FUNCTIONAL RELIABILITY IN RAPID APPLICATION DEVELOPMENT

Michael S. Wills, R.D.
The CBORD Group
Ithaca, New York USA

ABSTRACT

This paper describes the adaptation of Functional Reliability (FR) to the rapid application development (RAD) environment. Every third week since January 2000 we have cut, tested and evaluated for release a new version of our software products. Solutions to anticipated and unanticipated challenges involved in the modeling of reliability for this process are described with an emphasis on evaluation of risk pre-release and how field results are fed back into release decision process. For example, the defect closure standard was modified to accommodate new features that are desired by users and we model reliability of released product over a six month period by product, feature set, version and where in the development cycle a defect originated. We use this information for continuous improvement of our development, release decision and support processes.

Functional Reliability

Functional Reliability is a refinement of Orthogonal Defect Classification (Chillarege, 1995) that quantifies defect failures per CPU hour. From the user perspective, failures per CPU hour increase proportional to the extent the consequences of the defect reduce user productivity. That is why FR measures failures per *productive* CPU hour (PCPUh). PCPUh decrease with increasing defect consequence and approach zero for high consequence defects. The function of equation 1 (x = PCPU) leads to the improper integral of equation 2 and the realization that failures per PCPUh approach infinity for unresolved high consequence defects.

Equation 1

$$f(x) = \frac{failures}{x}$$

Equation 2

$$\int_0^x (failures / x)dx$$

This is why FR includes consequence as an adjustment to failures per CPU hour calculated from formal (Musa, 1987) or informal operational profiles. Instead of infinity, we chose an adjustment factor that causes all high consequence faults to exceed the threshold of .01 failures per PCPU hour that we chose as the point which a defect becomes high risk. Alternatively, the adjustment factor could be derived from equation 2 if the duration during the the defect was unresolved is substituted for 0 → x, but this derivation is beyond the scope of this paper. The FR formula and algorithm are published (Wills, 1999) and a summary is given in Addenda A.

FR is well suited for use in the RAD environment because:

- FR can be used to inform discussions about the meaning of a defect.
- FR parameters can be adjusted as a better understanding of a defect is achieved with a corresponding adjustment of that defect's contribution to the total failures per PCPUh for the software module.
- FR requires the application of a consistent, but not necessarily formal, operational profile. This translates to a metric that is descriptive and flexible. Implementation of FR depends as much upon a dialog between users, implemetors, developers and testers as it does upon software requirements.

This paper will explore these issues in the context of how FR was applied to the software development processes of the CBORD Group's Foodservice Management Division.

The Development Environment

CBORD provides software solutions to all foodservice markets, including colleges and universities, corporate, healthcare, restaurants, theme parks and supermarkets. Approximately 15,500 function points are contained in feature sets that encompass Replication, Purchasing, Production, Service, Items, Issuing, Service, Data conversion (from earlier versions), System Settings and Utilities, Nutrition Services, Nutrition Assessment. For the past four years we have

maintained a core feature set to which new features, enhancements and improvements are added.

Our client/server architecture is two-tier using Powerbuilder with Sybase databases. Software defects are logged on a Clarify database that we have customized with, among other information, our functional reliability parameters, algorithms and formulas.

Prior to January 2000 we released product about once every six months, when the targeted feature sets were complete and reliability goals were met for the entire product. Since January 2000, we have released product in three week cycles, based upon product completion, client needs, analysis of reliability statistics. We made this change in order to be more responsive to client requests and to provide new product on a scheduled basis.

Workaround Strategies and Field Experience

In both release schedules we faced the task of deciding which defects to correct. Then, as now, we use FR as the framework for discussions between development, testing and implementation. If the status of a defect is questioned, we step through the FR parameters and then either come to an agreement on a parameter or refine the nature of our disagreement.

