NISTIR 6137

The EFFective Manager Tool for Software Developers

Dolores Wallace
Mark Zimmerman

U.S. DEPARTMENT OF COMMERCE
Software Diagnostics and
Conformance Testing Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

NIST

NISTIR 6137

The EFFective Manager Tool for Software Developers

Dolores Wallace
Mark Zimmerman

U.S. DEPARTMENT OF COMMERCE
Software Diagnostics and
Conformance Testing Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

April 1998

U.S. DEPARTMENT OF COMMERCE
William M. Daley, Secretary

TECHNOLOGY ADMINISTRATION
Gary R. Bachula, Acting Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Raymond G. Kammer, Director

Abstract

The collection and analysis of software error, fault, and failure data from many high integrity systems may yield reference data for matching development and assurance methods to characteristics of a specific system. Profiles derived from the data may help researchers to identify areas where new methods of error prevention and detection are most needed. The National Institute of Standards and Technology has initiated a program on error, fault, and failure data to address these topics. An initial data collection and analysis tool has been developed for this project.

Keywords

Data; error; fault; failure; high integrity software; reference data; software quality; taxonomy; world wide web (WWW).

TABLE OF CONTENTS

TABLES

FIGURES

1. INTRODUCTION

The development and assurance of software for high integrity systems requires methods to prevent or detect software faults[1] during development and potential system faults and failures before they result in operational failure. It is difficult to predict how well development and assurance methods succeed in prevention and detection. Because introducing new technologies is costly, companies are reluctant to change unless they have confidence that the new methods will benefit them. Failures in high integrity systems are rare (and usually costly), and a single system usually does not accumulate enough data to permit meaningful statistical evaluations. Without sufficient data from many projects in various domains, researchers have difficulty identifying the types of problems for which new methods are needed. The results of a Call for White Papers issued by NIST revealed a strong need for an objective organization to address these problems [NIST95].

The mission of the Information Technology Laboratory (ITL) at NIST is to stimulate U.S. economic growth and industrial competitiveness through technical leadership and collaborative research in critical infrastructure technology (e.g., tests and test methods) to promote better development and use of information technology. ITL will provide tests and test methods to facilitate a usable, scalable, interoperable, and secure information technology infrastructure. One of the primary goals of ITL is to assure that U.S. industry, academia, and government have access to accurate and reliable test methods, data, and reference material.

Software researchers need project data on errors, faults and failures from many projects to identify characteristics and to develop benchmarks and profiles[2] for selecting methods and software tools. Providing such data is very closely related to ITL's mission. Consequently, ITL has initiated a project for error, fault, and failure data collection and analysis. While the project name is Reference Data: Software Error, Fault , Failure Data Collection & Analysis Repository Project, it is usually referred to as the EFF project. The EFF project recognizes the data needs for the development of high integrity systems and supports the mission of ITL.

The EFF project is collecting and analyzing data from the development and maintenance of software products or during the operation of a delivered computer system. The information technology industry may use the resulting reference data to develop software methods and tools and to build better end-user products. NIST encourages companies to consider the benefits of a public data base. NIST will accept new or existing data to augment the repository. All identifying

---

[1] In this paper, "error" is the human action that produces the incorrect result; fault is the manifestation of an error in an artifact; and failure is the result of a fault that has been activated during operation of a system.

[2] A profile provides a generalized characterization while a benchmark is a measurement; both may be used as reference values. An example profile may be "class of faults generally found for this set of defined characteristics in a specific application class"; an example benchmark may be "n faults for a program of this size, this language, this application domain."

information on data accepted by NIST will be removed before being included in the repository.

Several World Wide Web (WWW) tools are being developed by NIST to assist anyone collecting data for their internal analysis and to provide public access to data. The first tool developed for the EFF project is the EFFective Manager Tool (EFFTool), a WWW software tool for fault and failure data discovered during the development or maintenance of software. The EFFTool is a public domain tool that contains a fault management component to provide a benefit to any company who uses this tool. The tool enables a company to track the status of faults and failures and includes a simple analysis tool for tabulating the status of several fault and failure attributes.

Other tools are in the design stages. One is a data collection tool similar to the EFFTool but with data fields consistent with its purpose: collection of failure data from systems in operation. Both data collection tools will be used by industry on servers at their sites and the data may be provided to NIST whenever the contributor chooses. A WWW accessible data base system will provide access to sanitized data, other repositories and public domain analytic tools. Data for public access will have company identifiers removed. Existing statistical and graphical tools are being explored for their feasibility for analyzing and displaying data.

While the primary purpose of this report is to describe the EFFTool, it provides a complete project description (Section 2) and a discussion of research on error, fault, and failure data collection and analysis (Section 3). Section 4 provides an overview of using fault and failure data as a project management tool. Section 5 contains a description of the EFF project toolset, with Section 6 providing general description of the public domain EFFTool. APPENDIX A provides operational details for the data component of the EFFTool and APPENDIX B contains operational details of the analysis component and examples of displaying EFFTool data graphically.

2. THE EFF PROJECT

Successful prediction, risk assessment, and planning are crucial elements for saving millions of dollars per year in the software industry. But, project managers do not always collect the data on errors, faults, and failures that will help them to perform these tasks. On a larger scale, there is a fundamental lack of actual project data on errors, faults, and failures in a public repository. Without such data, industry and government agencies lack benchmark information against which to measure software program quality and to determine the software methods most appropriate for their software development environment.

The purpose of the EFF project is to provide reference data from software development and maintenance projects; fault and failure profiles and benchmarks derived from that data; analytic methods and tools; and metrics for measuring effectiveness of software development methods.

The EFF project will help industry and researchers[3] assess software system quality by collecting, analyzing, and providing error, fault, and failure data and by providing data collection and statistical methods and tools for the analysis of software systems.

Project data are needed to determine trends on broader concepts such as:

- Software error profiles: prevalent types (e.g., unachievable path, initial value, control-flow)
- Root technical causes of the errors and the development and assurance methods likely to prevent or detect those errors
- Types of problems requiring fault tolerance provisions in the software, and
- Error, fault, and failure problems not solvable or measurable by current methods or tools.

Data from many individual projects are needed to develop these and other benchmarks and to provide researchers with sufficient samples to develop new analytic methods and to identify where new methods are needed. Projects and their sponsoring companies need similar data to understand where specific error types are likely to occur and the frequency with which they occur. From various analysis methods, developers may locate troublesome parts of their programs and may adjust their development methods, adapt their testing processes, and maintain records for controlling their product quality.

Possible benefits to individual companies include:

- A public domain tool for collecting fault and failure data,
- Analysis tools to tabulate attributes of that data, and
- Methods to understand and compare projects within the company and against similar projects of other companies.

Possible benefits to industry from the EFF project include:

- Reference data for evaluating and selecting methods and software tools
- Taxonomy[4] and frequency profiles of errors, faults, and failures
- Reference methods for analyzing software data, and
- Data collection and preliminary analysis tools, using WWW technology.

By making data from various domains available to researchers, benefits to research may include:

- Software error, fault, and failure data available for analysis
- Qualitative and statistical methods for data analysis and measurement

---

[3]The term "industry" includes anyone developing software, including government, and the term "researchers" includes both academia and researchers within industry and government.

[4]A taxonomy is an organized classification scheme, in this instance, to identify types of faults and failures.

- Statistical basis to help with understanding error, fault, and failure data.

The EFF project involves the following tasks:

- Generate a standard data collection structure derived from IEEE and industry nomenclature and formats. Provide for anonymity of data and removal of any proprietary information.

