A Literate Approach for Improving the Verifiability, Reusability and Reproducibility of Scientific Computing Software

Spencer Smith, Jacques Carette, Dan Szymczak, Steven Palmer

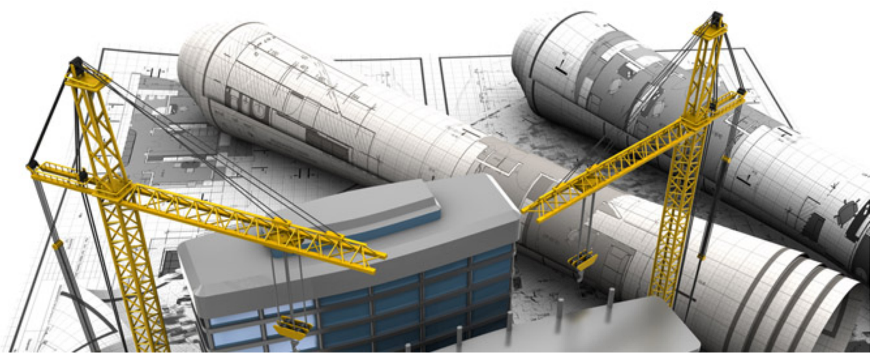
Computing and Software Department
Faculty of Engineering
McMaster University

CAIMS 2017, Third Canadian Symposium in Numerical Analysis and Scientific Computing (CSNASC): Simulation, July 18, 2017

Abstract

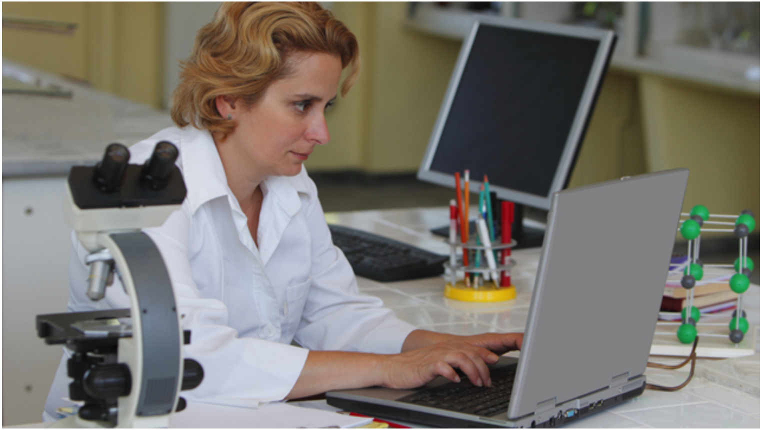
- **Goal** – Improve quality of SCS
- **Idea** – Adapt ideas from SE
- **Document Driven Design**
  - Good – improves quality
  - Bad – “manual” approach is too much work
- **Solution**
  - Capture knowledge
  - Generate all things
  - Avoid duplication
  - Traceability
- **Showing great promise**
  - Significant work yet to do
  - Looking for examples/partners

