

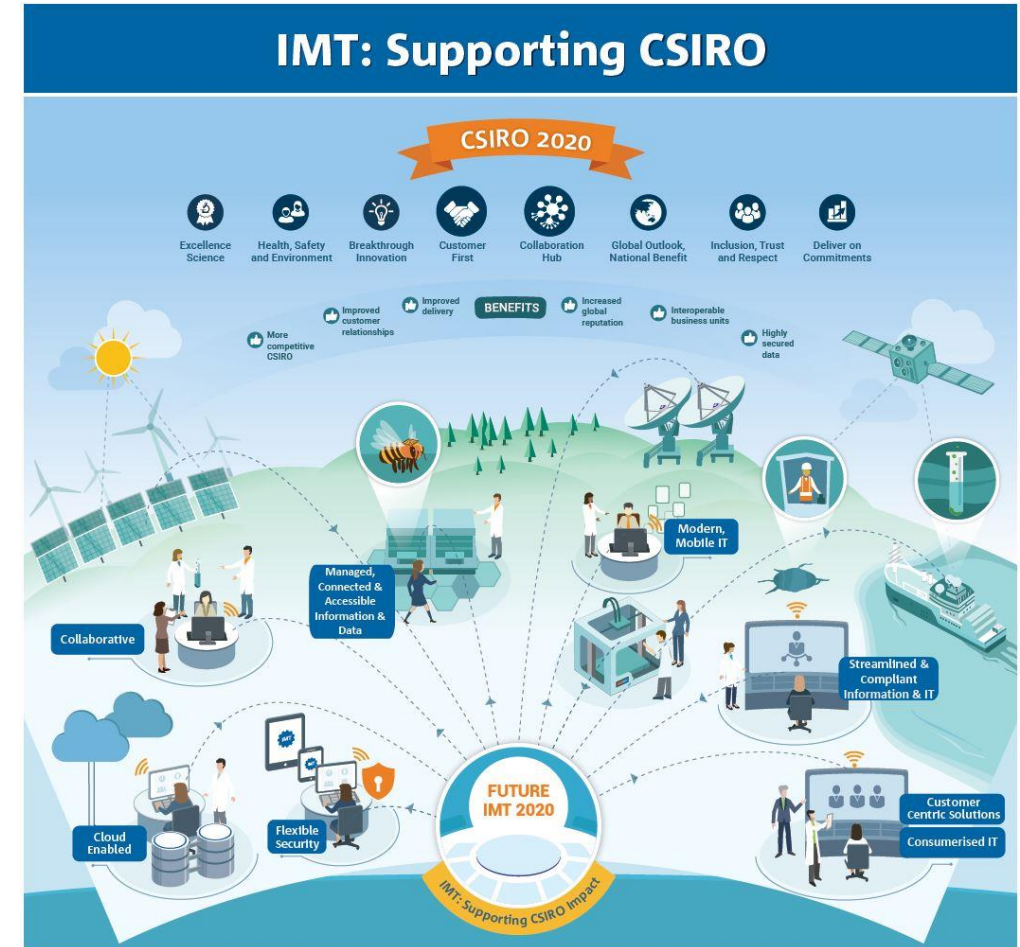


Transforming Research Code to A Modern Web Architecture - Pipetools

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)

```
1538 // \return True if it succeeds, false the number of iterations are exceeded.
1539
1540 bool CSuspension::Wilson2LDeposLocPoint(CPipe *pp, double c, double &V, double &dPdx)
1541 {
1542     double beta = FindBeta(cb() * m_ap.m_dBedFrac, c);
1543
1544     // check to see if the yield stress is sufficient to cause bed movement
1545     // if so pressure will equal the wall shear stress over the hydraulic diameter
1546     if (YieldDepLoc(pp, c))
1547     {
1548         V = 0.0;
1549         dPdx = 4.0 * m_f.YieldStress() / HydraulicDia(pp->Dia(), beta);
1550         return true;
1551     }
1552
1553     double D = m_ap.m_bHydDia ? HydraulicDia(pp->Dia(), beta) : pp->Dia();
1554     double Va = 4.0;
1555     double Vanew = 1.5 * Va;
1556     double lamda0, Jos, ReGen;
1557     double eta;
1558     int n = 0;
1559
1560     do
1561     {
1562         Va = Vanew;
1563         ReGen = m_f.Re(Va, D);
1564         lamda0 = m_f.Frict(ReGen, D, pp->Rough() * D / pp->Dia()); // Use pipe relative roughness
1565         eta = m_f.Frict(ReGen, D, m_ap.m_dIntCharDia * D / pp->Dia()) / lamda0;
1566         Jos = Wilson2LXs(beta, eta) * PlugGradient(pp, 0.0);
1567
1568         V = sqrt(2.0 * D * Jos / (m_f.Density() * lamda0));
1569         Vanew = V / A(beta);
1570     } while ((fabs((Vanew - Va) / Va) > m_ap.m_dEPS) && (n++ < m_ap.m_nMAXITS));
1571
1572     // dPdx = Wilson2LYs(beta, eta) * PlugGradient(pp, 0.0);
1573     dPdx = Wilson2LYs(beta, eta) * PlugGradient(pp, 0.0) + rhof() * 9.81 * sin(pp->Theta());
1574
1575     // If percolation is specified use Eyler & Lombardo's extension using the Ergun equation
1576     // to calculate slip, Ideally you should iterate on the slip velocity as well but in
1577     // reality it converges almost immediately so just use the value estimated from the
1578     // no flow case above.
1579
1580     if (m_ap.m_bBedPerc && (n < m_ap.m_nMAXITS))
1581     {
1582         double Vslip = Ergun(dPdx); // estimate of the slip velocity
1583         double Rep = m_f.Re(Vslip, d()); // characteristic particle Reynolds number
1584         double Cd = GetCharSolid()->Cd(Rep); // drag coefficient on this
1585         double a = A(beta); // area above the bed
1586         double Xs, Ys; // new coordinate forms
1587         // double pi = 4.0 * atan(1.0); // guess
```