For example, a sequence of user responses and mouse clicks triggers a defect that causes the client to close with an error message. Such a defect could warrant a high consequence setting and a hold on product release until it is resolved. The tester who discovered the sequence rates it as affecting a small number of users every day (because a simpler and more accessible sequence provides the same user benefit) with the computer used 3 hours a day. FR is calculated as .01 failures per PCPUh with a high consequence when these FR parameters are applied to the FR equation in addenda A: F = 1, U = 3, C = .01, V = 1, con = 3.

The software developer responsible for the affected feature may question these FR parameters. If the tester and developer cannot agree and, upon review by implementation, it is decided that those users who do trigger the error can be provided with a workaround solution, the adjusted FR is .0000333 failures per PCPUh with a moderate consequence (F = .01, U = 3, C = .01, V = 1, con = 1). Frequency is .01 because, once a user learns the workaround, the issue does not recur. Consequence is moderate because an approved workaround exists. This defect would not hold up product release and it could be fixed at a later time along with other defects in related code.

We explore the possibility of workaround solutions for all high consequence defects and tolerable consequence defects with an FR of .15 or more. Generally all workaround solutions are published on the web, but each needs to be approved before publication. Approval is the coresponsibility of the divisional quality assurance director and senior editor.

Client support has access to all workaround strategies and can link unresolved cases to unacceptable workaround strategies. A workaround that accumulates unresolved user cases is reviewed and FR is adjusted if appropriate.

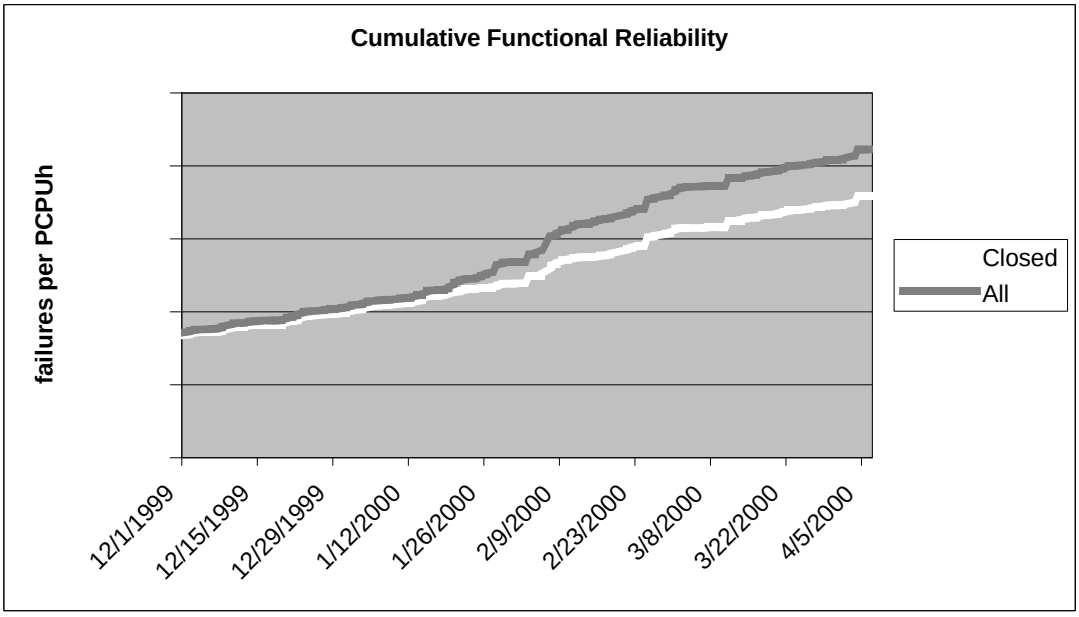
Defects accumulate unresolved cases in the same way. This allows use to monitor the effect of defects released to the field.

Reliability and Rapid Application Development (RAD)

Prior to the launching of a three week development cycle, we could not consistently predict release dates because meeting our reliability goals took precendence. Both our external and internal (implementation and sales) customers perceived the development group as unreliable and we concluded that the unpredictable release dates were the cause. This situation had the potential for reducing customer confidence in our ability to deliver on promises and was the motivation for adopting a three week RAD development cycle.

