Literate Scientific Software

Spencer Smith, Dan Szymczak and Jacques Carette

Computing and Software Department
Faculty of Engineering
McMaster University

PASC, MS06, June 16, 2016

Literate Scientific Software

1. Background
2. Literate Scientific Software
3. Drasil
4. Next Steps
5. Conclusions

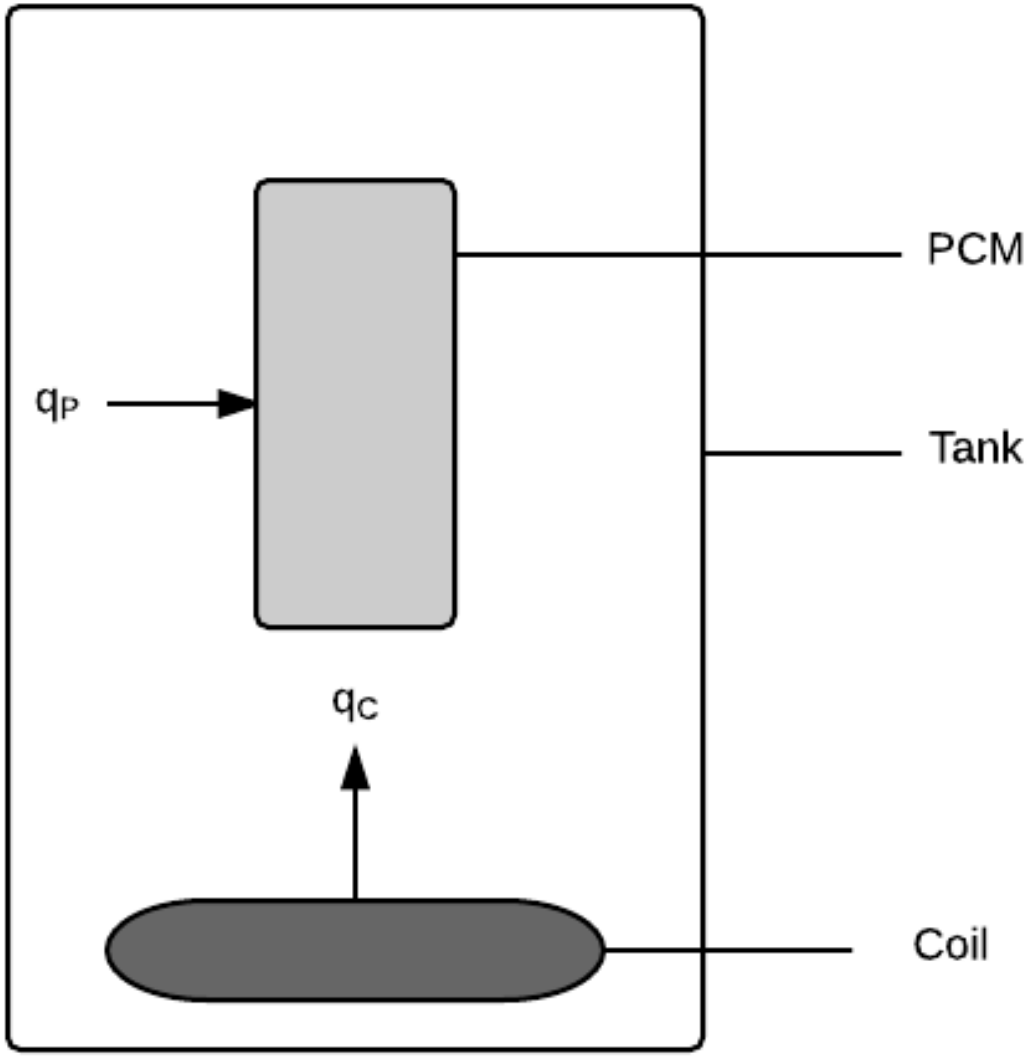
