Employing Software Defect Measurement for Improvement at JPL

John D. Powell

Caltech, Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena CA 91109-8099

The Jet Propulsion Laboratory's (JPL) Software Quality Improvement Measurement and Benchmarking subgroup is charged with the responsibility of developing, infusing and supporting quantitative analysis of software metrics to foster software improvement of existing and future software projects and the institution as a whole. A specific thrust within this effort is the measurement, analysis and improvement of defect trends during development of software at JPL. This paper will describe the approach being taken towards those ends.

The iterative software development lifecycle that has largely replaced the waterfall lifecycle at JPL requires a new rationale for the treatment of defects. This new rationale views defects as a valuable (and necessary) development and management tool as opposed to viewing them simply as an unwanted development byproduct. The defect employment approach is predicated on the iterative lifecycle's approach to learning about a software system while building it. Employing defects within JPL's current style of software development seeks to use defects as a source of learning throughout the lifecycle as well as a means of prediction across a project's lifecycle iterations and across multiple projects over time. This defect philosophy is in sharp contrast to defect measurement and analysis approaches that fit the waterfall lifecycle's concentration of learning in the requirements phase before building the software system. The waterfall lifecycle defect approaches are forced to treat iterations in JPL's iterative lifecycle as a series of consecutive waterfalls. However, this adaptation of the waterfall lifecycle does not fully or accurately capture and analyze defect data in a manner that allows JPL software development teams to fully benefit from new information gained (learning) during development and the inherent flexibilities within the iterative lifecycle to of maximize quality while remaining on-time and within budget. The new approach (Employment of Defects) centers around two main foci, 1) the *employment* of defects beyond simply a focus of elimination will lead to improvements in software quality and dramatic reduction of programmatic risks such as cost and schedule, 2) crafting an approach to effective address the development in software in the specific iterative style commonly used at JPL today. Many facets of the new approach formally integrate informal development team knowledge currently used (but not captured) to augment the waterfall rational for software defect analysis in an iterative development environment. The new approach presented in this paper represents a critical step forward towards the goals by specifically accounting for deferral of defect repairs until a later lifecycle iteration, distinctions between defects that provide critical new information about the system (learning defects) and defects that must simply be eliminated as well as a various analyses regarding the related repairs that must be performed in relation to the number, types and timing of defects encountered.