With the first release from the RAD cycle (late January 2000) we experienced an **apparent** degredation in reliability along with the benefits of scheduled releases (Figure 1). The figure 1 chart x-axis is the database defect record creation date. The All and Closed lines represent defect status. The gap between the lines represents unresolved defect, measured in FR.

Figure 1, Time prior to RAD through the third RAD release



Figures 2 and 3 provide more detail and perspective for the same time period. The chart of figure 2 is a break down of the gap in figure 1. It shows high consequence failures as the major contributor to the widening gap. Figure 3 add the perspective that our testing continued to find a high proportion of failures prior to release. In fact, the majority of post release failures were discovered within development. Less than 3% of failures were first discovered by implementation and users. This leaves us with the questions:

- What is the nature of the unresolved failures?
- Do the benefits of frequent scheduled releases exceed the cost of unresolved failures?

Figure 2, Breakdown of unresolved FR by consequence

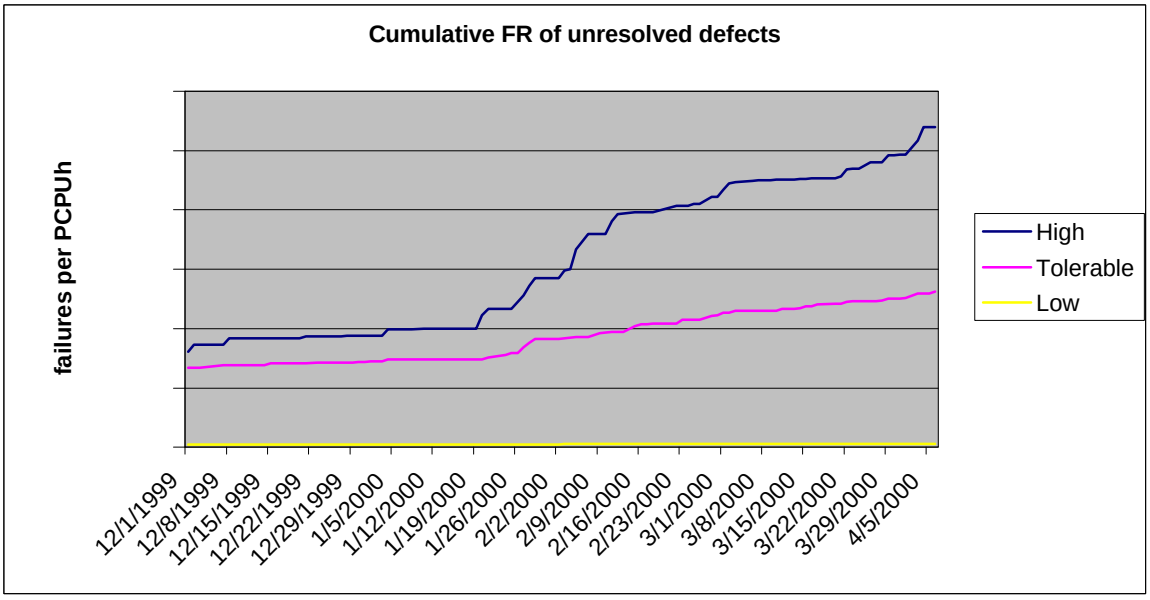
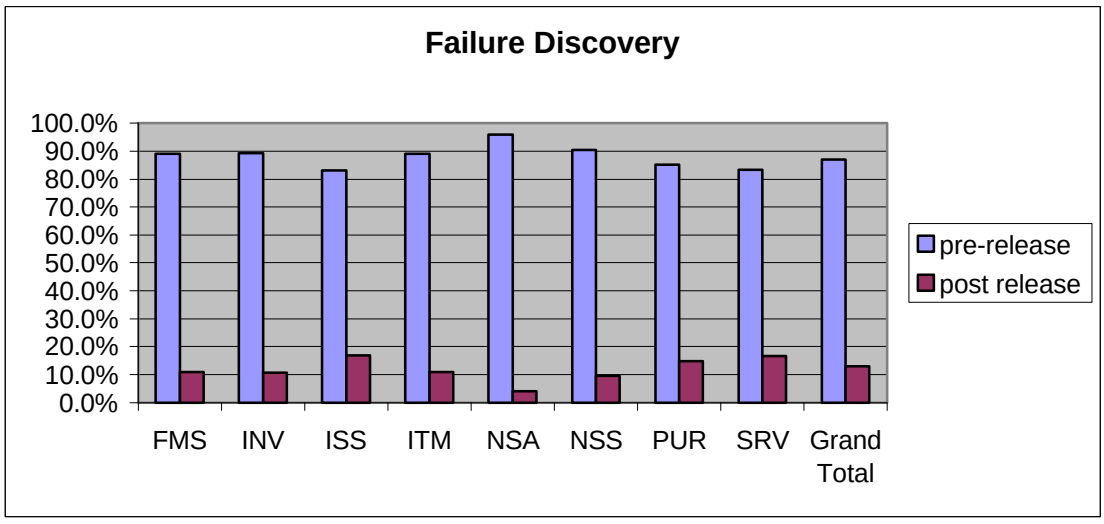


