Project Leap: Addressing Measurement Dysfunction in Review

Carleton A. Moore

February 4, 1999

The software industry and academia believe that software review, specifically Formal Technical Review (FTR), is a powerful method for improving the quality of software. FTR traditionally is a manual process, recently computer mediated support for review has had a large impact on review. Computer support for FTR reduces the overhead of conducting reviews for reviewers and managers. This reduction in overhead increases the likelihood that software development organizations will adopt FTR. Computer support of FTR also allow for the easy collection of empirical measurement of process and products of software review. These measurements allow researchers or reviewers to gain valuable insights into the review process. With these measurements reviewers can also derive a simple measure of review efficiency. A very natural process improvement goal might be to improve the numerical value of review efficiency over time.

My research group, the Collaborative Software Development Laboratory (CSDL), developed a computer supported review system called Collaborative Software Review System (CSRS)[5]. CSRS is a computer supported software review system implemented in UNIX with an Emacs front-end. It is fully instrumented and automatically records the effort of the participants of the review. The moderator of a review using CSRS may define and implement their own review process. Dr. Tjahjono and Dr. Johnson used CSRS to investigate the effectiveness of group meetings in FTR[4]. CSRS and systems like it induce measurement dysfunction due to a lack of attention to the human factors of automated collection of review metrics.

After looking closely at review metrics we started to worry about *measurement dysfunction[1]* and reviews. Measurement dysfunction is a situation in which the act of measurement affects the organization in a counter-productive fashion, which leads to results directly counter to those intended by the organization for the measurement. Some of the different types of measurement dysfunction that can occur in software review are defect severity inflation/deflation, defect density inflation/deflation, artificial defect closure, and reviewer preparation time inflation/deflation[3]. Reviewers or project managers may feel pressure to report good defect severity or density levels. They can miss report the severity or number of defects to affect the metric reported to management. I have witnessed defect severity reduction in industry. An organization's policy was a project could not pass any milestone with any open defects of critical severity. The project manager talked to the developers and got them to reclassify all the critical defects as high so the project could pass the milestone on date. Since all the critical defects were now high they did not get fixed first. This

reduced the quality of the overall product.

How can we reduce the threat of measurement dysfunction in software review without loosing the benefits of metrics collection? Project LEAP[2] is our attempt at to answer this question. It investigates the design, implementation, and evaluation of tools and methods for empirically-based, individualized software developer improvement. A major factor in developer improvement is receiving external reviews of products. I developed the Leap Toolkit, a Java based personal process improvement tool that supports distributed review of documents. It also support the generation of review and process checklists and patterns.

The Leap toolkit gives the developer or reviewer full control on the information they share with others. In small groups they trust, developers can share all their data. In larger groups that the developer does not trust they have three options: sharing the data they are comfortable with, such as checklists and patterns, obfuscating their data, or falsifying their data. The Leap Toolkit give the developer complete control over the data they share with others. It will also provide an obfuscater. Once the data is obfuscated the identity of the developer or reviewer is removed. The last option, falsifying data, is possible since Leap shows the user exactly what will be sent and allows the user to change any entry.

We are evaluating Project Leap in two ways. First, we are introducing the Leap Toolkit to industry and academia. The adoption of the Leap toolkit in industry will be an interesting case study. The design features that reduce measurement dysfunction allow the users to produce "good" answers to management. The case study may provide evidence that developers keep accurate personal data even while maintaining an organizational set of data.

Second, we are building the Leap data obfuscater and Web based shared data repository. The Leap data obfuscater removes all the identifying information from Leap data files, yet retains the accuracy of the data. We hope that Leap users will upload their obfuscated data files to our web data repository and share their insights into review and development. We will have a repository of checklists and patterns for software development.

Project Leap represents an explicit look at human factors issues in empirically based measurement, in the attempt to collect more accurate and useful data for personal process improvement.

References

[1] Robert D. Austin. *Measuring and Managing Performance in Organizations*. Dorset House Publishing, 1996.

[2] Philip Johnson and Cam Moore. Project leap: Personal process improvement for the differently disciplined. URL http://csdl.ics.hawaii.edu/Research/LEAP/LEAP.html.

[3] Philip M. Johnson. Measurement dysfunction in formal technical review. Technical Report ICS-TR-96-16, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii 96822, 1996.

[4] Philip M. Johnson and Danu Tjahjono. Does every inspection really need a meeting? *Journal of Empirical Software Engineering*, 4(1):9–35, January 1998.

[5] Danu Tjahjono. *Exploring the effectiveness of formal technical review factors with CSRS, a collaborative software review system*. Ph.D. thesis, Department of Information and Computer Sciences, University of Hawaii, August 1996.