

CAS 741 (Development of Scientific Computing Software)

Winter 2023

Artifact Generation

Dr. Spencer Smith

Faculty of Engineering, McMaster University

March 23, 2023



Artifact Generation

- Administrative details
- Finish Assurance Case review
- Artifact generation (Drasil)

Administrative Details: Report Deadlines

Final Documentation Week 13 Apr 12

- The written deliverables will be graded based on the repo contents as of 11:59 pm of the due date
- If you need an extension for a **written** doc, please ask
- When ready, assign issues to your primary and secondary reviewers
- GitHub issues due two days after assignment deadlines
- From Drasil Code onward, Drasil projects no longer need to maintain traditional SRS

Presentation Schedule

- Drasil Project Present (25 min each)
 - ▶ **Mar 30: Karen, Sam, Jason**

Presentation Schedule

- Test or Impl. Present (25 min each)
 - ▶ Apr 5: Lesley, Deesha, Volunteer?
 - ▶ Apr 6: Mina, Joachim, Maryam
- 4 presentations each (please check)
- If you will miss a presentation, please trade with someone else
- Implementation presentation could be used to run a code review, or code walkthrough

Course Evaluations

- Course experience surveys will close on Wednesday, April 12, 2023 11:59 PM
- <https://mcmaster.bluera.com/mcmaster/>

Questions?

- Questions on administrative details?
- Questions on final documentation?
- Questions on reflection document?
- Questions on final implementation?
- Questions on VnV report?
- Other questions?

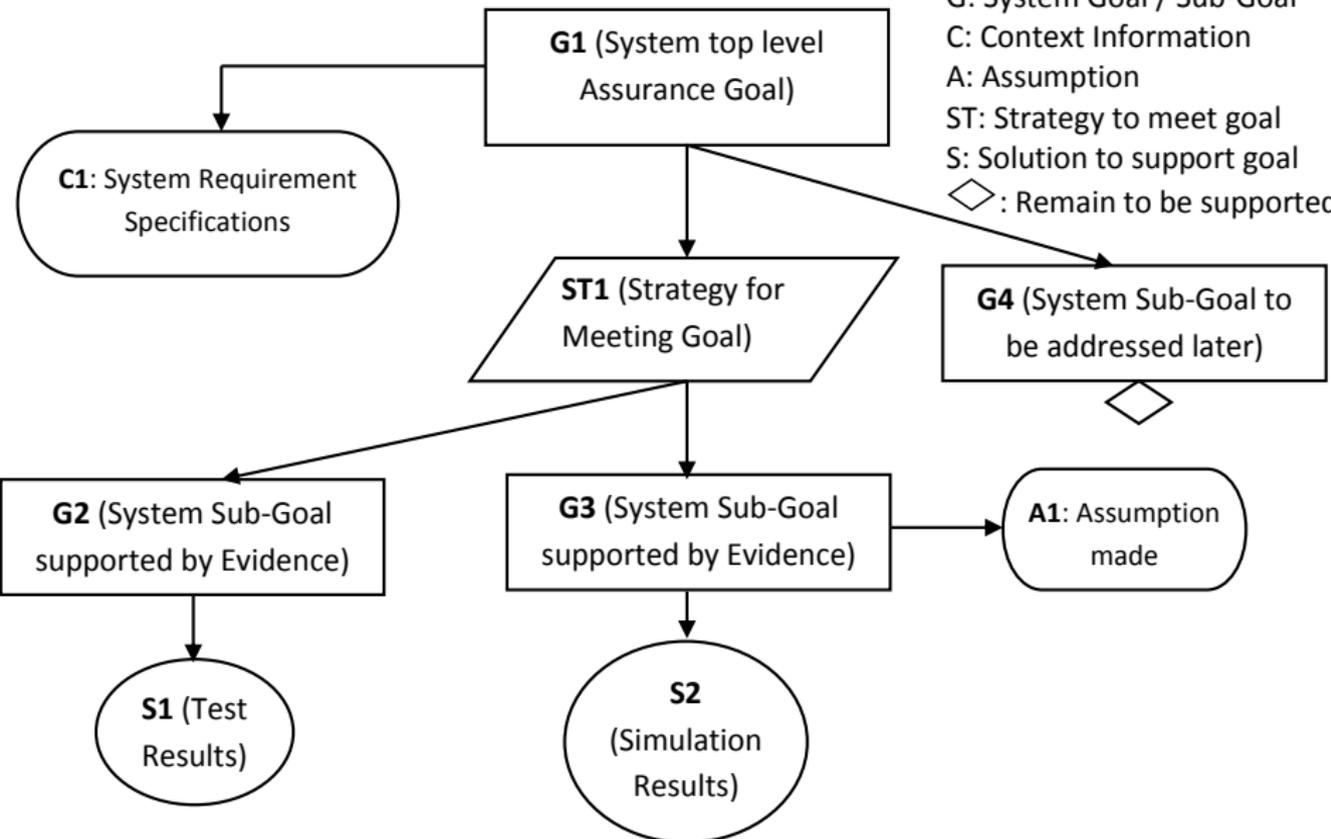
Assurance Cases in Scientific Computing [14, 13]