- Seek industry, government, academic collaborators/contributors to populate the data repository.
  - Provide a WWW-based data collection and analysis tool for individual contributors for use at their site.
  - Accept and adapt data collected by other mechanisms.
  - Address privacy issues. Perform initial summary and analysis. Index data and summarize by common descriptive analysis.

- Make sanitized data publicly available through WWW-based facilities at NIST. Validate and sanitize data from contributors. Identify and procure commercial database management system. Refine the data collection and classification methodology as needed.

- Develop methods and tools for qualitative and statistical analysis. Identify or develop methods or tools for viewing data, for analyzing data, for measuring impact of methods on software quality, and for assessing relationships of project factors to software quality.

- Conduct analyses of collected data. Develop frequency profiles. Conduct analyses to provide understanding of impact of various development and diagnostics methods on failures. Report results/findings.

The plan for the EFF project is aggressive; a primary risk is that the group of willing data contributors will be very small. The fact that NIST's traditional role in defining standards and measures for industry includes objectivity and the ability to protect any proprietary information may help to overcome industry reluctance to provide data. The data collection tools with tracking and management capabilities may be an incentive for contributors. From the research perspective, another risk lies in normalizing data from diverse environments; this is part of the research problem of this project.

Several EFF tasks are progressing simultaneously. NIST has formed a collaborative relationship with SoHaR, Inc. under a Cooperative Research and Development Agreement (CRADA) for which SoHaR, Inc. will be an active participant in the project. Another collaborative activity included a meeting in September, 1996, of researchers and industry representatives to discuss problems likely to be encountered and the results of NIST research in identifying a draft data

structure, or model. Indeed, the management component of the EFFTool was a consequence of this meeting. These industry representatives and researchers will continue to provide guidance.

As of the date of this report, another CRADA is under development, and several companies are negotiating the mechanics of providing data. The EFF project is continuously seeking contributors of data.

3. CONCEPTS OF EFF DATA COLLECTION AND ANALYSIS

Research on software faults, and hence on data collection and analysis, is almost as old as software itself. One early paper asks several questions for which the EFFTool is seeking data [ENDR]. These questions, pertinent today, are shown in Table 3.1 and indicate that considerable information about an error is needed to learn from it. Such information includes data about discovery of the problem (e.g., version, date, name of discoverer), description of the problem, resolution (date, name of resolver, version where change made), area or artifact of actual error cause, and the description of change.

RELEVANT QUESTIONS	
.	Where was the error made?
.	When was the error made?
.	Who (generic) made the error?
.	What was done wrong?
.	Why was the particular error made?
.	What could have been done to prevent this error?
.	If an error could not be prevented, what detection method could detect it?

Table 3.1 Questions for Software Error Analysis

Basili [BASI] proposed a highly organized approach for data collection and analysis, shown in Table 3.2. With respect to the EFF project, steps 1 and 2 are relatively easy. For step 3, some data categories have been easy to establish, but classifying the symptoms that revealed errors and ultimately the cause of each error is difficult and terms may be changed in the second version of the taxonomy. While a small group has reviewed the data concepts for the EFF project (Step 4), broader usage may require changes to it. Validation of the data (Step 5) will be extremely difficult because accurate (translate to possibly time-consuming) reporting of data is needed, the contributor must understand the data fields of the NIST tool, and NIST must carefully adapt the contributor's data fields to the NIST nomenclature. Implementation of Step 6 relies on contributed data and availability of the data to researchers.

Several potential contributors plan to provide data from existing collections. Such data will vary in content and must be translated into the data categories evolving in this project. Special care will be exercised when analyzing data collected by a mechanism different from the NIST tool. In some cases, a link may be provided to an existing data base.

BASILI'S GUIDELINES	
1.	Establish the goals of the data collection
2.	Develop a list of questions of interest
3.	Establish data categories
4.	Design and test the data types
5.	Collect and validate data
6.	Analyze data

Table 3.2 Process for Data Collection and Analysis

At the September 1996 meeting, attendees discussed taxonomies for faults and failures and some models for descriptive data about a project and its errors, faults, and failures (Table 3.3). The diversity among the taxonomies is great. The research occurred in different domains, problem sizes, languages, and other variables. For example Endres' interest was in an operating system while Beizer collected data from many projects of varying types and languages. And, worse, much of the data that lead to these specific taxonomies was collected before the existence of some languages (e.g., Ada, C++, JAVA) and the use of software tools aiding development. Do classifications apply to *all* languages equally? To *all* types of software? Has the advent of design tools, analysis tools, and other parameters changed the nature of errors and hence the faults manifested in the artifacts? And, has the entry of more complex systems added failures that couldn't have been dreamed of before networks? These issues need to be addressed when developing new taxonomies. Unfortunately, data are needed to develop the taxonomies, and collecting the data with a predefined taxonomy imposes a problem.

Key lessons learned from other researchers who have influenced the EFF project include the following:

- Attributes from large projects and small projects differ and will yield different fault frequencies.
- Normalization across projects needs to account for variability in project environments.
- Existing taxonomies have varying levels of details that make synchronization difficult.
- Fairly complex projects are needed for meaningful data.
- Data requests must be kept simple.

TAXONOMIES AND DATA MODELS		
Simple, few elements	Rubey Glass Weiss/Basili Grady Chillarege Fenton/Pfleeger	late 1970's 1981 1985 1992 1993 1996
Several groups, details	Endres Knuth	1975 1989
Security-oriented	Landwehr Aslam	1995 1995
CMU experiment: need data from complex projects	Greenberg/Siewiorek	1996
Maintenance	Stark	1997
Detailed, life cycle oriented	IEEE Standard 1044 Beizer Hecht/Wallace	1993 1990 1996

Table 3.3 Diversity Among Taxonomies

4. FAULT and FAILURE DATA AS A PROJECT MANAGEMENT TOOL

To be competitive, companies need to get their products out the door. To remain competitive, they need to learn from the current project and apply those lessons if possible during the current project and certainly to the next one. Collecting and analyzing data on faults and failures provides support to these objectives. While other issues must be addressed for overall process improvement, the EFF project is concerned only with the collection and analysis of fault data.

The assignment to get the product out should translate to getting a *quality* product built and delivered *within budget, on time*. Because the term quality may have varying definitions, the assumption is that the project manager knows the organizations's definition and has guidelines for the judging acceptability of the product[5]. Regardless of the definition of quality, the project manager needs to know the status of the faults and failures and resolve them.

The fault or failure may have a priority for resolution and a person assigned to its resolution. The

[5]Standard profiles of faults and failures for specific application domains would assist in defining acceptability, and in bettering the "standard" to beat competition.

status may be either open or resolved where resolution of a fault or failure may result in any of these states:

- corrected,
- deferral of correction until another version of the program,
- correction resulting from a correction to another problem,
- dismissal, that is, upon examination, no correction needed.

Typical questions that the project manager may ask:

- How many faults with highest priority are open?
- What is the average number of days to resolve a fault?
- How many faults/ failures were discovered in a specific month (e.g., May 1997) ?
- How many faults/ failures were resolved in a specific month (e.g., May 1997)?
- What is the symptom of each open fault? What is its priority?

These general questions comprise a subset of many questions whose answers provide in-depth information about the project. For this subset, the data elements needed to answer them are basic: the date each fault is found, the date of its resolution, its priority for resolution, and its symptom. The answers assist project management in keeping a tracking file that provides a simple count of faults yet to be resolved, along with their priority.

