Spring 5-2001

Criteria for Designing Testability in Software Systems

Andrew Kinoshita Peterson
University of Tennessee-Knoxville

Appendix D - **UNIVERSITY HONORS PROGRAM**
SENIOR PROJECT - APPROVAL

Name: Andrew K. Petersen

College: A&S Department: Computer Science

Faculty Mentor: J. H. POORE

PROJECT TITLE: Criteria for Designing Testability in Software Systems

I have reviewed this completed senior honors thesis with this student and certify that it is a project commensurate with honors level undergraduate research in this field.

Signed: [illegible], ~~Fa~~Faculty Mentor

Date: 12/12/2000

Comments (Optional):

CRITERIA FOR DESIGNING TESTABILITY IN SOFTWARE SYSTEMS

Prepared for

Dr. Jesse Poore

by

Andrew Petersen

2 December 2000

ABSTRACT

Despite the use of good programming practices, software projects continue to run over-budget and past deadlines, often because of the increasing amount of time and resources needed to test and repair the code. To decrease the cost of software and keep development time at a reasonable level, programmers must adopt new methods for detecting and isolating faults. Other disciplines have, especially in recent decades, begun to integrate the idea of testability into their designs. The following report examines the tenets of "designing for testability" developed in other fields, in the hope that the essential ideas and proven methods from other fields can be adopted to deal with the testing problem in software development today.

TABLE OF CONTENTS

PROBLEM STATEMENT

The science of developing software is facing similar problems to those encountered in circuit design in the past two decades: as it becomes possible to pack more information into a smaller space, the problem of identifying and locating errors becomes increasingly difficult. The testing phase of software development projects is fast becoming a dominant portion of the budget and schedule. Part of this problem can be traced to the lack of acceptance of more formal development methods, which can make code manageable and reduce errors. However, the majority of the responsibility is borne by the increasing complexity and changing focus of systems under development today. Because of this, testability will have to become an important criterion to consider in the design process. Furthermore, the methods used in testing and the reasons for performing tests will have to change to meet new demands.

THE REASONS FOR THE INADEQUACY OF CURRENT TESTING SCHEMES

In the past, testing could often be performed on a single system, where the program could be isolated in its environment. Test cases were crafted and then applied to the system being tested after confirmation that the software installed and performed basic operations correctly. Additional testing was performed on an ad hoc basis. Any errors found were reported to the developers, who attempted to recreate the error, locate its source, and then repair it. Now, however, test managers report that with the increased complexity of projects as well as environments that prohibit testing a system in isolation, errors have become more difficult to detect, let alone locate or repair (Carlsgaard).

Accumulating Complexity

Historically, it has been much easier and cheaper to design a piece of hardware to perform some function. That fact has changed in the past several years, as embedded systems have been recognized as being more reliable, cheaper to manufacture, and simpler to "enhance" with additional features (Prowell). For example, early microwaves (and until very recently, most washers and dryers) used dials to set times and temperatures. Current models utilize embedded software and offer "advanced features" (preset values for steaming vegetables, for instance) to increase usability and perceived value. Software is being used on a larger scale in a more diverse range of applications, and much of it requires constant modification as the hardware being used changes and new features are added to the product line to accommodate changing customer demands.

However, these reasons should not, by themselves, increase program complexity. The root cause is the lack of use of good programming techniques, often because of perceived conflicts between good practice and impending deadlines, which help keep code simple. As more features are added and changes are made, a project often becomes less a planned, systematic development and more a series of hastily cobbled patches and extensions. If the code being modified and the project architecture are not fully understood by the programmer making the changes, he may introduce unintended side effects not only in that functionality being modified, but also in other parts of the program that use the modified code or variables modified by that code (Basili). In addition, the sheer amount of code required for many projects (several million lines in a typical cellular telephone) adds to the level of difficulty.

