1996

# Orthogonal defect classification applied to a multidisciplinary design

Alex Amezquita
*Carnegie Mellon University*

Daniel P. Siewiorek

# Orthogonal Defect Classification Applied to a Multidisciplinary Design

Alex Amezquita and Daniel P. Siewiorek

EDRC 05-100-96

## Abstract

**Orthogonal Defect Classification (ODC) is a tool used in software engineering to manage, control, and improve a design process. ODC enables administration and designers to gain feedback on the design during the development process. This paper provides a basis for expanding ODC from a single discipline (i.e. software) to a multiple discipline environment (i.e. software, hardware, and mechanical). This paper**

**- describes ODC and the multiple discipline environment;**

**- defines the structure of the attributes used in the extended implementation of ODC;**

**- provides the taxonomy for each attribute with definitions and examples; and**

**- presents preliminary data from two complete product design cycles.**

# 1. Introduction

## 1.1. Introduction to ODC

Orthogonal Defect Classification (ODC) is a methodology that extracts information and provides feedback about a development process from the defects that occur during the design cycle. ODC assists in-process decisions by measurement and analysis techniques. There are many tools that attempt to assist in-process decisions ranging from statistical defect models to causal analysis [Chillarege and Bhandari, 1992; Chillarege,1994].

Statistical defect models use counting techniques and growth curve modelling to predict product maturity. When using statistical defect models, all defect types are treated equivalently and do not give the design team insight to remedy design process failures.

Causal analysis thoroughly explores all the details of a small quantity of defects. This technique is very time consuming, expensive, and does not necessarily capture a wide enough range of defect types.

In the middle of these two extremes, ODC possesses the benefits of both techniques. ODC provides feedback with a high level of detail from a wide range of defects, while still taking advantage of inexpensive implementation costs and a broad range of mathematical tools to perform complex analysis [Chillarege and Biyani, 1994]. ODC tracks attributes of the defects during the entire development process. Under each defect attribute there is a list of categories that represent situations that may occur. These categories are qualitatively disjoint to each other, meaning

there is one and only one category that will describe each defect that is discovered. By tracking the distributions of the categories under each attribute over time, signatures occur for each product. Similar products and different generations of the same product will have very similar signatures. Management can use information from these signatures to gain insight to the design cycle and how to efficiently execute corrective actions to failures in the development process. Also note that since similar products have similar signatures, a single ODC taxonomy can be used for a number of products and generations.

ODC is currently being used at IBM Watson Research with substantial success [Lyu and Chillarege, 1996]. Using ODC, IBM has achieved cost reduction over causal analysis by a factor of ten, cycle time reduction by a factor of three, and defect reduction by a factor of eighty over a period of five years. This paper expands on the idea used at IBM, by using ODC in a multidisciplinary environment.

### 1.2. Environment of EDRC

The Engineering Design Research Center (EDRC), located at Carnegie Mellon University, is composed of designers from a wide range of expertise. It is this quality that makes EDRC an excellent test case for multidisciplinary ODC. The wearable computer project [Smailagic et al., 1995] has evolved through seven generations of design at EDRC. Wearable computer design includes electrical, computer, software and mechanical engineering; human computer interaction, and industrial design. These disciplines range from quantitative, structured design to qualitative, abstract design.

1.3. Introduction to the Wearable Computer

A wearable computer includes a computer worn on the body and a display that is attached to a band that is placed on the head [Smailagic and Siewiorek, 1996]. One generation is shown in Figures 1 and 2. The computer contains the processor, memory, and PCMCIA slots that are used for a variety of options (speech recognition hardware, wireless LAN, additional memory, etc.). In some designs the input device is part of the main processing section, and in other designs the input device is attached via a cable. A docking station stage enables the wearable computer to upload information it has recorded into a database on a desktop computer for processing. During the docking stage, the unit will also recharge its batteries and undergo any maintenance if necessary.

**Figure 1** (above) displays a wearable computer in the docking station stage. The device in the top center of the screen recharges the batteries and transfers information. The head mounted display is shown in the top right portion of the picture. The center piece of the picture is the computer with an input device (the dial) attached to it. **Figure 2** (left) shows a wearable computer in operation. The computer is strapped to the users waist and the display is located over the left eye. The input device is a dial that is being used with the left hand.

### 1.4. Goals

A design process involving many disciplines can become very difficult to manage efficiently. ODC may help control and improve the development process of wearable computers as well as other products involving multiple disciplines. The robustness of ODC is a promising feature in an environment that possesses such a wide range of disciplines and defects. The goals for implementing ODC in the wearable computer project are two-fold: 1) to guide in-process decisions enabling more efficient use of resources, a decrease in design cycle time, and a decrease in failures seen by the end-user and 2) to analyze signatures of the defect distributions to forecast the maturity of the product.

The next section describes an extended, multidisciplinary ODC while Section 3 defines the categories on each dimension of the ODC structure. Section 4 illustrates the classification of actual wearable computer design defects. Section 5 presents data from two wearable computer design cycles and Section 6 briefly describes and comments on the software tool used to implement multiple discipline ODC.