Additional information, such as the type of unresolved fault or the activity in which the fault was discovered, may enable decisions about methods for developing the software and discovering faults. Large numbers of faults relative to project size, wild schedule swings due to many faults and rework, and many faults not revealing themselves until they become failures in system test are among the many types of signals suggesting that a project needs some correction. The manager may not even know these circumstances exist if data had not been collected and then reviewed. The manager needs more information about the faults and failures. Once data are reviewed, any number of actions may be taken, depending on the specific data. Examples include: reconsidering why inspections were not conducted for the artifacts with the faults, or why a lower priority was assigned to some open faults that are obviously very serious? Different managers on different projects may make different decisions.[6]

The EFFTool provides an opportunity to collect and analyze data to assist in getting a quality product delivered on time. By keeping track of fault attributes, amount of effort to correct a fault or failure, numbers of days to achieve the correction, and other features, managers can better

---

[6]For example, in one case, an inappropriate design tool for the application domain may lead to timing faults. In another, not having sufficient traceability data leads to not assigning higher priority to some faults. In yet another, time was running out, inspections were bypassed, unit test was hopelessly bogged down. What can be done in these situations? On this project? On the next project? This report is not intended to be a complete tutorial on using fault and failure data to assist in project management; many books and technical journals contain considerable information.

understand what is occurring on the project. With this understanding, managers can assess the types of changes and perhaps more effectively manage progress.

When the company collects and keeps the data from a project, that company will have valuable information for the next version of the product. The data provide lessons learned about methods for developing the software and for finding faults. Profiles can be generted to indicate frequency of faults found during specific activities, or the frequency of types of faults being found and when they are usually found. By examining profiles of the previous version or a similar product, project managers may be able to avoid some problems via better staff training, improved checklists that address the most frequent faults (and guidance on how to use the checklists), and more accurate test scheduling.

5. THE EFF PROJECT TOOLSET

The EFF project plan calls for several tools for the collection and analysis of fault and failure data:

- a public data base facility at NIST - the EFFPublicData Tool
- a data collection and (simple) analysis tool for fault and failure data during development and maintenance - the EFFective Manager Tool (EFFTool)
- a data collection and (simple) analysis tool for failures of systems in operation - the EFFSystem Tool, and
- access to public domain methods and tools providing more complex statistical and graphical capabilities, which may be developed by NIST or other researchers.

The high integrity software system assurance (HISSA) page at http://hissa.nist.gov/ links to the taxonomy-based Reference Information for Software Quality (RISQ) [NIST97]. RISQ provides direct access to artifacts for software quality. Among the artifacts (e.g., documents, tools, code) is one called data. The EFF data will be a data artifact accessible through RISQ. The data will reside in a database system residing on the HISSA server. Any contributed data will have been sanitized on a non-publicly-accessible system first, and then will be entered into the data base. Statistical functions, fault and failure tracking capabilities, and interfaces to publicly-available tools for examining the data will be available on the public system. Users of the data will also be able to download and analyze the data with their own tools.

Important criteria imposed for data collection tools for the EFF project discussed at the September, 1996, meeting included the need to keep data as simple as possible and to avoid the use of ambiguous terms. Contributors must understand the meaning of each type of data. A condition on the EFF tools that evolved at the meeting is that any data collection and analysis tool should provide a service to the contributors that will make collecting the data worthwhile to them during and following the collection process. The two collection tools provide searching and computational capabilities to enable managers to track their projects. The EFFTool could be used personally by individuals for tracking and improving their own processes.

The two data collection tools are to be WWW tools that a contributor could easily install on an organization's server. Almost everyone has access to the WWW and can download Perl. While the first version of the EFFTool was built for a unix server, the tool will be adapted for installation for other operating systems. The amount of data requested has been kept minimal, and data is generally entered via a click of the mouse. Definitions of terms are provided via Help buttons to prevent ambiguous responses. Other assumptions are that for every project, there will be one set of descriptive data about the project environment while there may be many unique faults and failures. The fault and failure data may be entered by one or more people for one project, an advantage of using a WWW tool. The data, as collected by the organization on its server, are available only to that organization. Only when installed at NIST will the data become publicly available.

The data collection tools, the EFFTool for data during development and maintenance activities and the EFFSystem Tool for failure data collected during operation of a software product, each consist of a project or system environment file, a file for each fault or failure, and finally a capability to search and analyze the fault and failure data according to their characteristics. It is this analysis component that provides the contributors with a value-added benefit for using the tools.

NIST receives data when a contributing organization is ready to release the data. NIST will examine and validate the data, remove any public identification of the contributing organization, and install the data on the public EFF data base. If requested, the organization's contact will be notified of the identity of its data within the EFF data base.

6. THE EFFTool

The EFFTool data consists of some general project information and specific information related to the discovery and resolution of faults and failures occurring during the development or maintenance processes. One purpose is to provide a facility that over many projects will contain enough information to further understanding of how faults and failures occurred, how they could have been prevented, and how they could have been detected earlier. From the current project manager's perspective, an important purpose is to contribute to individual project managers' capability to manage the project by examining the collected fault and failure data. Complete project management data, such as schedules, milestones, personnel skills and experience, are not requested because many methods and tools exist to understand process itself, and companies may consider supplying such data to be redundant. Such data for the EFF project goals may not be needed. The EFF project seeks data at a lower level and focuses on product in order to understand methods to be used in the development and maintenance processes.

The EFFTool is an interactive WWW tool which receives data by text entry, radio buttons, pull-down menus, and check boxes. While APPENDIX A contains operational information with details on the user interfaces and elements of the data categories, this section describes, in general, the data requirements and some research issues concerning them.

6.1 The EFFTool COLLECTION COMPONENT

The EFFTool collection component receives data about the project and about each fault or failure. The data fields were selected by trying to foresee questions that researchers and organizations might ask. Obviously, foreseeing every question is impossible, but the requested data allow for a multitude of questions; this topic is expanded in Section 6.2.

Within an organization, environment features for projects may be the same and may be known to all who work on the project. These features include elements such as development processes, the standards which govern the project, the programming language, and quality practices. For a single project, drawing conclusions about whether a specific method worked well may be relatively easy, provided other experimentation practices are followed [ZELK]. When data are collected across many projects and different organizations, these environment and organization elements are likely to differ. Researchers may combine data from like elements but they must be aware of differences and establish how they will treat the differences in their analyses.

To aid with the problem of normalizing data from many environments, the EFFTool requires a fairly extensive project description. Some questions address the company, such as how many years the company has developed software in the application domain. Others, such as primary software language, are critical to understanding certain project information. Program size for source lines of code (SLOC) may be computed differently for different languages. Size, the types of standards or quality practices applied to the project, and language are only a few characteristics for which profiles may change across projects. When drawing conclusions about the efficacy of methods, researchers need to take project differences into account. The general data categories for project information are shown in Table 6.1; most categories expand via a pull-down menu into set of choices, with "Other" provided to accommodate omissions from the set. While project data are entered only once, they may be edited. But the persons entering fault and failure data need not be concerned with the project data after the first entry.

