Quality Excellence for Suppliers of Telecommunications Forum (QuEST Forum)

TL 9000 Quality Management System

Measurements Handbook

SFQ Examples

R5.0 and Earlier

8.1 SFQ Examples

8.1.1 – SFQ Example

The following example illustrates calculation of the software fix quality measurement. Software fixes are counted regardless of the method used to package/deliver the fix.

An organization has three active releases, 1.0, 1.1, and 1.2. Software fixes have been made available are as shown in Table 8.1.1-1.

Table 8.1.1-1 Example SFQ Release/Fix Data

Generic/Base Release	Release	GA Date	Number of Software Fixes
1.0	1.0	Nov 2009	n/a (first release)
	1.0.1	Jan 2010	5
	1.0.1.1	Feb 2010	1
	1.0.1.2	Jun 2010	1
1.1	1.1	Jan 2010	15
	1.1.0.1	Feb 2010	4
	1.1.0.2	Mar 2010	2
	1.1.1	Apr 2010	8
	1.1.1.1	Jun 2010	1
1.2	1.2	Jun 2010	10
	1.2.0.1	Jun 2010	1
	1.2.0.2	Jun 2010	1

The number of software fixes reported each month for the Software Fix Quality measurement would be determined as shown in Table 8.1.1-2.

Table 8.1.1-2 Example SFQ Monthly Software Fix Data

	Jan 2010	Feb 2010	Mar 2010	Apr 2010	May 2010	Jun 2010
Number of Software Fixes						
Release 1.0	5	1	0	0	0	1
Release 1.1	15	4	2	8	0	1
Release 1.2	0	0	0	0	0	12
Total	20	5	2	8	0	14

In February 2010, during the installation of Release 1.0.1.1, it becomes evident that the fix could not be installed.

In May 2010 a major problem determined to be caused by one of the fixes in Release 1.1.1, made available in April 2010, is reported.

In June, after Release 1.0.1.2 is made available, additional internal testing by the organization determined that the fix doesn't completely correct the intended problem.

Also in June 2010, a critical problem determined to be caused by a fix in Release 1.2, made available earlier in June 2010, is reported.

The number of defective software fixes reported each month for the Software Fix Quality measurement would be determined as shown in Table 8.1.1-3.

Table 8.1.1-3 Example SFQ Monthly Defective Fixes

	Jan 2010	Feb 2010	Mar 2010	Apr 2010	May 2010	Jun 2010
Number of Defective Software Fixes						
Release 1.0	0	1	0	0	0	1
Release 1.1	0	0	0	0	1	0
Release 1.2	0	0	0	0	0	1
Total	0	1	0	0	1	2

The resulting monthly source data and measurement calculations are shown in Table 8.1.1-4.

Table 8.1.1-4 Example SFQ Monthly Source Data and Measurement Calculation

	Jan 2010	Feb 2010	Mar 2010	Apr 2010	May 2010	Jun 2010
Number of Software Fixes	20	5	2	8	0	14
Number of Defective Fixes	0	1	0	0	1	2
%Defective	0%	20%	0%	0%	100%	14.3%

For the month of June 2010, the TL 9000 SFQ data reported is shown in Table 8.1.1-5.

Table 8.1.1-5 SFQ Data Table Report

Identifier	Value
MeasurementID	SFQ
DFc	2
Fc	14

8.1.2 – Frequently Asked Questions

8.1.2.1 How do I count software fixes?

Organizations have one or more means by which software fixes are delivered or made available to the customer for implementation. These different types of delivery mechanisms include (but are not limited to) patches, files, maintenance releases, updates, dot releases, fix releases, etc. Although the actual implementation method differs, each of these means would include some type of notification of the availability of the software change and information on what fix(es) are included, such as a release letter or product bulletin.

Organizations can use the customer notification of the software change to obtain the number of fixes to be included in the SFQ measurement. The information must be descriptive enough to ensure only fixes to problems requiring changes in the product software are counted. Fixes associated with paper documentation or enhancement requests would not be counted. The customer notification would need to include fixes to problems in the delivered software found by the organization as well as those found by customers; otherwise, these fixes would need to be identified and counted separately and the internal and customer fix counts added together to obtain the reported SFQ counts.

An alternate means of identifying fixes to be included in the SFQ measurement is by utilizing the organization's problem tracking tool to identify problems fixed. As with the previous method, the tool must be able to identify what release(s) the problem is being fixed in, distinguish between defects and enhancements as well as distinguishing between fixes requiring product software changes and those that do not (for example, paper documentation changes, third-party software changes). If separate tools are used to track customer problems and internally found problems, counts from the two systems would need to be added together to obtain the reported SFQ counts.

8.1.2.2 How can I use the SFQ Measurement?

The Software Fix Quality measurement is the percentage of software fixes determined to be defective. The higher the percentage, the greater is the risk that the installation of a software fix to correct a problem will introduce additional problems into the network.

When using the SFQ measurement, especially to set goals and drive continuous improvement, it is important to consider the TL 9000 Performance Data Reports smoothing rules and use the smoothed averages. Monthly snapshots may demonstrate too much variability to provide an accurate representation of the software fix quality trend due to the delay in the discovery of defective fixes. This variability will be more obvious in products that only release fixes a few times a year.

The SFQ measurement should trend downward, eventually reaching 0. Where this is not happening, organizations should consider performing defect analysis on the defective fixes to identify possible process improvements. The analysis should be focused on why the fix provided did not work, not on what caused the original problem. A detailed defect analysis should identify the root cause of the problem, measures that could have detected the defective fix before it was made available and measures that could have prevented the defective fix from being introduced into the software.

If there is a spike in the percentage of defective fixes, the organization could perform a high-level defect analysis to see if the problems are related to a particular release, customer, product platform, etc. This could help identify possible areas for improvement at a lower cost than a more detailed analysis of individual defects.