Erika S. Mesh – Proposed Research

In 2001, Dr. Geoffery Chang published groundbreaking research mapping the structure of a protein found in cell membranes. Then, "the dream turned into a nightmare." Chang and his lab retracted five papers due to a software bug that had swapped two columns of data (Miller, 2006). In commercial organizations, such an error may still occur, but would likely be detected before release. Unfortunately, while traditional software engineering (SE) practices such as unit testing have been shown to enhance the quality of scientific software (Sletholt and Hannay, 2012), adoption of modern SE techniques has been limited and a gap has emerged between the scientific and SE communities (Kelly, 2007).

Past research offers some explanations for this gap. A survey by Heaton et al. (2012) suggests that lack of practice adoption is tied to a lack of knowledge regarding specific techniques. Kelly discusses how scientific software must adapt quickly to advances in the domain while also being carefully managed to ensure data integrity. These mixed goals make the determination of an appropriate SE process challenging (Kelly, 2011). Proposed solutions include enumerations of recommended practices (Heroux et al., 2009), customizations of specific techniques (Li et al., 2011), and increased training (Kelly, 2007).

While valuable, these approaches address only the short-term issues. Lists of practices offer a good starting point, but similar situations in other industries have shown that exhaustive manuals are an ineffective means to institute process improvement (Orr, 1996). Custom-tailored practices are effective, but create a maintenance problem and dependence on SE experts. Targeted training is useful, but we cannot presume to train all scientists to also be software engineers. To address the long-term needs of scientific researchers, I propose incorporating existing efforts into a larger strategy that will close the chasm and allow scientists to drive their own software process improvement (SPI) activities.

To provide scientists with the support they require, we must first understand the applicability of existing SPI strategies and the characteristics that must be considered to guide scientific SPI. Specifically, I will address the following:

1. Of traditional SE project characteristics, which are applicable as attributes to describe scientific research projects?
2. What other characteristics can be used to define specific instances of scientific research projects and how software is utilized in these projects?
3. What correlations exist between project characteristics (or sets thereof) and the successful enactment of specific SE practices?

Answering these questions will provide the necessary inputs to construct a theoretical model of the factors that define scientific software projects. This includes any relevant organizational and technical characteristics. However, in order to ensure the validity of any studies based upon this model, the results must be grounded in established SE principles and account for many possible combinations of variables. For example, it is reasonable to hypothesize that the applicability of source control practices will correlate to the size of the research project; however, this is only one of many possible hypotheses. Conducting experiments to test each hypothesis would be inefficient and may overlook non-traditional characteristics. A more effective methodology is to discover these relationships via a broader, deductive study. To this end, my research will leverage a grounded theory approach to ensure that each research construct, data collection procedure and conclusion is clearly derived from its predecessors.

Formal grounded theory (Corbin & Strauss, 1990) suggests that the research process itself is important and that hypotheses should evolve over time. In order to support this systematic evolution of theory, data collection and analysis are executed simultaneously to prevent planned analysis from creating bias in in the results. Preliminary research results have demonstrated the usefulness of grounded theory in ensuring construct validity in SE process research by providing traceability between mainstream SE process principles and survey design (Mesh, 2012).

In order to gain additional insight into the defining characteristics of scientific software projects, data sources beyond surveys must be considered. Open and axial coding of related work will provide a strong conceptual base. Similarly, longitudinal case studies of scientific research projects will uncover trends not visible via the snapshots provided by surveys.

Individually, these studies will provide useful correlations and immediate SPI lessons learned for the scientific community. Unified via consistent coding and analysis, my approach will establish protocols and a base of strong empirical data. This data can then be used to generate and test new hypotheses regarding SE process improvement for scientists. This incremental delivery of value and methodology refinement is also inline with my personal goals: to provide both short- and long-term value to the scientific and academic communities.

References

J. Corbin and A. Strauss, "Grounded Theory Research: Procedures, Canons, and Evaluative Criteria," *Qualitative Sociology*, vol. 13, no. 1, 1990.

D. Heaton, et al., "The Relationship between Development Problems and Use of Software Engineering Practices in Computational Science & Engineering: A Survey," in *Proceedings of the First Workshop on Maintainable Software Practices in e-Science, Co-located with 8th IEEE International Conference on eScience*, 2012.

M. A. Heroux and J. M. Willenbring, "Barely sufficient software engineering: 10 practices to improve your CSE software," in *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, 2009, pp. 15–21.

D. Kelly, "A Software Chasm: Software Engineering and Scientific Computing," *Software, IEEE*, pp. 0–2, 2007.

D. Kelly, "An Analysis of Process Characteristics for Developing Scientific Software," *Journal of Organizational and End User Computing*, vol. 23, no. 4, pp. 64–79, 2011.

Y. Li, N. Narayan, J. Helming, and M. Koegel, "A domain specific requirements model for scientific computing," in *Proceeding of the 33rd international conference on Software engineering - ICSE '11*, 2011, p. 848.

E. Mesh. "Development of a Software Process Maturity Model for Scientific Research Projects." M.S. Experience Report. Software Engineering Department, Rochester Institute of Technology, August 2012.

G. Miller, "A Scientist's Nightmare□: Software Problem Leads to Five Retractions," *Science*, vol. 314, no. 5807, p. 1856, 2006.

Orr, J. E. *Talking About Machines: An Ethnography of a Modern Job (Collection on Technology and Work)*. Cornell University Press, 1996.

M. Sletholt and J. Hannay, "What Do We Know about Scientific Software Development's Agile Practices?" *Computing in Science and Engineering*, vol. 14, no. 2, pp. 24–37, 2012.