TYPE OF PROJECT DATA REQUESTED	
Project name	CMM level
Company name & contact	Primary software language
Brief project description	Size of new & reused code
Date project began	% of COTS code
Development or maintenance	Requirements, design methods, automation
Perceived type of consequence of failure of the completed system	Performer of QA / VV
Perceived criticality of failure	Company experience: in domain
Relation to hardware	Company experience: w/ software in domain
Generic domain	Company experience: w/ software in general
Specific application	Company quality practices
Contractual requirements	ISO 9000

Table 6.1 The EFFTool Project Information

The form for fault and failure data will be used for every fault or failure. Details are separated into 3 groups: administrative, discovery and resolution, with most categories expanding via pull-down menus into a set of choices. Some of the requested details are intended to enable project managers to analyze the progress of their project and to make adjustments. Most of the information will be useful in the analysis of faults and failures over several projects. The requested data will be very important in enabling researchers to develop profiles, benchmarks, and quality measures from the public data base.

The administrative data types are simple. A number is assigned to each fault or failure. A short text description is requested (e.g., incorrect parameter passed to invoice sum program). A field for the actual name of a person exists; this is for company purposes so that other members of the organization may ask questions about the fault or failure either during the project or, later, for lessons learned or other company purposes. The name could be the project manager, the test manager, the discoverer, resolver, or anyone familiar with that aspect of the project. The name will not be used in the NIST data base. Finally, the administrative section contains the status field: open, open-assigned, resolved-corrected, resolved-deferred, resolved- no correction needed, resolved - corrected by another resolution.

Conflicting goals for the EFFTool arise when collection and use of fault and failure data are taken into account. The programmer, typically under pressure to finish, may be willing to provide a little information while investing as little time as possible into the fault documentation process. On the other hand, the user of this information wants a significant amount of characteristics for project planning and for further process changes. Asking for too much information from the fault or failure discoverer could result in no information being provided

(i.e., why bother- I don't have that kind of information at hand anyhow). Because of this problem, only fault/failure data related directly to the fault or failure are included. Information such as module complexity, module size and other attributes is sacrificed in the desire to obtain as accurate and complete fault data as possible.

For purposes of simplicity, any usage of the term "fault" in the remainder of this report includes both fault and failure. A summary of the data categories for discovery is shown in Table 6.2. Basic questions like "when found, where found, who found" are asked, and generic attributes are provided from which to select. The user has the option to specify more details on where a fault was discovered. For example, if the fault was discovered in the design, then the user may also enter an identifier in a text field for the generic artifact type, like for design, SUM_MONEY. The user supplies input on the possible consequence of this fault being allowed to remain in the system, and on the impact on the development schedule. The user assigns a priority to resolution of this fault. Assigning the priority may be a part of resolution in some organizations, or may be changed during resolution.

The data types for fault discovery allow for a description of the discovery method, (e.g., inspections focused on initialization). This category may iterate into looking more like Chillarege's triggers [CHIL]. Chillarege considers a trigger to be an activation process that activates a software failure from a fault. This is not quite the same as a method for discovering a fault, but seems closely aligned. Chillarege has identified triggers for inspection, function test, and system test. It may be that both categories will be useful for fault and failure analysis, or that, if enough data from enough projects are collected, perhaps a set of principal triggers satisfying both meanings can be identified.

Another data type presenting difficulty is the symptom of a problem. The symptom is the visible indication to the discoverer that something is wrong. Examples of symptoms include "ambiguity" and "execution stopped." When data are submitted to NIST, the "Other" field for the symptom category may indicate that the current list of symptoms needs to change. A similar problem occurs with fault classification in the resolution data, where the fault classification is the name of the actual cause. An example of the difference may be that an incorrect parameter is the visible symptom but the cause may have been a typing error or an incorrect specification. (And, the analyst needs to discover what caused the incorrect specification!) These two fields, like the trigger, are expected to iterate over time as more data arrive, especially from project using modern technology.

FAULT (and FAILURE) DISCOVERY	
Date of fault discovery	Potential severity if not fixed
Type of person who found fault	Impact on schedule
Specific location where fault discovered	Priority for resolution[7]
Generic artifact where fault discovered	Discovery method
Activity during which fault discovered	Discovery method effort
Symptom	Degree method automated

Table 6.2 The EFFTool Discovery Data

The resolution data types (Table 6.3) are similar to the discovery data in most data fields but there are differences. In resolution, the artifacts requested are those where the changes were made, which may lead to the actual source of the fault. For example, if the fault is found in the code, resolution may find that the source is in the design, and changes were made in both artifacts. The project manager will likely want to ensure that the error has been corrected, and that any other faults on other locations have been corrected. Researchers using the data to understand effectiveness of methods would in this case look for more errors in using the design method of this project and other projects using the same design method.

FAULT (and FAILURE) RESOLUTION
Date resolution completed
Resolver (generic)
Classification
Generic artifacts fixed - also text fields for specific ids
Was this caused by a previous fix?
Project activity during which resolution is made
Resolution method
Degree method is automated
Resolution method effort

Table 6.3 The EFFTool Resolution Data

To reiterate, one major challenge is the classification of faults and symptoms. The classifications must be easily understood and mutually exclusive. They need to apply to, or at least be understood with, all modern technologies. Also, a long selection list may tax the patience of contributors. The problem is to synthesize terms from the many existing taxonomies. If usage of this tool indicates that contributors are willing to scroll through multi-level lists, then it may be

[7]Priority may not always be assigned immediately upon discovery - may be part of resolution process.

feasible to go to more fine-grained fault classifications similar to Beizer's taxonomy [BEIZ]. Another question asks if the fault occurred because of resolving some other problem. Tracking the answers to this question may lead to better understanding of the maintenance process while helping to identify fault classifications . The expectation is that data and comments from users of this first version of the EFFTool will be useful in defining a better taxonomy for the second version.

6.2 THE EFFTool ANALYSIS COMPONENT

The purpose of the analysis component of the EFFTool is to give the company using the tool added value to make it worthwhile to collect data in the first place. The analysis component provides capability to sort and count faults having certain characteristics selected from the data elements and to perform various calculations and comparisons on those faults. The types of questions asked in Table 3.1, and many other queries, can be answered by using the features of the analysis component. The project (or test or quality assurance) manager should be able to use the analysis capability to monitor the project and to identify where to make changes related to methods, schedules, staff and other project concerns.

The analysis component enables a search to identify only those faults or failures with a certain set of characteristics, or attributes. The analyst composes a query to narrow the set of attributes of interest. The tool provides capability to sort and count information about the faults and failures according to selection criteria and enables some computations on numbers of faults and time. It enables monitoring the status of open and resolved faults. The profiles derived from the analysis capability enable visibility to project managers and staff to establish priorities for resolving faults / failures, status of project as a whole, effectiveness of development or testing methods, training needs, technology needs, and other issues serving to improve current and future projects.

The user can locate information on a specific collection of faults and failures by successively narrowing the attributes that identify a fault. The queries are basically in two parts, first, selecting fault/ failures by record number groups or by time periods, and second, selecting specific features of a fault/ failure. Then, the record numbers and status are displayed for that group of records. Additional category information may be displayed as selected.

A typical query might include identifying the most urgent faults found during inspection after December 1, 1997 and before January 10, 1998 that are still open. Or, the analyst may have wanted to know instead which of those have been resolved and the classification of each. Or, the analyst may have wanted to know the average number of days the faults in that group were open.

These are the types of queries for which profiles will be generated from the public data base, which will also allow queries on project characteristics. The queries may be saved and edited when invoked again. Specific details on building queries and examples of EFFTool output are provided in APPENDIX A.