Scope: Large/Multiyear

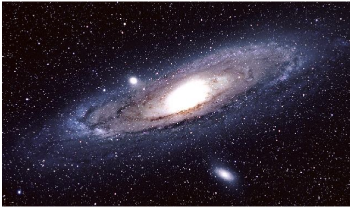


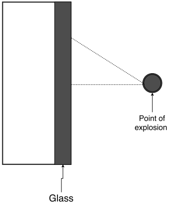
Scope: Program Families

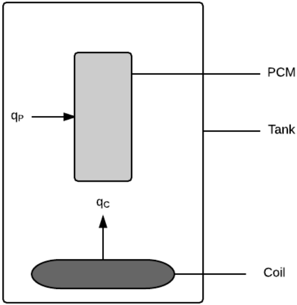
Scope: End User Developers

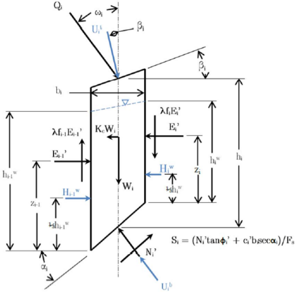


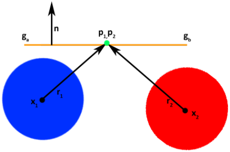
Scope: Physical Science

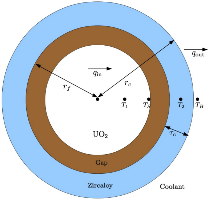




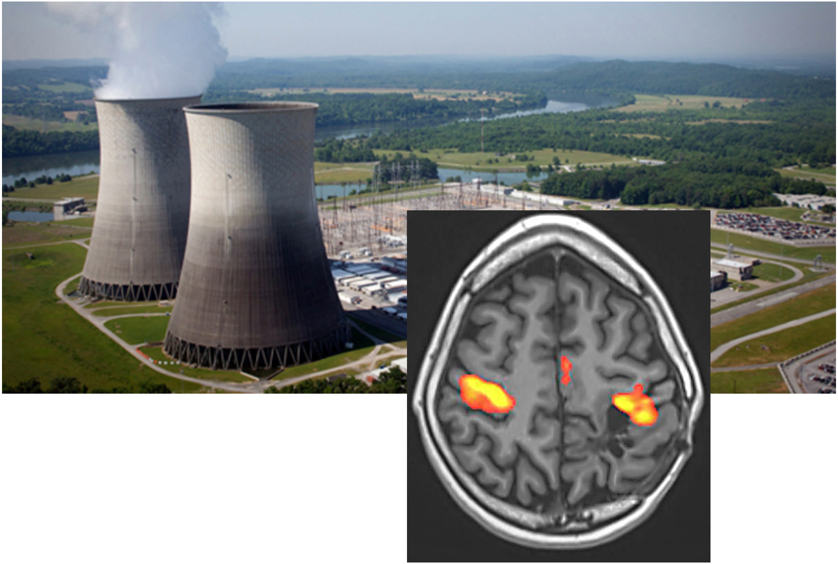






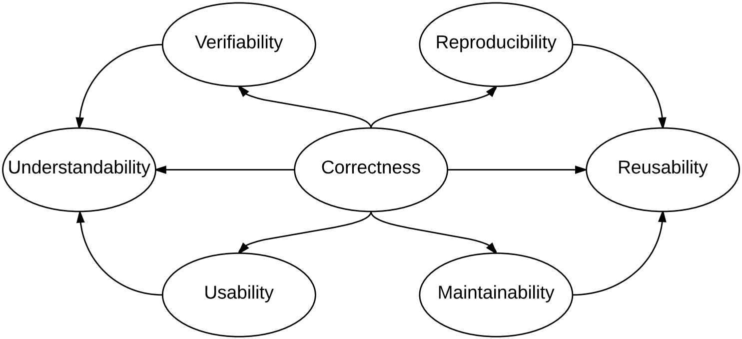


Motivation: Safety

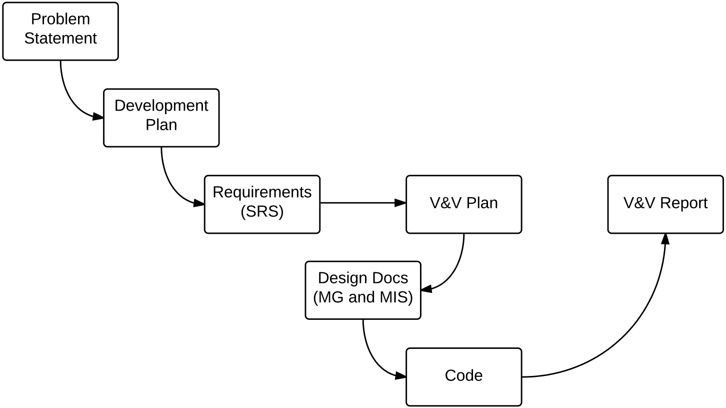


Motivation: (Re)certification

Motivation: Improve Quality



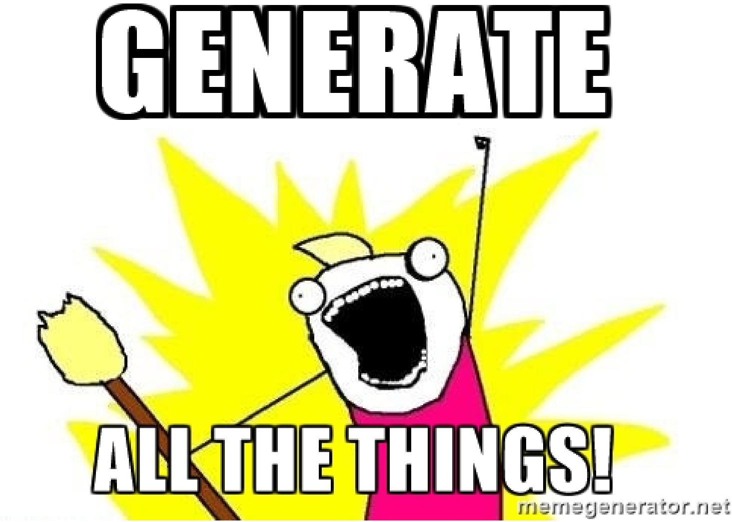
"Faked" Rational Design Process



SWHS example at https://github.com/smiths/swhs

