The Drasil Framework for Literate Scientific Software

Spencer Smith, Jacques Carette, Dan Szymczak,
Steven Palmer

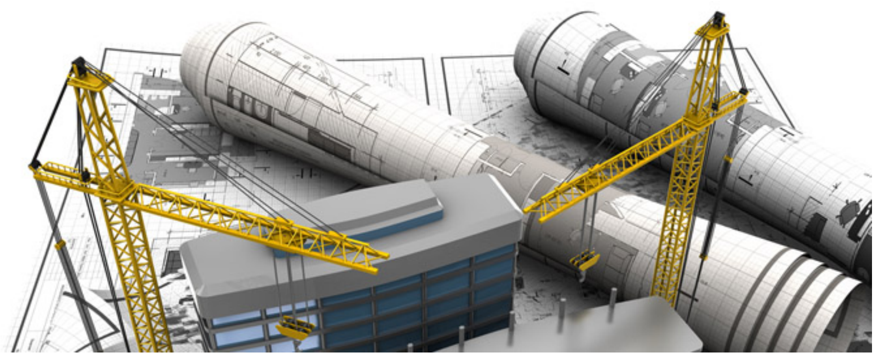
Computing and Software Department
Faculty of Engineering
McMaster University

CSE/SHARCNET Seminar in Scientific Computing,
November 23, 2016

Abstract

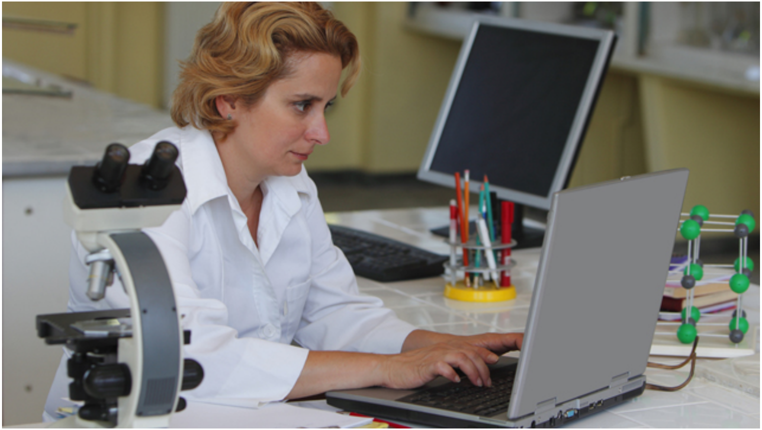
- **Goal** – Improve quality of SCS
- **Idea** – Adapt ideas from SE
- **Document Driven Design**
  - Good – improves quality
  - Bad – “manual” approach is too much work
- **Solution**
  - Capture knowledge
  - Generate all things
  - Traceability
- **Showing great promise**
  - Significant work yet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