Building The Application

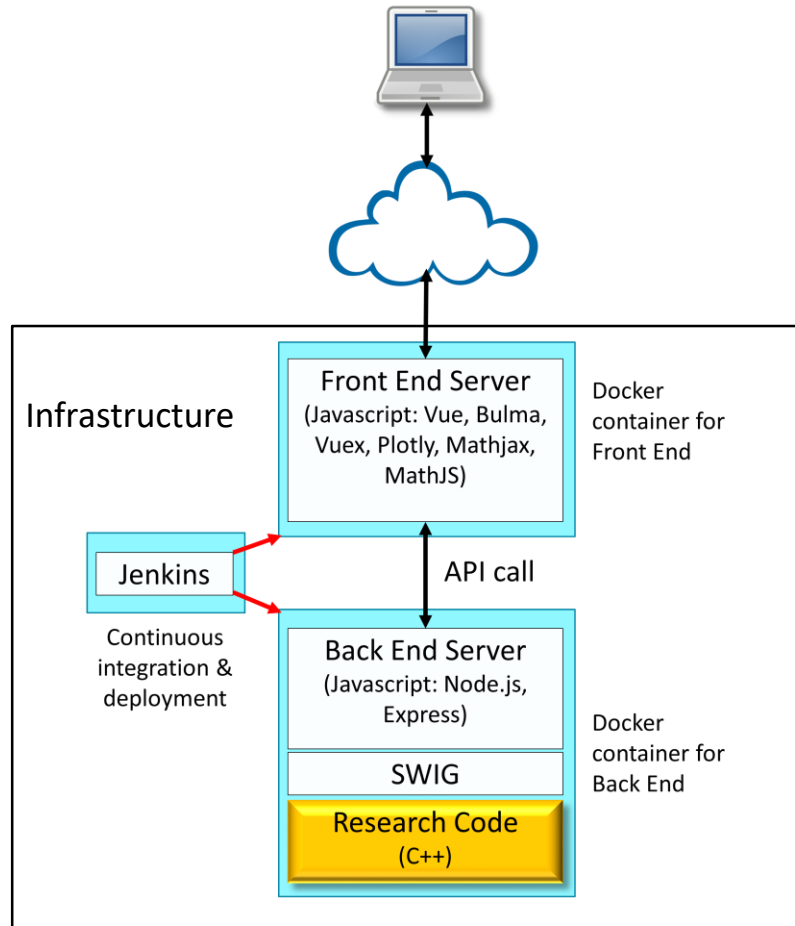
What we did

- 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

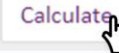
Technologies

- Docker
- Node.js (Javascript)
- Nginx
- Swig to interface C++ and Javascript
- Jenkins for continuous deployment
- Git in BitBucket

The Application's Layout



GUI



API call from Front End to Back End

`http://host_name:port_number/calc/dpdx`

Javascript on the Back End

```
const funcMap = {
  'dpdx': dpDx,
  ...
};
...
result[i] = pipe.dpDx(model.suspension, parseFloat(x)) / 1000.0;
```


