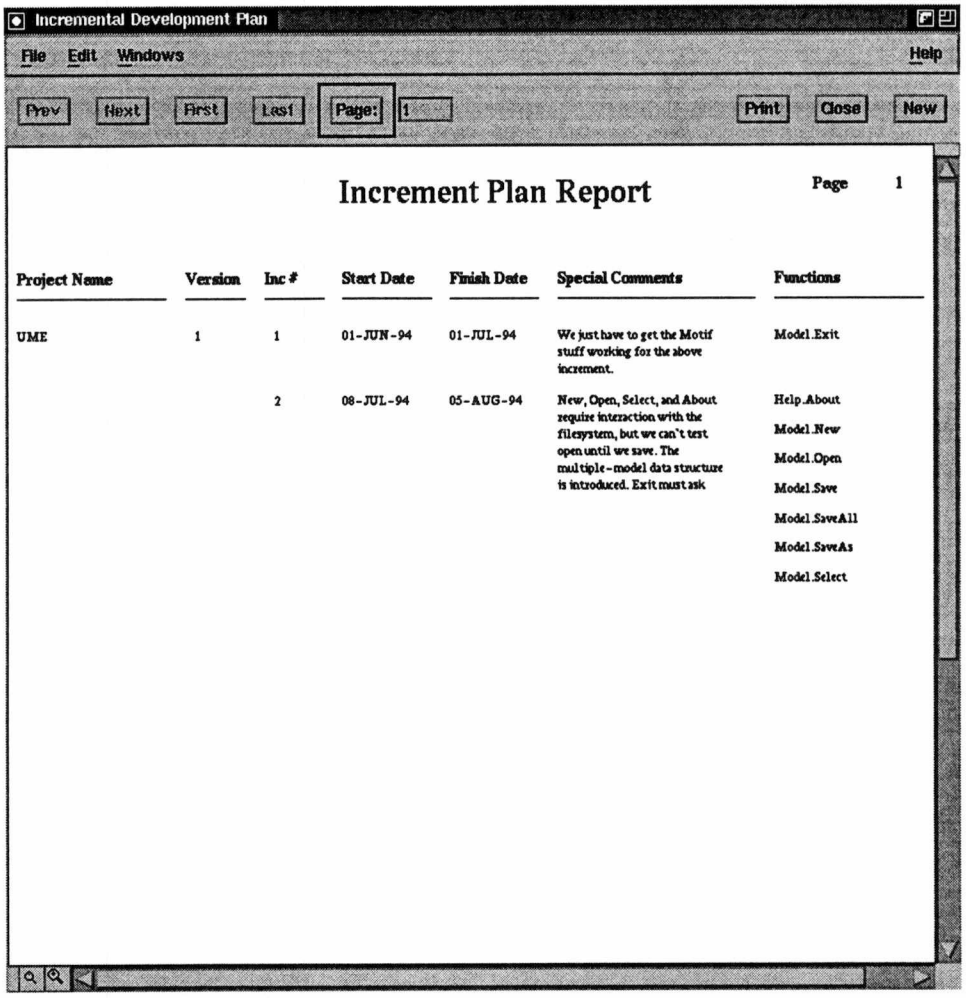

Figure 6.44: Incremental Development Plan

6.6 Shell Utilities

A shell script named *insert_rcs_info* has been developed to strip out information from the RCS log and insert the information into the database. It is intended that this script be run at the beginning of a verification session, after files have been checked-in with RCS. The script takes as arguments the project name to which the information shall relate and a list of RCS ,*v* filenames from which to strip information from the RCS log. This

script first runs a PERL script named *rlog_strip* which strips out and formats the RCS log information, and second runs the Oracle SQL*Load utility to load the information.

This *rlog_strip* script takes the list of RCS log files and strips out the filename and file description of each. Then, for each version of the RCS file checked in, the RCS version number, author, date checked in and description of changes since the last revision are stripped from the log. This information is written to a file, the format of which is specified in a control file. The control file is interpreted by the Oracle utility SQL*Load and loaded into the RCS_SUMMARY and RCS_VER_INFO tables.

6.7 Usage Model for the CPD

Table 6.1 represents a "state-of-use" usage model for the CPD with stimuli listed on the arcs. A list of all possible to-states are listed for each from-state with the stimuli responsible for the state transition represented by the arc. The usage model appears in graphical form in Appendix B.

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
1. INVOCATION	2. Logon Screen	<Signal:CPD>

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
2. Logon Screen	3. Logon Alert	<Select:CPD.LS.Connect, *Inv Env*>
	4. Main Window	<Select:CPD.LS.Connect, *Val Env*>
	5. Browse Processes [Without Structure]	
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	TERMINATION	<Select:CPD.LS.Connect, *Inv Env(3)*>
	TERMINATION	<Select:CPD.LS.Cancel, *CallWin*>
	4. Main Window	
	5. Browse Processes [Without Structure]	
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
3. Logon Alert	2. Logon Screen	<Select:CPD.LA.OK>
4. Main Window	2. Logon Screen	<Select:CPD.MW.Action.Connect>
	8. Projects...	<Select:CPD.MW.Action.Projects...>
	10. Increments...	<Select:CPD.MW.Action.Increments...>
	TERMINATION	<Select:CPD.MW.Action.Exit>
	12. About	<Select:CPD.MW.Help.About>
	7. Browse Artifacts	<Select:CPD.MW.Artifacts>
	5. Browse Processes	<Select:CPD.MW.Processes>

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
5. Browse Processes [Without Structure]	2. Logon Screen	<Select:CPD.BP.Action.Connect>
	8. Projects...	<Select:CPD.BP.Action.Projects...>
	10. Increments...	<Select:CPD.BP.Action.Increments...>
	13. Process Definitions	<Select:CPD.BP.Action.Process Definitions>
	14. Artifact Categories	<Select:CPD.BP.Action.Artifact Categories>
	15. File Types	<Select:CPD.BP.Action.File Types>
	TERMINATION	<Select:CPD.BP.Action.Exit>
	12. About	<Select:CPD.BP.Help.About>
	6. Browse Processes [With Structure]	<Select:CPD.BP.Show Structure>
	16. File Summary	<Select:CPD.BP.Select, *Artifact Type*>
	18. a) Report Parameter Form	
	19. Run Form	
	17. a) Class List	
	4. Main Window	<Select:CPD.BP.Cancel>
	5. Browse Processes [Without Structure]	<Select:CPD.BP.ProcList>
	5. Browse Processes [Without Structure]	<Select:CPD.BP.ArtList>
	5. Browse Processes [Without Structure]	<Select:CPD.BP.ArrowRt>
	5. Browse Processes [Without Structure]	<Select:CPD.BP.ArrowLt>

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
6. Browse Processes [With Structure]	2. Logon Screen	<Select:CPD.BP.Action.Connect>
	8. Projects...	<Select:CPD.BP.Action.Projects...>
	10. Increments...	<Select:CPD.BP.Action.Increments...>
	13. Process Definitions	<Select:CPD.BP.Action.Process Definitions>
	14. Artifact Categories	<Select:CPD.BP.Action.Artifact Categories>
	15. File Types	<Select:CPD.BP.Action.File Types>
	TERMINATION	<Select:CPD.BP.Action.Exit>
	12. About	<Select:CPD.BP.Help.About>
	5. Browse Processes [Without Structure]	<Select:CPD.SSCH.Hide Structure>
	16. File Summary	<Select:CPD.BP.Select, *Artifact Type*>
	18. a) Report Parameter Form	
	19. Run Form	
	17. a) Class List	
	4. Main Window	<Select:CPD.BP.Cancel>
	6. Browse Processes [With Structure]	<Select:CPD.BP.ProcList>
	6. Browse Processes [With Structure]	<Select:CPD.BP.ArtList>
	6. Browse Processes [With Structure]	<Select:CPD.BP.ArrowRt>
	6. Browse Processes [With Structure]	<Select:CPD.BP.ArrowLt>