- Assurance cases
 - ▶ Organized and explicit argument for correctness
 - ▶ Successfully used for safety critical systems
- Advantages for SC
 - ▶ Engaging domain experts
 - ▶ Producing necessary and relevant documentation
 - ▶ Evidence that can be verified/replicated by a third party
- Example of 3dfim+
 - ▶ No errors found
 - ▶ However
 - ▶ Documentation ambiguities
 - ▶ No warning about parametric statistical model

Assurance Cases in SC Motivation

- Do we put too much trust in the quality of SCS?
- Are enough checks and balances in place, especially for safety related software?
- Problems with imposing external requirements for certification
 - ▶ External body does not have expertise
 - ▶ SCS developers dislike documentation
- Solution – Assurance Cases by experts
 - ▶ Experts engaged
 - ▶ Relevant documentation
- Current techniques of development and testing still used, but arguments will no longer be ad hoc and incompletely documented

G: System Goal / Sub-Goal
C: Context Information
A: Assumption
ST: Strategy to meet goal
S: Solution to support goal
◇: Remain to be supported



[A] AFNI: tmp/LRtap/mdef3d_01+ori

[Order: RAI=DICOM]
 x = -39.500 mm [R]
 y = 31.500 mm [P]
 z = 45.500 mm [S]

Xhairs Multi X+

Color green

Gap 5 Wrap

Index

Axial Image Graph

Sagittal Image Graph

Coronal Image Graph

◆ Original View
 ◆ AC-PC Aligned
 ◆ Talairach View

See Markers

See OverLay

[A] AFNI: tmp/LRtap/m

Colr
 Swap
 Norm

c
b
r
g
l
9
z

pan
crop

141

Left=Right float [2%-98%]

[A] AFNI: tmp/LRtap/mdef3d_01+ori

Colr
 Swap
 Norm

c
b
r
g
l
9
z

pan
crop

135

Float [2%-98%]