The EFFTool is intended to be used by companies at their sites. Because commercial tools exist to perform many statistical analyses and related graphics, the tool does not provide those capabilities but does provide simple filtering and counting capabilities. The output of the EFFTool is an ASCII file, for which we provide a script to translate the data to be acceptable to most spreadsheets, which usually have some graphics capability for the most common statistical measures.

7. SUMMARY

The development and assurance of software for high integrity systems requires methods to prevent or detect software faults during development and potential system faults and failures before they result in operational failure. NIST initiated a program for fault and failure data to enable researchers and practitioners to evaluate how well methods support prevention and detection of faults and failures. The data will also enable development of benchmarks and profiles that indicate to companies how well they are developing and maintaining software systems relative to other similar systems.

NIST will accept new or existing private data to augment the repository. Such data may be from either the software development or maintenance process, or may consist of failure data derived from systems already in service. All identifying information on data accepted by NIST will be removed any before being included in the repository.

As a further service, NIST is making tools available via the Web site for use by anyone that may want to collect data for their own internal analysis. The EFFTool provides for relatively easy data entry and in addition has the capability to extract information derived from specific characteristics of the entered data. Use of the derived information can be used by project, test, or quality assurance managers to monitor their projects and to make adjustments in their processes (e.g., methods, schedules, other resources) as needed.

The EFFTool is a public domain tool which may be downloaded from NIST at `http://hissa.nist.gov/toolkit/eff.html`, or one may request to `dwallace@nist.gov` that the tool be sent on a diskette.

8. ACKNOWLEDGMENT

We are grateful to the participants in the September 1996 meeting at NIST: Dr. V. Basili, University of Maryland; Dr. S. L. Pfleeger, Howard University; Dr. P. Keiller, Howard University; T. Rhodes, NIST; Dr. M. Zelkowitz, University of Maryland & NIST; Dr. C. Michael, RST Corporation; J. Calvert, Nuclear Regulatory Commission; J. Gaffney, Lockheed-Martin; Dr. N.F.Zhang, NIST; S. Hissam, CARDS. Affiliations were applicable at time of meeting.

9. REFERENCES

[BASI] V. R. Basili and D. M. Weiss, "A methodology for collecting valid software engineering data," *TSE 10*, Number 6, November, 1984, 728-738.

[BEIZ] Boris Beizer, *Software Testing Techniques*, International Thomson Computer Press, 1990.

[CHIL] Ram Chillarege and Karen A. Bassin, "Software Triggers as a function of time - ODC on field faults," Fifth IFIP Conference on Dependable Computing for Critical Applications, IEEE Computer Society, 1995.

[ENDR] Albert Endres, "An Analysis of Errors and Their Causes in System Programs," *IEEE Transactions on Software Engineering,* Vol.SE-1, No.2 June 1975, pp.140-149.

[HECHT] Herbert Hecht, Dolores Wallace, "Project Data to Support High Integrity Methods, " Nuclear Plant Instrumentation, Control and Human Interface Technologies Conference May 6-9, 1996, Pennsylvania State University, State College, PA.

[NIST95] Dolores Wallace and Marvin Zelkowitz, NISTIR 5677, "Center for High Integrity Software System Assurance-Initial Goals and Activities," U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology, June 1995.

[NIST97] Charles B. Weinstock and Dolores R. Wallace, NISTIR 5954, "RISQ: A WWW-Based Tool for Referencing Information on Software Quality," U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology, January 1997.

[ZELK] Zelkowitz, Marvin V., and Dolores R. Wallace, "Experimental Models for Software Diagnosis," NIST IR 5889, September, 1996

APPENDIX A. USING THE EFFTool

The EFFective Manager Tool (EFFTool) is a World Wide Web (WWW) tool designed to collect and analyze data on faults and failures. This tool has 2 components associated with it: the data collection component and the data analysis component. Both components may be accessed through the entry menu. This APPENDIX provides guidance for using the tool.

A.1 Collection Component Menus

The data collection component consists of the entry menu containing the project data collection functions, access to the fault/failure collection functions, and the analysis component (Figure A.1). The data collection functions are similar for both project and fault/ failure data; the user may enter a record of data, edit it, view it, and delete it. While probably only one user will be responsible for entering the project data, those users entering fault/ failure data may occasionally want to verify that the project information is correct. Otherwise these users will go immediately to the Fault/Failure menu. Throughout this component and the analysis component, buttons marked HELP or H display online help files designed to assist and guide the user through various aspects of the tool.

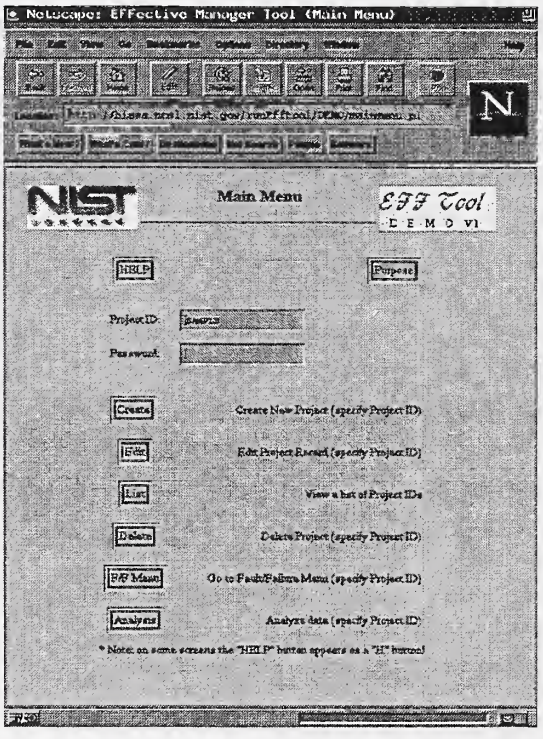

Figure A.1 Project (entry) menu

The entry, or project, menu provides services for the user to:

1. Create a project record and enter project data
2. Edit project data
3. View a list of project names
4. Delete a project
5. Go to the Fault/Failure (level) menu
6. Go to the analysis component of the tool.

The user may choose not to provide most of the project data but both the project name and a password must be supplied to gain entry to the fault/ failure data. And, of course, if a company is using this tool for more than one project, the password will prevent persons on one project from examining data from another project. Project name and password are required to review the project information. For companies planning to provide data to NIST, the project information will be essential in developing profiles over many projects. Only one project record has been entered into a demonstration data base and the execution of "View a list of project names" yields the output shown in Figure A.2.

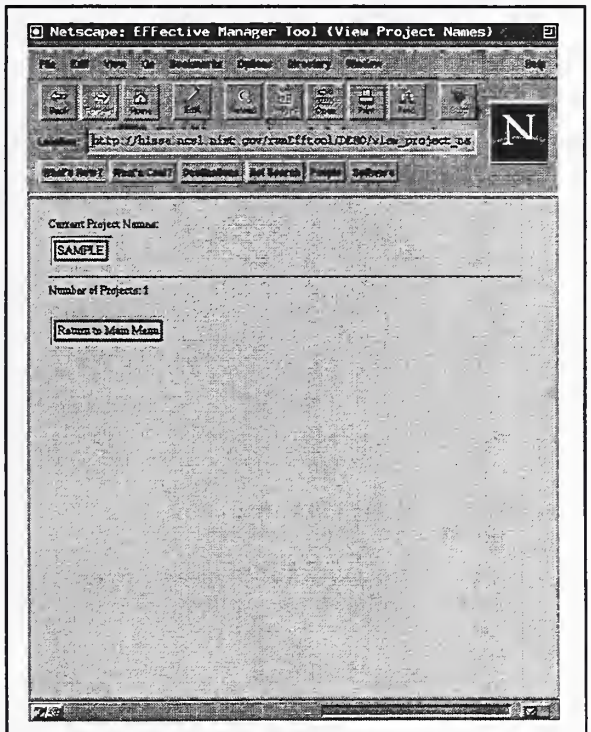

Figure A.2 The display as indicated above

The fault/failure menu (Figure A.3) provides functions for fault/ failure records similar to those for the project record. The complete fault / failure menu allows the user to:

1. Enter fault/failure data
2. Edit fault/failure data
3. View a list of Fault numbers and Generic Description
4. View fault /failure data (and select a record to edit from this function)
5. Delete a fault/failure record
6. Go to analysis component
7. Return to main menu.

