Orthogonal Defect Classification v. 5.2

Extensions for Defects in GUI, User Documentation, Build and National Language Support (NLS)

To be used in conjunction with
Orthogonal Defect Classification v. 5.2 for Design & Code

September 12, 2013

This document is made available for general interest and informational purposes only.
IBM assumes no responsibility for its usage.

1. Information Development (User Documentation Development)

The defect found is with the written description contained in user guides, installation manuals, on-line help, user messages.

Examples include:

1) An installation problem is experienced because there was no user documentation of the data sets to be allocated before the installation was attempted.

Opener Section (These attributes are usually available when the defect is opened.)			**Closer Section** (These attributes are usually available when the defect is fixed.)					
Defect Removal Activities	*Triggers*	*Impact*	*Target*	*Defect Type*	*Qualifier*	*Age*	*Source*	*Content Type*
ID Review	• Accuracy • Clarity • Completeness • Consistency • Organization • Retrievability • Style • Task Orientation • Graphic Design/Aesthetics	The same as that for Design and Code	Information Development	• Editorial • Technical • Navigational	The same as for Design/Code			• Reference-Information • Tasks • Examples • Presentation • Concepts

1.1 Opener Section: (These are attributes you can classify when you find a defect.)

1.1.1 Activity: Information Development (ID) Review

Development and Inspection of external user documentation, help text, user's guide, screen information

Examples:

1) The error message was spelled incorrectly.

2) The help text listed 3 steps to add a record but there were 4.

1.1.2 ID Triggers

1.1.2.1 Accuracy:

The information does not describe the product correctly; there are factual errors.

1.1.2.2 Clarity:

The information is confusing or difficult to understand.

1.1.2.3 Completeness:

Necessary information is missing.

1.1.2.4 Consistency:

The information is not displayed in a consistent manner. There are different explanations, spellings, wording, etc... for the same piece of information.

1.1.2.5 Organization:

The relationship between parts or between a part and the whole is not conveyed.

1.1.2.6 Retrievability:

The information is difficult to find.

1.1.2.7 Style:

The manner of expression is inappropriate or difficult to understand.

1.1.2.7.1 Task Orientation:

The presentation of why and how to perform a task is inappropriate.

1.1.2.8 Graphic Design/Aesthetics:

The appearance and layout of the information is inappropriate.

1.2 Closer Section: (These are attributes you can classify when you fix a defect.)

1.2.1 Defect Type: Represents the actual correction that was made.

1.2.1.1 Editorial

Defects relates to grammar, spelling, punctuation, organization, etc... necessary to build the information describing a product.

Examples:

1) The help text that is displayed has a spelling error.

1.2.1.2 Technical

Defects related to the description of a product and its interfaces.

Examples:

1) The Installation Guide shows an incorrect wiring diagram for connecting the modem to a computer.

1.2.1.3 Navigational

Defects that prevent users from finding needed information about a product including errors in mechanisms used to connect information (i.e. hyperlinks, indices, tables of contents, cross references, book shelves, etc...).

1) When the user clicked on a hyperlink on the web, it got the message "Netscape is Unable to Locate the Server..."

1.2.2 Information Development Content Type

This captures the nature of the information content that had the defect.

1.2.2.1 Reference Information

Defect contained in detailed descriptive information such as definitions, dictionary entries, glossary items, parameter explanations, etc...

1.2.2.2 Tasks:

Defect contained in guidance information that describes the user tasks of the underlying product.

Examples:

Defect contained in special descriptive information of a detailed nature showing specific applications of the described product.

1.2.2.3 Presentation:

Defect contained in graphical and other elements used to present the information, such as music, fonts, colors, layout, packaging, tables, etc...

1.2.2.4 Concepts

Defect contained in high level overview and conceptual information describing important characteristics of the product like function or performance.

2. Graphic User Interface: This only addresses the visual aspects of GUI and not the underlying functionality.

Opener Section (These attributes are usually available when the defect is opened.)			Closer Section (These attributes are usually available when the defect is fixed.)				
Defect Removal Activities	*Triggers*	*Impact*	*Target*	*Defect Type*	*Qualifier*	*Age*	*Source*
GUI Review	• Design Conformance • Widget/Icon Appearance • Screen Text/Characters • Input Devices • Navigation • Widget/GUI Behavior	The same as that for Design and Code	Design/Code	The same as for Design/Code			

2.1.1 GUI Review Triggers

2.1.1.1 Design Conformance:

Defects found while reviewing GUI design documents.

2.1.1.2 Widget/Icon Appearance:

Defects found while reviewing widget and icon appearance (wrong color, too big, etc...).