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
7. Browse Artifacts	2. Logon Screen	<Select:CPD.BA.Action.Connect>
	8. Projects...	<Select:CPD.BA.Action.Projects...>
	10. Increments...	<Select:CPD.BA.Action.Increments...>
	14. Artifact Categories	<Select:CPD.BA.Action.Artifact Categories>
	15. File Types	<Select:CPD.BA.Action.File Types>
	TERMINATION	<Select:CPD.BA.Action.Exit>
	12. About	<Select:CPD.BA.Help.About>
	7. Browse Artifacts	<Select:CPD.BA.Query>
	16. File Summary	<Select:CPD.BA.Select, *ArtType*>
	18. a) Report Parameter Form	
	19. Run Form	
	17. a) Class List	
	4. Main Window	<Select:CPD.BA.Cancel>
	7. Browse Artifacts	<Select:CPD.BA.Scope>
	7. Browse Artifacts	<Select:CPD.BA.Type>
	7. Browse Artifacts	<Select:CPD.BA.ArtList>

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
8. Projects...	4. Main Window	<Select:CPD.PRJ.Select, *CallWin*>
	5. Browse Processes [Without Structure]	
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	9. Modify Project	<Select:CPD.PRJ.Modify>
	4. Main Window	<Select:CPD.PRJ.Cancel, *CallWin*>
	5. Browse Processes [Without Structure]	
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	9. a) New Project	<Select:CPD.PRJ.New>
	8. Projects	<Select:CPD.PRJ.ProjectList>
9. Modify Project	8. Projects...	<Select:CPD.MODP.OK>
	8. Projects	<Select:CPD.MODP.Cancel>
9. a) New Project	8. Projects...	<Select:CPD.NEWP.OK>
	8. Projects	<Select:CPD.NEWP.Cancel>

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
10. Increments...	4. Main Window	<Select:CPD.INC.Select, *CallWin*>
	5. Browse Processes [Without Structure]	
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	11. Modify Increment	<Select:CPD.INC.Modify>
	4. Main Window	<Select:CPD.INC.Cancel, *CallWin*>
	5. Browse Processes [Without Structure]	
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	10. Increments...	<Select:CPD.INC.IncList>
	11. a) New Increment	<Select:CPD.INC.New>
11. Modify Increment	10. Increments...	<Select:CPD.MODI.OK>
	10. Increments...	<Select:CPD.MODI.Cancel>
11. a) New Increment	10. Increments...	<Select:CPD.NEWI.OK>
	10. Increments...	<Select:CPD.NEWI.Cancel>
12. About	4. Main Window	<Open:CPD.ABT, *CallWin*>
	5. Browse Processes [Without Structure]	
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
13. Process Definitions	5. Browse Processes [Without Structure]	<Select:CPD.PROC.OK>
	6. Browse Processes [With Structure]	
	5. Browse Processes [Without Structure]	<Select:CPD.PROC.Cancel>
	6. Browse Processes [With Structure]	
	17. b) Process List	<Select:CPD.PROC.Process>
	17. b) Process List	<Select:CPD.PROC.ParProcess>
14. Artifact Categories	5. Browse Processes [Without Structure]	<Select:CPD.ARTCAT.OK, *CallWin*>
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	5. Browse Processes [Without Structure]	<Select:CPD.ARTCAT.Cancel, *CallWin*>
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	14. Artifact Categories	<Select:CPD.ARTCAT.Scope>
	14. Artifact Categories	<Select:CPD.ARTCAT.Type>
	17. c) Project List	<Select:CPD.ARTCAT.ProjList>
	17. e) Increment List	<Select:CPD.ARTCAT.IncList>
	17. b) Process List	<Select:CPD.ARTCAT.ProcList>
	17. a) Class List	<Select:CPD.ARTCAT.ClassList>
	17. d) In/Out List	<Select:CPD.ARTCAT.InOutList>

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
15. File Types	5. Browse Processes [Without Structure]	<Select:CPD.FT.OK, *CallWin*>
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	5. Browse Processes [Without Structure]	<Select:CPD.FT.Cancel, *CallWin*>
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
16. File Summary	5. Browse Processes [Without Structure]	<Select:CPD.FSUM.Action.Cancel, *CallWin*>
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	16. File Summary	<Select:CPD.FSUM.Delete>
	21. Update File	<Select:CPD.FSUM.Update>
	20. New File	<Select:CPD.FSUM.New>
	23. Extract To	<Select:CPD.FSUM.Extract>
	16. File Summary	<Select:CPD.FSUM.Query>
	16. File Summary	<Select:CPD.FSUM.FileList>
	17. c) Project List	<Select:CPD.FSUM.ProjName>
	17. e) Increment List	<Select:CPD.FSUM.IncNum>
	17. f) File Type List	<Select:CPD.FSUM.FileType>

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
17. a) Class List	17. a) Class List	<Select:CPD.CLW.Find>
	17. a) Class List	<Select:CPD.CLW.OK, *CallWin*>
	5. Browse Processes [Without Structure]	
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	14. Artifact Categories	
	5. Browse Processes [Without Structure]	<Select:CPD.CLW.Cancel, *CallWin*>
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	14. Artifact Categories	
17. b) Process List	17. b) Process List	<Select:CPD.PROCLW.Find>
	17. b) Process List	<Select:CPD.PROCLW.OK, *CallWin*>
	13. Process Definitions	
	14. Artifact Categories	
	5. Browse Processes [Without Structure]	<Select:CPD.PROCLW.Cancel, *CallWin*>
	13. Process Definitions	
	14. Artifact Categories	

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
17. c) Project List	17. c) Project List	<Select:CPD.PROJLW.Find>
	17. c) Project List	<Select:CPD.PROJLW.OK, *CallWin*>
	14. Artifact Categories	
	16. File Summary	
	20. New File	
	21. Update File	
	5. Browse Processes [Without Structure]	<Select:CPD.PROJLW.Cancel, *CallWin*>
	14. Artifact Categories	
	16. File Summary	
	20. New File	
	21. Update File	
17. d) In/Out List	17. d) In/Out List	<Select:CPD.IOLW.Find>
	17. d) In/Out List	<Select:CPD.IOLW.OK, *CallWin*>
	14. Artifact Categories	
	5. Browse Processes [Without Structure]	<Select:CPD.IOLW.Cancel, *CallWin*>
	14. Artifact Categories	

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
17. e) Increment List	17. e) Increment List	<Select:CPD.INCLW.Find>
	17. e) Increment List	<Select:CPD.INCLW.OK, *CallWin*>
	14. Artifact Categories	
	16. File Summary	
	20. New File	
	21. Update File	
	5. Browse Processes [Without Structure]	<Select:CPD.INCLW.Cancel, *CallWin*>
	14. Artifact Categories	
	16. File Summary	
	20. New File	
	21. Update File	
17. f) File Type List	17. f) File Type List	<Select:CPD.FTLW.Find>
	17. f) File Type List	<Select:CPD.FTLW.OK, *CallWin*>
	16. File Summary	
	20. New File	
	21. Update File	
	5. Browse Processes [Without Structure]	<Select:CPD.FTLW.Cancel, *CallWin*>
	16. File Summary	
	20. New File	
	21. Update File	