Figure 3, Pre vs. Post



Adapting Functional Reliability to RAD

We used two techniques to manage the widening gap in figure 1:

- Extract from the gap those defects with accurate FR, but which did not affect users.
- Create a process to resolve high risk defects within a defined period of time.

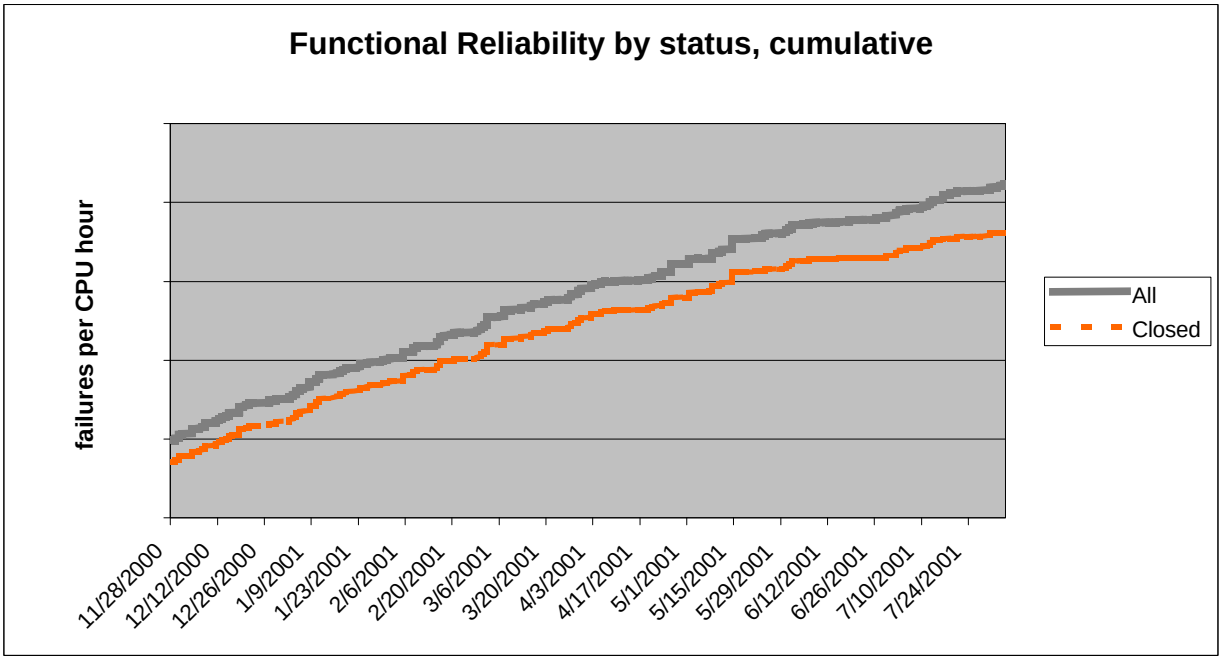
We implemented a release committee as part of the RAD development process. It was composed of representatives from senior management, client services, sales, development and quality assurance. The committee met upon the completion of the test cycle to consider the affect on users of each new release's enhancements and improvements, especially the users to whom commitments were made. High risk defects (Wills, 1999) are assessed, both those introduced in the release and the implications of defects introduced in a earlier versions.