Changing Focus

In addition to being applied to a wider range of applications, software developed today has a different focus than in the past. Past development efforts often produced code designed to function as a self-sufficient unit. While this continues to be the case, there has been a trend toward developing multi-component systems. The result of a large industrial project, for example, may well be composed of ten or even twenty components, running on multiple platforms that must interact to provide some service or perform some function (Carlsgaard). Furthermore, consumer products are beginning to exhibit this behavior on an even larger scale. With the development of flexible and scalable networks powered by wireless communication systems like the proposed Bluetooth, the code for formerly self-sufficient devices (starting with laptops and hand-held computers and progressing, perhaps, even to microwaves and toasters) must be developed to deal with a more complex system environment (What is Bluetooth 1).

The Effect on Current Testing Schemes

Current testing schemes detect errors by testing basic functions and observing output. However, due to the modifications and extensions present in code produced today, it has become difficult to locate faults simply by observing output. A fault may not cause an observable problem unless very specific conditions are met. Furthermore, once an error is detected, it may not be immediately or easily reproducible, since its observability may rely on conditions set by a previous test case. In effect, the developers may have made their program appear nondeterministic at the level from which they can observe.

Furthermore, with the shift to co-dependent systems, it becomes difficult to test a program in isolation. Instead, an error, once detected, may have to be traced through multiple components to determine if it originated within the program being tested or propagated through an interface with another element of the system (Carlsgaard). System boundaries must be drawn very carefully: if too much is included, the error may be hidden in the "black box" of the system. If the boundary is too small, too many interfaces between the system and elements in the environment -- which cannot be tested at the same time -- will exist (Prowell).

THE REASONS FOR PERFORMING TESTING

The most obvious reason for performing testing is "to find the errors in the software." This implies that testing need only be performed at specific times -- when errors are reported or immediately after development. Another reason, currently cited mainly in safety-critical systems, is "to insure that the system is operating correctly." This takes a broader approach to testing: one that suggests that testing should occur more frequently, to insure that the software continues to operate correctly. With systems becoming more reliant on multiple elements in the environment, the reasons for performing testing will have to change. Instead of assuming that the goal of testing is merely to find errors in the code, developers will, for the benefit of the user, have to develop tests to gage the environment and insure that the system can continue to operate as specified (Turino 11).

ANALYSIS OF APPROACHES TAKEN BY OTHER DISCIPLINES

My research focused on three major disciplines: circuit design, chemical engineering, and facility management. The fundamental ideas in each discipline are summarized and then analyzed based on the kinds of results obtained, the purpose for performing testing, the amount of contamination introduced or efficiency lost, and the integration of testing issues into the design process.

DESCRIPTION OF CRITERIA USED IN ANALYSIS

Even if the fundamental reasoning behind the idea of designing for testability might be similar across most disciplines, each one has different priorities that cause these ideas to be implemented differently. To successfully import new ideas into the discipline, the factors involved in its use must be clear.

Results Obtained

Some methods of testing yield precise information about inputs or results, while others return estimates. In some cases, testing methods also provide a way to manipulate input values or isolate specific variables. The results obtained in testing are analyzed to determine if it would be useful in software development to test for information that is currently believed to be unimportant or if data believed to be difficult to obtain is, in fact, commonly gathered in other disciplines.

Purpose for Testing

In software development, testing is usually performed in or soon after development, in a controlled environment that simulates various specific usage conditions. Also, the purpose of the testing is to determine whether or not faults exist in the code and to locate them once detected. In contrast, most of the other disciplines surveyed use testing in the field or during production to diagnose known problems or to insure that the object being tested continues to work properly. The reasons for building testability into a system or process can provide insight as to how the results will (or can) be used. Evaluating the motivation for performing any process is valuable, because it offers the opportunity to change perspectives and question whether it is useful to ask additional questions or use new methods.

Process Contamination

One important factor to consider whenever a test is proposed is the effect that the sampling will have on functionality, efficiency, and quality. The benefits of, in this case, a method of testing must be compared to any error or reduction in efficiency that it introduces. Examining how many costs other disciplines are willing to bear in exchange for an increase in ease of testability is also important.

