Paper Review CS846, Jan. 16
Software analytics and its application in practice

Wenhan Zhu (Cosmos)
w65zhu@uwaterloo.ca
University of Waterloo
Waterloo, Canada

ABSTRACT

This week's paper reviewed here are: **Software analytics and its application in practice** by *Dongmei Zhang, Tao Xie* and **No silver bullet: Essence and accidents of software engineering** by *Fred Brooks*.

Zhang and Xie talked about the aspects of software analytics and how to approach data and apply the results in practice. *Brooks* article expressed his view on the problem of solving software related issues and his view on that the problem will always exists and there will not be a method that will end this issue.

1 SUMMARY

The chosen combined readings address different problems in the field of software engineering. *Zhang and Xie* talked about the different perspectives of software analytics and their recent experiences in applying software analytics in practice. They categorized software analytics in 6 perspectives, research topics, target audience input, output, technology pillars and connections to practice. The first 4 perspectives are very straight forward. The later two are further discussed in detail by the authors by they belong to the perspectives of software analytics. Technology Pillar is the techniques used by software analytic. These techniques evolve over time and the technologies have their own advantages and disadvantages. There are active research to improve the used technologies. Connection to practice describes the perspective that software analytics are tied with four real elements in the scope of data, problems, users and tools. The authors gave some valuable learnings from putting software analytics in action. The most important part is to identify the essential problem then have an active development feedback cycle to iteratively create solutions. Finally it is essential for the researchers to have a deep understanding of the domain of the subject to perform the analytic.

Brooks article has many of his views of the software engineering world a few decades ago. Many of his predictions were true and some of his insights are still very valuable in today's standard. *Brooks* made a metaphor of software evolving into an unmanageable mess as people turning into werewolves. However, unlike the folklore, there are no silver bullet to the problem of software artifacts. He decomposed software engineering to two parts, one is the artifact or side-effects of current software engineering tools that is tied to a specific set of circumstances and often does not apply between programming languages and practices. The other is the essential problems that is unavoidable as long as we are still producing software. He gave his view on the advancements of technology at the time and they could not solve essence problems of software engineering. In the end, he advocates for the need for great designers of problems. He believes that it is the great designers that creates produces great software.

2 THOUGHTS

Many idea from *Zhang and Xie* seem to originates from the open source community and the agile development protocol. The description about the learning they found from software analytics in practice seems to suggest that the software analytic itself is a form of software development. I think the perspectives they gave captures the scope of software analytics and shows how it is used and what information it needs and provides in a generic setting.

I really liked *Brooks* paper. It showed great insight on what he though about software engineering. His arguments about object oriented programming predicted the problems it solved and the problems it couldn't address. But I think there are still some points that he missed. One is the graphical user interface. He failed to predict convenience graphical user interface brought to everyday computing and more people can learn programming more easier then before. But he did see that "smart" programming environments (IDE's of today) does not solve essence problems. From my point of view, I agree that these tools only scratch itches of the development process but does not address essential problems on software engineering. Large system still needs to build from ground up and the design is more important than the smaller details. And based on the time the article was published, the current open source development model did not appear in the software engineering world. By looking at history, it seems to me that open source project was able to introduce many new ideas and give solutions to some of the old software engineering problems. With the involvement of more people is seems like a ecosystem that wasn't something imaginable at the time given that most software are still hardware specific. Many historical affects are still valid today that software like *xterm* needs to support terminal standards that are decades old and probably no one still uses them.

I think the open source community really introduced a new way to look at software development. Compared to the traditional way, open source development have contributors that will work on problems that they want to solve and software will evolve to what people need. So instead of having to evaluate what software to develop and what features people what, in the open source world people will decide on their own. And in such process, the essence part of software engineering in my mind follows the pattern of natural evolution such that the good ones will stay and the bad ones will become irrelevant or extinct.

3 RATINGS

I would give *Zhang and Xie*'s paper a 3 out of 5. It talks about their view on software analytics and their learnings of applying it in practice. However, I don't find it giving me detailed understanding of what could software analytics provide when used in practice.

I really enjoyed *Brooks* article. I would rate it a 5. It's an excellence paper that discusses many ideas which are still applicable today.

4 QUESTIONS

I've attended Daniel Berry's talk on his view of software engineering. He mentioned that in the beginning of time, people believed that by designing better programming languages people could produce good software. But it failed and people eventually realized that it's a own field of study of how to produce good software. Then he moved to the study of requirement engineering which works on how to extract specific requirements from language descriptions to software engineering problems which can then be done by software engineers. As of today we still do not have a good metric to tell good software from bad ones. And the type of software is also shifting, from business practices from the early days to mobile apps about a decade ago and now web applications. In these context the type of software changes and so does many things. Do essence also shift in this manner? A concrete example would be it's probably a huge problem in the past where memory is wasted but now as long as there are no memory leak and the magnitude of memory usage is still on the same level no one would care if it takes more memory as long as it gets the job done.