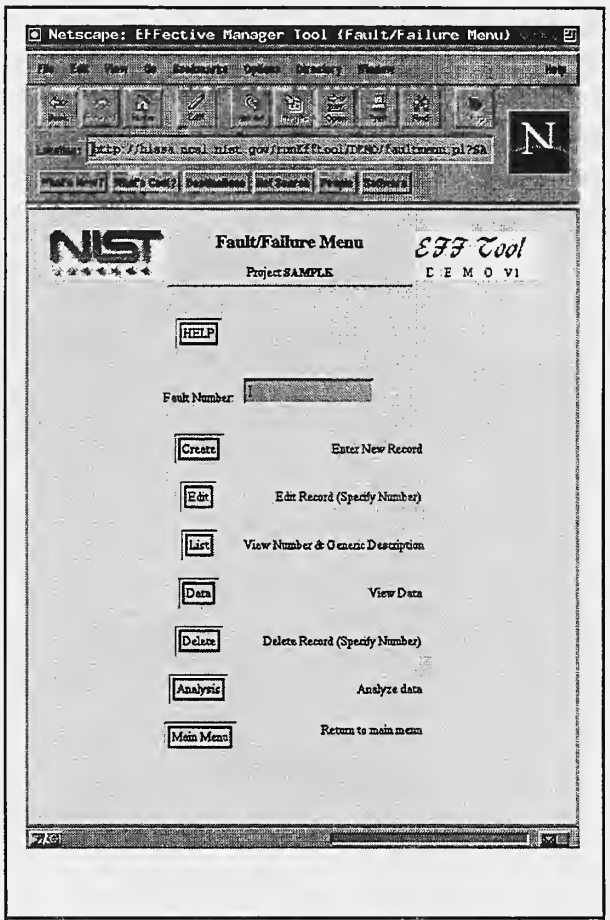

Figure A.3 Fault/ failure menu

A user may enter a new record or may view or edit a record. Similar to the requirement to provide a project name to gain entry to the project data, the user must provide a record number to edit a record. When the user does not know exactly which record to review, he can use the capability to display and scroll through all the fault /failure records in a condensed format rather than in the data entry format. Then the user can click on the record number to go immediately to

edit mode (Figure A.4).

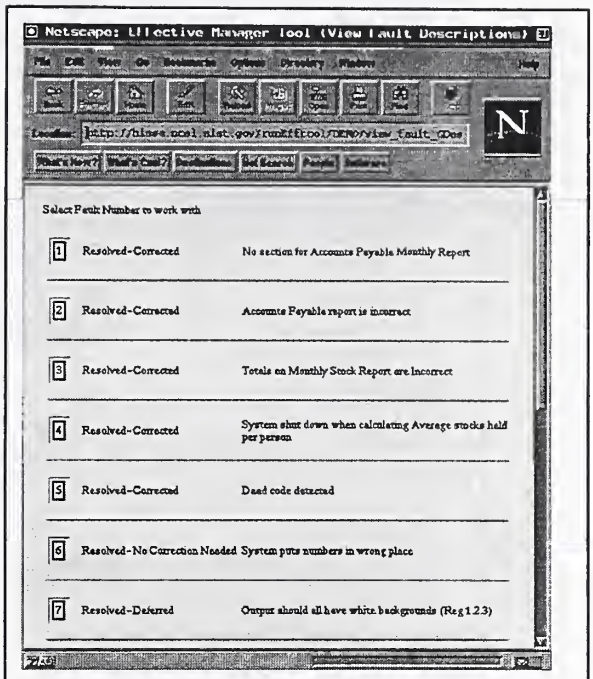

Figure A.4 A one record display from the View f/f/data

Data are generally entered using pull-down menus, radio button selection, or in a few cases, text entry. An example of a pull-down menu and a radio button selection is shown in Figure A.5 . Typically, these lists also provide a choice "Other" with a related text field in case the provided options do not apply. The radio buttons are used when more than one selection may be appropriate.

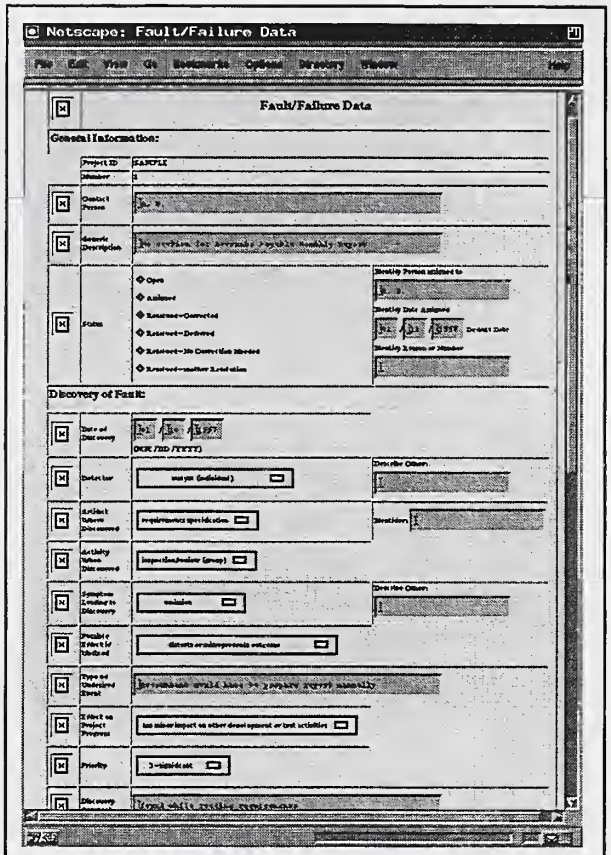

Figure A.5 Part of Fault/Failure Form

A.2 Analysis Component Menus and Displays

The Analysis Component of the EFFTool provides a query process to sort, count, and display fault and failure records according to attributes selected by the user. The Analyze Data Menu (Figure A.6) is the first screen of the Analysis Component. This component allows the user to save, edit, and execute queries. Managers (e.g., project, test, quality assurance) may use the analysis capability to monitor the project and to identify where to make changes related to methods, schedules, staff and other project concerns.

While the EFFTool allows a user to see all the fault and failure records as an option on the fault/failure menu, this list may not be very useful for a large number of records . Instead, the user may prefer to view only a selection of records, related to individual attributes. The analysis component allows the user to construct a query consisting of two parts: computations and attributes. Attributes are the characterizing elements selected from categories on the fault and failure data form. Computations are selection criteria based on Fault Numbers, Discovery Dates, Date Resolved, and Number of days in open status.