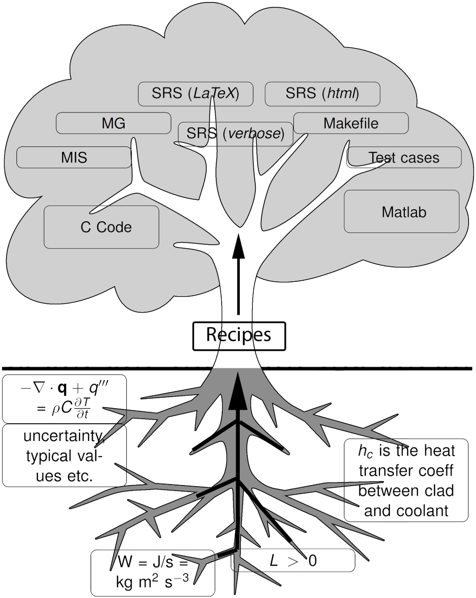
The Challenge

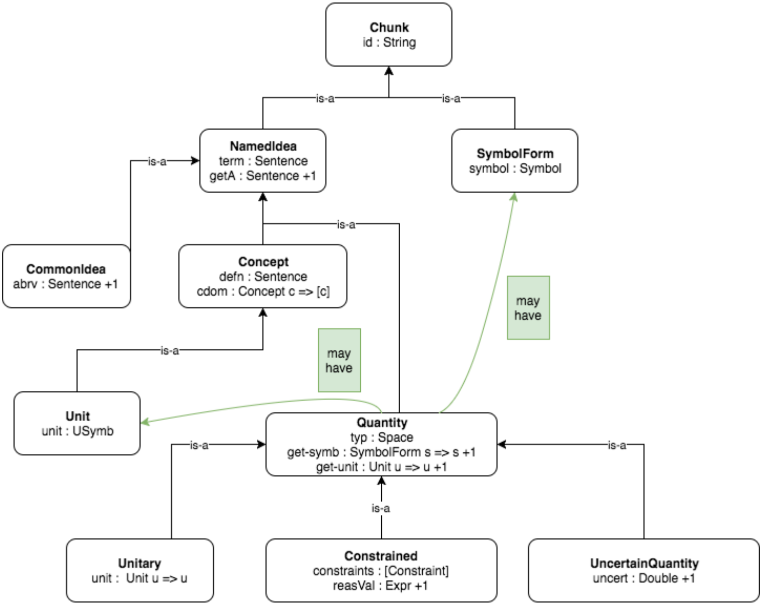
- Documentation provides advantages
  - Improves verifiability, reusability, reproducibility, etc.
  - From Parnas (2010)
    - easier reuse of old designs
    - better communication about requirements
    - more useful design reviews
    - etc.
  - New doc found 27 errors (Smith and Koothoor, 2016)
  - Developers see advantage (Smith et al., 2016)
- But documentation is felt to be ...
  - Too long
  - Too difficult to maintain
  - Not amenable to change
  - Too tied to waterfall process
  - Reports counterproductive (Roache, 1998)
- **The Solution?**


