

# CAS 741, CES 741 (Development of Scientific Computing Software)

Fall 2018

## 21 Artifact Generation

Dr. Spencer Smith

Faculty of Engineering, McMaster University

November 21, 2018



# Artifact Generation

- Administrative details
- Questions about MIS?
- Artifact generation (Drasil)

# Administrative Details

- For final documentation, make sure you have **addressed and closed** all open issues
- MIS Marking Scheme
  - ▶ On Avenue
  - ▶ Not all of the spec has to be formal
  - ▶ First steps
    - ▶ Syntax of access programs
    - ▶ State variables?
    - ▶ Environment variables?
- Course evaluation
  - ▶ Thurs, Nov 22, 10:00 am to Thurs, Dec 6, 11:59 pm
  - ▶ <https://evals.mcmaster.ca>
- No class on Friday, Nov 30

# Administrative Details: Deadlines

MIS	Week 11	Nov 23
Unit VnV or Impl. Present	Week 12, 13	Wed, Nov 28, Dec 5
Unit VnV Plan	Week 13	Dec 3
Final Doc	Week 14	Dec 10

Combine Unit VnV Plan deadline with Final Doc deadline?

# Administrative Details: Presentation Schedule

- Unit VnV Plan or Impl. Present
  - ▶ Wednesday (Nov 28): Brooks, Vajiheh
  - ▶ Wednesday (Dec 5): Olu, Karol
- Can present anything related to the implementation or testing
  - ▶ Code
  - ▶ Tools used
  - ▶ Testing
  - ▶ As always it is fine to show work in progress
  - ▶ Good to bring questions to the class

# Questions?

- Questions about MIS

# 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

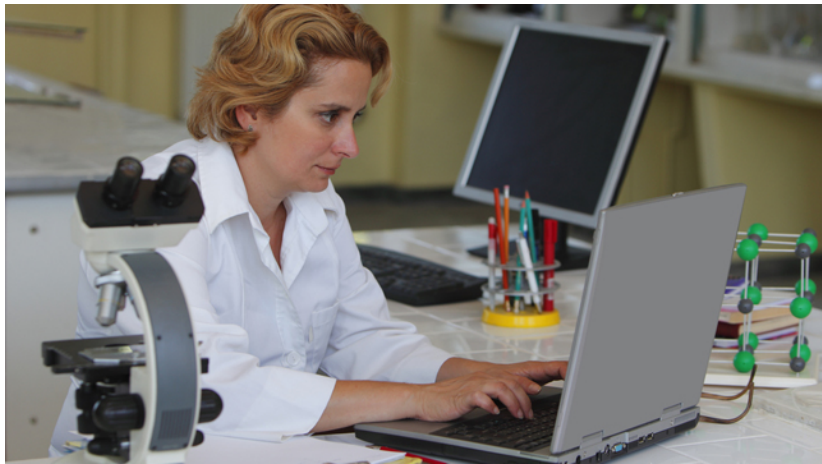




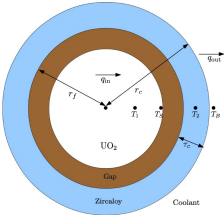
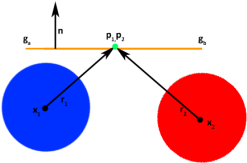
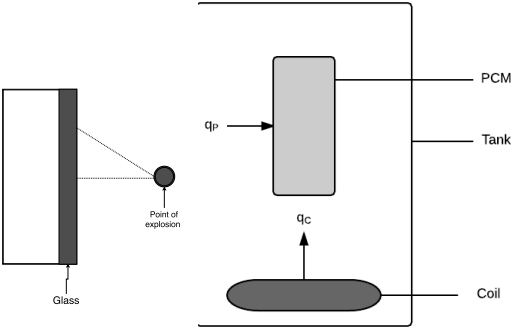
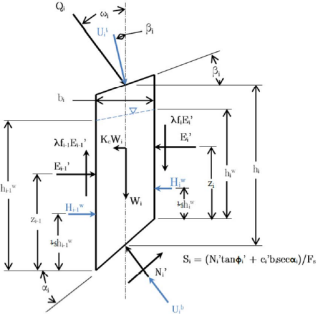
# Scope: Program Families