Scope: Large/Multiyear

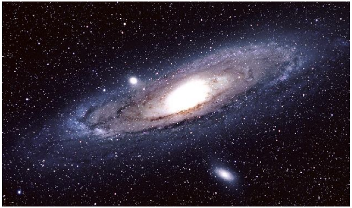


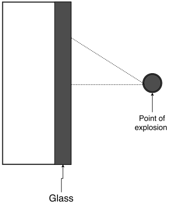
Scope: Program Families

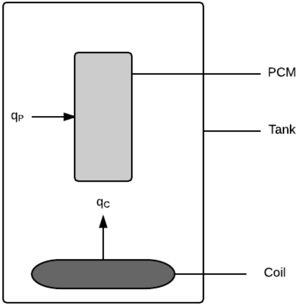
Scope: End User Developers

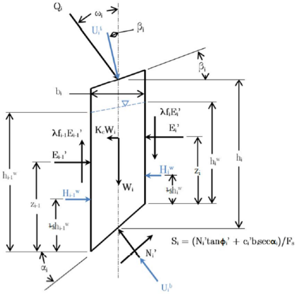


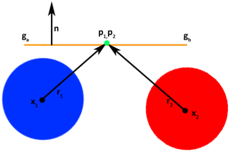
Scope: Physical Science

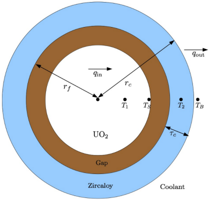




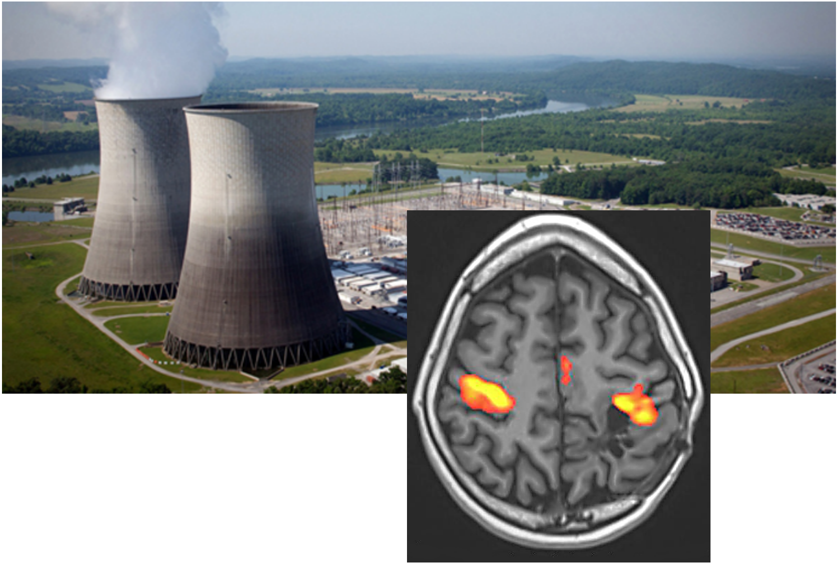






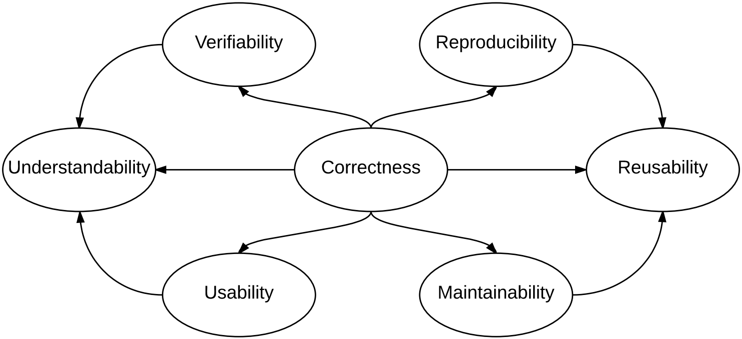


Motivation: Safety



Motivation: (Re)certification

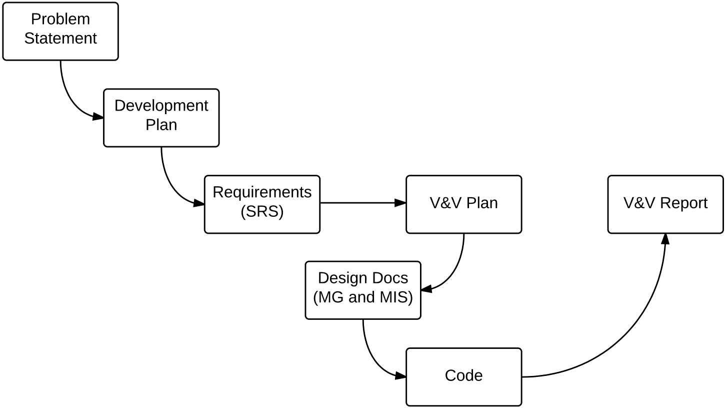
Motivation: Improve Quality



Current Approach

- Agile like (Carver et al., 2007)
- Amethododical (Kelly, 2013)
- Knowledge acquisition driven (Kelly, 2015)
- Each stage reports counterproductive (Roache, 1998)
- Limited tool use (Wilson, 2006)
- Limited testing of code (Kelly and Sanders, 2008)
- Lack of understanding of testing (Merali, 2010)
- Missed opportunities for reuse (Owen, 1998)
- Emphasis on:
  1. Science (Kelly, 2007)
  2. Code

"Faked" Rational Design Process



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

Reusability

Num.	**T1**
Label	**Conservation of energy**
Eq	$-\nabla \cdot \mathbf{q} + q''' = \rho C \frac{\partial T}{\partial t}$
Descrip	The above equation gives the conservation of energy for time varying heat transfer in a material of specific heat capacity C and density ρ, where $\mathbf{q}$ is the thermal flux vector, q''' is the volumetric heat generation, T is the temperature, ∇ is the del operator and t is the time.