## 2. ODC Structure

### 2.1 ODC Terminology

In ODC, characteristics of a defect, defined as an **attribute,** are recorded. There is a list of **categories** that span the set of all possible descriptions for each attribute. An attribute may have categories that have similar qualities and therefore are placed in **groups.**

### 2.2. Organization of Defect Attributes

This approach uses seven defect attributes: **phase found, defect type, defect class, trigger, source, impact,** and **environment** [Bhandari et al., 1994]. The first attribute, **phase found,**

describes the stage of the development process when the defect is discovered. The stage when the defect is found will not necessarily be the stage where the defect was introduced. The second attribute, **defect type,** identifies what needs to be corrected in the product in order to eliminate the problem. The list of categories under defect type is broken down into four groups: software, hardware, industrial and mechanical design, and general. The third attribute, **defect class,** illustrates if the discovered defect was incorrectly addressed or if the defect was due to an oversight. Next, the **trigger** attribute describes the action that led to the discovery of the defect. Under the trigger attribute, there are three group headings corresponding to three phases of the design cycle: review and inspection, unit test, and system and field test. Triggers under the review and inspection group are selected based on what the designer was thinking about during the inspection of the design. The type of stimulus being applied to the unit is the basis for categorizing triggers in the unit test group. Third, triggers in the system and field test group represent the environment that the product is being subjected to when the defect surfaces. The fifth attribute, source, states the part of the product in which the defect resides. The sixth attribute, **impact,** describes what effect the defect would have on the user if it had not been discovered and the last attribute, **environment,** specifies the setting that the defect will affect.

The addition of attributes is feasible. For example, a phase introduced or component/subsystem attribute may also provide useful insight. However, it is equally important to have a structure that provides the user and the analyst with a tool that can be used efficiently. The addition of attributes will be more time consuming for both the user and the analyst and may not provide insight that is worth the extra time or effort. Also, a structure that is very large in nature may be intimidating to a user and less productive to an organization that is implementing ODC for the first time. The structure of this implementation of ODC was chosen to be quick, efficient, and

thorough enough to gain accurate feedback to the development process. It is possible to implement ODC with a different structures. The next section defines the categories for each of the attributes used in this structure.

## 3. ODC Taxonomy Defined

This section is based upon the IBM ODC taxonomy [Chillarege and Bassin, 1995; Laprie, 1985; Lyu and Chillarege, 1996]. Enhancements to expand the structure to multiple discipline design are indicated by an asterisk. The illustrative examples actually occurred during the design cycle of two generations of wearable computers.

### 3.1. PHASE FOUND

**Table 1: Phase Found Categories**

| Phase Found |
|---|
| Configuration |
| Detailed Design |
| Integration |
| Operation |

Configuration - This stage incorporates the time when the product is conceived to the time when the conceived product is broken down into all of its separate blocks. The necessary requirements and specifications for each block are determined.

Detailed Design - During this stage, the design, construction, and implementation of each block is realized according to the specifications of the configuration phase.

Integration - The integration phase is when the separate blocks merge into one functioning unit. The merging of separate blocks may or may not be of the same discipline.

Operation - Usage of the product in the field for final testing and usage by the client.

3.2. DEFECT TYPE

Table 2 summarizes the defect type attribute. The table is organized so that similar defect types across groups are place in the same row. Also, the table is arranged top to bottom from abstract to physical.

**Table 2: Defect Type Attribute**

| Group | | | |
|---|---|---|---|
| Hardware | Software | Industrial/Mechanical | General |
| Function | Function | Perceptual | Problem Solution |
| Interface | Interface | Interaction | |
| | Algorithm | | |
| Logical | | | |
| Timing | Timing/Serialization | | |
| | Assignment | | |
| Schematic | Syntax | Translation | |
| | | Assembly Design | |
| | | Human Factors | |
| Physical Design | Build/Package/ Merge | Physical Design | |
| Faulty Component | | | |
| | | Thermodynamics | |
| | | Durability | |
| | | | Documentation |

Hardware

•Function - Defect that requires a formal design change because it affects significant capability, end user interfaces, product interface, or interface with software architecture. Examples are: a pull-up resistor is missing on an input/output port, a chip is specified in the design

but it is unavailable to the design team, or a different version of a processor was installed that did not function with the current software.

•Interface - Communication problem between a component and another component or programming tool. Examples are: An EPROM can not be properly programmed with the current device, the tolerance of a 300 Baud data rate may be 5% and the deviation is actually 10%, or a receiver requires parity and the data contains no parity.

*Logic Fault - Incorrect signal level on a particular net. Examples are: A reset line is at a logic low when it should be at a logic high, or a control line needs to be either a logic high or a logic low and it is left in an undefined state.

•Timing - Defect related to the timing of a signal. Examples are: A signal has a rise time, fall time, or hold time that causes an error, the data bus is accessed before it is ready, or the dynamic RAM is refreshed at too slow of a rate.

•Schematic Design - Defect in the schematic of the design. Examples are: connection made to the anode of a diode when it should have been the cathode, a connection between two signals was never made and it should have been, or power is connected to the wrong pin number.

•Physical Design - Defect in the physical design of the printed circuit board (PCB) or hardware component. Examples are: a connection was not made on the PCB and it was on the schematic, the pad size on the PCB is too small for the chip to be soldered, cold solder joint, solder bridge, a broken wire, or connection was made to a pin that was different from the schematic.

•Faulty Component - The component is defective and does not function as expected.