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
18. a) Report Parameter Form	5. Browse Processes [Without Structure]	<Select:CPD.RPT.PARAM.File.Exit, *CallWin*>
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	5. Browse Processes [Without Structure]	<Select:CPD.RPT.PARAM.File.Cancel, *CallWin*>
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
	18. b) Report Previewer	<Select:CPD.PARAM.Run>
18. b). Report Previewer	5. Browse Processes [Without Structure]	<Select:CPD.RPT.PRV.File.Exit, *CallWin*>
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
19. Run Form	5. Browse Processes [Without Structure]	<Select:CPD.FRM.Action.Exit, *CallWin*>
	6. Browse Processes [With Structure]	
	7. Browse Artifacts	
20. New File	24. Insert From	<Select:CPD.NF.OK>
	16. File Summary	<Select:CPD.NF.Cancel>
	17. c) Project List	<Select:CPD.NF.ProjName>
	17. e) Increment List	<Select:CPD.NF.IncNum>
	17. f) File Type List	<Select:CPD.NF.FileType>

Table 6.1: List of Usage Model States and Arcs

From State	To State	Arc
21. Update File	16. File Summary	<Select:CPD.UF.OK>
	24. Insert From	
	16. File Summary	<Select:CPD.UF.Cancel>
	17. c) Project List	<Select:CPD.UF.ProjName>
	17. e) Increment List	<Select:CPD.UF.IncNum>
	17. f) File Type List	<Select:CPD.UF.FileType>
	21. Update File	<Select:CPD.UF.Yes>
	21. Update File	<Select:CPD.UF.No>
23. Extract To	16. File Summary	<Select:CPD.ETL.OK>
	16.File Summary	<Select:CPD.ETL.Cancel>
24. Insert From	16. File Summary	<Select:CPD.IFL.OK>
	16. File Summary	<Select:CPD.IFL.Cancel>

6.8 Mapping of Forms and Windows to Database Tables

Table 6.2 provides a mapping from Cleanroom artifact categories of *form* type to the corresponding tables of the CPD.

Table 6.2: Mapping Forms and Windows to Tables

Form	Tables / Views	Description
Browse Processes	PROCESS	Query-Only
	PROC_ART_TYPE	Query-Only
Browse Artifacts	BROWSE_ART	Query-Only
Projects	PROJECT	Query-only
Modify Project	PROJECT	Update
New Project	PROJECT	Query, Insert, Delete, Update
Increments	INCREMENT	Query-only
Modify Increment	INCREMENT	Update

Table 6.2: Mapping Forms and Windows to Tables

Form	Tables / Views	Description
New Increment	INCREMENT	Query, Insert, Delete, Update
Process Definitions	PROCESS	Query, Insert, Delete, Update
Artifact Categories	ARTIFACT_CATEGORY	Query, Insert, Delete, Update
	PROJ_SPEC_ART_CAT	Query, Insert, Delete, Update
	INC_SPEC_ART_CAT	Query, Insert, Delete, Update
	ARTIFACT_CLASS	Query, Insert, Delete, Update
	PROCESS_ARTIFACT	Query, Insert, Delete, Update
	PROJECT	Query-Only
	INCREMENT	Query-Only
	PROCESS	Query-Only
	ARTIFACT_CATEGORY	Query-Only
File Types	ART_TYPE	Query, Insert, Delete, Update
File Summary	BROWSE_ART_INST	Query-Only
New File	ARTIFACT_FILE_INSTANCE	Query, Insert, Delete, Update
	PROJ_ART_FILE_INST	Query, Insert, Delete, Update
	INC_ART_FILE_INST	Query, Insert, Delete, Update
Update File	ARTIFACT_FILE_INSTANCE	Query, Insert, Delete, Update
	PROJ_ART_FILE_INST	Query, Insert, Delete, Update
	INC_ART_FILE_INST	Query, Insert, Delete, Update
Tasking	TASKING	Query, Insert, Delete, Update
	TASKING_SQRL	Query, Insert, Delete, Update
	TASKING_PROJ	Query, Insert, Delete, Update
	TASKING_INC	Query, Insert, Delete, Update
	PROJECT	Query-Only
	INCREMENT	Query-Only

Table 6.2: Mapping Forms and Windows to Tables

Form	Tables / Views	Description
Verification Meeting	VER_MEET	Query, Insert, Delete, Update
	VER_MEET_VERIFIERS	Query, Insert, Delete, Update
	VER_MEET_FILENAMES	Query, Insert, Delete, Update
	PROJECT	Query-Only
	INCREMENT	Query-Only
	COMP_MEET	Query-Only
	SQRL_TASKS	Query-Only
	RCS_VER_INFO	Query-Only
	RCS_SUMMARY	Query-Only
Verification Fault	VERIFICATION	Query, Insert, Delete, Update
	VERF_FILENAMES	Query, Insert, Delete, Update
	VERF_FUNCTIONS	Query, Insert, Delete, Update
	VER_MEET_FILENAMES	Query, Insert, Delete, Update
	PROJECT	Query-Only
	INCREMENT	Query-Only
	COMP_MEET	Query-Only
Compilation Session	COMP_MEET	Query, Insert, Delete, Update
	SQRL_TASKS	Query-Only
	PROJECT	Query-Only
	INCREMENT	Query-Only
Compilation Fault	COMPILATION	Query, Insert, Delete, Update
	ENGINEERING_CHANGES	Query, Insert, Delete, Update
	ENG_CH_FILENAMES	Query, Insert, Delete, Update
	ENG_CH_FUNCTIONS	Query, Insert, Delete, Update
	VER_MEET_FILENAMES	Query, Insert, Delete, Update
	PROJECT	Query-Only
	INCREMENT	Query-Only
	COMP_MEET	Query-Only

Table 6.2: Mapping Forms and Windows to Tables

Form	Tables / Views	Description
Testing Failure	TESTING	Query, Insert, Delete, Update
	ENGINEERING_CHANGES	Query, Insert, Delete, Update
	ENG_CH_FILENAMES	Query, Insert, Delete, Update
	ENG_CH_FUNCTIONS	Query, Insert, Delete, Update
	VER_MEET_FILENAMES	Query, Insert, Delete, Update
	PROJECT	Query-Only
	INCREMENT	Query-Only
	COMP_MEET	Query-Only
	SQRL_TASKS	Query-Only
Incremental Development Plan	INC_PLAN	Query, Insert, Delete, Update
	INC_PLAN_FUNC	Query, Insert, Delete, Update
	PROJECT	Query-Only
	INCREMENT	Query-Only
Customer Interaction	CUST_INT	Query, Insert, Delete, Update
	CUST_SQRL_ATTEND	Query, Insert, Delete, Update
	CUSTOMER_ATTEND	Query, Insert, Delete, Update
Process Control Meeting	CONTROL_CHECK	Query, Insert, Delete, Update
	CONTROL_SQRL_ATTEND	Query, Insert, Delete, Update

6.9 Mapping of Reports to Database Tables

Table 6.3 provides a mapping from the reports to the corresponding CPD tables.