Integration of Testability into Design

Currently, testability is not a major issue in the software design process. This criterion evaluates the amount of emphasis placed on testability within the design processes of other disciplines. The motivations and benefits of integrating testability into the project and design are also examined.

CIRCUIT DESIGN

Circuits are, due to their mathematical nature as well as by virtue of being a platform on which software is designed and run, closely related to software. The process by which software is designed, however, has diverged significantly from the circuit design process. Part of this dichotomy can be explained by differences in the materials and tools being used (silicon and doping versus languages and compilers) or by the final product (a wafer that may contain physical defects, versus an executable that can only contain errors of logic or intent). However, with the popularity of computer-based circuit design programs and the increasing demand for embedded systems, these differences, merely accidents of design, are disappearing. The real reason for the difference, then, lies with the unique problems that circuit designers have faced.

Like software developers today, electrical engineers encountered inadequacies in their testing methods in the early 1970's. Their situation was remarkably similar to that facing software development now (Siddiqui 1). By examining the similar problems faced by circuit designers and the processes used to solve them, summarizing the principles and methods used to implement testability in circuit designs, and analyzing these solutions with respect to the criteria mentioned earlier, a process for solving the problems facing software development may be revealed.

Testability Problems Encountered in Circuit Design

Prior to the late 1970's, the primary aim of a circuit design project was to produce a functional piece of hardware with minimal complexity at minimal cost. Testing was performed solely with equipment external to the board. For example, the "bed of nails" approach involves the construction of a testbed that reads and manipulates the values of specific lines through physical contact between the line and a "nail" (probe). However, "the increasing complexity of new products and the proliferation of new . . . fabrication and packaging technologies . . . made testability a necessary product performance attribute" (Turino 1). In the case of the "bed of nails" approach, new fabrication techniques allowed for a density of gates and lines that required an impractical amount of precision in the best case and made many lines utterly inaccessible. To continue testing with a similar approach, test points (additional circuitry) had to be built into the design (Turino 37).

The difficulties encountered in testing mounted even as other facets of development became easier. The cost of testing rose even as hardware costs fell, and increasingly larger proportions of circuit development projects were involved in the testing process. In the end, the development of large-scale and very-large-scale integration (LSI and VLSI) created a testing "wall of 'intractability.'" This generated general interest in "'design for testability' techniques" and provided the impetus needed to make testability a design *criterion* instead of merely a phase in the development process (Fujiwara ix). With the aim of reducing monetary and schedule costs, engineers introduced methods to simplify the testability problem.

Summary of Testability Principles

"Testability is a system design characteristic." It must be considered as "a measure of the effectiveness of the system," as "designing for it allows for the recognition and location of failures" (Siddiqui 1). As such, to be effectively implemented, testability must be considered throughout the entire design cycle, and tradeoffs with other important criteria must be made to maximize the total benefits provided. Turino lists three principles for designing testability into a circuit: partitioning, controllability, and visibility (7).

Siddiqui adds another consideration: to be effective, testing procedures should require the "minimum human interaction and external tester support" (2). This becomes especially important in autonomous systems, which, by definition, operate independently as much as possible (Birk 1).

Of these four criteria, partitioning is the most familiar to the software developer. It refers to the practice of dividing a chain of logic into small or easily understandable blocks. This increases testability only if the inputs and outputs to each of these functions are easily accessible, which is addressed by the principle of visibility. This idea requires that important inputs and outputs be made easily accessible, often by adding extra structures. Finally, the principle of controllability refers to the manipulation of input variables to allow patterns for testing to be easily produced. By allowing input lines to be changed as needed, each function can be tested separately and verified to be correct (Turino 7-13). Then, using the idea of referential transparency, the entire circuit can be verified to be correct if each component part is correct and has been integrated properly into the whole (Meunier 2.2). The additional consideration, that the testing element of the system is built to require as little human interaction as possible, addresses the problem that trained, experienced personnel are becoming difficult to find. This idea requires a group of experienced engineers to design the system, but then less experienced technicians and engineers can, with the assistance of the automated testing mechanism, diagnose and repair problems as they are encountered (Siddiqui 6-7).