**Software**

**Function - Defect that will require a formal design change because it affects significant capability, end user interfaces, product interfaces, interface with hardware architecture, or global data structures. Examples are: end user application does not include vital information, or application does not save state in the event of an unexpected shut down.**

**Interface - There is a communication defect between a certain portion of code and another portion of code or software tool. Examples are: a call statement includes the wrong parameters, a function calls another functions and gives the data in an inappropriate format, or a macro is used incorrectly.**

**Algorithm - Defect in the correctness or efficiency to perform a particular task. The defect can be fixed by changing an algorithm or local data structure without the need for a design change. Examples are: a linked list was used instead of an array, or the display required bytes to be given horizontally line by line and the picture was given vertically.**

**Timing/Serialization - Defect in the sequence of using resources. Example: the processor's transmitter is used before it is initialized.**

**Assignment - Assigning the incorrect value or the absence of assigning a value to a variable. Note that multiple assignment defects may be an algorithm defect. Examples are: a variable is assigned to the incorrect variable, a constant is given the wrong value, or a failure to initialize a variable correctly.**

**•Syntax - A language defect in the code. For example, an integer type is printed when it should be a character type.**

**Build/Package/Merge - Problems encountered during the driver build process, in library systems, or with management of change or version control. Examples are: functions that are used**

**are missing from a library file, or an application program is written using language updates but the compiler uses an older version of the language.**

**Industrial and Mechanical Design**

**•Perceptual - The product does not have the correct feel, vision, or sound. Defects in this category pertain to the senses of the user. Examples are: the input device makes a sound that gives the illusion of a cheap product, the head band feels like it is going to break, or the form of the product does not portray the intended vision.**

**•Interaction - Defects due to misunderstanding or lack of understanding how to use the product. Also, the inability to use the product efficiently. Examples are: the input device is difficult to use, the options to select on the display are confusing, entering data into the unit is too laborious, it is not clear where to attach the docking station, or the user is not sure if there was a response when pressing the power button.**

**•Translation - A defect occurs due to a change in the design because information is interpreted differently from one person or machine to another. Examples: a manufacture interprets the design plans differently than intended from the specifications, transferring an idea from foam board to a CAD drawing causes an error in shape, a curve is changed when transferring data from one CAD program to another, or a design plan is altered due to language misinterpretation from one discipline to another.**

**•Assembly Design - Defects related to the design of the assembly or disassembly of the product. Examples are: the connector for the docking station is difficult to attach and remove, replacing batteries is troublesome due to the compartment lid, or a part on the PCB was not given enough clearance for the unit to close.**

*Human Factors - The user has certain anatomic characteristics that do no allow convenient or satisfying use of the product. Examples are: the processing section of the unit will not fit on the back of users under 5'6", or the input devices can not be used with large hands.

•Physical Design - A defect found in the actual assembly mechanism or material used in the product. Examples are: a screw has the wrong thread size, the plastic cures differently than expected, or the material alters the color differently than expected,

•Thermodynamics - Defects caused by thermal management of the product. Examples are: the heat sink is rated for 5 W and the unit needs 7 W, or the processor can not operate over 90 degrees F and the temperature is 100 degrees.

•Durability - Defects related to the lifetime of the product. These defects may be related to the strength, shock resistance, water resistance, etc. of the product. Examples are: the unit cannot take a drop over 3' and it is used in high places, or the unit is not water resistant and it is used in maritime situations.

General

•Problem Solution - The defect occurs because a component is in a situation that the specification of the design does not solve. The omission of the design specification may be intentional or unintentional. Examples are: It is assumed that the batteries would not be charged for over three hours but there is no way of shutting off the battery charger after three hours, or the application program jumps to a part of code that was not expected.

•Documentation - The problem is with the written description contained in user guides, installation manuals, or comments. Examples are: the user guide give instructions that are incorrect, a help screen is displayed with incorrect information, a software designer

reviews some code and the comments do not explain the function correctly, or information on a particular feature is missing.

## 3.3. DEFECT CLASS

**Table 3: Defect Class Categories**

| Defect Class |
|---|
| Omission |
| Commission |

Omission - The defect was never addressed in the conception of the component or product. To correct the defect, additional consideration may be needed in the design specifications.

Commission - The cause of the defect was addressed but not properly managed. In order to correct the defect, existing design plans must be readdressed.

## 3.4. TRIGGER

**Table 4: Trigger Categories**

| Group | | |
|---|---|---|
| Review and Inspection | Unit Test | System and Field Test |
| Design Conformance | Simple Test Coverage | Normal Stress |
| Defect Fix | Complex Test Coverage | High Stress |
| Backward Compatibility | Side Effect | Side Effect |
| Lateral Compatibility | | |
| Understanding Details | | |

Review and Inspection

Design Conformance - Comparing the design element with its specifications.

**Defect Fix - Defect discovered while correcting another.**

**Backward Compatibility - Project member's product knowledge of an earlier version detects an incompatibility.**

**Lateral Compatibility - Project member's product knowledge of another system with which it must operate detects an incompatibility.**

**•Understanding Details - Defect found while examining and comprehending the specifications of a component.**

**Unit Test**

**Simple Test Coverage - The test case that found the defect was a straightforward attempt to exercise a single function with a single input.**

**Complex Test Coverage - The test case that found the defect was a straightforward attempt to exercise a single function with many inputs.**

**Side Effect - Defect found because of some unanticipated behavior which was not the reason for the test.**

**System and Field Test**

**High Stress - The system is being tested under conditions that are pushing the resource limits.**

**Normal Stress - System is being tested under normal conditions well within its resources.**

**Side Effects - Defect found because of some unanticipated behavior which was not the reason for the test.**

3.5. SOURCE

**Table 5: Source Categories**

| Source |
|---|
| Vendor Software |
| New Software |
| Reused Software |
| Vendor Hardware |
| New Hardware |
| Reused Hardware |
| Physical |

Vendor Software - The defect was introduced by code written by a vendor.