Important SS Qualities

- Reusability
- Maintainability
- Verifiability
- Reproducibility

Challenges

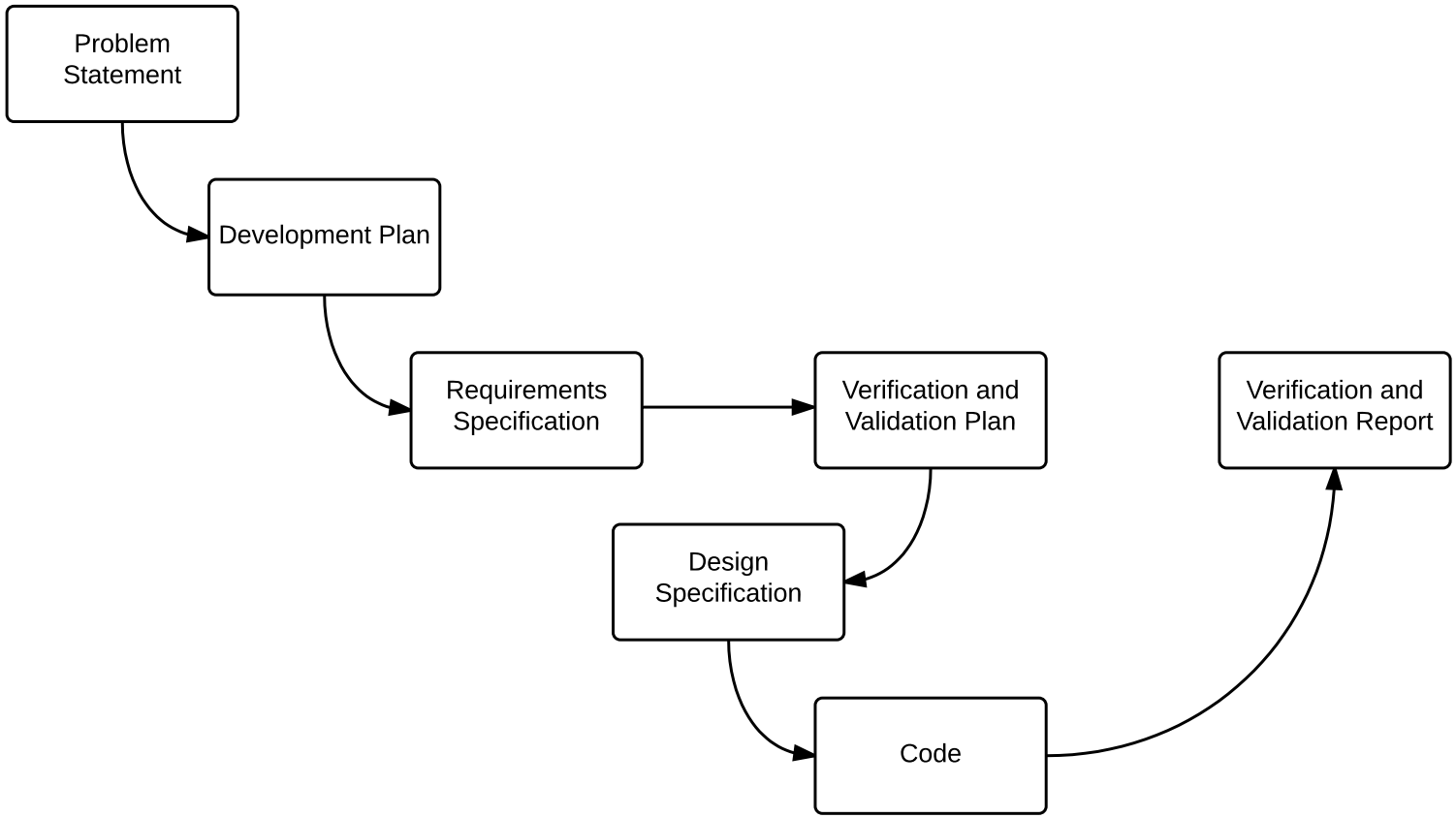
- Up front requirements
- Rapid change for numerical algorithms
- Information duplication
- Synchronization headaches between artifacts
- Perceived over-emphasis on non-executable artifacts