Results of Testing

The principles for testing in circuit design define a methodology that expects very specific data from multiple points within the circuit. Furthermore, the principle of controllability requires that some method for manipulating the inputs to any single function exists. In combination, this allows for the precise isolation of faults and a simple method for running specific test cases. The more conscientious the designers have been in implementing testability into the system, the more likely it is that any single fault can be identified using only the test equipment already built into the chip.

Purpose for Designing Testability

In circuit design, testing needs to be performed both in laboratory situations and in the field. The aim of circuit designers is to improve the ease of testability in both situations. The situation is further complicated because the goal of testing is different in both cases. Laboratory testing is normally conducted to detect faults and verify correctness immediately after development, and field-testing is performed, often by less trained personnel using less sophisticated equipment, to locate and repair known errors (Turino 3-7). However, in both cases, traversing all paths and isolating an error, once detected, are the two most difficult problems (Fujiwara 12-14). By designing testability into a system, these problems become easier, and fewer resources are required to solve them in a satisfactory manner.

Self-checking and status reporting are two motivations that are considered less often when implementing testability into a design. A less trained or experienced user can use a system that reports when it fails. Furthermore, a system that visually reports the result of some self-test (the memory check at the beginning of a computer's boot sequence, for example) raises user confidence in the system and may also increase monetary value. "Unless it is a built-in test that benefits the customer as well as the producer, test adds no value to a product; it just adds cost" (Turino 2). In addition to making testing by the developers and maintenance personnel easier and less expensive, designing for testability can be used to increase the scope of demand for a product and raise its market value (Turino 15-16).

Contamination Introduced by Procedures

Introducing additional hardware to implement testability in the design runs the risk of introducing new errors, and furthermore, the extra circuitry will affect the properties of the circuit (resistance, capacitance, etc.), which can detract from performance if not accounted for early in the design process. By considering the effect of these factors during the design process, the effects of this additional hardware on cost and speed can be minimized. Adding structures to increase testability to an already complete design is a much more expensive process in terms of performance, cost, and the risk of unintended side effects. In any case, designers agree that with the rapid advances in the speed and the decreasing cost of hardware, the loss of speed and higher cost incurred by designing testability into a circuit is negligible. Furthermore, the investment returns a high yield during design testing and later, during maintenance (Turino 2-3).

Integration of Testability into Design

Due to several spectacular failures (the error in the Pentium III, for example) and the increasing intractability of the testing problem, designing testability has become an accepted concept in circuit design, as exhibited by the paper topics at the International Test Conference. However, as evidenced by the lack of education in this area -- the Electrical Engineering department at the University of Tennessee does not address this topic in its undergraduate curriculum, for example -- it is still a developing idea and has not yet become a universal concept. Nevertheless, the idea of designing for testability, while often, in the past, merely an afterthought in the process, has become a much more critical phase in the past fifteen years and when utilized, is often one of the most important criteria. “The key lies in the *prevention* of testability problems” (Turino 3).

CHEMICAL ENGINEERING

The manufacture of chemicals requires both precision and careful control, to ensure the quality of the compound being produced and to minimize risks associated with its production. The need for these qualities is magnified in industry, since the large size of the batches being produced introduces problems with mixing and micro versus macro reaction issues as well as increasing the magnitude of any accident. The material being sampled complicates the testing issue, too. To physically test a large batch of chemicals, whether or not it is stable or in a reactive state, personnel are often placed at risk. This factor drives the need to consider testability in the design step: remote or automated test equipment must be built into any plan that involves the need for testability in a possibly hazardous process (Baasel 163-168).

Basic Tenets

In the design of a chemical process, engineers seek to divide the reactions into a series of independent steps. The condition of the batch before and after each step should be easily measurable, and each phase should be controllable, such that any unwanted reactions or products can be neutralized or removed without contaminating the entire batch. After partitioning the process, test equipment is chosen that can monitor the conditions of material entering and exiting the step, at least, and occasionally, in the middle of a phase. This equipment allows personnel to observe the reaction at the optimum times and to take appropriate action to keep selected "variables" (acidity, the concentration of a specific compound, hydration, etc.) within appropriate bounds (Baasel 163-164).