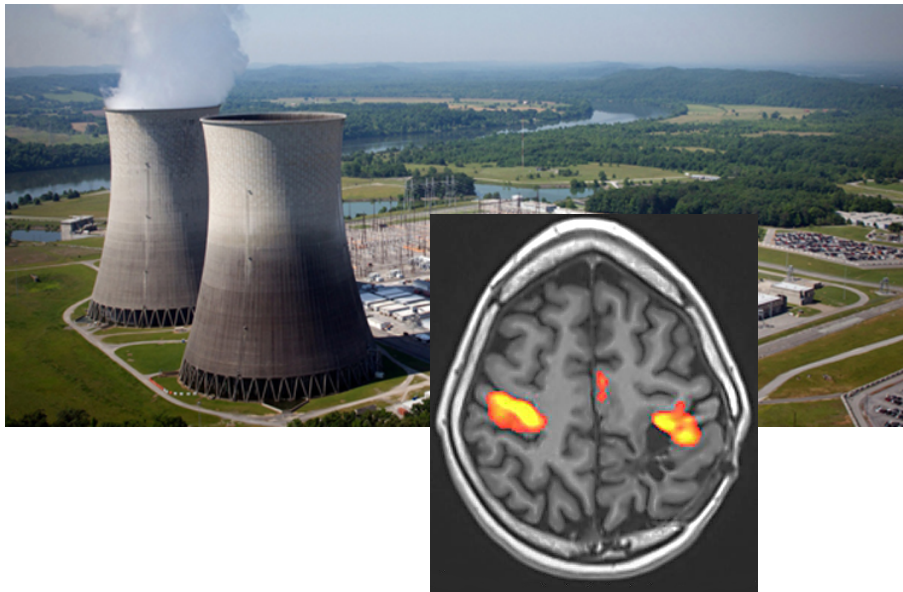
## Scope: End User Developers



# Scope: Physical Science



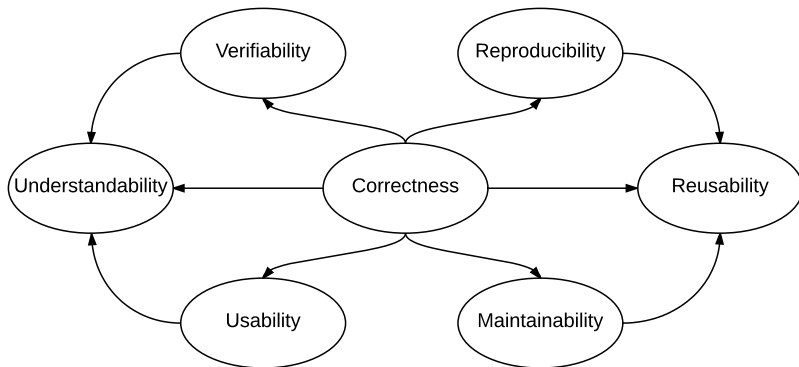
# Motivation: Safety



Motivation: (Re)certification



# Motivation: Improve Quality



# Current Approach

- Agile like [1]
- Amethododical [3]
- Knowledge acquisition driven [4]
- Each stage reports counterproductive [10]
- Limited tool use [13]
- Limited testing of code [5]
- Lack of understanding of testing [7]
- Missed opportunities for reuse [8]
- Emphasis on:
  1. Science [6]
  2. Code

# Documentation Advantages

- Improves verifiability, reusability, reproducibility, etc.
- From [9]
  - ▶ easier reuse of old designs
  - ▶ better communication about requirements
  - ▶ more useful design reviews
  - ▶ easier integration of separately written modules
  - ▶ more effective code inspection
  - ▶ more effective testing
  - ▶ more efficient corrections and improvements
- New doc found 27 errors [12]
- Developers see advantage [11]