[A] AFNI: tmp/LRtap/m

Colr
 Swap
 Norm

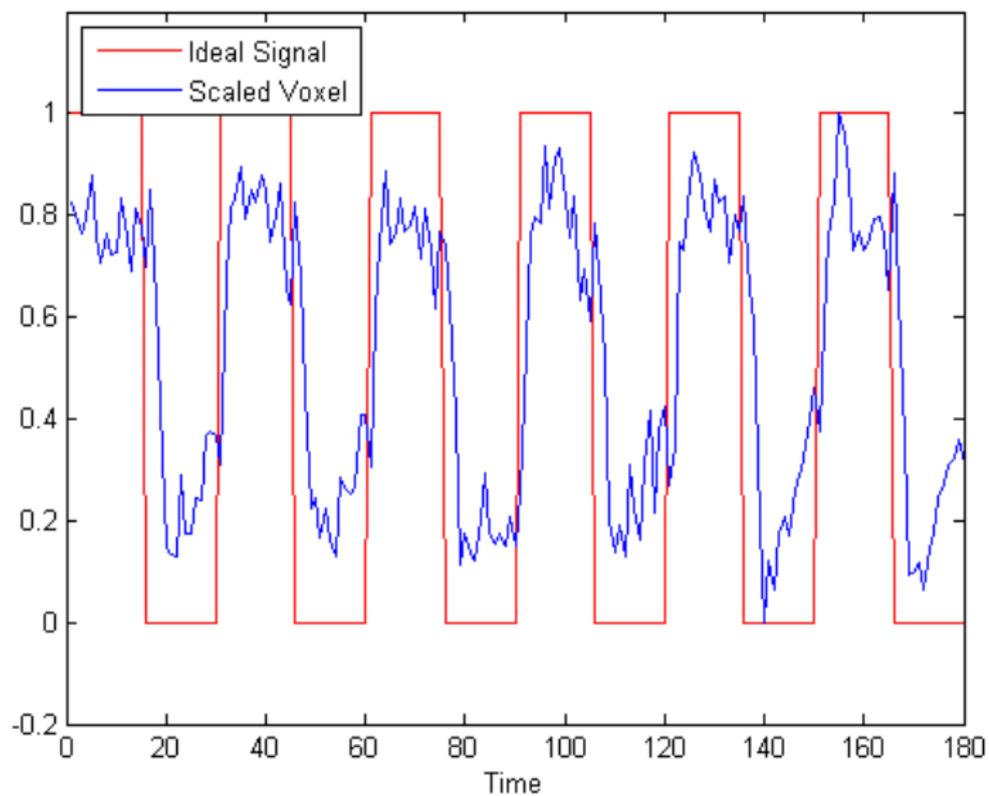
c
b
r
g
l
9
z

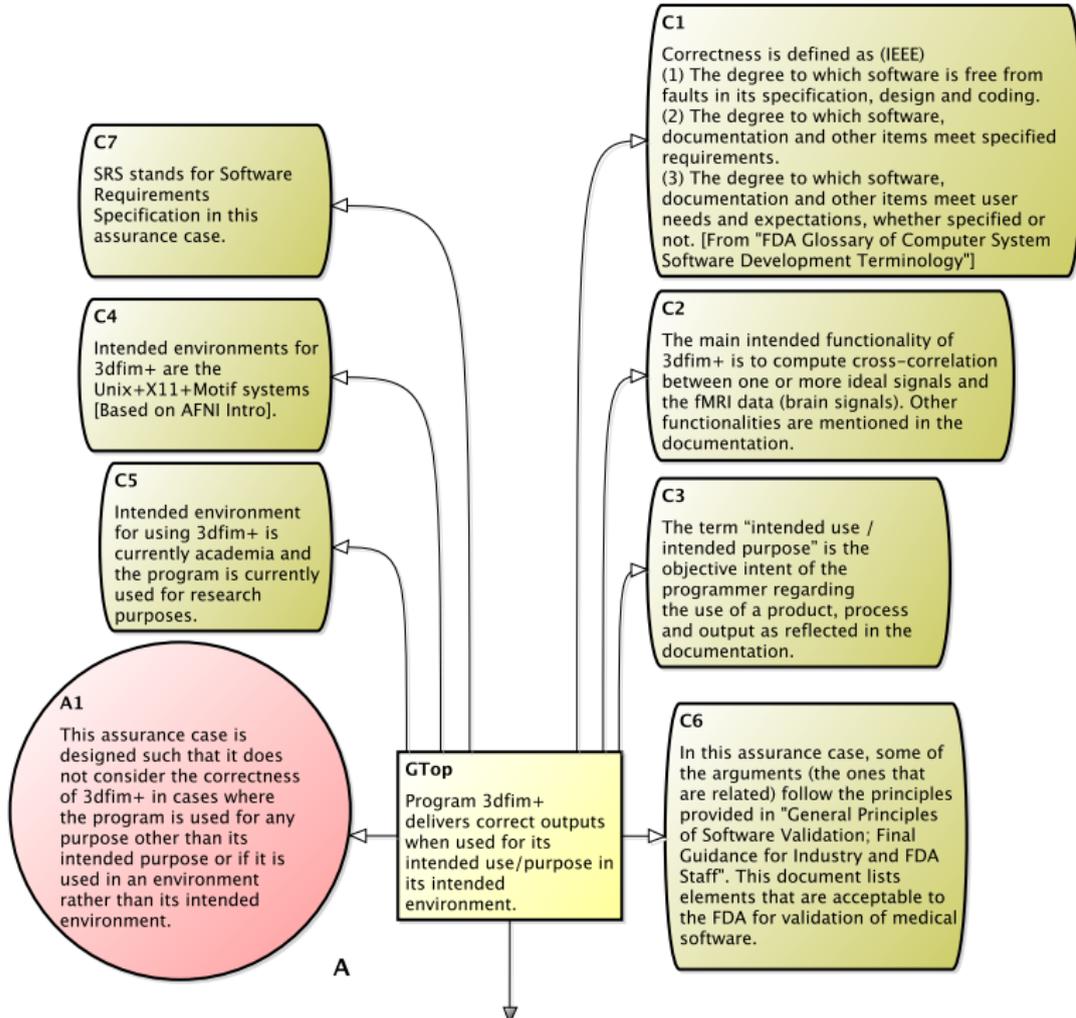
pan
crop

159

18%

Scaled Voxel (23,27,22) and Ideal Signal over time





G_{Top}
Program 3dfim+ delivers correct outputs when used for its intended use/purpose in its intended environment.

S_{Top}
G can be decomposed into:
GR. 3dfim+ requirements are documented and documentation of the requirements is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GD. The