New Software - The defect was introduced by code designed for current product.

Reused Software - The defect was introduced by code designed for earlier version.

•Vendor Hardware - The defect was introduced by hardware designed by a vendor.

*New Hardware - The defect was introduced by hardware designed for current product.

*Reused Hardware - The defect was introduced by hardware designed for earlier version.

•Physical - The defect was introduced by the mechanics or form of the product.

3.6. IMPACT

**Table 6: Impact Categories**

| Impact |
|---|
| Critical Defect |
| Performance |
| Reliability |
| User Guide |
| Integrity |
| Lost Objective |
| Usability |
| Maintainability |

•Critical Defect - The product will not perform the intended application.

Performance - The product performs all functions correctly, but not as efficiently as expected.

Reliability - The ability to consistently provide a service without unplanned interruption.

User Guide - Aspects of the product are not correctly represented in the documentation.

Integrity - Data or software that is held in memory and/or transferred during docking stage may be erroneous from inadvertent or malicious means.

*Lost Objective - Defect that does not allow the product to meet a design goal.

Usability - The defect causes the product to be hard to understand or inconvenient to use.

Maintainability - The ease with which the product can be serviced.

3.7. ENVIRONMENT

**Table 7: Environment Categories**

| Environment |
|---|
| Field |
| Docking |
| Both |

*Field - The product is being used for the utility it was designed.

•Docking - The product is back from the field and is either recharging or transferring data.

*Both - The defect will affect the product in both docking and the field.

# 4. Examples of Defect Classification

The definitions of the taxonomy in Section 3 give the designer basics to classify any defect that may occur. However, there are situations that may seem difficult to classify. The difficulty in assigning certain defects to one specific category may come from the abstract nature of the early phases of design and human computer interaction. Most likely, hardware, software, and mechanical engineering defects can be clearly classified without much deliberation. However, when dealing with the conceptual aspect of design, classification of defects may require more thought. This section is dedicated to clarifying the differences between categories by examining some examples in detail.

**Example 1.** At a demonstration, a new docking station (See Figure 1) is left to charge the batteries overnight, when the documentation states that the batteries should not be left charging for over three hours. As a result, the batteries leak acid and destroy the unit. The phase that the defect is discovered is the **operation** phase. The defect type is **problem solution** under the gen-

eral group. Problem solution is the correct category because the design does not address what happens if the unit is docked for longer than three hours. The design solution assume that instructing a user not to charge the batteries for over three hours insures that it will never happen. The team that derived the solution may have considered this situation and neglected to design for it because of time, resources, or expense. The defect class is **commission** because the situation was considered but not properly managed. The trigger is **high stress** under the group system and field test. The trigger is **high stress** because the product was operated under conditions that exceeded its resource limits. The source of the defect is **new hardware** because the docking station design is new for this generation. The impact of the problem is **critical defect.** Finally, the environment that the defect surfaced is the **docking** stage. If the scenario changes slightly to the batteries leaking acid after charging for two hours with no damage to the unit (proper operation of the docking station), we have very different results. The phase found, source, and environment attributes remain the same. The defect type is **faulty component** because the batteries did not behave according to expectations. The defect class is **omission** due to the lack of design for this situation. Since the batteries were being charged within their resource limits, the trigger is **normal stress** under system and field test. The impact on the product is under the **maintainability** category due to the additional effort to service the unit.

**Example 2.** The receptacle for the display cable connector is mounted on the printed circuit board. While closing the housing around the printed circuit board, a wire strand in the display cable breaks. The receptacle is close to the housing, causing a sharp bend in the cable, and leading to the wire snapping. The phase found for this defect is **integration.** The defect type for this problem may be under three categories: **physical design** under the hardware group, **physical design** under the industrial and mechanical design group, or **assembly design** under the industrial

and mechanical design group. If the cable connector or receptacle was not the size that the product was designed for, we have a **physical design** defect under the industrial and mechanical design group and a defect class of **commission.** If the size of the equipment was correct according to the design specifications, but the proper clearance needed was miscalculated, the defect type is **assembly design** with a defect class of **commission.** If the size of the equipment was correct, but the proper clearance was never considered, we still have an **assembly design** defect, but the defect class is **omission.** Finally, if the equipment is the right size, and the correct clearance was calculated, but the receptacle was not placed in the proper area on the printed circuit board, the defect type is **physical design** under the hardware group with a defect class of **commission.** The trigger is **design conformance.** The source is **new hardware** if the receptacle was placed incorrectly or **physical** if the equipment size was wrong or improperly considered. The impact is a **critical error** because the display will not work which yields the unit dysfunctional. The environment that is affected is the **field.** This is because the display is only needed in field use and not during the docking stage.

**Example** 3. While testing the input device, a user has difficulty manipulating the dial (See Figure 2) that is used to select various options. The phase found is **detailed design,** the trigger is **simple test coverage** under unit test, the source is **physical,** the impact is **usability,** and the environment is **field.** The defect type attribute may be of **human factors** or **interaction.** If the user has difficulty because the dial is an incompatible size for the hand, the defect type is **human factors.** However, if the user is having difficulty manipulating the dial because it is unclear when the dial activates a command, there is ambiguity in using the dial to select an option, or it is just unclear how to use the dial, the defect type is of the **interaction** category. The defect class would be selected based on if the input device team considered the defect or not in the design plan. If the

user did not have trouble using the dial, but did not like using the dial, the defect type is in the **perceptual** category. Possible reasons the user may not have liked the dial are: the dial made a disturbing sound when it turned, or the texture of the dial did not feel pleasant.