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].

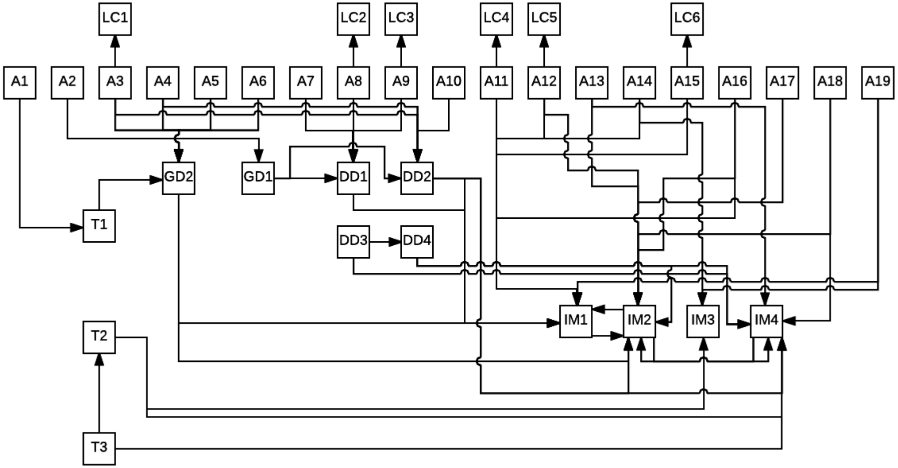
A2: All heat transfer coefficients are constant over time [GD1].

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.

SWHS Traceability Graph



Verifiability

Var	Constraints	Typical Value	Uncertainty
L	$L > 0$	1.5 m	10%
D	$D > 0$	0.412 m	10%
V_P	$V_P > 0$	0.05 m^3	10%
A_P	$A_P > 0$	1.2 m^2	10%
ρ_P	$\rho_P > 0$	1007 $\mathrm{kg/m}^3$	10%

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

Reproducibility

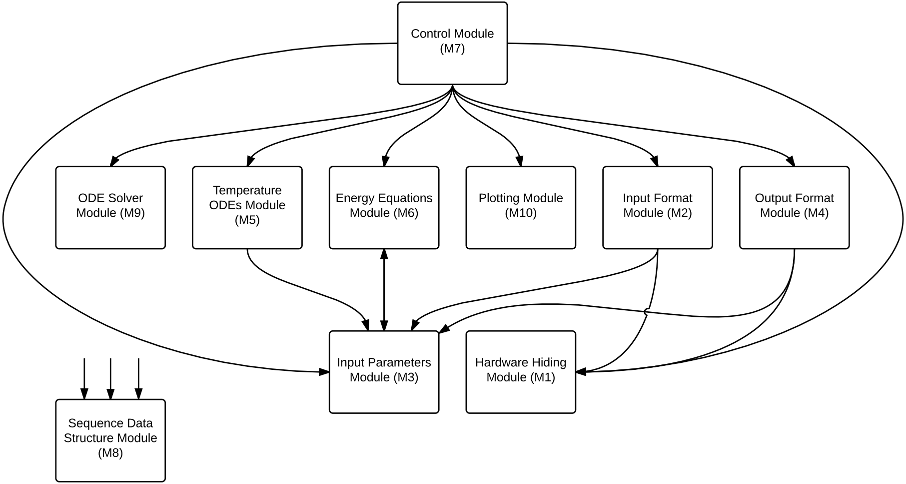
Ionescu and Jansson (2012) show reproducibility challenges due to undocumented:

- Assumptions
- Modifications
- Hacks

Complete Documentation

Input	m_P, C_P^S, C_P^L, h_P, A_P, t_{final}, T_{init}, T_{melt}^P, $T_W(t)$ from IM1
Output	$T_P(t)$, $0 \leq t \leq t_{\text{final}}$, with initial conditions, $T_W(0) = T_P(0) = T_{\text{init}}$ (A12), and $T_W(t)$ from IM1, such that the following governing ODE is satisfied. The specific ODE depends on T_P as follows: $\frac{dT_P}{dt} = \begin{cases} \frac{dT_P}{dt} = \frac{1}{\tau_P^S}(T_W(t) - T_P(t)) & \text{if } T_P < T_{\text{melt}}^P \\ \frac{dT_P}{dt} = \frac{1}{\tau_P^L}(T_W(t) - T_P(t)) & \text{if } T_P > T_{\text{melt}}^P \\ 0 & \text{if } T_P = T_{\text{melt}}^P \\ & \text{and } 0 < \phi < 1 \end{cases}$