Results of Testing

Testing, in the domain of chemical engineering, provides increased visibility into the process being tested. In general, only estimates or interpolations can be generated, since only representative samples of the material can be tested and unless absolutely stable, mixtures of chemicals are often not homogeneous. Nevertheless, the data gathered is representative of the whole, and in situations where precision is absolutely critical and an accurate test is too expensive, "backup" test equipment can be activated to obtain more accurate measurements if initial tests indicate abnormal readings.

In addition to these limitations, each test can usually measure only one specific "variable." The ability to measure multiple variables in combination is not thought to be useful, as it may not provide specific enough information to make an informed decision (Baasel 164-167). However, many test devices are placed in combination with an optional input control device. If the values measured are found to be out of range, the device can manipulate the environment by reducing the flow or adding chemicals that will react with the sample being tested, to bring it within bounds. This may cause a safety or quality risk, however, so such devices often only recommend action, with human intervention required to actually cause it to occur (Baasel 101-107).

Purpose for Designing Testability

When building testability into a chemical process, chemical engineers are attempting to allow operators to easily monitor the state of the processes occurring within the chain of chemical reactions. In addition, they are providing a set of points at which corrective action can be taken to keep the reaction within safe and efficient boundaries. They believe that by designing testability into the system, they are significantly enhancing the reliability and controllability of the entire process.

Contamination Introduced by Procedures

Normally, chemical engineers select test equipment or procedures that introduce as little contamination as possible. However, almost by definition, because testing chemicals usually requires that some part of the material being tested react with some other compound, testing introduces some error. This error can compound as, later in the process, other testing equipment introduces more error. Therefore, it is important to carefully select equipment that introduces as little error as possible and to know an estimate of how much was introduced, so that it can be counteracted (Baasel 82-84).

However, minimally invasive procedures are often prone to false alarms. Therefore, redundant tests are built into critical phases, and a separate laboratory, whose expertise may be called upon when an alarm has been triggered, is often placed onsite. While sending samples to the laboratory for test will slow production, the data collected will be of higher accuracy. By paying a higher initial monetary cost, then, a high level of safety and efficiency can be produced without sacrificing too much purity or quality from any given batch.

Integration of Testability into Design

Partly for economic reasons and partly due to regulations, the testability of a process is an important design criterion for chemical engineers. Building testability into a system reduces production costs and increases reliability and safety. Furthermore, by integrating it into the development process, testing can be much less intrusive, and more of the testing in the production process can be automated. Because of these benefits, most chemical process design projects design testability into the process early in the development cycle and appropriate a fairly high portion of available resources to the effort.

FACILITY DESIGN

When designing a new building or complex, the final goal is to increase efficiency while reducing cost. The majority of the cost associated with a facility is not involved in its construction. Instead, it lies in the money spent to maintain the building and in the unexpected expenses incurred if an accident (error) occurs. Incorporating features that enhance the testability of the various systems (climate control, security, fire protection, and even maintenance management, among others) reduce both the risk of a major accident and the overall costs of maintenance.

Summary of Testability Principles

The introduction of testability into facility design is referred to as total facility control. This approach integrates data collection sensors and control points from all systems into a master control arrangement that includes monitoring and control of the building environment, as well as fire protection and security (Cherry 1). The aim of this integration is to provide an interface by which a small number of people can detect any problem, however small, and deal with it before it has a chance to become unmanageable. In addition, this system should, ideally, provide a mechanism for managing resources. Integrating data collection and monitoring equipment can simplify equipment or process control and energy management.

The entire process of designing testability into a facility plan reduces to three steps: selecting appropriate equipment, effectively placing the equipment, and concentrating control in one or more nodes. Equipment is selected based primarily on cost and the specific needs of the facility being built, which include such concerns as unobtrusiveness (or its opposite), the level of physical security required or expected, and the danger of fire.