**Example 4.** This example is intended to clarify the differences between a few of the categories under the impact attribute. There exists a defect that affects the display. If the defect does not allow the display to ever operate, this is a **critical defect.** If the defect allows the display to operate normally, but occasionally the screen freezes for ten seconds at a time, this is a **reliability** defect. If the defect causes the display to use four times more power than needed, the defect impacts the **performance.** If the defect disables part of the screen that contains the battery monitor, the defect causes a **lost objective.** If the defect causes the display to bother a user's eyes, the impact is under the **usability** category.

## 5. Preliminary Data

Through an exhaustive interview process of designers that worked on the past two generations of wearable computers, a data set of defects were compiled. Both generations produced a single data set of approximately eighty defects. This section will examine the distributions of these defects with respect to each attribute. The appendix lists the complete set of defects.

Two of the main goals of analyzing this data set are to be sure this implementation meets the necessary and the sufficient condition of ODC. The two conditions must be met in order for ODC to produce valid information [Chillarege, et al. 1992]. The necessary condition states that there must be a classification of defects such that its distribution, as a function of process activities, changes as the product advances through the process. The sufficient condition states that the categories under an attribute must describe any situation that can occur under that attribute

space. As the data was collected and processed through a software program, it was clear that this implementation of ODC met the sufficient condition. Through the entire data entry process, there did not exist an example of a defect characteristic not being able to be specifically described by an attribute category. The necessary condition is also met for this implementation of ODC. This can be verified by examining the distributions of each defect attribute over the development process below. The time line of the development process is derived from the phase found attribute.

In the following tables (Table 8-13), the distribution of the categories under each attribute are given as percentages (total may not sum to 100% due to rounding). They are followed in parenthesis by the quantity of defects filed in that category.

5.1. DISTRIBUTION OF DEFECT TYPE ATTRIBUTE

**Table 8: Defect Type Attribute**

| | Stage | | | |
|---|---|---|---|---|
| Defect Type | Configuration (12) | Detailed Des. (27) | Integration (18) | Operation (25) |
| Logic Fault | | 4% (1) | | |
| Faulty Component | | 11% (3) | 11% (2) | |
| Hardware Interface | 17% (2) | 11% (3) | | 4% (1) |
| Schematic Design | 17% (2) | 19% (5) | 17% (3) | 4% (1) |
| Hardware Physical Design | 33% (4) | 19% (5) | 44% (8) | 16% (4) |
| Assignment | | 15% (4) | | |
| Software Interface | 8% (1) | | 11% (2) | 4% (1) |
| Assembly | 8% (1) | 4% (1) | 11% (2) | 16% (4) |
| Mechanical Physical Design | | 4% (1) | 6% (1) | 8% (2) |
| Documentation | | | | 12% (3) |
| Problem Solution | | 4% (1) | | 28% (7) |
| Other | 17% (2) | 11% (3) | | 8% (2) |

The distributions of the characteristics in the defect type attribute meet the necessary condition. Table 8 shows the change in distributions through the process development. The distributions of the characteristics also follow a very logical pattern. For example, the assembly design attribute has a distribution that increases as the process advances. This is because as the development process continues, a greater number of components are merged together to make a functioning unit. The most assembly design errors occur during the operation stage where the unit is put together for system and field tests and then disassembled when a defect occurs. Also note the faulty component characteristic. Faulty components defects are most prevalent when subsystems are being put together and tested. Also, notice the problem solution characteristic shows a large increase in the operation phase. This is because situations arise from field tests that were never thought of during the design of the product.

When this implementation of ODC is used during an actual development process, there will be a much larger set of defects for analysis. However, even from this small data set, the signature of the defect type attribute can be derived as can the rest of the attributes.

## 5.2. DISTRIBUTION OF DEFECT CLASS ATTRIBUTE

**Table 9: Defect Class Attribute**

| | Stage | | | |
|---|---|---|---|---|
| Defect Class | Configuration (12) | Detailed Des. (27) | Integration (18) | Operation (25) |
| Omission | 25% (3) | 67% (18) | 61% (11) | 40% (10) |
| Commission | 75% (9) | 33% (9) | 39% (7) | 60% (15) |

The defect class attribute meets the necessary condition due to the fluctuation of the distributions with respect to design phase. These distributions are logical if the design process is con-

sidered. There is a small amount of omission errors at the beginning of the design process because any defect that occurs will be due to new ideas. It is not until the detailed design phase that omission defects will surface. It is also logical to think that omission errors will dominate the detailed design and integration stages. When working on a project, many defects occur that are not discovered until subsystems are actually designed and made to work with other subsystems.