GENERATE
ALL THE THINGS!
memegenerator.net

Knowledge Capture

Drasil


SRS (LaTeX)
SRS (html)
MG
SRS (verbose)
Makefile
MIS
Test cases
C Code
Matlab
Recipes
$-\nabla \cdot \mathbf{q} + q''' = \rho C \frac{\partial T}{\partial t}$
uncertainty, typical values etc.
h_c is the heat transfer coeff between clad and coolant
W = J/s = kg m^2 s^{-3}
$L > 0$


Chunk
id : String
is-a
is-a
NamedIdea
term : Sentence
getA : Sentence +1
SymbolForm
symbol : Symbol
is-a
is-a
CommonIdea
abrv : Sentence +1
Concept
defn : Sentence
cdom : Concept c => [c]
may have
is-a
may have
Unit
unit : USymb
Quantity
typ : Space
get-symb : SymbolForm s => s +1
get-unit : Unit u => u +1
is-a
is-a
is-a
Unitary
unit : Unit u => u
Constrained
constraints : [Constraint]
reasVal : Expr +1
UncertainQuantity
uncert : Double +1

J_{tol} in SRS.pdf

Refname	**DD:sdf.tol**
Label	J_{tol}
Units	
Equation	$J_{tol} = \log\left(\log\left(\frac{1}{1-P_{btol}}\right)\frac{\left(\frac{a}{1000}\frac{b}{1000}\right)^{m-1}}{k\left((E*1000)\left(\frac{h}{1000}\right)^2\right)^m * LDF}\right)$
Description	J_{tol} is the stress distribution factor (Function) based on Pbtol P_{btol} is the tolerable probability of breakage a is the plate length (long dimension) b is the plate width (short dimension) m is the surface flaw parameter k is the surface flaw parameter E is the modulus of elasticity of glass h is the actual thickness LDF is the load duration factor

J_{tol} in SRS.tex

```
\noindent \begin{minipage}{\textwidth}
\begin{tabular}{p{0.2\textwidth} p{0.73\textwidth}}
\toprule \textbf{Refname} & \textbf{DD:sdf.tol}
\phantomsection
\label{DD:sdf.tol}
\\ \midrule \\
Label & $J_{tol}$
\\ \midrule \\
Units &
\\ \midrule \\
Equation & $J_{tol}$ = $\log\left(\log\left(\frac{1}{1-P_
    {btol}}\right)\frac{\left(\frac{a}{1000}\frac{b
    }{1000}\right)^{m-1}}{k\left(\left(E*1000\right)\left
    (\frac{h}{1000}\right)^{2}\right)^{m}*LDF}\right)$
\\ \midrule \\
Description & $J_{tol}$ is the stress distribution factor
     (Function) based on
              Pbtol\newline$P_{btol}$ is the tolerable
                 probability of breakage ...
\end{minipage}\\
```

J_{tol} in SRS.html

```
<a id="">
<div class="equation">
<em>J<sub>tol</sub></em> = log(log(<div class="fraction">
<span class="fup">
1
</span>
<span class="fdn">
1 &minus; <em>P<sub>btol</sub></em>
</span>
</div>)<div class="fraction">
<span class="fup">
(<div class="fraction">
<span class="fup">
<em>a</em>
</span>
<span class="fdn">
1000
</span>
</div><div class="fraction">
...
```

J_{tol} in Python

```
def calc_j_tol(inparams):
    j_tol = math.log((math.log(1.0 / (1.0 - inparams.
        pbtol))) * ((((inparams.a / 1000.0) * (inparams.b
         / 1000.0)) ** (inparams.m - 1.0)) / ((inparams.k
         * (((inparams.E * 1000.0) * ((inparams.h /
        1000.0) ** 2.0)) ** inparams.m)) * inparams.ldf))
        )
    return j_tol
```

J_{tol} in Java

```
public static double calc_j_tol(InputParameters inparams)
    {
        double j_tol = Math.log((Math.log(1.0 / (1.0 -
            inparams.pbtol))) * ((Math.pow((inparams.a /
            1000.0) * (inparams.b / 1000.0), inparams.m -
             1.0)) / ((inparams.k * (Math.pow((inparams.E
             * 1000.0) * (Math.pow(inparams.h / 1000.0,
            2.0)), inparams.m))) * inparams.ldf)));
        return j_tol;
    }
```