Table 6.3: Mapping Reports to Tables

Report	Tables / Views
Fault and Failure Distribution	VERIFICATION
	COMPILATION
	TESTING

Table 6.3: Mapping Reports to Tables

Report	Tables / Views
Project Summary	PROJECT
	INCREMENT
	SQRL_TASKS
Verification Summary	VERIFICATION
Compilation Summary	COMPILATION
Testing Summary	TESTING
Increment Plan	INC_PLAN
	INC_PLAN_FUNC

Chapter 7

Future Work

This chapter contains suggestions for future enhancements to the CPD.

7.1 Mechanism to Show Process and Artifact Structure

The *Browse Processes Window* currently has a *Show Structure* button which displays the process schema as identified in the Process Description. This button could be renamed *Show Schema* and a new button, *Show Structure* added to the window. The *Show Structure* button would display the process hierarchy as displayed in the Process Description. The additional functionality of being able to select a process from this hierarchy and have that process become the currently selected process would be more convenient than the current method whereby the user must navigate step by step down the process tree in order to reach a leaf process.

The *Browse Artifacts Window* could have an extra feature added to display the artifact hierarchy in the same manner as the Show Structure button of the Browse Processes

Window. The artifacts have been organized in a hierarchical manner in the Process Description.

7.2 Dynamic Process Schema/Structure Generation

Additional information is required for high-level processes in order to mirror the process schema contained in the Process Description. For a given high-level process, the control flow among sub-processes must be specified. This may take the value of concurrent, sequential, alternative, or iterative. High-level schemas may also contain internal blocks representing control flow among sub-processes. These internal control flow blocks can also be maintained in the database. Once all of these process properties are specified in the database, this data may be used to automatically generate the process schema. These more complex process relationships are not considered in this implementation of the database. Allowing additional properties of processes to be maintained within the database would allow for the dynamic creation of the schema. This feature would eliminate the necessity of redeveloping the *Process Structure Window* each time the process is altered. This is an important extension to the CPD interface since Cleanroom is continually undergoing review and improvement.

7.3 Enhanced Process-Driven Interface

The interface to the CPD may be enhanced in the future to provide more specific guidance to the user in terms of being driven by the Cleanroom Process. The tool could recognize which stages of the Cleanroom Process for a given project had been completed and prompt the user with the next step of the process which is to follow according to the Process Description. Thus, upon selection of a process from the process list, the user could

be prompted with the appropriate artifact without having to specifically select an artifact from the list.

The tool could also be designed so as to keep track of the last process completed by a given team member and, upon recognizing that particular user logging on to the CPD, return that user to the state at which they left the tool.

7.4 Default Project and Increment Selection

The database could be updated so as to keep track of the last project on which a given user was working at the completion of a CPD session. The default value of the current project could then be automatically assigned to that project when the user initiates a session. The default current project for a first time user would be the project which has most recently begun, as calculated by the start date for its most recent increment.

Similarly, the database could be updated so as to keep track of the last increment of a project on which a given user was working. The default value of the current increment could then be automatically assigned to that increment of the project when the user initiates a session. The default current increment for a first time user would be the increment of the current project which has most recently begun, as calculated by the start date for its most recent increment.

7.5 Forms for Future Addition

The *Customer Interaction Meeting Form*, *Process Control Meeting Form*, *File Summary Window*, *New File Window*, *Update File Window*, *Extract File Browser*, *Insert File Browser*, and *Testing Summary* have yet to be implemented. The tables on which they will be based are listed in Chapter 4 and template forms are included in Chapter 6. Additional reports could easily be designed from the data currently maintained by the CPD.

7.6 Expanded Artifact Query Criteria

The criteria for querying artifact categories in the CPD are too restrictive. Either additional artifact category properties need to be included, or the definitions of the existing properties need to be expanded. The *report* artifact category classification currently includes only reports generated by Oracle's Report generation tool. The report classification could be expanded to include ascii reports generated by other tools.

7.7 Clustering CPD Tables

Should the performance of the CPD become problematic in the future, clustering the tables of the CPD should be considered as one option for improving the data access speed.

7.8 Restricting Access to Artifacts Based on Type of Use

In the current version of the CPD, the type of use of an artifact category at a given process step is displayed for information only. In the future, the interface could be expanded to restrict access to the artifacts based on type of use. Artifacts that are used as *input* to a Cleanroom process would only be used as reference. That is, the data contained in these artifacts may be queried, extracted to the filesystem (as in the case of any files stored in the database), output to the screen (such as in the case of forms) or printed (such as in the case of reports) but would not be permitted to be modified or created at that step. *Inout* artifacts could be used for reference or updated at a given process step. *Output* artifacts could be created at the given step.

7.9 Updating Tables Directly Through Other Tools

The CPD can eventually be directly updated through interfaces to the database by other applications. The option to save/retrieve files to/from the database or to the filesystem could be built into other tools.

A script could be built to capture and load the output of a code analyzer into the database in the same fashion as the RCS log file information has been captured. The PROJECT_TOTALS (Figure 4.35) and INCREMENT_TOTALS (Figure 4.36) tables could be loaded in this fashion.

7.10 Automated RCS Log Insertion

Relevant RCS information is currently stripped from the RCS log and inserted into the database through the *insert_rcs_info* script. This action is performed at the onset of a verification meeting. This functionality could be conveniently built into the *Verification Session Form.*

A new button labeled *Strip RCS Log* could be added to the *Verification Session Form.* Upon selection of this button, the user would be prompted with a directory browser though which the location of the files maintained by RCS for the selected project could be specified. The directory and project selection could be passed to the *rcs_info_script* which would proceed to load the information into the database.

BIBLIOGRAPHY

[1] Bobrowski, Steven M. (1994). Mastering Oracle7 & Client/Server Computing. Sybex, Inc.

[2] Crisp, Russell L. (1992). Database Design for the Cleanroom Software Engineering Environment.

[3] Dyer, Michael. (1993). The Cleanroom Approach to Software Development. John Wiley and Sons, Inc.

[4] Hausler, Philip A., Linger, Richard C., and Trammell, Carmen J. (1994, January). "Adopting Cleanroom Software Engineering with a Phased Approach." IBM Systems Journal.

[5] Koch, Geirge. (1993). Oracle7 The Complete Reference. McGraw-Hill, Inc.

[6] Oracle Corporation. ORACLE Documentation Set. Parts No. A10324-1, A10323-1, and A11854-1, 6694-70, 6693-70, 778-70, 800-20, 5411-70, 3602-70, 3605-70, 5142-31, A14000, 5452-15, A11986-2, A11987-2, A11988-2, A11989-2, A11990-2.

[7] Poore, J.H. (1990, August). "The Cleanroom Approach to Six Sigma Software." Technical Vitality Seminar, IBM Westlake Laboratory.

[8] Prowell, Stacy J. (1994). Specification for the Usage Model Editor.

[9] SQRL. (1994). Software Quality Research Laboratory, Cleanroom Process Description.

APPENDICES

Appendix A

Database Functions and User Privileges

Table A.1: CPD Functions by Control Flow and User

	Main			Process			Artifact		
	Admin	**SQRL**	**Joe**	**Admin**	**SQRL**	**Joe**	**Admin**	**SQRL**	**Joe**
Define/Modify Artifact Category				X			X		
View Artifact Category Definition				X	X	X	X	X	X
Define/Modify Artifact File Type				X			X		
View Artifact File Type Definition				X	X	X	X	X	X
Define/Modify Process Hierarchy				X					
View Process Hierarchy				X	X	X			
Define/Modify Project	X	X		X	X		X	X	
View Project Description / Select Project	X	X	X	X	X	X	X	X	X
Define/Modify Increment	X	X		X	X		X	X	
View Increment Description / Select Increment	X	X	X	X	X	X	X	X	X
Insert/Modify/Extract Artifact Instance				X	X		X	X	
Extract/View/Print Artifact Instance				X	X	X	X	X	X

Not available to any user.