design of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GI. The implementation of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
GA. Inputs to 3dfim+ satisfy the defined operational assumptions.

Reasoning Proof:
Premise: GR, GD, GI and GA are true.
Conclusion: G_{Top} is valid.

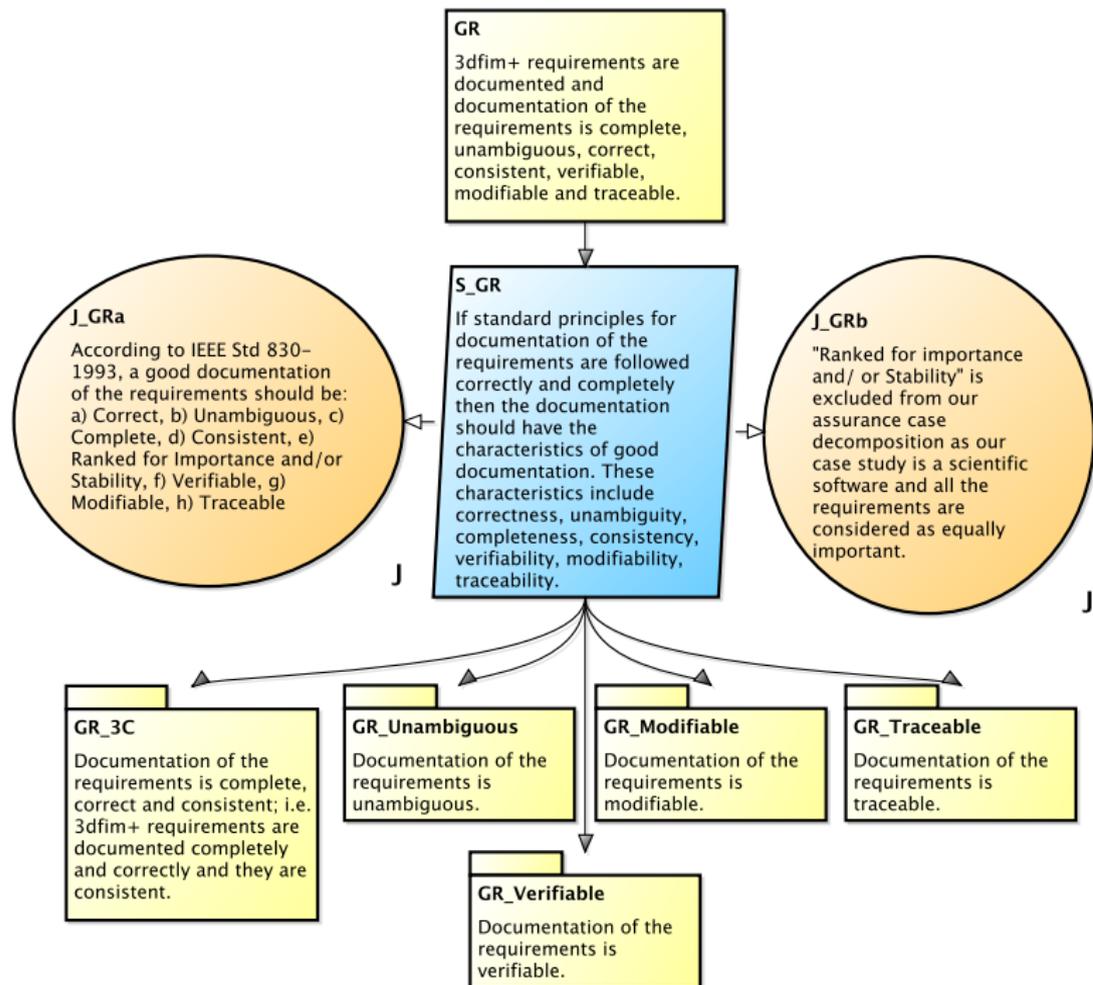
GR
3dfim+ requirements are documented and documentation of the requirements is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