SWHS Uses Hierarchy



Verification and Validation

- Compare to closed-form solutions
- Method of manufactured solutions
- Interval arithmetic
- Convergence studies
- Compare to another program
- Mutation testing
- Metamorphic testing
- Code inspections

Tools and Development Practices

- Unit Testing
- Version control
- Issue tracking
- Performance measurement
- Virtual machines
- Follow best practices (Wilson et al., 2014)

B.6.1 Computing q'_N, T_2 and k_c

The input relative fuel power (q'_{NFRAC}) is changed to linear element power (q'_N) by multiplying it with the initial linear element rating ($q'_{\mathrm{N_{max}}}$) as given by DD25 of the SRS.

$$q'_N = q'_{\mathrm{NFRAC}} q'_{N_{\mathrm{max}}}; \tag{B.8}$$

This q'_N is used to determine the relevant temperatures for the fuelpin. We evaluate linear element power as

17 $\langle$Calculation of q'_N 17$\rangle \equiv$

```
*q_N = *q_NFRAC * (*q_Nmax);
```

This code is used in chunks 15 and 57

Smith and Koothoor (2016)

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

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

- Uncovered 27 issues with the previous documentation
  - Incompleteness (R_{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

Smith et al. (2016)

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

Summary of Case Studies

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

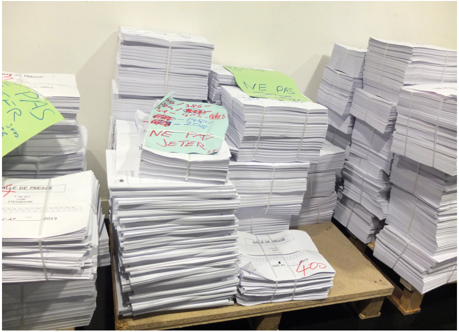
Advantages

- 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

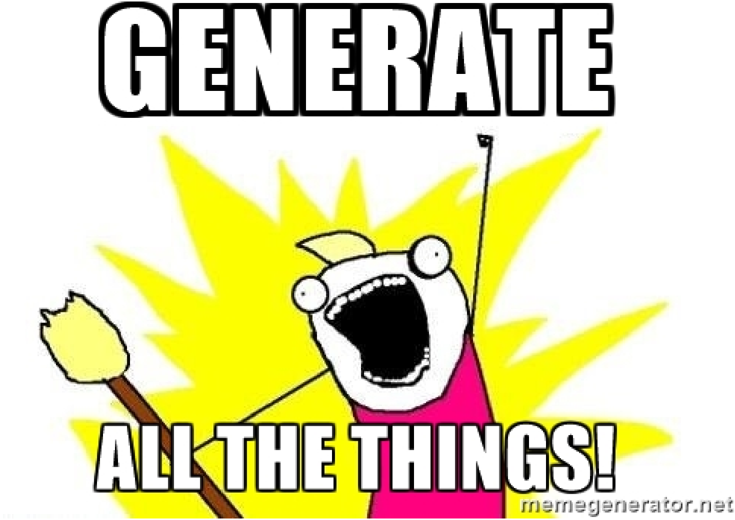
Disadvantages (Perceived and Real)