# Study Of Documentation in SC [11]

1. Select 5 small to medium size SCS
2. Interview code owners
3. Redevelop using Document Driven Design (DDD)
4. Interview code owners
5. Analyze responses

# Summary of Case Studies

	LOC	Lng	ND	Ag	SE	Prg	Tst	VC	Bug
<b>SWHS</b>	1000	F77	1	5	✗	✓	✗	✗	✗
<b>Astro</b>	5000	C	2	10	✗	✓	✗	✗	✗
<b>Glass</b>	1300	F90	1	<1	✗	✓	✗	✗	✗
<b>Soil</b>	800	M	1	5	✓	✓	✓	✓	✗
<b>Neuro</b>	1000	M	1	5	✓	✓	✗	✓	✗
<b>Acoust</b>	200	M	4	2.5	✗	✓	✗	✗	✗

# Perceived Advantages from Participants

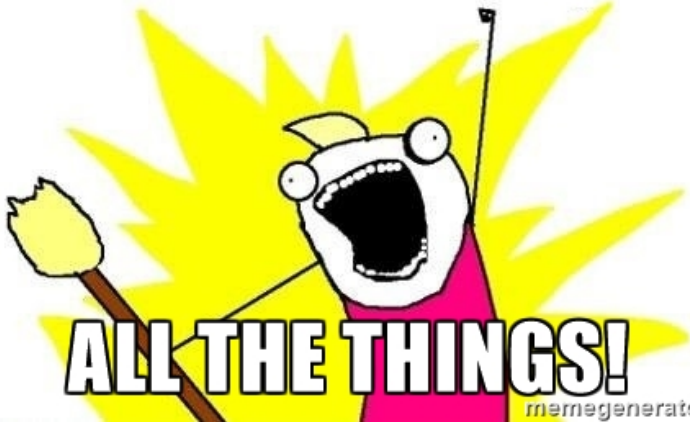
- Documentation of assumptions
- All variables have explicit units
- SRS helpful with new graduate students
- Modules result in more user friendly code
- Traceability between modules and requirements useful
- Better organized code
- Information sharing on design choices
- Detailed record of knowledge capital
- Code is produced to make testing easier

# Disadvantages (Perceived and Real)

- SRS is too long
- SRS is not necessary
- DDD will not work in reality, since needs upfront requirements
- Too much SE jargon
- Difficult without a team of people
- Too difficult to maintain
- Not amenable to change
- Too tied to waterfall process
- Reports counterproductive [[10](#)]

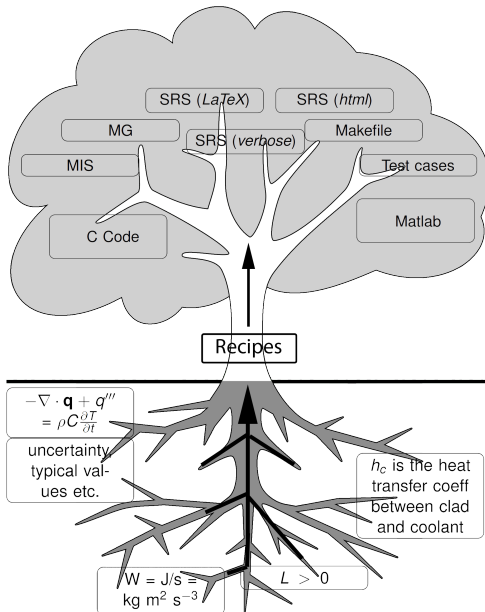
## The Solution?

