

Typical uses of defect records in Cisco

Mod Marathe
Distinguished Engineer
Cisco Systems
February 2007

Interesting fields in defect records

<i>Interesting Fields in the defect database</i>	
Field	Description
Component	Each functional component consists of a set of source files and is mapped to a manager. This field is set by the submitter of the defect but can be updated as the defect is analyzed by engineers.
Version	This is the version number for the software against which the defect report is being submitted. If this report is being submitted against multiple versions, all versions are entered in the field. This field is set by the submitter of the defect.
Found by	Shows which function found defect (unit test, code review, system test, customer, etc.). This field is set by the submitter of the defect.
Development-Escape	This flag will capture information on whether the defect is a development escape or not. If flagged, also includes the earliest phase where this defect <i>should</i> have been found. This field is set by the engineer fixing the defect after the fix has been created.
Regression	The Regression flag captures whether the defect represents regression in functionality from a previous version of software. That is, functionality that was working before stopped working due to a bad bug fix or other changes. This field is set by the submitter of the defect.
ODC Fields	There are five fields supporting Orthogonal Defect Classification: Impact and Triggers, filled in by submitter; Origin, Category, and Reason, filled in by engineer fixing the defect.

Use by programmers

- to track the evaluation and the fix for the defect
- the Development-Escape and Regression fields are used by programmers for self-learning – that is, if a large number of defects are found in subsequent testing that should have been found during unit testing, a better unit test needs to be developed as well as a better review of the unit test plan needs to be conducted.

Use by first level management

- Phase containment – are defects being found in the right dev/testing phase
- MTTR for customer found and internally found defects
- SRE curves (cumulative defects vs. testing effort)
- Root cause analysis
- failure-prone modules or files
- Development process adoption

Use by upper management

- % of total defects found by customers
- % of customer found defects that are regressions
- MTTR for customer found defects
- Severity 123 backlog
- organizational productivity (bug finding and bug fixing)
- comparison of metrics between different groups
- tracking improvement goals they have set for their teams.

Research opportunities for using the defect database

The defect database represents defects after the fact. It is better if these defects were prevented from entering the code in the first place.

- **Prevention of defects**
Use pattern matching of previous defects to identify defects in newly written or existing code. Also, if semantic information about the defects can be captured at the time of the fix, this can be used in automatic code review tools to identify defects or at least point out potential code areas to inspect manually.
- **Fixing similar defects**
Once a fix has been developed, there are usually other places in the code where a similar defect may be present. The programmer fixing the defect already thinks of such places but if the code size is large, this is not always possible. Any automated tools (syntactic and semantic) to help with this identification will be very useful.



CISCO