Project Leap: Addressing Measurement Dysfunction in Review

Carleton Moore
University of Hawaii, Manoa

1 A Problem with Reviews

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 allows 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) (Tjahjono 1996). 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 (Johnson and Tjahjono 1998). Our experiences with CSRS caused us to start thinking about the issues of measurement dysfunction.

After looking closely at review metrics, we started to worry about *measurement dysfunction* (Austin 1996) 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 where the severity of a defect is raised because the reviewers want it fixed faster; Defect severity deflation, where the severity is lowered so the project may complete a milestone. Defect density inflation, where many minor defects are reported or the size of the document reduced to bring the document up to norms. Defect density deflation, where the number of defects are under reported or the size is over reported to make the document look better. Other types are artificial defect closure, and reviewer preparation time inflation/deflation (Johnson 1996). 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.

2 Project LEAP

How can we reduce the threat of measurement dysfunction in software review without losing the benefits of metrics collection? Project LEAP (Johnson and Moore 1999) 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. Project Leap has produced a toolkit for personal process improvement that supports distributed review of documents. It also supports 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 that 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. It will also provide an obfuscater. The developer runs their original data file through the obfuscater and all identifying data is changed so there is no way to associate the Leap data with the developer. 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. For example, a developer can copy their accurate data and modify their productivity to match some corporate standards.

3 Evaluation

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 containing different values.

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.

4 Preliminary Results

The Leap toolkit is in use in a senior level ICS class this spring at the University of Hawaii. 19 students are using it to track their software engineering work. So far we have had good results from our students. We will track them to see if they continue to use Leap after they finish the class.

We are using Leap for document review in CSDL. We have even used the Leap toolkit to review the Leap toolkit. One of my colleagues in CSDL feels uncomfortable about sharing their time data with the rest our group, yet they are very comfortable using the Leap toolkit for reviewing since it allows them to not reveal their time data to the rest of the group. During one particular review, my colleague collected very accurate time and defect data. Leap allowed my colleague to share the defects with the author of the document while keeping their time data private.

Our use of the Leap toolkit will enable us to improve our own review practices. Another colleague made the following remark during one of our weekly meetings. "I have gathered 38 defects on the PSP journal article [using Leap]. I am interested in seeing what defects the external reviewers find, and comparing

these two datasets to learn how we can improve our own review process." By using the Leap toolkit we are able to easily compare the defects we find in review against the defects the journal reviewers find. We should be able to develop checklists to help us find and remove the defects found by those external reviewers.

5 Current Status

Leap is publicly available at http://csdl.ics.hawai.edu/Tools/LEAP/LEAP.html. Since we publicly announced the release of Leap in December 1998, several individuals, not associated with CSDL, have adopted Leap.

We are looking for industrial partners who are willing to use Leap and allow us to record their experiences. We are going to use Leap to help teach a graduate software engineering class in the fall of 1999. The students will be using Leap to conduct reviews of their fellow classmate's programs.

We are developing the Leap obfuscater and designing the Leap data repository. These two tools should help with the maintenance and distribution of valuable review checklists.

6 References

Austin, R. D. (1996), *Measuring and Managing Performance in Organizations*. New York, Dorset House.

Johnson, P. M. & Moore, C. (1999). Project leap: Personal process improvement for the differently disciplined., http://csdl.ics.hawaii.edu/Research/LEAP/LEAP.html.

Johnson, P. M. (1996). Measurement Dysfunction in Formal Technical Review. Technical Report ICS-TR-96-16, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii 96822.

Johnson, P. M. & Tjahjono, D. (1998). Does every inspection really need a meeting? *Journal of Empirical Software Engineering*, 4 (1), 9–35.

Tjahjono, D. (1996) *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.