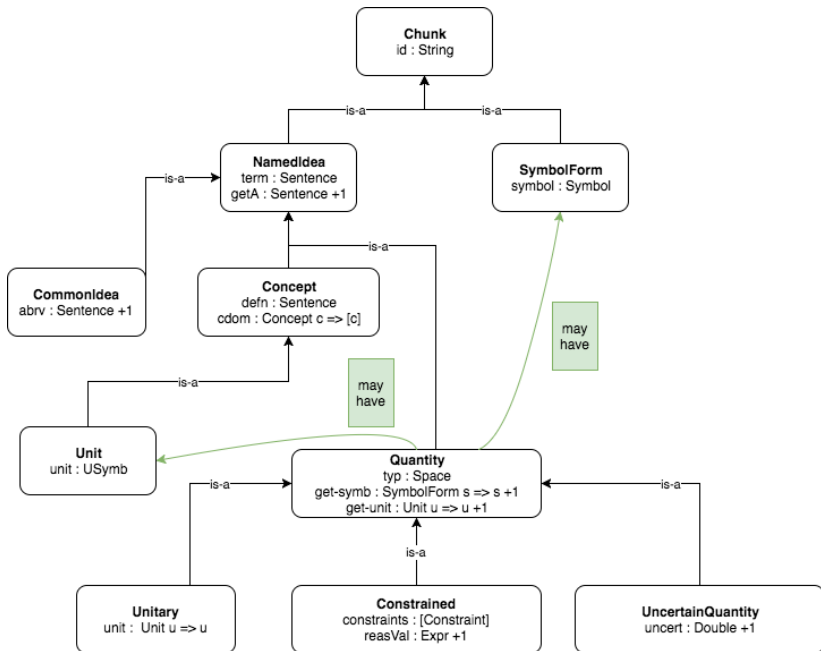
# GENERATE



# Knowledge Capture









# $J_{tol}$ in SRS.pdf

Refname	DD:sdf.tol
Label	Stress Distribution Factor (Function) Based on Pbtol
Units	Unitless
Equation	$J_{tol} = \log \left( \log \left( \frac{1}{1-P_{btol}} \right) \frac{\left( \frac{a}{1000} \frac{b}{1000} \right)^{m-1}}{k \left( \left( E \cdot 1000 \left( \frac{h}{1000} \right)^2 \right) \right)^m \cdot LDF} \right)$
Description	<p><math>J_{tol}</math> is the stress distribution factor (Function) based on Pbtol</p> <p><math>P_{btol}</math> is the tolerable probability of breakage</p> <p><math>a</math> is the plate length (long dimension) (m)</p> <p><math>b</math> is the plate width (short dimension) (m)</p> <p><math>m</math> is the surface flaw parameter (<math>\frac{m^{12}}{N^7}</math>)</p> <p><math>k</math> is the surface flaw parameter (<math>\frac{m^{12}}{N^7}</math>)</p> <p><math>E</math> is the modulus of elasticity of glass (Pa)</p> <p><math>h</math> is the actual thickness (m)</p> <p><math>LDF</math> is the load duration factor</p>

## $J_{tol}$ 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 &  $J_{tol} =$ 

$$\log\left(\log\left(\frac{1}{1-P_{btol}}\right)\frac{\left(\frac{a}{1000}\right)\frac{b}{1000}}{\left(E*1000\right)\left(\frac{h}{1000}\right)^2}\right)^{m*LDF}$$

\\ \midrule \\
Description &  $J_{tol}$  is the stress distribution
factor (Function) based on
```

# $J_{\text{tol}}$ 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>
```

## $J_{tol}$ 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.pb_tol))) * ((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)