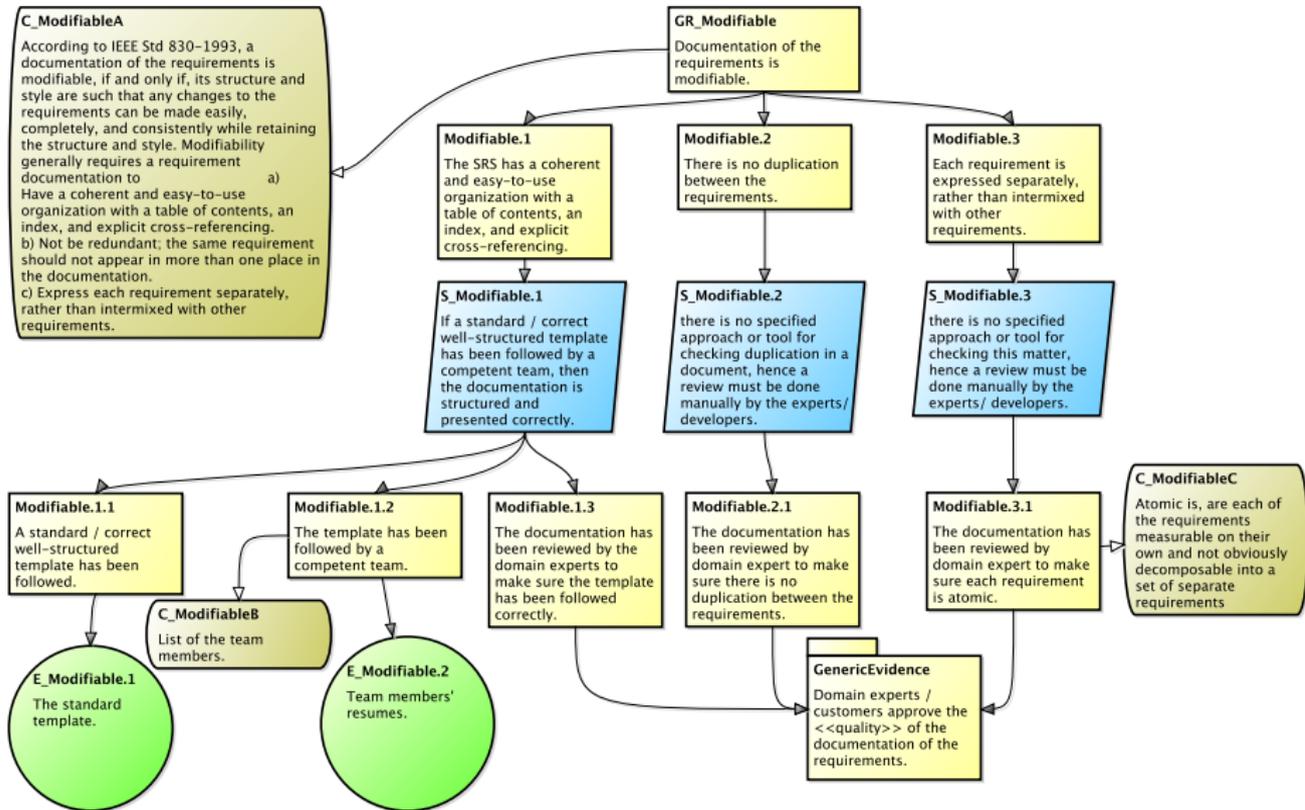
GD
The design of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