Solar Water Heating Tank

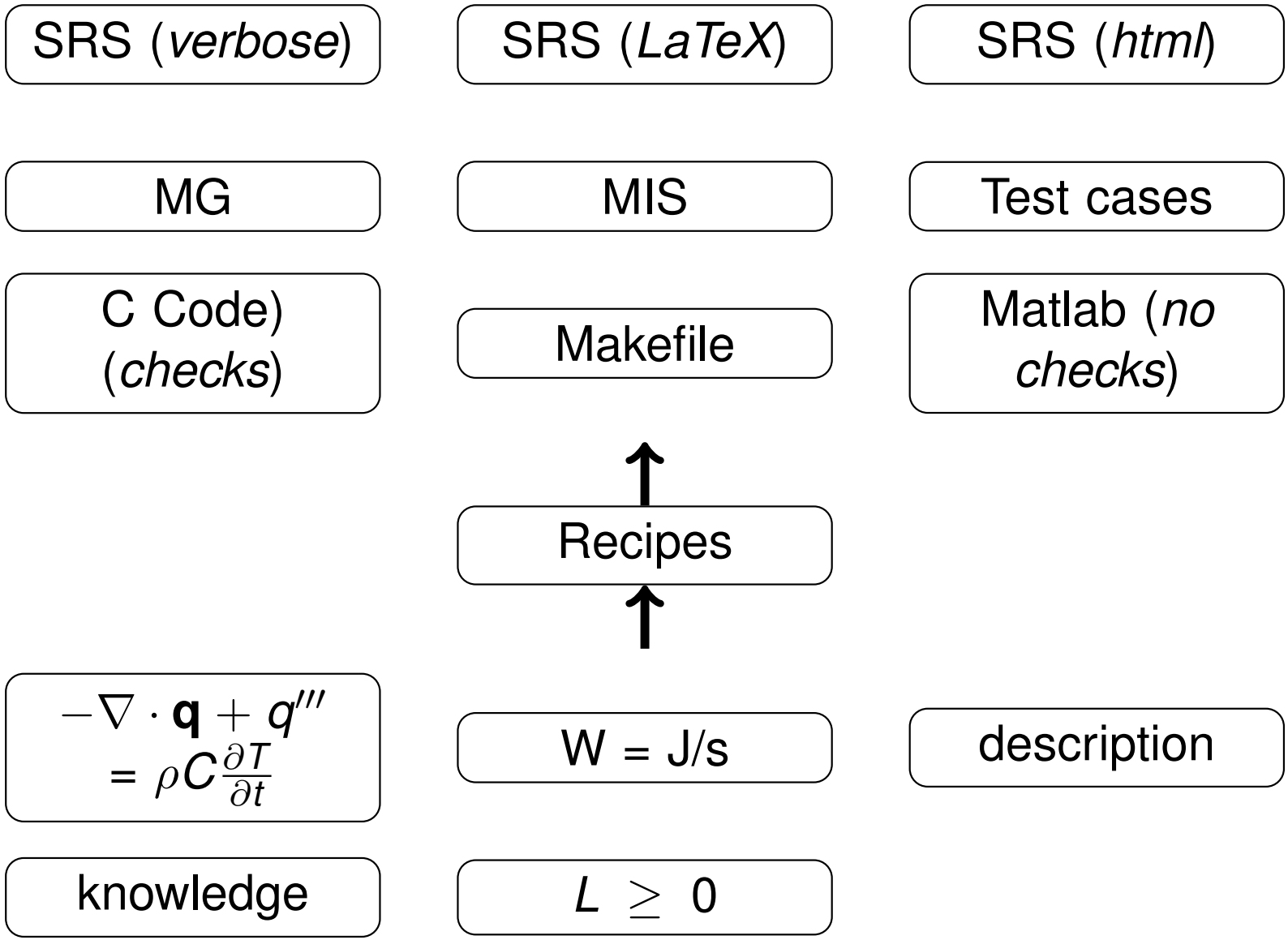


https://github.com/smiths/swhs

“Faked” Rational Design Process



Literate Scientific Software



How Addresses Challenges

- 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

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 kg/m^3	10%

- ??Add latest from Brooks??
- Sanity checks captured and reused
- Generate guards against invalid input
- Generate test cases

Reusability

Number	**T1**
Label	**Conservation of energy**
Equation	$-\nabla \cdot \mathbf{q} + q''' = \rho C \frac{\partial T}{\partial t}$
Description	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.