## 5.3. DISTRIBUTION OF TRIGGER ATTRIBUTE

**Table 10:** Trigger **Attribute**

| Trigger | | Stage | | | |
|---|---|---|---|---|---|
| | | Configuration (12) | Detailed Des. (27) | Integration (18) | Operation (25) |
| Review and Inspection | Design Conformance | 17% (2) | 33% (9) | 33% (6) | 12% (3) |
| | Defect Fix | | 11% (3) | 11% (2) | |
| | Backward Compatibility | 33% (4) | 4% (1) | | |
| | Lateral Compatibility | | 4% (1) | | |
| | Understanding Details | 50% (6) | 11% (3) | 6% (1) | 12% (3) |
| Unit Test | Simple Test Coverage | | 22% (6) | 17% (3) | |
| | Complex Test Coverage | | 4% (1) | 11% (2) | 8% (2) |
| | Side Effect Unit Test | | 11% (3) | 11% (2) | 4% (1) |
| System and Field Test | High Stress | | | | 12% (3) |
| | Normal Stress | | | 11% (2) | 24% (6) |
| | Side Effect Field Test | | | | 28% (7) |

The trigger attribute distributions change very drastically with design phase. This quality adheres to the necessary condition of ODC. It is also reasonable to think that review and inspection triggers will be high during the beginning to middle of the development process, unit test trigger will be frequent during the middle to end of the design cycle, and system and field test will be present at the end of the process. Also note the high concentration of system test side effects at

the operation phase. This is very sensible since many defects will occur during operation of the unit that are not expected.

## 5.4. DISTRIBUTION OF SOURCE ATTRIBUTE

**Table 11: Source Attribute**

| | Stage | | | |
|---|---|---|---|---|
| Source | Configuration (12) | Detailed Des. (27) | Integration (18) | Operation (25) |
| New Software | 17% (2) | 19% (5) | 11% (2) | 24% (6) |
| New Hardware | 33% (4) | 44% (12) | 56% (10) | 40% (10) |
| Reused Hardware | 50% (6) | 26% (7) | 11% (2) | 12% (3) |
| Physical | | 11% (3) | 22% (4) | 24% (6) |

The source attribute meets the requirements of the necessary condition. Table 11 illustrates the varying distributions over the process cycle. When designing with reused hardware, only a few changes need to be made so that there is compatibility with the current product. Generally, after the first few changes the reused hardware operates with little trouble. This characteristic is clearly illustrated in the distribution of the reused hardware characteristic in the source attribute. There will be very few defects with the form or physical structure of the product until it is assembled and tested in the field. This distribution trait is also exemplified in the physical category.

## 5.5. DISTRIBUTION OF IMPACT ATTRIBUTE

**Table 12: Impact Attribute**

| | Stage | | | |
|---|---|---|---|---|
| Impact | Configuration (12) | Detailed Des. (27) | Integration (18) | Operation (25) |
| Critical Error | 33% (4) | 33% (9) | 39% (7) | 12% (3) |
| Performance | 17% (2) | 11% (3) | | 4% (1) |
| Reliability | | 7% (2) | 28% (5) | 24% (6) |
| Lost Objective | 33% (4) | 41% (11) | 6% (1) | 36% (9) |
| Usability | | 7% (2) | | 24% (6) |
| Maintenance | 17% (2) | | 28% (5) | |

The impact attribute meets the necessary condition due to the variations in the statistics of the categories. The critical error characteristic illustrates a very logical distribution. There are high levels of critical errors until the operation phase. This declination in critical defects is natural since it is assumed that the product is operating with enough functionality to perform system and field tests. Also, the usability characteristic shows an expected distribution. Defects that pertain to critiquing the ease of use of a product will not happen until the product is functional.

## 5.6. DISTRIBUTION OF ENVIRONMENT ATTRIBUTE

**Table 13: Environment Attribute**

| | Stage | | | |
|---|---|---|---|---|
| Environment | Configuration (12) | Detailed Des. (27) | Integration (18) | Operation (25) |
| Field | 33% (4) | 22% (6) | 28% (5) | 36% (9) |
| Docking | | 11% (3) | | 40% (10) |
| Both | 67% (8) | 67% (18) | 72% (13) | 24% (6) |

The environment attribute meets the necessary condition and also possesses a logical pattern. Defects occurring in the docking stage category are very uncommon until the operation phase. This is valid since docking stage defects will not occur until the unit is mature enough to be able to go out in the field and collect data. Also, the characteristic that represent both environments has a sharp reduction in the operation phase. This reduction is related to the docking stage characteristic in that defects that are discovered in the operation phase are representative of a product that is nearing full maturity. Defects that occur when a product is close to maturity will be very specific to certain environments.

# 6. ODC Implementation

## 6.1. Software Implementation

The analysis of the data in Section 5 was aided by a custom designed software tool coded in standard ANSIIC for a UNIX operating system. The software tool records the category selection of each of the seven attributes for a given defect. This data is transferred to a central database where analysis may be performed. Each entry is also given a unique identification number when submitted to the database.

The end user interface is an all text display. A list of categories for an attribute is displayed and an end user inputs a letter corresponding to the appropriate category. The user may also go to a help section that provides the definitions for the current attribute. After each selection, the list of categories for the next attribute is displayed and the process continues. After the seventh attribute, a review of the defect entry is displayed and confirmation is requested. If the defect entry is confirmed positively, the entry is transferred to a database, and if the defect is con-

firmed negatively, the entry is discarded.

### 6.2. Implementation Comments

For designers and team members on a product to effectively use this tool, it must not be time consuming or an annoyance. When summarizing data from post process interviews, the data entry process was very quick and effortless to use. Once the end user is familiar with each attribute and the categories that are listed under each attribute, an average defect entry of 45 seconds to a minute was achieved.

The quantity of defects gathered for this paper was relatively small compared to the amount of defects that actually occurred. This is due to the small amount of accurate details about a defect that each interviewed designer could recall. ODC will capture significantly higher quantities of defects when it is implemented in conjunction with the design process. This is because an easy to use tool will be more conducive to capturing defects while they are fresh in the mind of the designers.