- 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

Information Duplication

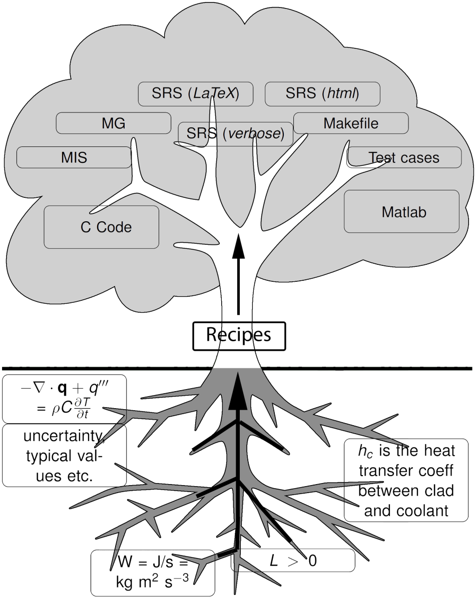


- Challenging to maintain
- Wastes resources


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}
$L > 0$


NO
SILVER

Advantages of Drasil

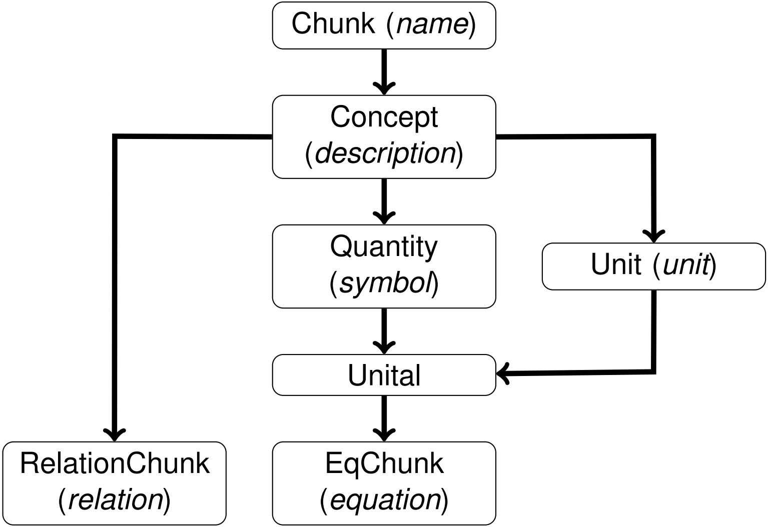
- Supports changing requirements and design
  - Generation
  - Automated traceability
- Supports duplication
  - Knowledge is entered once, generated/transformed
  - Eases maintenance
  - If incorrect, incorrect everywhere
- Non-executable artifacts are generated

Design

Drasil is currently being implemented as a combination of six eDSLs:

- Expression
- Expression Layout
- Document Layout
- C Representation
- LaTeX Representation
- HTML Representation

Chunks


Chunk (*name*)
Concept (*description*)
Quantity (*symbol*)
Unit (*unit*)
Unital
RelationChunk (*relation*)
EqChunk (*equation*)

Simple SRS from LaTeX

SRS from LaTeX
SRS in HTML

```haskell
vars :: [EqChunk]
vars = [h_g, h_c]

s1, s2, s3, s4 :: LayoutObj
s1=table_of_units si_units
s2=table_of_symbols vars
s3=Section 0 (S "Data Definitions") $ map (Definition.Data) vars
s4=Section 0 (S "Code") $ map (CodeBlock.toCode CLang Calc) [h_c]

srs :: Quantity s => [s] -> String -> [LayoutObj] -> Document
srs ls author body =
  Document ((S "SRS for ") :+:
    (foldr1 (:+:) (intersperse (S " and ")
    (map (\x -> U $ x ^. symbol) ls)))
    (S author) body

srsBody :: Document
srsBody = srs vars "Spencer Smith" [s1, s2, s3, s4]
```

```haskell
table_of_symbols :: (Unit s, Quantity s) => [s] -> LayoutObj
table_of_symbols ls=Section 0 (S "Table of Sym") [intro,table ls]

intro :: LayoutObj
intro = Paragraph $
  S "The table that follows ..."

table :: (Unit s, Quantity s) => [s] -> LayoutObj
table ls=Table [S "Symbol",S "Description",S "Units"] (mkTable
  [(\ch -> U (ch ^. symbol)),
   (\ch -> ch ^. descr),
   (\ch -> Sy $ ch ^. unit)] ls)
  (S "Table of Symbols") False
```

```
fundamentals :: [FundUnit]
fundamentals = [metre, kilogram, second, ...]

derived :: [DerUChunk]
derived = [centigrade, joule, watt, calorie, kilowatt]

si_units :: [UnitDefn]
si_units = map UU fundamentals ++ map UU derived

------------- Fundamental SI Units ------------------------------
fund :: String -> String -> String -> FundUnit
fund nam desc sym = UD (CC nam (S desc)) (UName $ Atomic sym)

metre, kilogram, second, ... :: FundUnit
metre    = fund "Metre"    "length"                "m"
kilogram = fund "Kilogram" "mass"                  "kg"
second   = fund "Second"   "time"                  "s"
kelvin   = fund "Kelvin"   "temperature"           "K"
mole     = fund "Mole"     "amount of substance"   "mol"
ampere   = fund "Ampere"   "electric current"      "A"
candela  = fund "Candela"  "luminous intensity"    "cd"
```