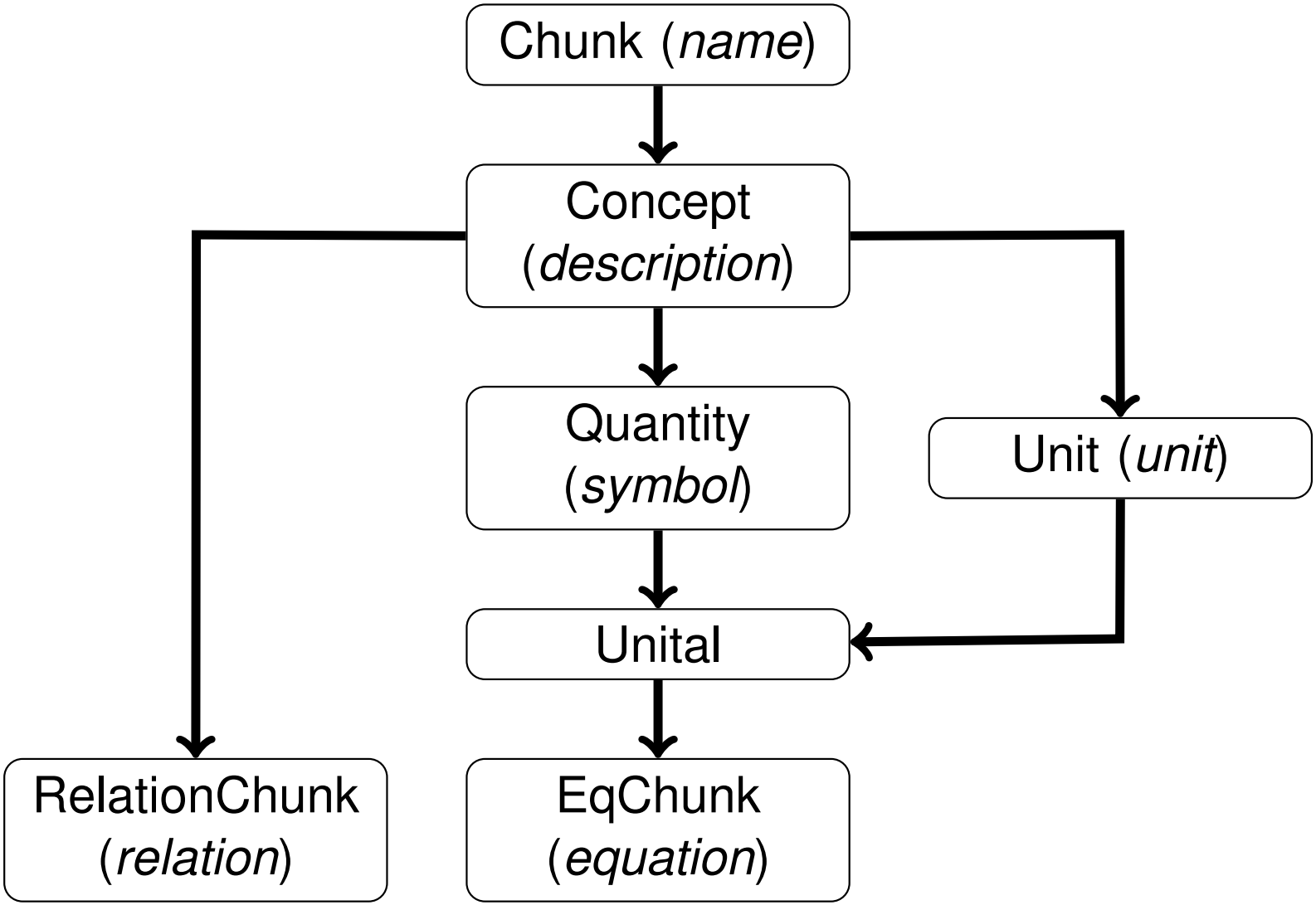
Maintainability

show assumptions

Reproducibility

- Knowledge is explicitly stored for the future
- Recipes can be use to regenerate any artifacts
- Recipes include build instructions

Drasil Framework Design



SRS for h_g and h_c

Spencer Smith

May 15, 2016

1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are employed as described below. For each unit, the symbol is given followed by a description of the unit with the SI name in parentheses.

Symbol	Description
m	length (metre)
kg	mass (kilogram)
s	time (second)
K	temperature (kelvin)
mol	amount of substance (mole)
A	electric current (ampere)

Example Recipe

```
srsBody = srs [h_g, h_c] "Spencer Smith" [s1,s2]

s1 = Section (S "Table of Units") [intro, table]

table = Table
 [S "Symbol", S "Description"] (mkTable
   [(\x -> Sy (x ^. unit)),
    (\x -> S (x ^. descr)) ] si_units)

intro = Paragraph (S "Throughout this ...")
```

Reusable Chunks

```
metre, second, kelvin :: FundUnit
metre  = fund "Metre"  "length (metre)"       "m"
second = fund "Second" "time (second)"        "s"
kelvin = fund "Kelvin" "temperature (kelvin)" "K"
```