The release committee discussions led to a definition of unresolved high risk defects that did not affect users in the released version or for which effective workaround strategies could be defined. Examples of such high risk defects are:

- There are features that affected a portion of the user base and this made it possible to negotiate an satisfactory compromise. One such case were reports that could not print on A4 paper. We communicated with affected users, identified the reports most important to them and fixed only those. This left one unresolved high risk defect per report.
- Features new to that version. If there were compelling reasons to release the version (if other new features were ready, for example) we made the new feature inaccessible or negotiated another approach with the implementation team.
- Features with a strong user demand, but with significant flaws. In one case many users requested advanced queries for reporting. It is possible for users to create complicated advanced queries that require a very long time to complete. We provided training materials and release notes that advise users of this situation.
- Systems related to the released product, but which are treated separately. Interfaces are one such system. An interface might be planned to be coordinated with a release
- Product managers take responsibility for deferring defects which, in their own judgement, will not adversly affect users. The test manager is responsible for reviewing these and, if there is a disagreement, the test manager can request reviews by client services, development management and senior management in that order.

We categorized these defects in the Clarify database as "special considerations" and remove them from the "All defects" data series to obtain the chart of figure 4. Special considerations defects are categorized by system and defect type (i.e, "A4 paper," "Advanced Query") and tracked in tabular format at total FR.

Figure 4, after adjustment for Special Considerations



The subsequent chart reveals the FR of defects in the current round of testing (on right) and aging defects (on left). Typically aging defects are difficult to fix or affect a portion of the system that requires a large testing investment if changed.

Development, quality assurance, testing and support/implementation are involved in version planning. During the version planning process we evaluate the following issues relating to defects:

- Which aging defects are identified by users and client services as needing a fix? Support links defects to user case records. Defects with more than two links rise to the top of everyone's list.
- Special consideration defects:
    - does each category still apply?
    - Has the FR for the category changed? If so, why?
- Which high risk defects are still open from recent version cycles? Which user sites are potentially affected by these defects?

During the version release process we evaluate:

- The change in the right side gap of the cumulative FR chart since the last release.
- The high risk defects opened during the last three release cycles, including the current cycle and excluding special consideration defects. What effect will these defects have on the users who are scheduled to receive the version?
- Has there been a change in FR for any system or special consideration category? If so, why?

Conclusions

The CBORD FMS division has a user group conference each summer that includes a "users only" meeting in which concerns and praise is aired. Our windows products are consistently rated as high quality and valuable for user operations.

References

Chillarege, R.. (1995). Orthogonal Defect Classification. Software Reliability Engineering, Chapter 5. Lyu, M (editor). IEEE Computer Society Press, Los Alamitos, California.

Musa, J.D., Iannino, A., and Okumoto, K., (1987) Software Reliability - Measurement, Prediction, Application. McGraw-Hill, New York, 1987.

Wills, M.S. (1999). Functional Reliability. Proceedings of the Ninth International Conference on Software Quality October 1999, 43-54.

FUNCTIONAL RELIABILITY EQUATION AND RULES

Equation 3, "Functional Reliability," combines failures per CPU hour (**F** divided by **U**) with adjustments for system usage weight (**C**), visibility (**V** divided by 3) and consequence (1 divided by **con**). Equation 3 reduces a fault's contribution to reliability if less than 100% of users are affected by the fault, if the fault is of less consequence, and if the fault is of a lower consequence and difficult to notice.

Senior CBORD management and myself set the values for the adjustment factors (**V** divided by 3), (1 divided by **con**) and system usage (**U**) of 3 hours per day. System usage can be tied to operation profiles.