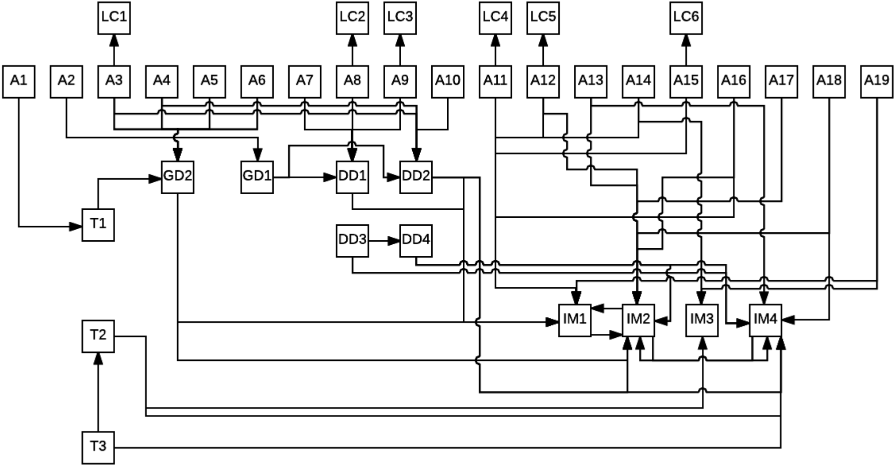
J_{tol} in Drasil (Haskell)

```
stressDistFac = makeVC "stressDistFac" (nounPhraseSP
  $ "stress distribution" ++ " factor (Function)") cJ

sdf_tol = makeVC "sdf_tol" (nounPhraseSP $
  "stress distribution" ++
  " factor (Function) based on Pbtol")
  (sub (stressDistFac ^. symbol) (Atomic "tol"))

tolStrDisFac_eq :: Expr
tolStrDisFac_eq = log (log ((1) / ((1) - (C pb_tol)))
  * ((Grouping (((C plate_len) / (1000)) * ((C
     plate_width) / (1000)))) :^
  ((C sflawParamM) - (1)) / ((C sflawParamK) *
  (Grouping (Grouping ((C mod_elas) * (1000)) *
  (square (Grouping ((C act_thick) / (1000))))
  )) :^ (C sflawParamM) * (C loadDF))))

tolStrDisFac :: QDefinition
tolStrDisFac = mkDataDef sdf_tol tolStrDisFac_eq
```

J_{tol} without Unit Conversion

```
tolStrDisFac_eq :: Expr
tolStrDisFac_eq = log (log ((1) / ((1) - (C pb_tol)))
  * ((Grouping ((C plate_len) * (C plate_width))) :^
  ((C sflawParamM) - (1)) / ((C sflawParamK) *
  (Grouping ((C mod_elas) * (square (C act_thick)))) :^ (
     C sflawParamM) * (C loadDF))))
```

Traceability Graph



Verifiability

Var	Constraints	Typical Value	Uncertainty
L	$L > 0$	1.5 m	10%
ρ_P	$\rho_P > 0$	1007 kg/m^3	10%

$$E_W = \int_0^t h_C A_C (T_C - T_W(t)) dt - \int_0^t h_P A_P (T_W(t) - T_P(t)) dt$$

- If wrong, wrong everywhere
- Sanity checks captured and reused
- Generate guards against invalid input
- Generate test cases
- Generate view suitable for inspection
- Traceability