Results of Testing

In facility design, testing is performed primarily to gather information: either to detect abnormal or unwanted situations (in the case of a fire or intrusion) or to provide basic data to improve efficiency or results (in the case of climate control, for example). This data is then, ideally, returned to a central location that either presents the operator with all of the information in a format that is easily digestible (a series of readouts, for example) or filters it and reports any abnormal conditions (in the case of an alarm system). The operator can also be provided with some way of manipulating the environment being tested, but in general, the structures used to allow for "ease of testability" do not, at the same time, act as tools for changing the environment being sampled.

Purpose for Designing Testability

Facility designers consider the issue of testability not only for economic reasons, but also because their goal is to build as efficient and safe a facility as is possible. By carefully selecting the structures designed into the system and centralizing control of the results, they have found that it becomes easier to maintain control and diagnose problems.

Contamination Introduced by Procedures

The testing equipment and processes used by building designers, in general, produce little or no interference. A fire alarm, for example, will probably not influence the work being performed daily at its location. Security devices and processes are a notable exception. Restricting access to an area produces some inconvenience, for example. Furthermore, designers must also worry about the impression caused by a given security device: a barbed wire fence may be acceptable at a factory, but it would be out of place at a downtown office building (Cherry 106).

Integration of Testability into Design

The fully integrated approach is not used in all facility design projects. In *Total Facility Control*, Cherry states that the total facility control process can be touted as a benefit that the competition does not offer in order to lure tenants into office space (3). He notes that the industry has not accepted the need for testability because, until recently, the cost of providing such control exceeded the benefits. The increasing power and decreasing cost of computers, he explains, have made this process cost-effective (Cherry 3-10). However, total facility control has yet to reach universal popularity in the facility design community, although its use has increased, and today, most new designs for large buildings and multi-building complexes utilize these ideas, at least in part, as they are recognized as being effective in reducing long-term costs and risk.

CONCLUSIONS

While each of the three disciplines examined implements testability into its designs in different ways with varying levels of commitment, each follows similar principles to reduce complexity and locate errors. The idea of centralizing control of all the test data, as championed in the domain of facility design, may be useful. In addition, the model adopted by circuit designers, due to the close mathematical relationship of circuits to software, is especially applicable. In this model, the principle ideas can be summarized by "partitioning, visibility, controllability" (Turino 7).

Software designers already use the idea of partitioning to reduce the complexity of their programs and to increase readability. However, criteria other than a focus on future testability are used to determine how the program is to be divided. To be truly effective, testability will have to be a, if not the, prime consideration when determining the scheme for dividing a program. The other three principles -- visibility, controllability, and centralized control -- can, arguably, be automatically added to any project with the proper use of a debugger such as gdb. However, the use of programs such as gdb is restricted to the laboratory, where the source code is available. As it will not be available in the production version, this does not provide any reassuring feedback for the end user or maintenance personnel (if this is a goal of the developers). Finally, freely available debugging programs such as gdb do not work on the multiple-component programs that are becoming much more common. Before new tools that overcome these problems can be generated, software developers will have to determine what features are worthwhile, and that can occur only after testability has been considered and developed into many software projects.

In addition to following similar principles, all of the disciplines examined implement testability into their designs for similar reasons. Chemical engineers and facility designers have found that their processes are more safe and controllable. Furthermore, practitioners of all three disciplines agree that, properly implemented, building testability into a design can actually lower not only maintenance, but also production costs, despite the additional expenditures required for the slightly longer design time and extra structures to implement the chosen functionality. With testing costs rapidly becoming a major expenditure in software development projects, I believe that software designers, contrary to expectation, will find that similar economic benefits can be realized. With the release of faster processors and the decreasing cost of storage, the other barrier to designing testability into software systems, efficiency, is minimized as well. The cost of the loss of a few clock cycles in any given function or a slightly larger program is negligible when combined with the advances in performance and cost offered by new hardware.