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

 PipeTools™ Beta

Material SpecPipeline CalcsPipeline SpecPlotsSupportLog out

Suspension ? ^ x

File *unsaved* v

ρ_m 2392 kg.m⁻³ v

c 0.2 - v

☒ v/v ☐ w/w ☐ w/v

Solids ? ^ x

File *unsaved* v

Particle size

☐ None ☒ Mono ☐ Poly

ρ_s 2600 kg.m⁻³ v

d_p 1 mm v

ψ 0.8

c_b 0.62

☒ Auto-calc velocities

Carrier Fluid ? ^ x

File *unsaved* v

Herschel Bulkley fluid v

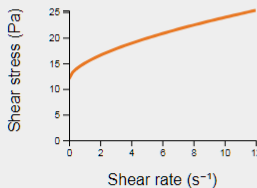
ρ_f 1200 kg.m⁻³ v

τ_y 12 Pa v

k 3 Pa.s v

n 0.6

T 20 °C v



Slurry ? ^ x

File *unsaved* v

ρ_m 2340 kg.m⁻³ v

c 0.3 - v

☒ v/v ☐ w/w ☐ w/v

Carrier Fluid Solids ? ^ x

File *unsaved* v

Particle size

☐ None ☒ Mono ☐ Poly

ρ_s 5000 kg.m⁻³ v

d_p 1 mm v

ψ 0.8

c_b 0.62

Pipe ? ^ x

Transport Gradient ? ^ x

☒ Single value ☐ Graph over range

Fluid Method Wilson Thomas v

Suspension Method Two Layer Slidi v

☒ Delivered ☐ Inline

Velocity v

V 1 m.s⁻¹ v

Property Pressure gradient v

Value kPa.m⁻ v

Calculate

Min Conveying Velocity ? ^ x

☐ Single value ☒ Graph over range

Pipe diameter v

D 0.01 m v





Sus

FI

Sol

SI

CSO

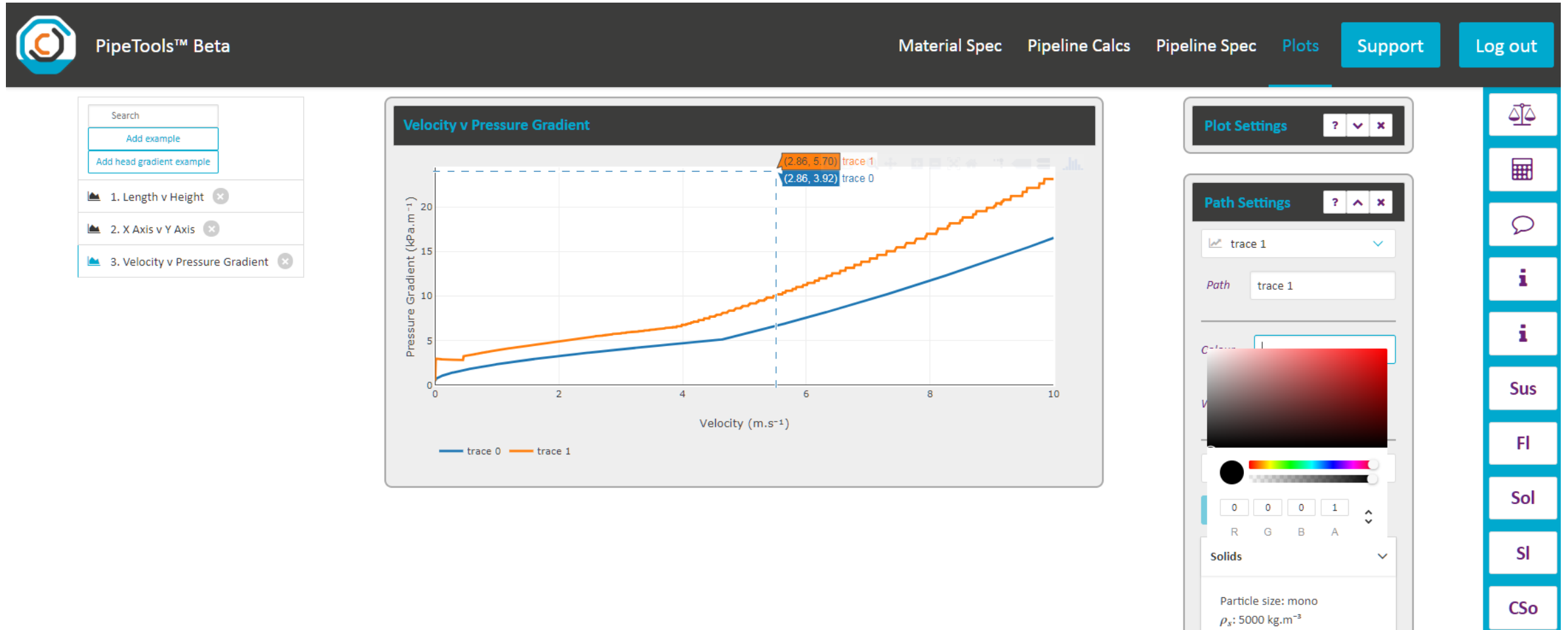
Tr

MCV

Pi

</>

GUI: Output



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)