Reusability

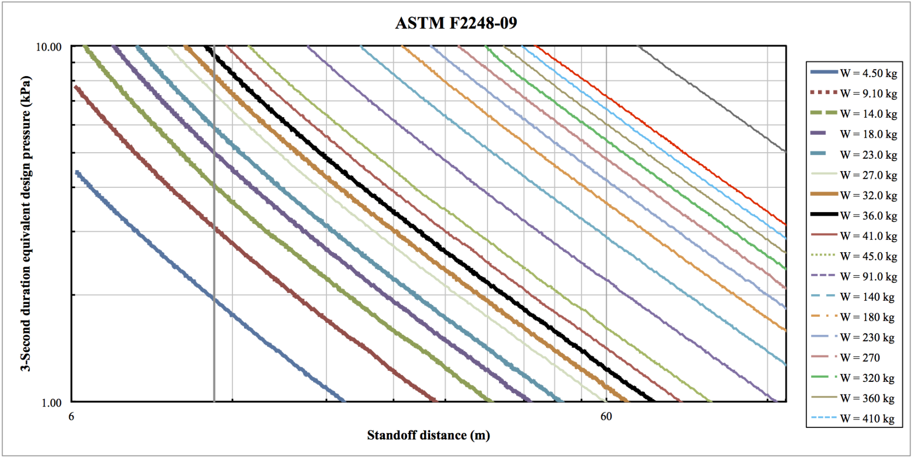
- De-embed knowledge
- Reuse throughout document
  - Units
  - Symbols
  - Descriptions
  - Traceability information
- Reuse between documents
  - SRS
  - MIS
  - Code
  - Test cases
- Reuse between projects
  - Knowledge reuse
  - For a family of related models or reuses of portions of knowledge
  - Conservation of thermal energy
  - Interpolation
  - Etc.

Reproducibility

- Usual emphasis is on reproducing code execution
- However, Ionescu and Jansson (2012) show reproducibility challenges due to undocumented:
  - Assumptions
  - Modifications
  - Hacks
- Shouldn't it be easier to independently replicate the work of others?
- Require theory, assumptions, equations, etc.
- Drasil can potentially check for completeness and consistency


NO
SILVER

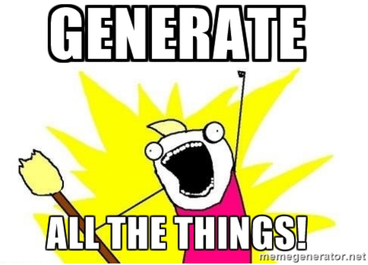
Future Work

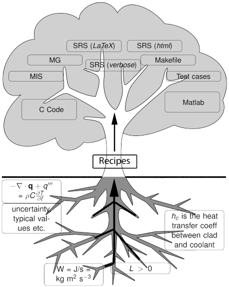


ASTM F2248-09
3-Second duration equivalent design pressure (kPa)
10.00
1.00
6
60
Standoff distance (m)
W = 4.50 kg
W = 9.10 kg
W = 14.0 kg
W = 18.0 kg
W = 23.0 kg
W = 27.0 kg
W = 32.0 kg
W = 36.0 kg
W = 41.0 kg
W = 45.0 kg
W = 91.0 kg
W = 140 kg
W = 180 kg
W = 230 kg
W = 270
W = 320 kg
W = 360 kg
W = 410 kg

Drasil Framework for LSS

- SCS has the opportunity to lead other software fields
- Document driven design is feasible
- Requires an investment of time
- Documentation does not have to be painful
- Develop/refactor via practical case studies
- Ontology may naturally emerge





References I

Cezar Ionescu and Patrik Jansson. Dependently-Typed Programming in Scientific Computing — Examples from Economic Modelling. In *Revised Selected Papers of the 24th International Symposium on Implementation and Application of Functional Languages*, volume 8241 of *Lecture Notes in Computer Science*, pages 140–156. Springer International Publishing, 2012. doi: 10.1007/978-3-642-41582-1_9.

David Lorge Parnas. Precise documentation: The key to better software. In *The Future of Software Engineering*, pages 125–148, 2010. doi: 10.1007/978-3-642-15187-3_8. URL http://dx.doi.org/10.1007/978-3-642-15187-3_8.

References II

Patrick J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, New Mexico, 1998.

W. Spencer Smith and Nirmitha Koothoor. A document-driven method for certifying scientific computing software for use in nuclear safety analysis. *Nuclear Engineering and Technology*, 48(2):404–418, April 2016. ISSN 1738-5733. doi: http://dx.doi.org/10.1016/j.net.2015.11.008. URL http://www.sciencedirect.com/science/article/pii/S1738573315002582.

References III

W. Spencer Smith, Thulasi Jegatheesan, and Diane F. Kelly. Advantages, disadvantages and misunderstandings about document driven design for scientific software. In *Proceedings of the Fourth International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering (SE-HPCCE)*, November 2016. 8 pp.