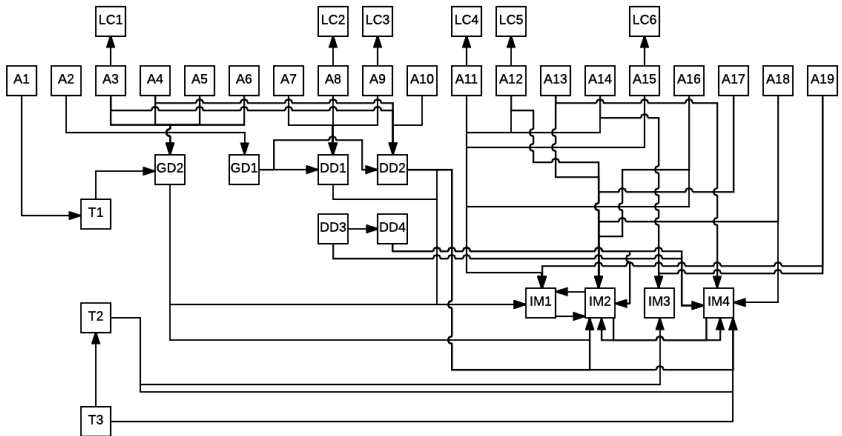
```
stressDistFac = makeVC "stressDistFac" (nounPhraseSP
  $ "stress distribution" ++ " factor (Function)"
  cJ
sdf_tol = makeVC "sdf_tol" (nounPhraseSP $
  "stress distribution" ++
  " factor (Function) based on Pbtol")
(sub (eqSymb stressDistFac) (Atomic "tol"))

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 lDurFac)))))
tolStrDisFac :: QDefinition
tolStrDisFac = mkDataDef' sdf_tol tolStrDisFac_eq
  (aGrtrThanB :+: hRef :+: ldfRef :+: pbTolUsr)
```