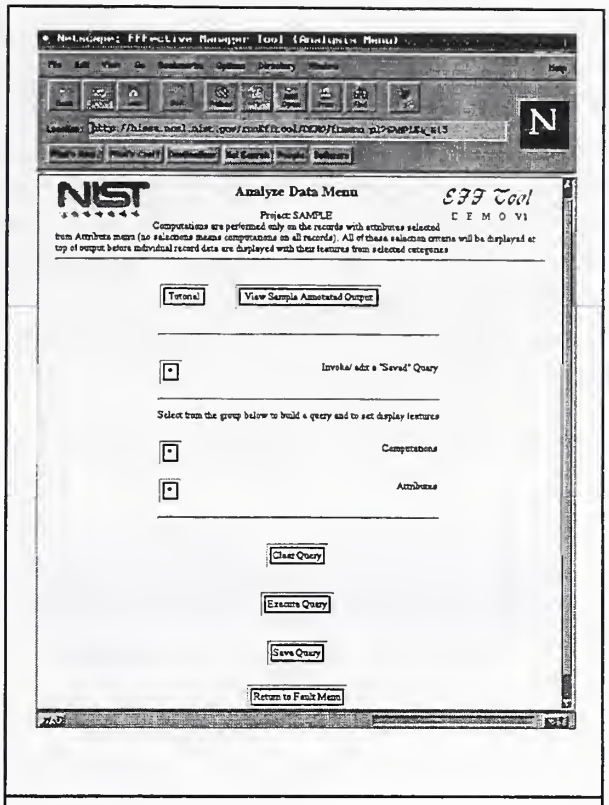

Figure A.6 Analysis Menu

When the user presses the Attributes button, a menu somewhat similar to the collection form appears on the screen. The difference is that the user has a choice of selecting any number of items from a category (left of the category) and selecting which categories will be displayed in the final output (right). The only faults or failures that will be displayed in the final output are those that match the selection criteria defined by the query.

Clicking on the box to the left of a category will display a list of characterizing items for that category. A fault or failure may have only one item per category associated with it. All items checked from this list will be added to the selection criteria, so that any records with those choices will be selected for display. Of course, when a selection from a category includes more than one item (e.g., 3 items), the displayed information will not identify which of those three items belongs to a specific record. Checking the box to the right of the category will add this category to the display option of the query and all selected records will be presented with the specific item from that category.

A query may be saved by clicking the save button at the bottom of the analysis component's main screen. The saved query can be retrieved automatically by clicking the Invoke/ edit a "Saved" Query button. The Clear Query button will remove the query definitions.

Examples of analysis

The user can locate information on a specific collection of faults and failures by successively narrowing characteristics about the collection. The queries are basically in two parts: 1) select fault/ failures by record number groups or by time periods, and 2) select specific features of a fault/ failure. Then, the record numbers and status are displayed for that group of records. Additional category information may be displayed as selected. Section 4 of this report provided a list of typical questions a project manager might ask. The following examples show how the EFFTool analysis component provides the answers. In the examples, the instruction set for each question is provided, with interactive "buttons" for the tool represented by text in the different font enclosed by brackets, e.g., `[name]` .

Example 1: How many faults with the most urgent priority are open?

```
[Attributes] on Analyze Data Menu
[*] Status
Mark Open.
[Return to Attribute Menu]
[*] Priority
Mark 1-urgent
[Return to Attribute Menu]
[Return to Analyze Data Menu]
[Execute Query]
```

In this example, the last line of the Display (Figure A.7) tells how many faults/failures are still open with the most urgent priority.

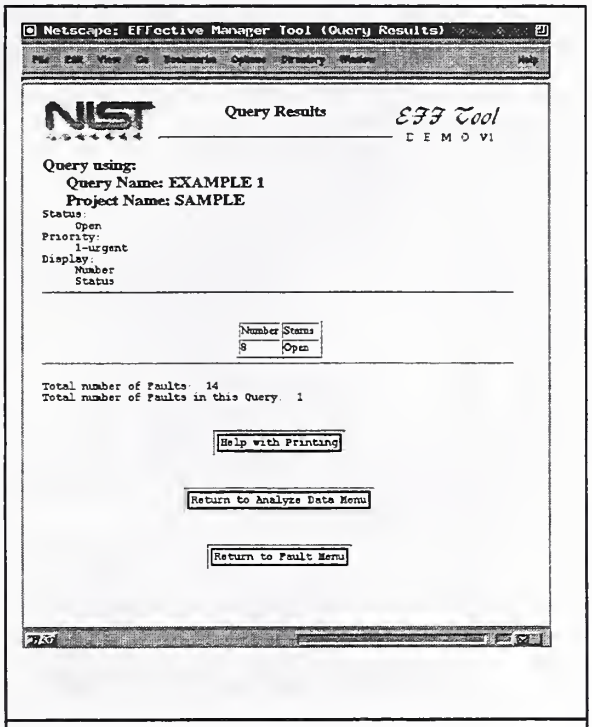

Figure A.7 Query results for Example 1

Example 2: What is the average number of days to resolve a fault ?

[Computations] on Analyze Data Menu
Mark box at bottom next to Show the
    average days open for each status
[Return to Analyze Data Menu]
[Attributes]
[*] status
Mark the 4 resolved statuses
[Return to Attribute Menu]
[Return to Analyze Data Menu]
[Execute Query]

The lower table in Figure A. 8 indicates the Average Number of days.

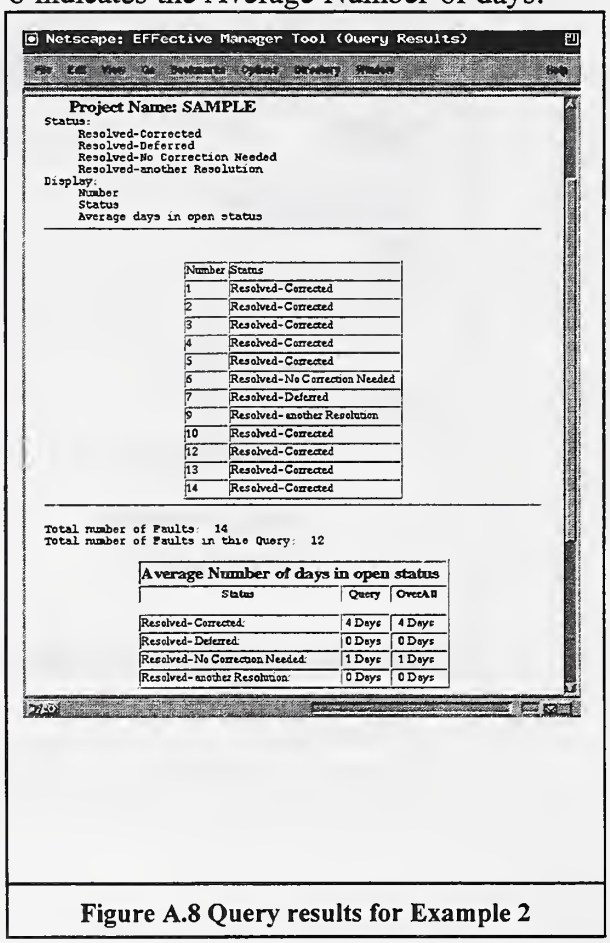

Figure A.8 Query results for Example 2

Example 3: How many faults/failures were discovered in May 1997?

[Computations] on Analyze Data Menu
Set Discovery Date After to 04/31/1997
Set Discovery Date Before to 06/1/1997
[Return to Analyze Data Menu]
[Execute Query]

The last line of the display in Figure A.9 indicates how many faults/failures were discovered between those dates.

