CVS Release History Data for Detecting Logical Couplings
By Harald Gall, Mehdi Jazayeri and Jacek Krajewski

Summary:

Taken unto themselves, the alterations tracked by a CVS system provide only the basic information for versioning of a software product. However when the information is tracked by CVS is filtered through a three tier system consisting of a Quantitative analysis for monitoring growth and changes in behavior, followed by Change Sequence analysis which reveals common changes across potentially disjoint modules and sub modules, and then finally composed with the information garnered in a Relation analysis which would compare classes based on the CVS release history and the prior information, flaws in the software design can be revealed.

Based on the 28 month study of PACS, a java project which by projects end had approximately one half million lines of code and had gone from two thousand to fifty-five hundred classes, the QCR approach based on the three phases described above was intended to reveal dependencies in software modules both within modules themselves and across module boundaries. To then validate the results suggested by the application of the above approach the findings were then shared with the software developers of the PACS project (i.e. the identified logical couplings). An added benefit of the approach would appear to be that the results can be derived without the inspection of individual lines of code by a user, as the metrics can be generated automatically assuming that the underlying infrastructure of the programming language lends itself to a module hierarchy, in this case classes. One of the key aspects of this investigation was to determine the logical couplings between classes. Prior to this, dependencies were identified by coupling or cohesion methods or simply by the knowledge of the developer involved, with no formal information recorded outside of the developer's domain knowledge. By analysis of the three phase system described above architectural flaws consisting of functional redundancy, god-classes with more than one thousand lines of code and other previously un-determined dependencies were exposed. The logical coupling itself was depicted graphically as well, with the classes themselves represented as nodes and the edges between as discovered logical couplings. Further to this the thickness of the lines was coupled to the number of discovered dependencies, i.e. the thicker the line the greater the number of dependencies. This depiction yielded candidates for further investigation. A further extension on this process was to also observe the changes in time of the logical couplings, to provide clues to the breakdown of the software architecture as well as to depict the evolution of the system design over the 28 month period.