Frequency (**F**) is based upon the system option(s) affected. It can be from an operation profile [31Musa], and adjusted as needed. For our implementation of Functional Reliability I used my knowledge of FMS to create an informal profile. Refer to Addendum C for rules to use in determining frequency. Variable operational profiles can be applied if standard rules are used to evaluate special considerations. See Addendum C, step 3 for examples.

Analysis of each fault is used to determine the system usage weight (**C**). Refer to Addendum B for the rules.

The rules for Visibility and Consequence weight are given in the Equation 3.

Equation 3

$$\sum_{s=2}^{3} (F \div U) * (C) * (V \div 3) * (1 \div con)$$

S = the status of the recorded fault. The defined range includes all valid, unresolved faults.

Failures per CPU hour: $F \div U$

F is the frequency, how many times a day. Frequency is a number between .01 and 1 or a multiple of 1.

U is the system usage in one day. System usage is measured in hours.

Usage Weight: C

C is a number between .01 and 1 used to reflect the number of feature users affected by th fault. See Addendum A for guidelines.

Visibility Weight. $V \div 3$

Visibility for faults of critical consequence is always the default value (1).

Not obvious to all users and does not affect essential functionality. A factor of .3333

Evident only to knowledgeable users and does not affect essential functionality. A factor of .6667

The default value. A factor of 1.

Consequence Weight:

$$1 \div CON$$

Consequence is the effect of fault on the user. The following numbers are used for CON:

1 **Mild**. At worst, the fault does not affect the user's work, but makes the work a little more difficult. For example, having to press the OK button an extra time.

2 **Tolerable**. At the lowest level, a serious fault would not affect the user's work, but would cause the user to distrust the system. For example, if a system error happens every time the user signs off after creating a new vendor item. At worst, the user's work is hindered and recovery is possible with loss of no more than 10 minutes.

3 **Critical.** At the lowest level, a critical fault hinders a user's work and requires more than ten minutes recovery time. Faults that cause data loss are critical. A fault that renders a feature non-operation is also critical.

Addendum B

SYSTEM USAGE WEIGHT

Step	Description
1	Consideration is activated by a frequency (F) above 0.
2	Weighting is one (1) if the fault causes the option to be inaccessible. Example: system crashes when option is accessed.
3	Weighting is one (1) if the fault causes the option to be unusable. Example: a report prints, but presents inaccurate data.
4	Weighting is one (1) if the fault is on an essential usage path for the option. Example: a query that populates a window is functional, is missing a required search criterion.
5	Weighting is one (1) if the fault is close to an essential path for the option. When estimating weight, use the highest possible number. Example: there are four possible paths for finding and applying menu items and one (1) is affected by a fault. The operational profile does not cover the relative frequency and insufficient field data is available.
6	Weighting is less than one (1) if steps one through five do not apply. Set the highest possible weighting supported by the facts. The following guidelines are used by CBORD, but may not be applicable to all circumstances:
	Guidelines: Help options are .25 by default. Rely upon the operation profile for multiple paths. Example: use .5 if a fault affects two of four paths used with identical frequency. If the affected path is unique and not covered by the operation profile, then estimate. Use your knowledge of the application and users. Place the weight on the high side of your estimate.

Addendum C

RULES FOR DETERMINING FREQUENCY

Steps	Description
1	Determine the affected option(s). Example: Create and Maintain Purchase Orders.
2	Refer to the operation profile the number of times the option is used in a day. For our implementation I limited frequency to once a day unless multiple options are affected. Example: Create and Maintain Purchase Orders are each used daily, for a total of 2.
3	Evaluate special considerations. Example: the setting for Menu Courses is typically set once. If the failure does not prevent the entry or use of Menu Courses, but only cause the creation of courses to be more difficult, then the user is only affected when Menu Courses are entered. Use .01 for such cases. If the failure prevents Menu Course entry, users are affected daily because courses are used to organize foods within menus. If variable operation profiles are used, special considerations could be conserved if frequencies of .01 and .99 are used as signals to not change frequency with changes of operational profiles.
4	Add the results of steps 2 and 3 for the options identified in step 1.