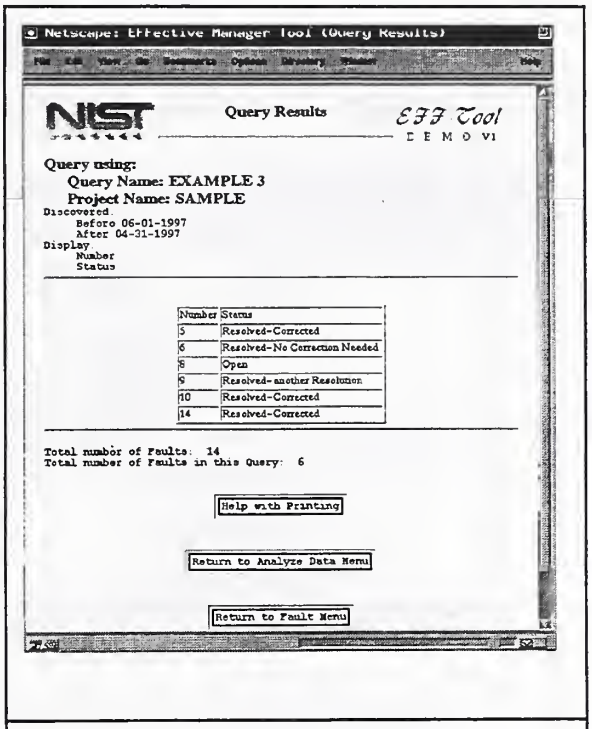

Figure A.9 Query results for Example 3

Example 4: How many faults/failure were Resolved in May 1997?

[Computations] on Analyze Data Menu
Set Date Resolved After to 04/31/1997
Set Date Resolved Before to 06/1/1997
[Return to Analyze Data Menu]
[Execute Query]

The last line of the display in Figure A. 10 indicates how many faults/failures were discovered between those dates.

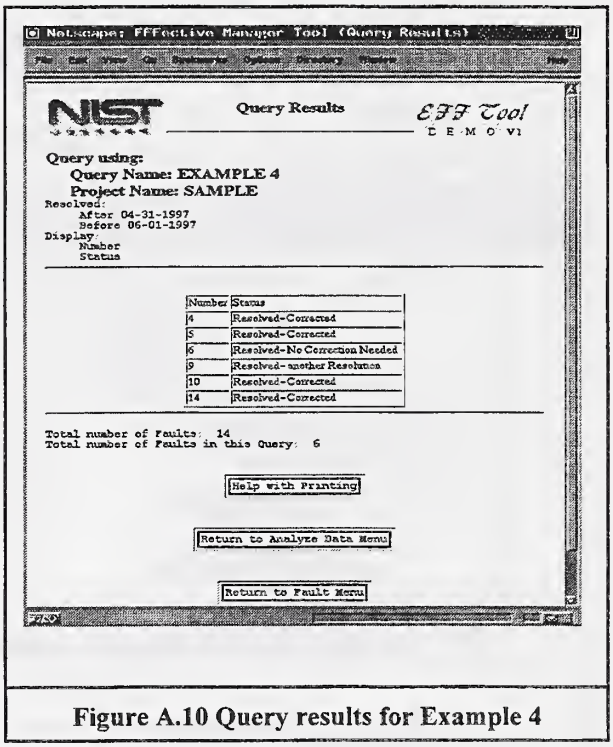

Figure A.10 Query results for Example 4

Example 5: How many faults/failures were discovered in May 1997 and Resolved in May 1997?

```
[Computations] on Analyze Data Menu
Set Discovery Date After to 04/31/1997
Set Discovery Date Before to 06/1/1997
Set Date Resolved After to 04/31/1997
Set Date Resolved Before to 06/1/1997
[Return to Analyze Data Menu]
[Execute Query]
```

The last line of the display in Figure A. 11 indicates how many faults/failures were discovered between those dates.

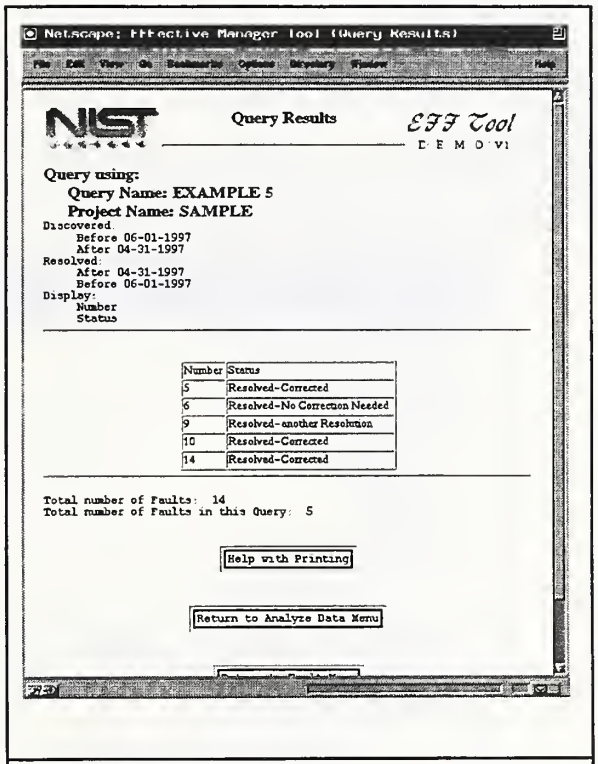

Figure A.11 Query results for Example 5

Example 6: Find all Faults discovered during requirements definition or design and resolved during any other activity. For these faults display fault number, detector, status, resolver, and the average days in open status?

[Attributes] on Analyze Data Menu
[*] beside Activity when Discovered
check requirements definition
check design
[Return to Attribute Menu]
[*] beside Activity When Resolved
check everything except requirements definition and design
[Return to Attribute Menu]
check detector and resolver (Note: Status is a default)
[Return to Analyze Data Menu]
[Computations]
check average days open for each status
[Return to Analyze Data Menu]
[Execute Query]

In this example, (Figure A.12) the first table displays the one fault that fits the selection criteria and the second table displays the average days in open status.

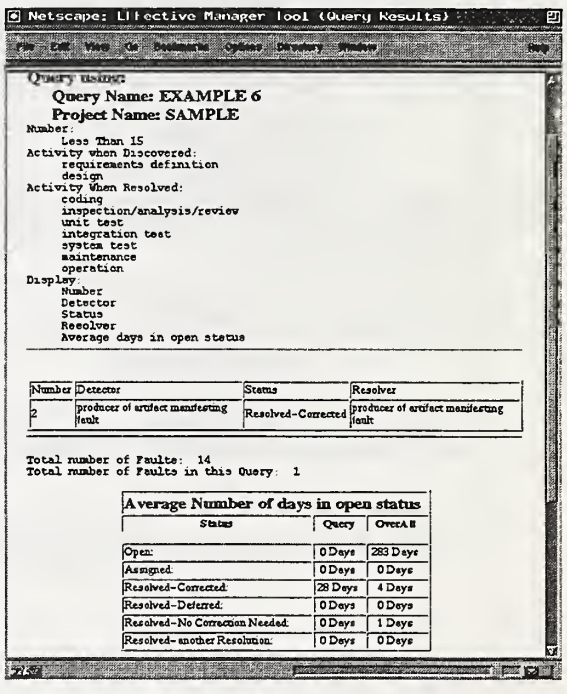

Figure A.12 Query results for Example 6