2.1.1.3 Screen Text/Characters:

Defects found while reviewing text or characters, for example, button mislabeled.

2.1.1.4 Input Devices:

Includes defects found while testing various input methods including mouse, keyboard, touch screen.

2.1.1.5 Navigation:

Defects found while attempting to navigate within or between windows, menus, or components.

2.1.1.6 Widget/GUI Behavior:

Defects found while testing the expected behavior of widgets and icons (widget doesn't get focus, multiple list items end up with selection emphasis, help button doesn't work).

2.2 Build/Package/Merge: This only addresses defects found in the Build and Packaging process only.

2.2.1 Build/Package/Merge

Problems encountered during the driver build process, in library systems, or with management of change or version control.

Examples include:

1) Part not defined to build process. Mapping macros required to assemble application programs missing from a public library.

2) Module compiled using a back-level version of a source code part.

Opener Section (These attributes are usually available when the defect is opened.)			Closer Section (These attributes are usually available when the defect is fixed.)				
Defect Removal Activities	*Triggers*	*Impact*	*Target*	*Defect Type*	*Qualifier*	*Age*	*Source*
Design Rev, Code Inspection, Unit test, Function Test, System Test.	The same as that for Design and Code		Build/Package	• Process Conformance • Code Integration • Maintain/Fix Dependencies • Shipped Files • Packaging Scripts • Install/Upgrade Dependencies • Media	The same as for Design/Code		

2.2.2 Defect Type for Target=Build/Package

2.2.2.1 Process Conformance

Problems with the specified procedure to be followed for the build.

2.2.2.2 Code Integration

Code from one area caused a build/execution break in another area. A dependency was missed. Also used with an incorrect merge of two source files.

2.2.2.3 Maintenance Application Dependencies

Incorrect or incomplete pre-requisite, co-requisite, or sequencing of dependencies is captured.

2.2.2.4 Shipped Files

Problems with the shipped files of a file set, such as dual ownership, system errors (bad size, checksums), etc...

2.2.2.5 Packaging Scripts

Problems with the scripts used by the file set during install, de-install, upgrade, migration, etc...

2.2.2.6 Feature Install /Upgrade Dependencies

Problems with the set of files required for installation or problems with requirements, such as missing, incorrectly specified, inappropriate, etc...

2.2.2.7 Media

Problems with the package used to distribute the software. This would include problems with a CD-ROM, tape drive, etc...

2.3 National Language Support

2.3.1 National Language Support

Problems encountered in the implementation of the product functions in languages other than English.

Examples include:

1) Message contains garbage for French only

2) Cannot print German specific characters

This only addresses the aspects related to the translation of the software to the user's natural language (German, Japanese, etc.) and does not consider underlying functionality.

Opener Section (These attributes are usually available when the defect is opened.)			Closer Section (These attributes are usually available when the defect is fixed.)				
Defect Removal Activities	*Triggers*	*Impact*	*Target*	*Defect Type*	*Qualifier*	*Age*	*Source*
Design Rev, Code Inspection, Unit test, Function Test, System Test.	The same as that for Design and Code		National Language Support	• Translation • Character Handling • User Interface • Language Expectation • Enablement	The same as for Design/Code		

2.3.2 Defect Type for Target=National Language Support

2.3.2.1 Translation

The translation was incorrectly done or was missing altogether.

2.3.2.2 Character Handling

The inability of the program to correctly handle characters associated with another language. Examples: Command/menu input does not accept multi-byte characters, array parsing assumes single byte characters only, or program strips 8th bit when processing data.

2.3.2.3 User Interface

The inability of the user interface to function or display correctly in another language. Examples: Motif application buttons expand entire window in Chinese, calendar icon does not display localized data, or web interface does not display Russian characters correctly (font problem, not character handling).

2.3.2.4 Language Expectation

When a command or program is checking for a specific input or output as an English or C locale would provide it.

2.3.2.5 Enablement

When the product was not designed or incorrectly designed to enable National Language Support.

3. REFERENCES

[1] Orthogonal Defect Classification for Design and Code v5.2

[2] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S.Moebus, B. K. Ray, and M.-Y. Wong, “Orthogonal Defect Classification: A Concept for In-Process Measurements,” IEEE Transactions on Software Engineering 18, No. 11, 943–956 (1992).

[3] K. A. Bassin, T. Kratschmer and P. Santhanam, "An Objective Approach to Evaluate Software Development", IEEE Software Magazine, November/December, Vol.15 No.6 pp. 66-74, 1998

[4] M. Butcher, H. Munro & M. Butcher, "Improving software testing via ODC: Three case studies", IBM Systems Journal, Vol. 41, No. 1, pp.31-44, 2002.