## 7. Conclusion

ODC is a tool that can help an organization rapidly acquire feedback from a design cycle. This feedback helps designers and management make in-process decisions that can reduce the length of the design cycle, use resources more efficiently, and reduce the amount of failures seen by the end user. Growth curve modelling and forecasting can also be incorporated into ODC for additional insight. A tool for implementing ODC in a multidisciplinary design course has been implemented. Preliminary results of this implementation indicate that this tool meets all the requirements for ODC to function in a multiple discipline design. These initial results also provide hope that incorporating multiple discipline ODC in the wearable computer project may provide the same success as seen by IBM Watson Research.

## Appendix

This list of defects is generated from post process interviews of designers from two generations of wearable computers. Each defect possesses a unique defect number. Abbreviations are used for the phase, defect class, source, and environment attribute. The abbreviations correspond to the category title under that attribute. Abbreviations are: Con - Configuration, DD - Detailed Design, Int - Integration, Op - Operation, O - Omission, C- Commission, NS - New Software, NH - New Hardware, RH - Reused Hardware, P - Physical, D - Docking, F - Field, B - Both.

**Table 14: Source List of Defect**

| No | Phase | Defect "type | Class | Trigger | Source | Impact | Env. |
|---|---|---|---|---|---|---|---|
| 1 | Con | Timing | O | Under. Detail | RH | Critical Error | B |
| 2 | Con | Hard. Physical | C | Under. Detail | RH | Lost Objective | B |
| 3 | Con | Schematic Design | C | Back. Compat. | RH | Lost Objective | F |
| 4 | Con | Hard. Interface | C | Design Conf. | NH | Critical Error | B |
| 5 | Con | Soft. Interface | C | Back. Compat. | NS | Critical Error | B |
| 6 | Con | Hard. Physical | C | Under. Detail | NH | Maintenance | F |
| 7 | Con | Assembly | O | Under. Detail | RH | Maintenance | F |
| 8 | Con | Hard. Physical | C | Back. Compat. | NH | Performance | B |
| 9 | Con | Soft. Function | O | Under. Detail | NS | Performance | B |
| 10 | Con | Hard. Interface | C | Back. Compat. | RH | Critical Error | B |
| 11 | Con | Schematic Design | C | Design Conf. | NH | Lost Objective | F |
| 12 | Con | Hard. Physical | C | Under. Detail | RH | Lost Objective | B |
| 13 | DD | Hard. Interface | 0 | Under. Detail | NH | Performance | F |
| 14 | DD | Hard. Interface | O | Simple Cover. | NH | Reliability | F |
| 15 | DD | Logic | O | Complex Cov. | NH | Performance | F |
| 16 | DD | Mechanical Phys. | O | Design Conf. | P | Lost Objective | B |
| 17 | DD | Perceptual | O | Design Conf. | P | Lost Objective | B |
| 18 | DD | Hard. Physical | C | Design Conf. | NH | Usability | D |

**Table 14: Source List of Defect**

| No | Phase | Defect Type | Class | Trigger | Source | Impact | Env. |
|---|---|---|---|---|---|---|---|
| 19 | DD | Faulty Comp. | O | Defect Fix | RH | Usability | B |
| 20 | DD | Faulty Comp. | O | Under. Detail | RH | Reliability | B |
| 21 | DD | Problem Solution | C | Simple Cover. | RH | Performance | F |
| 22 | DD | Schematic Design | C | Simple Cover. | NH | Critical Error | B |
| 23 | DD | Schematic Design | 0 | Simple Cover. | NH | Critical Error | B |
| 24 | DD | Soft. Function | C | Design Conf. | NS | Lost Objective | D |
| 25 | DD | Assignment | C | Simple Cover. | NS | Critical Error | B |
| 26 | DD | Assignment | 0 | Design Conf. | NS | Critical Error | B |
| 27 | DD | Assignment | C | Design Conf. | NS | Critical Error | B |
| 28 | DD | Hard. Physical | O | Unit Side Eff. | NH | Lost Objective | B |
| 29 | DD | Schematic Design | O | Lateral Comp. | RH | Critical Error | B |
| 30 | DD | Hard. Physical | C | Defect Fix | RH | Lost Objective | B |
| 31 | DD | Assembly | C | Unit Side Eff. | NH | Critical Error | B |
| 32 | DD | Faulty Comp. | 0 | Defect Fix | NH | Critical Error | B |
| 33 | DD | Hard. Interface | C | Simple Cover. | NH | Lost Objective | B |
| 34 | DD | Schematic Design | 0 | Design Conf. | NH | Lost Objective | B |
| 35 | DD | Hard. Physical | 0 | Design Conf. | NH | Critical Error | D |
| 36 | DD | Schematic Design | C | Under. Detail | RH | Lost Objective | B |
| 37 | DD | Assignment | 0 | Design Conf. | RH | Lost Objective | F |
| 38 | DD | Algorithm | 0 | Back. Compat. | NS | Lost Objective | F |
| 39 | DD | Translation | C | Unit Side Eff. | P | Lost Objective | B |
| 40 | Int | Hard. Physical | O | Design Conf. | NH | Critical Error | B |
| 41 | Int | Schematic Design | C | Under. Detail | NH | Lost Objective | F |
| 42 | Int | Hard. Physical | C | Design Conf. | NH | Critical Error | B |
| 43 | Int | Soft. Interface | C | Normal Stress | NS | Reliability | F |
| 44 | Int | Assembly | O | Design Conf. | P | Maintenance | B |
| 45 | Int | Faulty Comp. | O | Simple Cover. | RH | Critical Error | B |