Therefore, I believe that it will be profitable for software engineers to integrate the idea of testability into the entire design process, as, in particular, circuit designers have done. The principles of partitioning, controllability, and visibility are, if not already an acknowledged part of "good programming practice," at least compatible with the idea. By using good practice to generate clean, understandable code and keeping the goal of developing a testable system in mind, I believe that the reliability of software can be increased while reducing the burden on testing.

WORKS CITED

Baasel, William D. Preliminary Chemical Engineering Plant Design. 2nd ed. New York: Van Nostrand, 1990.

Basili, Victor R. and Harlan D. Mills. "Understanding and Documenting Programs." IEEE TSE SE-8, No.3 (May 1982).

Birk, Andreas. "Autonomous Systems as Distributed Embedded Devices." 17 Aug. 1999 <http://arti9.vub.ac.be/~cyrano/AUTOSYS/>. 21 Oct. 2000.

Carlsgaard, Marlene. Email interview. 12 Oct. 2000.

Cherry, Don T. Total Facility Control. Boston: Butterworths, 1986.

Fujiwara, Hideo. Logic Testing and Design for Testability. Cambridge: MIT Press, 1985.

International Test Conference. International Test Conference. <http://www.itctestweek.org/itc2000.html>. 21 Oct. 2000.

Meunier, Jeffrey A. "Functional Programming." 19 Dec. 1995. <http://www.engr.uconn.edu/~jeffm/FuncProg/Papers/funcprog.html>. 14 Nov. 2000.

Prowell, Stacy. Class Lecture. Computer Science 525. The U of Tennessee, Knoxville. Oct. 2000.

Siddiqui, Uzair. Designing Testability into a Digital System. Knoxville: 1985.

Turino, Jon. Design to Test. 2nd ed. New York: Van Nostrand, 1990.

What is Bluetooth? InfoTooth. 5 Nov. 2000 <http://www.infotooth.com/whatis.htm>. 12 Nov. 2000.

APPENDIX 1.

APPLICATION OF CRITERIA TO THE CLEANROOM PROCESS

INTEGRATION OF DESIGN FOR TESTABILITY INTO THE PROCESS

Testability can be built into any system from the beginning of the Cleanroom process. It is my belief that the most important contributions to the testability of the system can be made first during the establishment of requirements and the definition of the stimulus and response sets and later during the transition between the black and state boxes. The first of these hypotheses was developed by examining projects performed by students in an introductory software engineering course that stressed the Cleanroom process. The system being developed was a bicycle computer with a fairly simple environment: a set of three interrupts (one clock interrupt and two interrupts produced by a button decoder) and several registers used to receive information from various sensors and to display responses on a screen.

REDUCTION OF OBSERVABILITY CAUSED BY ABSTRACTIONS

In these projects, the students were asked to perform usage model based "simulated" testing at two different times, with the goal of identifying a set of errors. Both times, several errors escaped identification, primarily because the usage model for the system -- which is based on the stimulus set and the black box -- was incomplete, in terms of stimuli accepted or responses returned. In many cases, stimuli that caused errors were purposefully left out of the stimulus set, as no behavior was associated with their application, or abstracted away. For example, no group differentiated between a short press of the button C and a long press (defined to be "greater than two seconds") of the button C. Instead, both possibilities were included in the stimulus "C." However, the hardware used differentiated between the two and applied a different stimulus to the system being tested. Similarly, several errors escaped detection because the response returned by the system included too much information. The response issued was correct, but during its computation, an error occurred. These are cases of abstractions that restrict observability: too much information is hidden. Abstractions in the stimulus and response sets need to be considered very carefully.

THE ADDITION OF STATES TO ENHANCE TESTABILITY

The other part of the process which might be used to effectively enhance testability is the transition from the black to the state box. Certain sequences of stimuli may not provide enough of a response to effectively determine if the function is being correctly performed. Among other cases, this may occur when a black box is embedded within another black box. By carefully using partitioning and adding responses to clarify events that occur, the testability of the system may be enhanced. This might involve breaking states apart in the state machine or requiring that each transition between states has an associated response.