$$h_c = \frac{2k_c h_b}{2k_c + \tau_c h_b}$$

```
heat_transfer :: DerUChunk
heat_transfer = DUC (UD ht_con ht_symb) heat_transfer_eqn

ht_con :: ConceptChunk
ht_con = makeCC "Heat transfer" "Heat transfer"

ht_symb :: USymb
ht_symb = from_udefn heat_transfer_eqn

heat_transfer_eqn = USynonym (UProd
  [kilogram ^. unit, UPow (second ^. unit) (-3),
   UPow (centigrade ^. unit) (-1)])

h_c_eq :: Expr
h_c_eq = 2*(C k_c)*(C h_b)/(2*(C k_c)+(C tau_c)*(C h_b))

h_c :: EqChunk
h_c = fromEqn "h_c" (S "convective heat transfer ...")
  (lH `sub` lC) heat_transfer h_c_eq
```

Approach to Developing Drasil

- Case studies
  - Solar water heating tank
  - Slope stability analysis
  - Glass safety analysis
  - Game physics engine
- Practical
- Not trying to automate everything
- Small chunks of knowledge
- Look for patterns
- Tool support
  - Version control
  - Issue tracking
  - Regression testing

Refactor

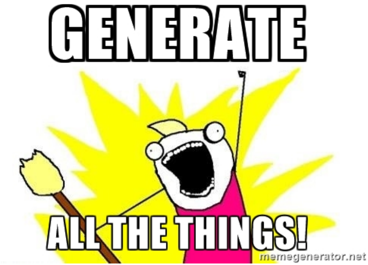
```
boiling = makeCC "Boiling"
  "Phase change from liquid to vapour"
phsChgMtrl  = makeCC "PCM" "Phase Change Material"
liquid  = makeCC "Liquid" "liquid state"
solid   = makeCC "Solid" "solid state"

...

Paragraph (S "This derivation does not consider the " :+:
(sMap (map toLower) (S (boiling ^. name))) :+: S " of the " :+:
S (phsChgMtrl ^. name) :+: S ", as the " :+: S (phsChgMtrl ^.
name) :+: S " is assumed to either be in" :+: S " a " :+:
(solid ^. descr) :+: S " or a " :+: (liquid ^. descr) :+:
S " (A18).")]
```

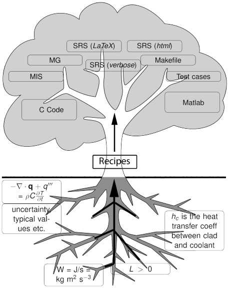
Future Work

- Cleaner separation between knowledge and recipes
- Generate additional software artifacts
- Capture design decisions
- Develop alternative recipes
- Assurance case for FMRI statistical correlation
- Predict solid fraction for metal alloy cooling
- Testing
  - Guards on input
  - Sanity checks
  - Metamorphic testing
  - Computational variabilty testing

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

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. ISBN 0-7695-2828-7. doi: http://dx.doi.org/10.1109/ICSE.2007.77.

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.

References II

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. URL `http://dl.acm.org/citation.cfm?id=2555523.2555555`.

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. doi: 10.1016/j.jss.2015.07.027. URL `http://dx.doi.org/10.1016/j.jss.2015.07.027`.

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 Softw.*, 24(6):120–119, 2007. ISSN 0740-7459. doi: http://dx.doi.org/10.1109/MS.2007.155.

Zeeya Merali. Computational science: ...error. *Nature*, 467:775–777, 2010.

Steven J. Owen. A survey of unstructured mesh generation technology. In *INTERNATIONAL MESHING ROUNDTABLE*, pages 239–267, 1998.

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

References IV

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`.

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.

References V

Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. Best practices for scientific computing. *PLoS Biol*, 12 (1):e1001745, 01 2014. doi: 10.1371/journal.pbio.1001745. URL `http://dx.doi.org/10.1371%2Fjournal.pbio.1001745`.

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. URL `http://www.americanscientist.org/issues/pub/wheres-the-real-bottleneck-in-scientific-co`