Not available to this user.

X — Available to this user.

Appendix B

Usage Models

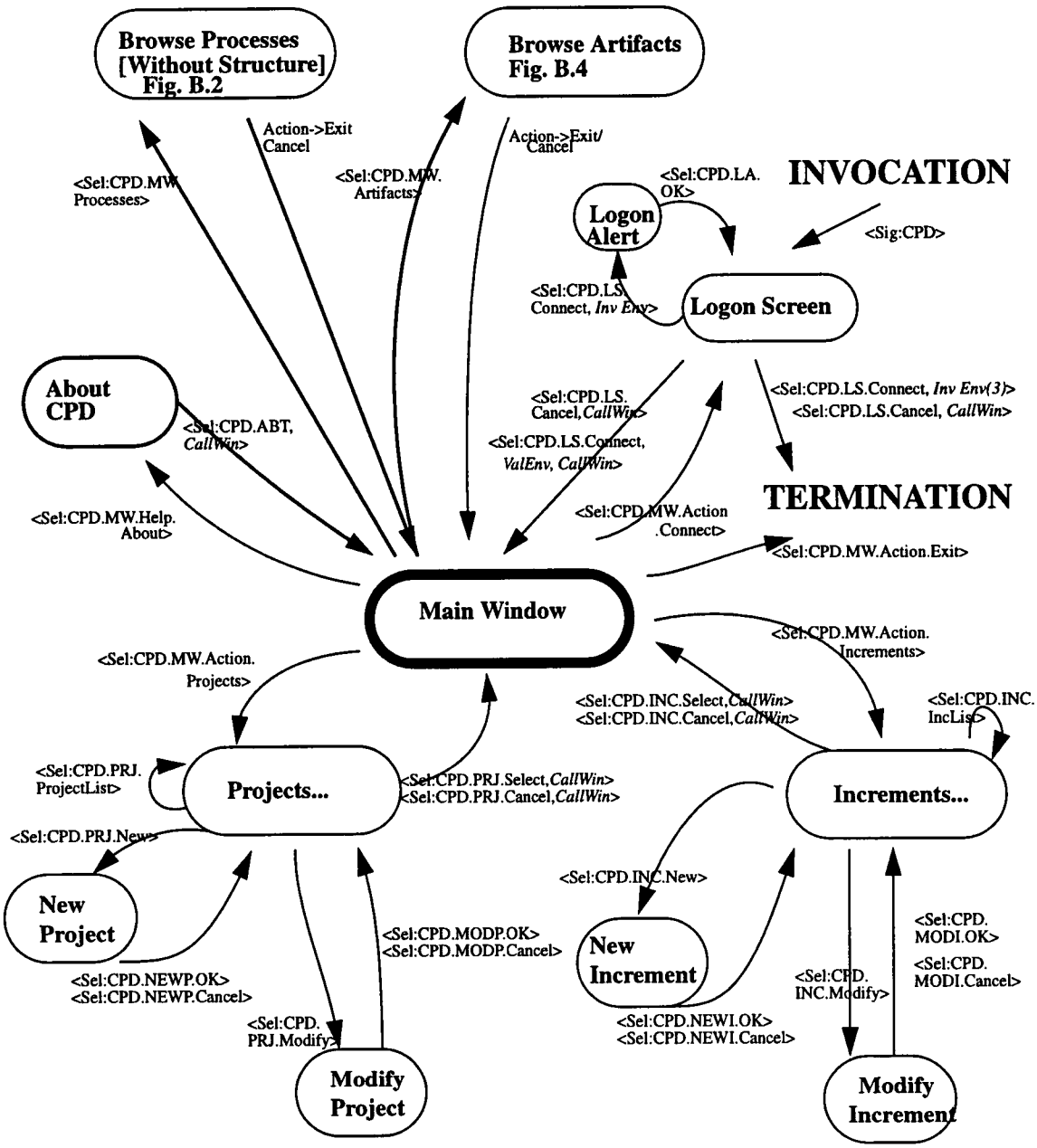

Figure B.1: Main Window Usage Model

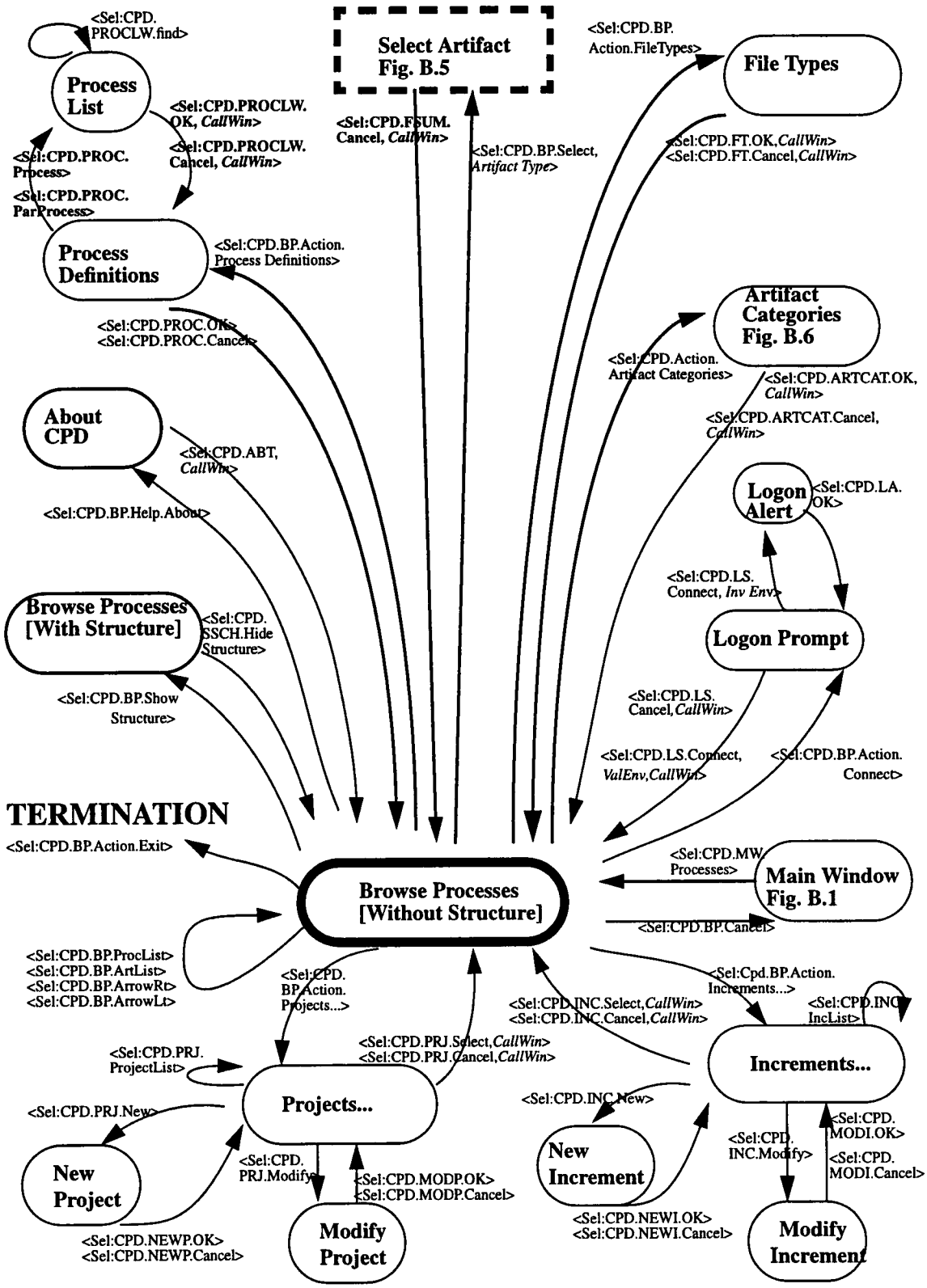

Figure B.2: Browse Processes Window [Without Structure] Usage Model

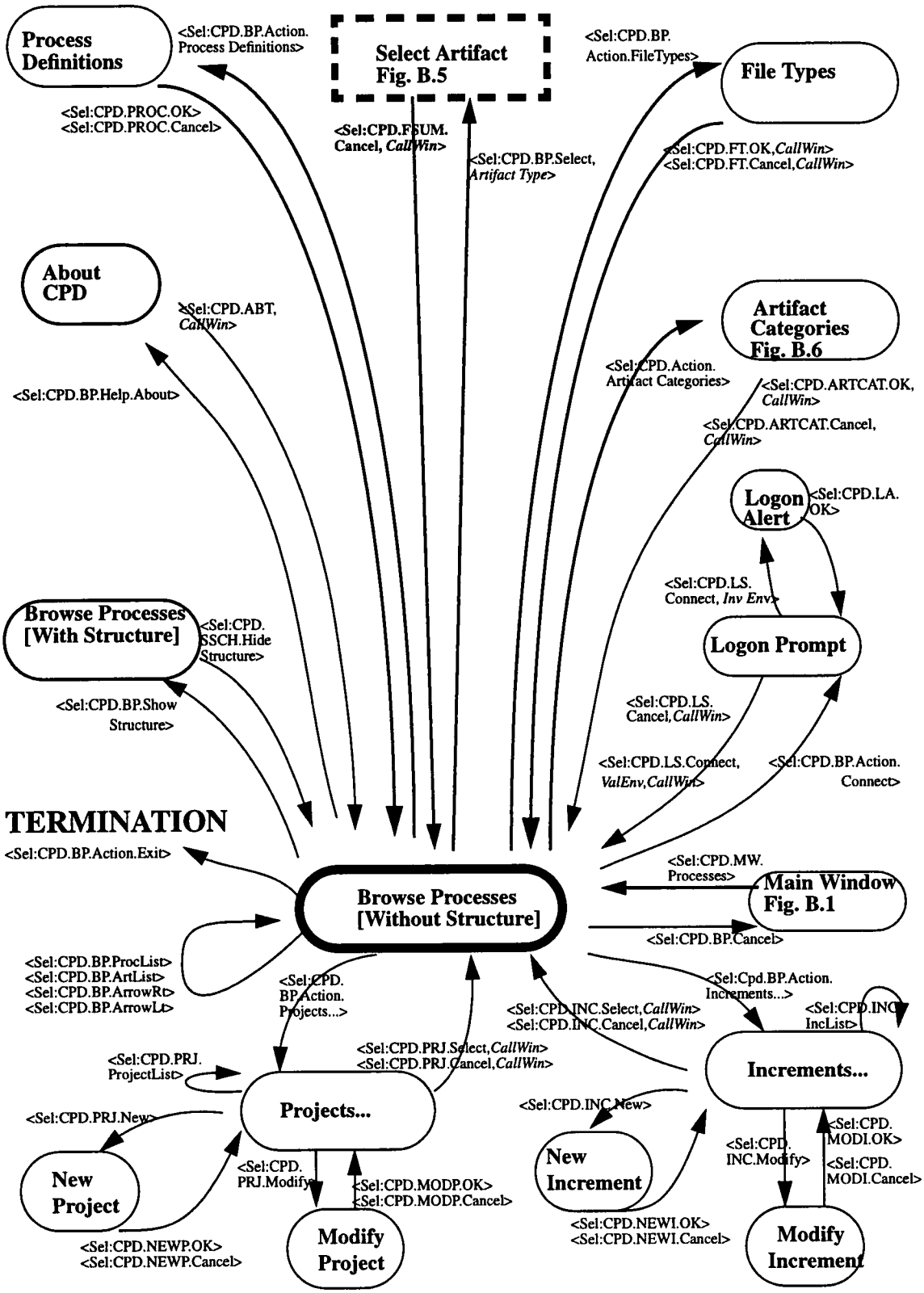

Figure B.3: Browse Processes Window [With Structure] Usage Model

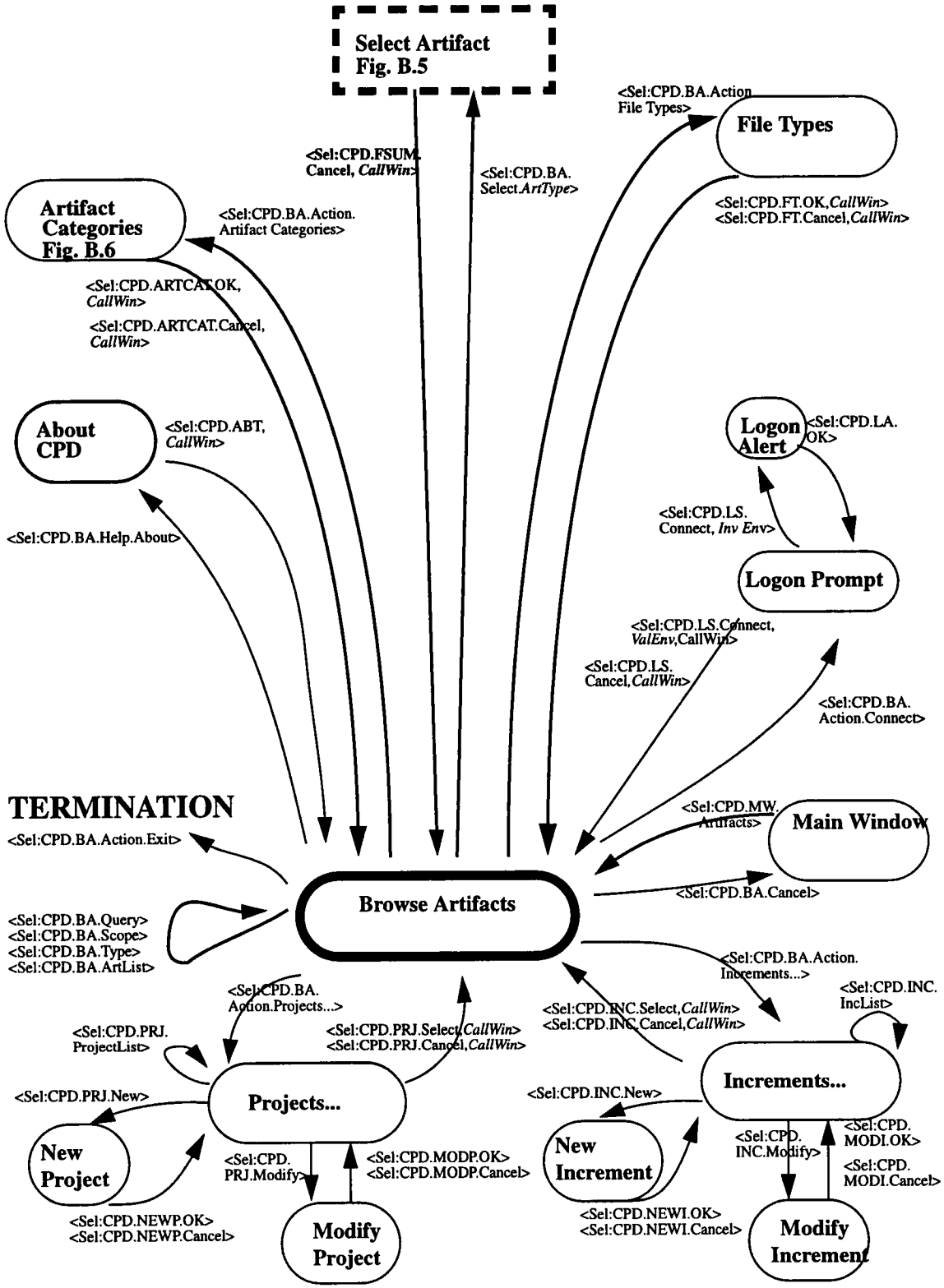

Figure B.4: Browse Artifacts Window Usage Model

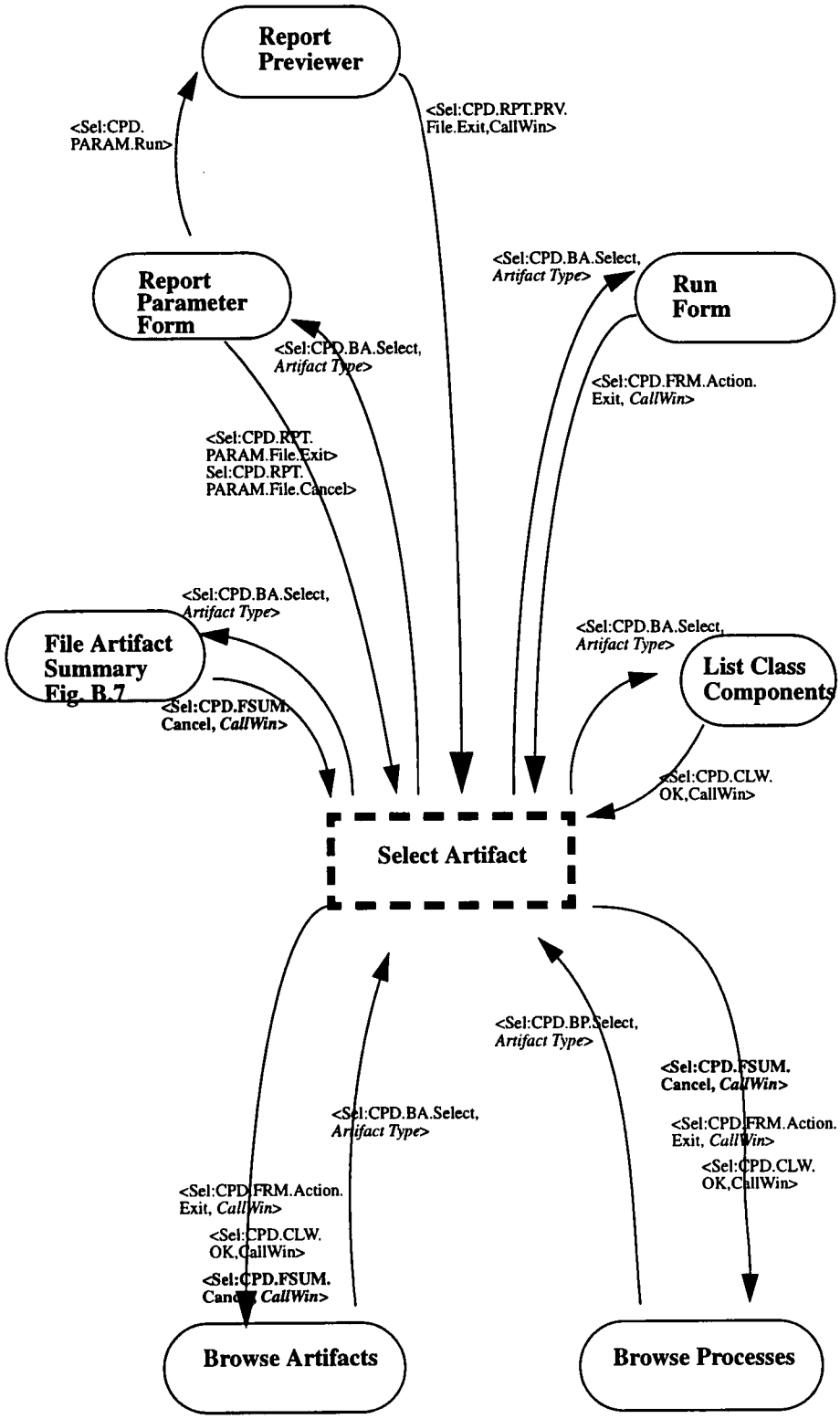

Figure B.5: Browse Artifacts/Browse Processes Artifact Selection Usage Model

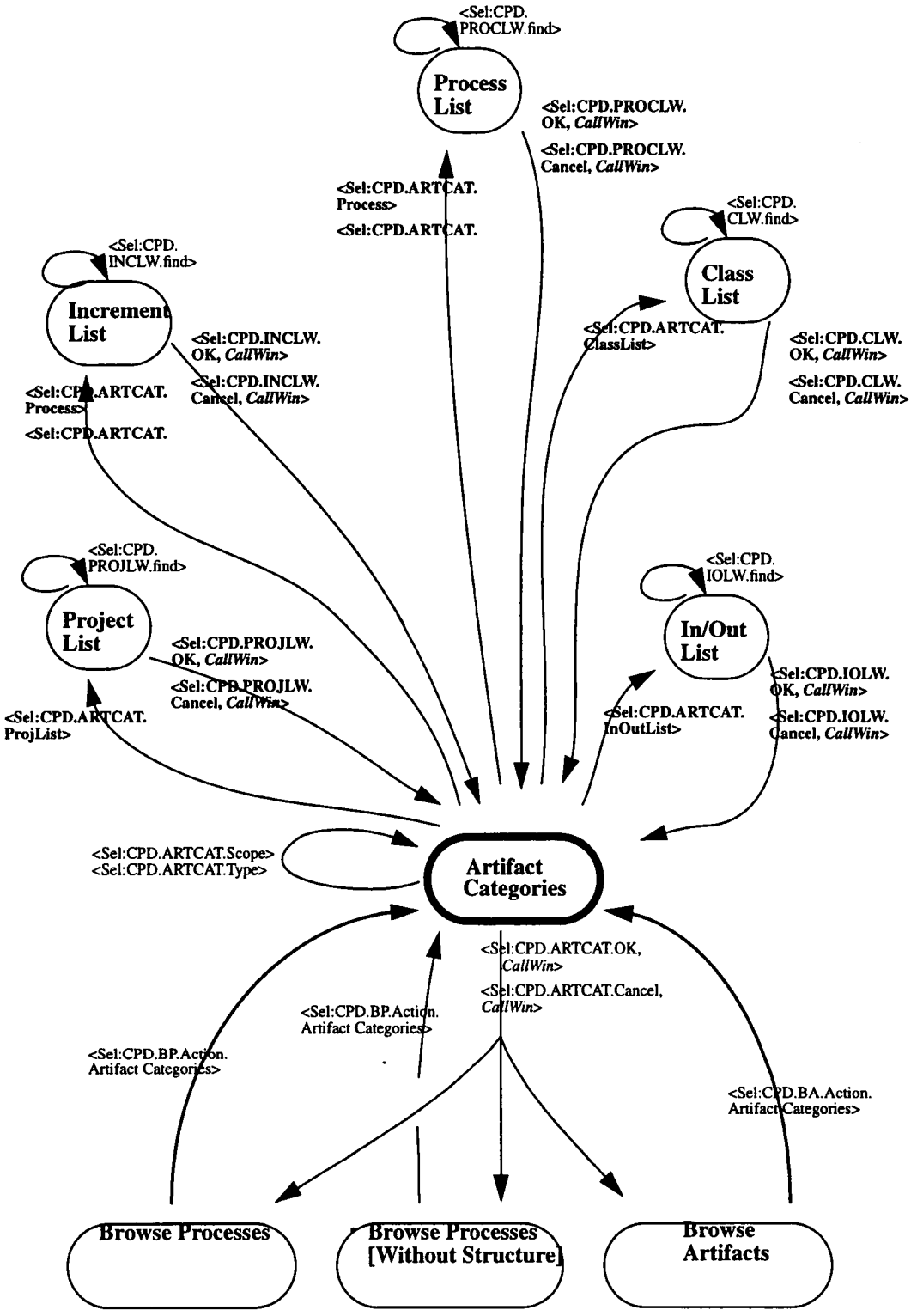