## $J_{tol}$ 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 (Grouping ((C mod_elas * 1000) *
    (square (Grouping (C act_thick)))))) :^
    (C sflawParamM) * (C lDurFac)))))
```

# Traceability Graph





# Maintainability

- A1: The only form of energy that is relevant for this problem is thermal energy. All other forms of energy, such as mechanical energy, are assumed to be negligible [T1].
- A2: All heat transfer coefficients are constant over time [GD1].
- A3: The water in the tank is fully mixed, so the temperature is the same throughout the entire tank [GD2, DD2].
- A4: The PCM has the same temperature throughout [GD2, DD2, LC1].
- A5: etc.

# Verifiability

Var	Constraints	Typical Value	Uncertainty
$L$	$L > 0$	1.5 m	10%
$\rho_P$	$\rho_P > 0$	1007 kg/m <sup>3</sup>	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 for verification of change

# Reusability

---

<b>Num.</b>	<b>T1</b>
-------------	-----------

---

<b>Label</b>	<b>Conservation of energy</b>
--------------	-------------------------------

---

<b>Eq</b>	$-\nabla \cdot \mathbf{q} + q''' = \rho C \frac{\partial T}{\partial t}$
-----------	--

---

<b>Descrip</b>	The above equation gives the conservation of energy for time varying heat transfer in a material of specific heat capacity $C$ and density $\rho$ , where $\mathbf{q}$ is the thermal flux vector, $q'''$ is the volumetric heat generation, $T$ is the temperature, $\nabla$ is the del operator and $t$ is the time.
----------------	--

---

# 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
  - ▶ A family of related models, or reuse of pieces
  - ▶ Conservation of thermal energy
  - ▶ Interpolation, Etc.

# Reproducibility

- Usual emphasis is on reproducing code execution
- However, [2] 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

# Smith and Koothoor (2016) [12]

$$R_1^{\text{code}} = \frac{f}{8\pi k_{\text{AV}}} + \frac{1}{2\pi r_f h_g} \quad (1)$$

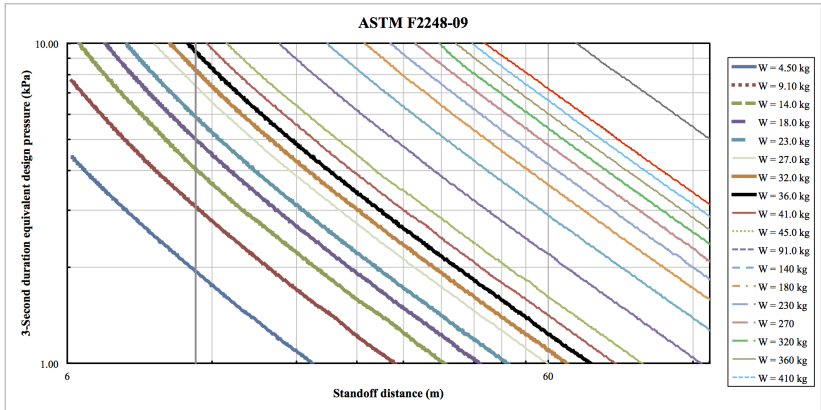
$$R_1^{\text{manual}} = \frac{f}{8\pi k_{\text{AV}}} + \frac{1}{2\pi r_f h_g} + \frac{\tau_c}{4\pi r_f k_c} \quad (2)$$

- Uncovered 27 issues with the previous documentation
  - ▶ Incompleteness ( $R_{\text{gap}}$ )
  - ▶ Inconsistency( $r, r_0, h_g$ )
  - ▶ Verifiability problems ( $R_1$ )
  - ▶ Lack of traceability (circuit analogy)
- Advantages of proposed approach
  - ▶ Abstract to concrete
  - ▶ Separation of concerns
  - ▶ Every equation, assumption, definition, model, derivation, source and traceability between them

**NO**



# Future Work

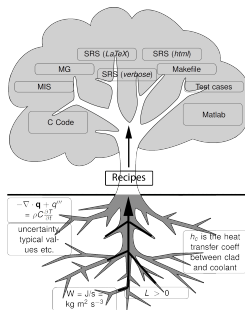
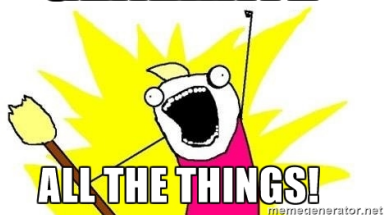




# Drasil Framework for LSS

- 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
- Open source Drasil [here](#)

# GENERATE



# References I



Jeffrey C. Carver, Richard P. Kendall, Susan E. Squires, and Douglass E. Post.

Software development environments for scientific and engineering software: A series of case studies.

*In ICSE '07: Proceedings of the 29th International Conference on Software Engineering*, pages 550–559, Washington, DC, USA, 2007. IEEE Computer Society.



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*

# References II

*Computer Science*, pages 140–156. Springer International Publishing, 2012.



Diane Kelly.

Industrial scientific software: A set of interviews on software development.

*In Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research, CASCON '13*, pages 299–310, Riverton, NJ, USA, 2013. IBM Corp.



Diane Kelly.

Scientific software development viewed as knowledge acquisition: Towards understanding the development of risk-averse scientific software.

*Journal of Systems and Software*, 109:50–61, 2015.

# References III



Diane Kelly and Rebecca Sanders.

The challenge of testing scientific software.

*In Proceedings of the Conference for the Association for Software Testing*, pages 30–36, 2008.



Diane F. Kelly.

A software chasm: Software engineering and scientific computing.

*IEEE Software*, 24(6):120–119, 2007.



Zeeya Merali.

Computational science: ...error.

*Nature*, 467:775–777, 2010.

# References IV



Steven J. Owen.

A survey of unstructured mesh generation technology.

In *INTERNATIONAL MESHING ROUNDTABLE*, pages 239–267, 1998.



David Lorge Parnas.

Precise documentation: The key to better software.

In *The Future of Software Engineering*, pages 125–148, 2010.



Patrick J. Roache.

*Verification and Validation in Computational Science and Engineering.*

Hermosa Publishers, Albuquerque, New Mexico, 1998.

# References V



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-HPCCCE), November 2016.*

8 pp.



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.

# References VI



Gregory V. Wilson.

Where's the real bottleneck in scientific computing?

Scientists would do well to pick some tools widely used in the software industry.

*American Scientist*, 94(1), 2006.