Merely Mining? Or Sharing Insight?

Tim Menzies Laurie Williams Tom Zimmermann

Come, sit down. We'd like to learn more about the world and, if you have a little spare time, we would like your help in achieving that goal.

What can we talk about? Well, we just edited a book on data science for software engineering and (if you are reading this) then its clear you just bought a book on that topic. So lets start there- what can we share about software analytics?

What we will say in this chapter there are problems sharing all the things we might want to share. To show that:

- First we will list all the things we might want to share (data, methods, models, insights);
- Next, we will discuss what is currently usually shared (method are often shared, data and models are sometimes shared, but insight is currently not shared very often).

This will lead to the main point of this chapter; i.e. for the next generation of software analytics it will be important to move beyond merely mining in order to explore how to share insights.

What might we share?

There are many things we might share with each other including stories of our children, etc. But if we are interested in software analytics, our conversation might turn to sharing data, methods, models, and insight.

Why share data? General models often need tuning with local data. Hence, we might offer to *share project data* with each other. This data sharing is particularly useful if one team is using a technology that is new to them, but has long been used by the other. Recent work on transfer learning shows that is now technically possible to transfer such lessons learned from project to project-- even when the projects use different terminology to describe their work [7].

Why share methods? Tt is not enough just to share data in order to share ideas. This data has to be summarized into actionable statements- which is the task of the data scientist. When two such scientists meet for lunch, they might spend some time discussing the tricks they use for different kinds of data mining problems. That is, we might *share analysis methods* for turning data into models.

Why share models? Another possibility is that we might be reading the software engineering literature and want to *share models* about software development. For example, suppose you have just read Barry Boehm's book on Software Process Dynamics [6]. That book documents a power law of software that states that larger software projects take exponentially longer to complete than smaller projects. Accordingly, we might discuss if development effort for larger projects can be tamed with some well-designed information hiding.

Why share insight? Finally, we might want to share our *insights* about software projects. For example, we might have just read Fred Brooks' book on The Mythical Man Month [5]. This book documents many aspects of software project management including the famous Brooks' Law which says "adding staff to a late software project makes it later". To share such insights about management, we might swap war stories on (e.g.) how upper management tried to save late projects by throwing more staff at it. Shaking their heads ruefully, we might remind each other that often the real problems are the early lifecycle decisions that crippled the original concept.

What do we share?

In theory, decades of research in empirical software engineering and mining software repositories lets us share all of the following:

- *Sharing insights*. Specific lessons learned or empirical findings. An example is that in Windows Vista it was possible to build high-quality software using distributed teams if the management is structured around code functionality [1].
- *Sharing models*. One of the early models was proposed by Fumio Akiyama and says that we should expect over a dozen bugs per 1,000 lines of code [2]. In addition to defect models, plenty of other models (for example effort estimation, retention and engagement) can be built for software.
- *Sharing methods*. Empirical findings such as insights and models are often context specific, e. g., depend on the project that was studied [3]. However, the method ("recipe") to create findings can often be applied across projects. We refer to "methods" as the techniques by which we can transform data into insight and models.
- *Sharing data* By sharing data, we can use and evolve methods to create better insight and models.

However, in practice, when a recent Dagstuhl seminar on software analytics [4] reviewed the literature, we found:

- Many initiatives devoted to *sharing data*;
- Much recent work on *sharing models*;
- There exists many texts and toolkits that let us *share methods*.

What we did not see were many attempts to *share insights*. In fact, this seems to be a gaping hole in the literature There are many excellent data mining textbooks that lead the reader through a detailed description of data mining algorithms. That said, those textbooks rarely step back to offer higher-level insights. Yet, at the Dagstuhl meeting, very little of our time was spend discussing algorithmic details. Instead, the participants there were focused on defining and refining a set of insights that define modern data science for software analytics. Further, when we looked for any text that listed all those insights, we found none.

References

1. Christian Bird, Nachiappan Nagappan, Premkumar Devanbu, Harald Gall, Brendan Murphy. 2009. Does distributed development affect software quality?: an empirical case study of Windows Vista. Communications of the ACM 52(8), pages 85-93.
2. Akiyama, F. (1971, August). An Example of Software System Debugging. In IFIP Congress (1) (Vol. 71, pp. 353-359).
3. Menzies, T.; Butcher, A.; Cok, D.; Marcus, A.; Layman, L.; Shull, F.; Turhan, B.; Zimmermann, T., "Local versus Global Lessons for Defect Prediction and Effort Estimation," in Software Engineering, IEEE Transactions on , vol.39, no.6, pp.822-834, June 2013 doi: http://dx.doi.org/10.1109/TSE.2012.83
4. Harald C. Gall, Tim Menzies, Laurie Williams, Thomas Zimmermann: Software Development Analytics (Dagstuhl Seminar 14261). Dagstuhl Reports 4(6): 64-83 (2014)
5. Frederick P. Brooks, Jr.. 1995. The Mythical Man-Month (Anniversary Ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
6. Barry Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, and Bert Steece. Software Cost Estimation with COCOMO II. Prentice-Hall, 2000. ISBN 0-13-026692-2
7. Jaechang Nam and Sunghun Kim. 2015. Heterogeneous defect prediction. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015). ACM, New York, NY, USA, 508-519. DOI=http://dx.doi.org/10.1145/2786805.2786814