**Table 14: Source List of Defect**

| No | Phase | Defect Type | Class | Trigger | Source | Impact | Env. |
|---|---|---|---|---|---|---|---|
| 46 | Int | Mechanical Phys. | C | Design Conf. | NH | Maintenance | F |
| 47 | Int | Hard. Physical | O | Complex Cov. | NH | Reliability | B |
| 48 | Int | Schematic Design | C | Normal Stress | NH | Reliability | B |
| 49 | Int | Hard. Physical | O | Simple Cover. | P | Critical Error | F |
| 50 | Int | Hard. Physical | O | Defect Fix | P | Critical Error | B |
| 51 | Int | Schematic Design | C | Design Conf. | NH | Maintenance | B |
| 52 | Int | Hard. Physical | O | Unit Side Eff. | P | Reliability | B |
| 53 | Int | Hard. Physical | C | Defect Fix | NH | Maintenance | F |
| 54 | Int | Faulty Comp. | 0 | Complex Cov. | NH | Critical Error | B |
| 55 | Int | Soft. Interface | O | Design Conf. | NS | Reliability | B |
| 56 | Int | Hard. Physical | O | Simple Cover. | NH | Critical Error | B |
| 57 | Int | Assembly | O | Unit Side Eff. | RH | Maintenance | B |
| 58 | Op | Schematic Design | O | Complex Cov. | NH | Lost Objective | B |
| 59 | Op | Hard. Physical | C | Normal Stress | RH | Lost Objective | D |
| 60 | Op | Problem Solution | C | Normal Stress | NH | Lost Objective | D |
| 61 | Op | Assembly | C | Normal Stress | P | Usability | D |
| 62 | Op | Problem Solution | C | Normal Stress | P | Usability | F |
| 63 | Op | Hard. Physical | O | Under. Detail | P | Lost Objective | B |
| 64 | Op | Assembly | C | High Stress | RH | Critical Error | F |
| 65 | Op | Hard. Physical | O | Field Side Eff. | NH | Reliability | D |
| 66 | Op | Problem Solution | O | Field Side Eff. | NS | Usability | D |
| 67 | Op | Assembly | C | Field Side Eff. | NH | Usability | D |
| 68 | Op | Documentation | C | Design Conf. | NS | Lost Objective | F |
| 69 | Op | Documentation | C | Design Conf. | NS | Lost Objective | B |
| 70 | Op | Soft. Interface | C | Field Side Eff. | NS | Performance | F |
| 71 | Op | Problem Solution | O | Field Side Eff. | NH | Critical Error | D |
| 72 | Op | Hard. Interface | C | Complex Cov. | NH | Reliability | F |

**Table 14: Source List of Defect**

| No | Phase | Defect Type | Class | Trigger | Source | Impact | Env. |
|---|---|---|---|---|---|---|---|
| 73 | Op | Mechanical Phys. | 0 | Field Side Eff. | NH | Reliability | B |
| 74 | Op | Problem Solution | O | Unit Side Eff. | NH | Reliability | F |
| 75 | Op | Hard. Physical | C | Under. Detail | P | Reliability | D |
| 76 | Op | Assembly | C | Normal Stress | RH | Lost Objective | D |
| 77 | Op | Mechanical Phys. | 0 | High Stress | P | Usability | B |
| 78 | Op | Documentation | C | Design Conf. | NH | Lost Objective | F |
| 79 | Op | Problem Solution | O | High Stress | NS | Reliability | F |
| 80 | Op | Problem Solution | 0 | Under. Detail | NH | Usability | F |
| 81 | Op | Build/Pack/Merge | C | Field Side Eff. | NS | Critical Error | B |
| 82 | Op | Perceptual | C | Normal Stress | P | Lost Objective | D |

## References


Bhandari, I., Halliday, M.J., et al. *"In-process improvement through defect data interpretation,"* IBM Systems Journal, Vol. 33, No. 1, 1994, pp. 182-214.

Chillarege, Ram. *"ODC for Process Measurement, Analysis and Control,"* The Fourth International Conference on Software Quality, Oct. 1994.

Chillarege, Ram and Bassin, Kathryn A. *"Software triggers as a function of time - ODC on field faults",* Dependable Computing for Critical Applications, Sept. 1995.

Chillarege, Ram and Biyani, Shriram. *"Identifying Risk Using ODC Based Growth Models,"* Fifth International Symposium on Software Reliability Engineering, Nov. 1994.

Chillarege, Ram, Bhandari, Inderpal S., et al. *"Orthogonal Defect Classification - A Concept for In-Process Measurements,"* IEEE Transactions on Software Engineering, Vol 18, No. 11, Nov. 1992.

Laprie, Jean-Claude. *"Dependable Computing and Fault Tolerance: Concepts and Terminology,"* The Fifteenth Annual International Symposium on Fault-Tolerant Computing. June 1985.

Lyu, Michael R., Chillarege, Ram. *"Orthogonal Defect Classification,"* Handbook of Software Reliability Engineering, McGraw-Hill, Chapter 9, 1996.

Smailagic, Asim and Siewiorek, Daniel P., *"Modalities of Interaction with CMU Wearable Computers,"* IEEE Personal Communications, vol. 3, no. 1, Feb. 1996, pp. 14-25.

Smailagic, Asim, Siewiorek, Daniel P., et al. *"Benchmarking An Interdisciplinary Concurrent Design Methodology for Electronic/Mechanical Systems,"* Engineering Design Research Center, Carnegie Mellon University, 1995.