The h_c Chunk

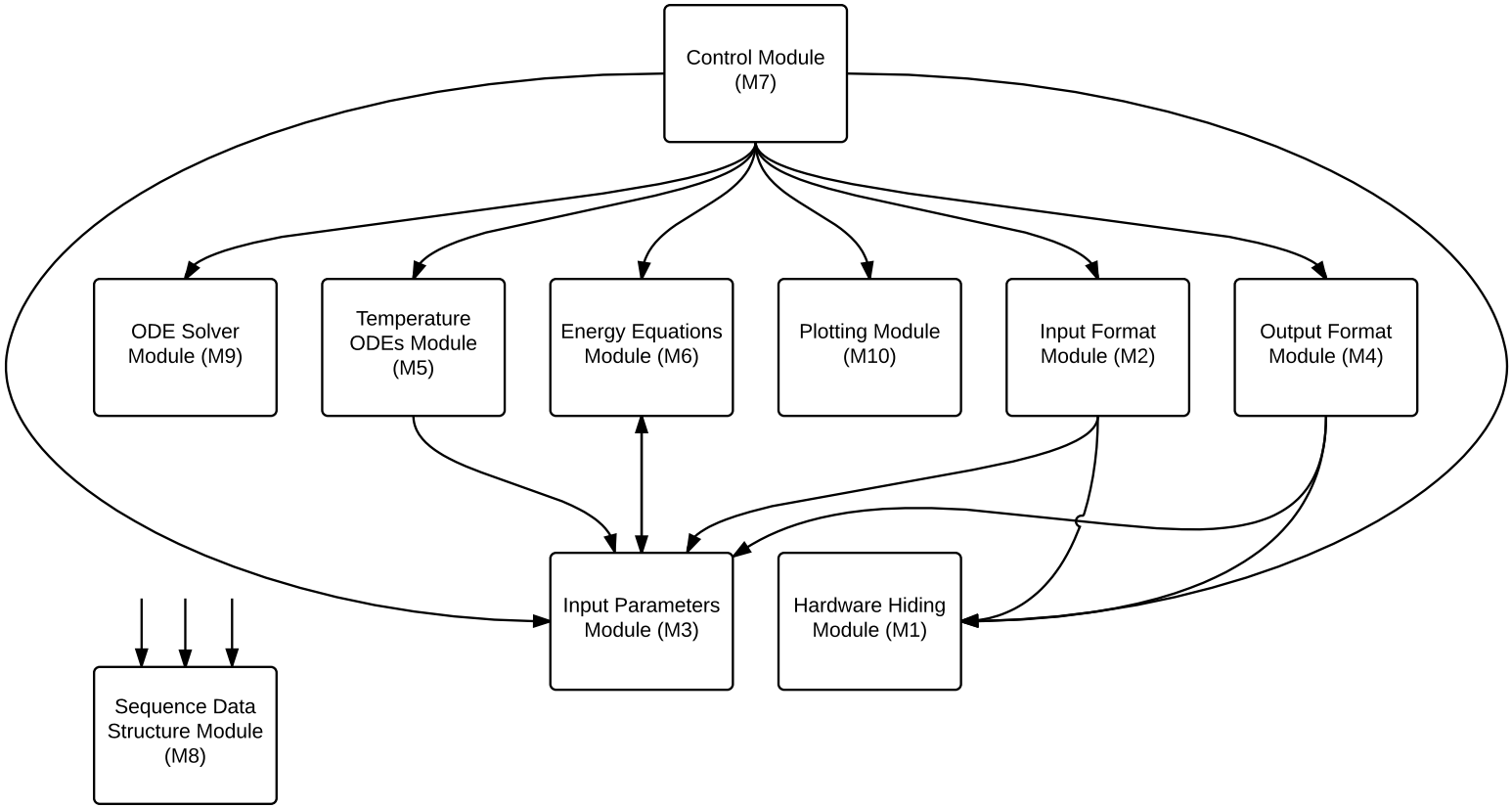
$$h_c = \frac{2k_c h_b}{2k_c + \tau_c h_b}$$

```
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"
 "convective heat transfer coefficient between
    clad and coolant"
   (sub h c) heat_transfer h_c_eq
```

Table of Symbols

```
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"
 "convective heat transfer coefficient between
    clad and coolant"
   (sub h c) heat_transfer h_c_eq
```

Next Steps: Design Documentation



Generate Code to Solve Instanced Models

part of IM2

Approach to Developing Drasil

- Case studies
  - Solar water heating tank
  - Slope stability analysis
  - Glass safety analysis
  - Game physics engine
- Practical
- Decompose into small chunks
- Look for patterns

Conclusions

- SCS has the opportunity to lead other software fields by leveraging its solid existing knowledge base
- DDD is feasible with a knowledge-based approach
- Documentation for QA and software certification does not have to be painful, expensive or time consuming
- Drasil will be developed via practical case studies