Figure B.6: Artifact Categories Usage Model

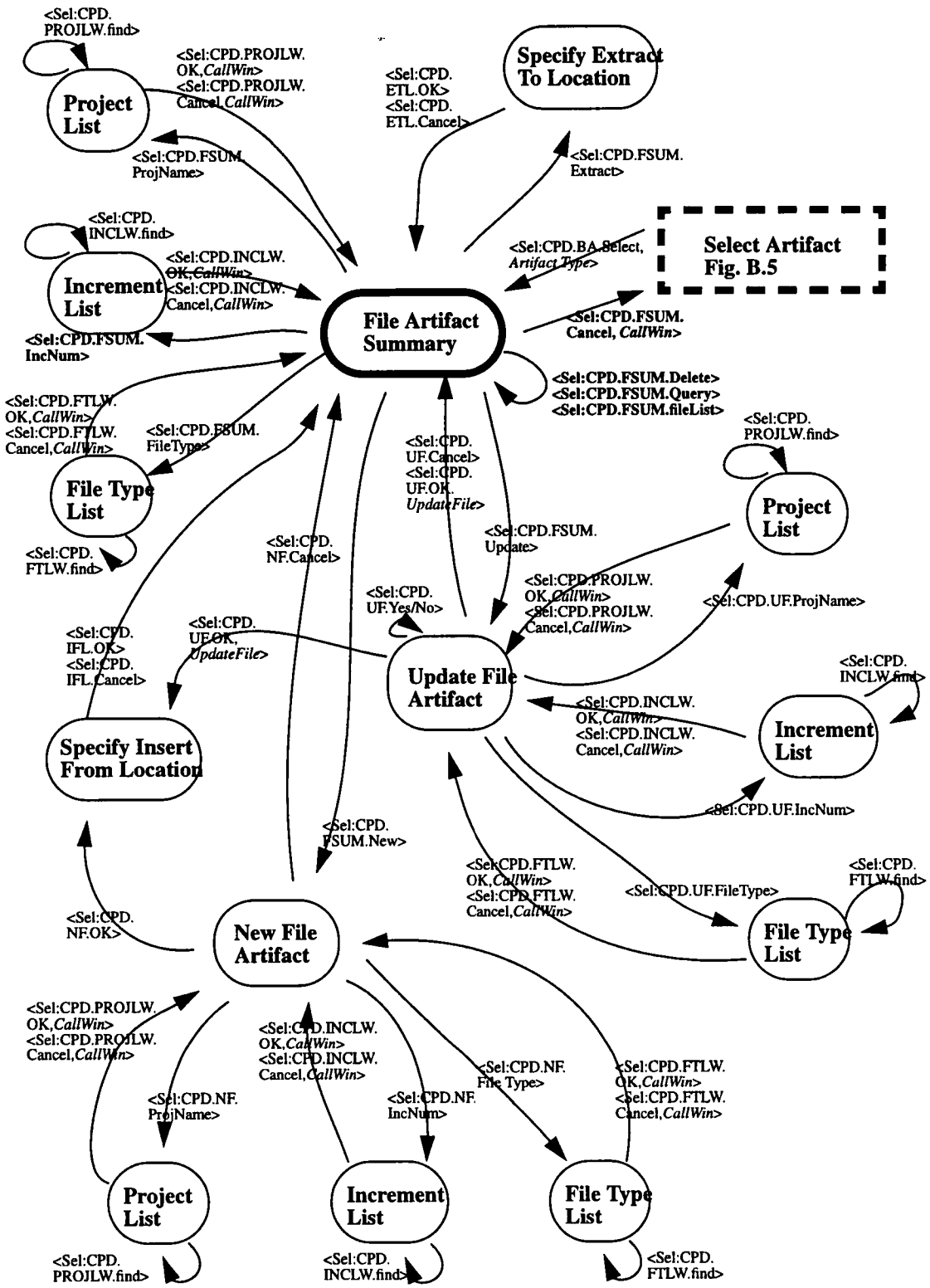

Figure B.7: File Summary, New/Update File Artifact Usage Model

VITA

John Gibson was born in Brisbane, Australia on June 6, 1967. He graduated from Brisbane State High School in November 1984. The following January he was granted a two year tennis scholarship to the Australian Institute of Sport in Canberra, Australia. While at the Sports Institute, Mr. Gibson took several introductory Computer Science classes at the Canberra College of Advanced Education.

In 1988, Mr. Gibson accepted a tennis scholarship at the University of Tennessee, Knoxville. Mr. Gibson received the awards of Academic All-SEC in tennis in 1988, 1990, and 1991. Mr. Gibson was elected to the Upsilon Pi Epsilon Computer Science Honor Society in 1992. That same year Mr. Gibson received his Bachelor of Science Degree with a major in Computer Science and a minor in Business. On December 19, 1992 Mr. Gibson married Amy Kay Gibson of Kingsport, Tennessee.

Upon graduation, Mr. Gibson accepted a position as Graduate Assistant System Administrator to the University of Tennessee Computer Science Department. Mr. Gibson's responsibilities included administration of the department's 32 node CM-5 super computer, monitoring network security, network backup, and installing, configuring, upgrading, and maintaining various hardware and software.

In February 1994, Mr. Gibson accepted a position as Research Assistant for the Software Quality Research Laboratory at the University of Tennessee. His responsibilities included the development of CASE tools for the Unix platform to support Cleanroom Software Engineering. He implemented the Cleanroom Process Database and was a member of a team of designers and developers implementing the Usage Model Editor. Mr. Gibson received the Master of Science Degree with a major in Computer Science in May 1995.