Transforming Research Code to A Modern Web Architecture - Pipetools
Paulus Lahur, IMT, CSIRO
About Us

- CSIRO
  >> Information Management and Technology (IMT)
  >> Scientific Computing
  >> Science Computing Applications
- Research Software Engineering
- 16 people in the team
- Supporting researchers
  - Wide range of domains
  - Collaborate (6-month cycle) or dedicated
- Demand for our service is far higher than supply
About The Project

• Transform existing research code into web-based application
  • Access on any device, interactive, highly responsive, rich set of features …
  • The research code is the implementation of mathematical models of mixed flow of fluid and solid inside a pipe
• People: 8 (2 on client side, 6 on IMT side). Effective: 2 people (IMT)
• Time: 1.5 years
• The project was cancelled due to lack of funding
• Nevertheless it’s a very valuable learning process ➔ lessons learned (at high level)?
The Physics

Complex flow of a mixture of liquid and solid particles (fine and coarse) inside pipeline
The Research Code

- The code is already functioning as Windows application
- C++
- Library: Numerical Recipe
- No unit tests ➔ This is very important!
- Documented (Doxygen)
Building The Application

What we did
• Explored solutions (beginning)
• Set up infrastructure (beginning)
• Implemented back end: Make API, interface with core code
• Implemented front end: Call API, GUI
• Improved the core code
  • Refactored the code towards best practices
  • Completed some functionalities
  • Debugged

Technologies
• Docker
• Node.js (Javascript)
• Nginx
• Swig to interface C++ and Javascript
• Jenkins for continuous deployment
• Git in BitBucket
The Application’s Layout

- **Infrastructure**
  - Jenkins
    - Continuous integration & deployment
  - Front End Server (JavaScript: Vue, Bulma, Vuex, Plotly, MathJax, MathJS)
    - Docker container for Front End
  - Back End Server (JavaScript: Node.js, Express)
    - Docker container for Back End
  - SWIG
  - Research Code (C++)

- **API call from Front End to Back End**
  - Javascript on the Back End:
    ```javascript
    const funcMap = {
      'dpdx': dpDx,
      ...
    };...
    result[i] = pipe.dPdx(model.suspension, parseFloat(x)) / 1000.0;
    ```

- **GUI**
  - Calculate

- **SWIG interface file**
  ```
  %module "pipetools"
  
  %include "include/Pipe.h"
  
  %include "include/Pipe.h"
  ```

- **C++ header file**
  ```
  virtual double dPdx(CSuspension *s, double v);
  ```
GUI: Input
Lessons Learned

• Resource estimate is really hard, especially for research code
  • Most of the time is spent at the early stage, so perhaps it’s better to split into 2 stages (exploratory stage and implementation stage)?

• People resource is very limited
  • Automate everything that can be automated (build, deployment)
  • Find generic solution that can be reused (at least a portion of it) → find common elements

• Wide range of coding proficiency among researchers, especially for languages like C++
  • Need to provide help at early stage of coding stage of the research code → how?

• From experience in other projects: research code quality is strongly influenced by the already existing code being used (framework / library).

• Idea: make a simple code framework that can be used as a starting point of research code as well as an education tool of best practices.
Thank you

Acknowledgements
Research code: Lionel Pullum (Mineral Resources, Clayton VIC)
Project manager: Andrew Chryss (Mineral Resources, Clayton VIC)
Team lead on IMT side: Daniel Collins (IMT, Kensington WA)
Front End: Kieran Lomas (IMT, Clayton VIC)
Back End: Paulus Lahur, Sam Moskwa (IMT, Clayton VIC)
Infrastructure: Dylan Graham, Andrew Spiers, Sam Moskwa (IMT, Clayton VIC)