

Candidate Metrics

Title	Definition	Comments	Depends On
Defect Inflow	Defect find rate - normally defects found per week	Indicates the level of quality in an area (e.g. a package) based on number & frequency of defects. All else being equal, the more defects, the lower the quality.	
Trend of Defect Inflow	Average Defect find rate (e.g. over 5 weeks, over 13 weeks)	As above. Unexpected changes in inflow may indicate the need for analysis to discover the cause.	
Defect Outflow (Fixes)	Defect fix rate - normally defects fixed per week	Indicates the level of quality in an area (e.g. a package) based on number & frequency of defects. Not susceptible to defects that are rejected.	
Trend of Defect Outflow (Fixes)	Average Defect fix rate (e.g. over 5 weeks, over 13 weeks)	As above. Unexpected changes in outflow may indicate the need for analysis to discover the cause.	
Open Defects	Number of Open defects	Indicates the backlog of work assigned to an area (e.g. a package). Process indicator.	
Trend of Open Defects	Average number of open defects (e.g. over 5 weeks, over 13 weeks)	As above. Less susceptible to short-term fluctuations. Process indicator.	
Age of Open Defects	Average age of Open defects	Indicates the maturity of backlog of work assigned to an area (e.g. a package). Process indicator	
Fix Turnaround Time	Average time from New to Verified: Fixed	Indicates package owner's ability to fix defects. Process indicator.	
<i>Reject Turnaround Time</i>	<i>Average time from New to Closed: not Fixed ??</i>	<i>Indicates package owner's ability to identify defects not requiring fixes. Process indicator.</i>	
<i>Resolve Turnaround Time</i>	<i>Average time from New to Resolved ??</i>	<i>Indicates package owner's ability to identify defect resolutions (Fix/Reject). Process indicator.</i>	
<i>Rejected Defects (Ratio)</i>	<i>The proportion of the total number of defects found that are Closed: not Fixed</i>	<i>Used in combination with other data to check for possible anomalies. More useful if subcategories are measured, e.g. WONTFIX, INVALID, etc.</i>	
<i>Defect Removal Effectiveness (Internal Test)</i>	<i>The proportion of all defects that have been found internally in comparison to those found externally (e.g. by the package owner vs. those found by others)</i>	<i>Indicates package owner's ability to prevent defects leaking out to others. Process indicator.</i>	

<i>Product Size (Source Code)</i>	<i>The number of lines of source code in a package version.</i>	<i>Used to normalise other measures, e.g. Defect Inflow => Relative Defect Inflow.</i>	
<i>Relative Defect Inflow (Hotspots)</i>	<i>Ratio of defect inflow to product size, normally by package.</i>	<i>The defect inflow figures are normalised by size, allowing a comparison to be made between different packages and with the SDPR Asset base.</i>	<i>Defect Inflow, Product Size</i>
<i>Code Change</i>	<i>The number of lines of source code added or changed. Blank lines & comments are excluded.</i>	<i>Indicates the amount of effort going into a package, and therefore the amount of defects being introduced. Also needed for estimating defect density.</i>	
<i>Defect Introduction Rate</i>	<i>The rate at which defects are introduced, per lines of source code added or changed</i>	<i>Indicates the quality of submitted code. Also needed for estimating defect density. See note 1.</i>	<i>Defect Outflow, Code Change</i>
<i>Found-to-Latent Ratio a.k.a. Defect Discovery Rate</i>	<i>Ratio of weekly defect find rate to the estimate of defects remaining</i>	<i>Used to forecast Estimated Defect Density (and ENUD). This is based on an estimate of defects remaining, so should be used with caution as an indicator.</i>	
<i>Estimated Number of Unresolved Defects (ENUD)</i>	<i>The estimated defects remaining in a product</i>	<i>Indicates the quality of the code for a package. See note 2.</i>	
<i>Estimated Defect Density</i>	<i>The estimated defects remaining in a product, divided by the product size</i>	<i>Indicates the quality of the code for a package. See note 2.</i>	<i>ENUD, Product Size</i>

Key (as currently perceived)

Easiest to implement

Moderately easy to implement

Hardest to implement

Note 1.

This metric is ONLY an indicator, not a precise measure. By measuring the amount of code change in a package, we can use the defect introduction rate to estimate the number of new defects introduced into the code. It is usually calculated over a sufficiently long period (e.g. six months), so that defects existing at the start of the period (legacy defects) and defects remaining at the end can be ignored (or assumed to be the same). However, this measure can vary due to factors other than quality of submitted code, e.g. level of testing and level of legacy defects.

Note 2.

This metric is ONLY an indicator, not a precise measure. This ongoing estimate is validated, using an exponential decay rate model, during a period where code change is low (i.e. defect fixes only) for a period of 12 weeks or longer, usually at the end of the project. At this point, the Defect Introduction Rate is also re-calculated - over the whole project.

Note 2. continued

The estimate based on several assumptions:

- The estimated number of unresolved defects (ENUD) in a package version, at the point the version is first created, is the same as the that of the most recent previous version, i.e. the number of legacy defects inherited.
- The code change for a package has a fairly uniform defect introduction rate during its lifetime. We check this assumption by measuring the Defect Introduction Rate.
- The Code Change information for an OS product is complete and correct.

Metric Areas Not Covered:

1. Development Effort
2. Test Effort
3. Test Cases
4. Test Results
5. Build Results
6. Requirements
7. Submissions
8. Code Ownership (e.g. package ownership, e.g. proportions of SF asset base owned by different SF member organisations)