Principles for collaboration between Researchers and Software Development

Brendon Bradley & Sung Bae

November 2017

Context:

One of the strengths of our earthquake resilience research collaboration at UC is the close connection between researchers (grad students, postdocs, academic staff) and software developers. The fruitfulness of this collaboration is reflected in the values and culture that we express toward each other and the research endeavour. Because personnel will come and go this document aims to provide the underpinning principles to ensure the most fruitful interactions.

- **We are all in this together:** It is our collective efforts and research outputs that make our endeavours ultimately successful – no one person can achieve without the significant prior work that others have contributed (be it research outcomes or code-base development), and the active collaboration that occurs during undertaking research. Significant active collaboration should be reflected in joint authorship of research publications, and minor collaboration or prior contributions via paper acknowledgements. The role of software developers in co-authorship should be seen in the same manner as for research students of academic supervisors.
- **We are research outcome-driven:** There are many activites that are interesting, but it is important that we retain focus on the topics that are both interesting and of research relevance. You may not realize it (because you are purposefully sheltered from it!), but there are significant financial pressures to obtain and maintain the funding to undertake our research, and it is only by delivering on such research expectations (through publications principally, but also other means) that such funding can continue to be obtained. This applies both for software development and for investigator-led research. If you have an interesting idea that is not currently tasked then convince others that it needs to be considered (or we need to get funding to do so).
- **Rapid prototyping – production software:** As a research enterprise we focus on developing disruptive technologies. This poses challenges in the development of software workflows – because the aims of the research endeavour mean that our needs continually change. Therefore, there is conventionally a need for rapid prototyping to test many ideas, and very few of them will snowball into the development of production software. Rapid protyping also will often only involve a few people, but further development will naturally require involvement of others to spread expertise, for resourcing flexibility and long-term productive increases.
- **Functionality -> integration -> efficiency:** Prototyping code should initially focus solely on functionality required to undertake a research outcome-driven task. Revisions to the prototype to seemlessly incorporate it into a code-base will focus on integration and abstraction. When, and if identified necessary, the final step will be computational efficiency.
- **Prioritisation:** Its a brick-by-brick process to achieve technological advancement. Tasks that provide significant stability/reliability of critical workflows and advancing new workflows to achieve mission critical research tasks will receive the greatest priorities. If a task needs to be prioritised or de-prioritised in your opinion then talk to, and convince others to come to concensus.