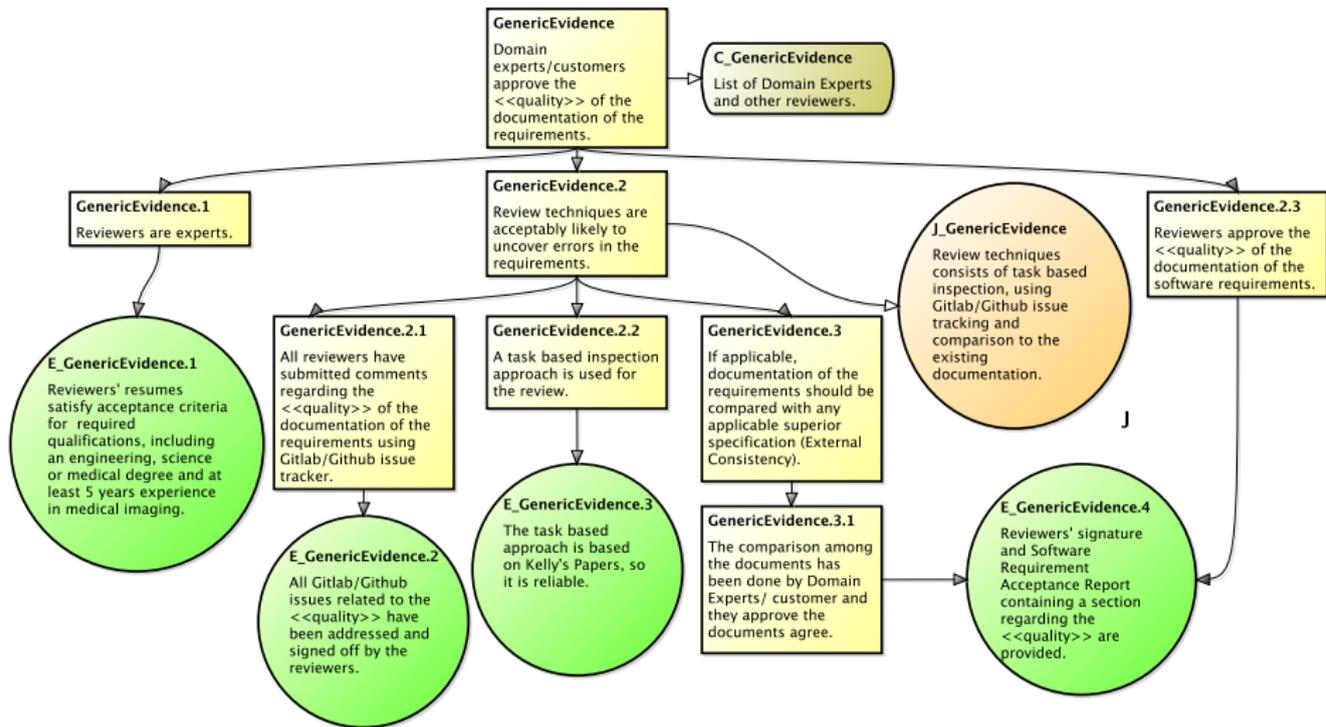
GI
The implementation of 3dfim+ complies with its requirements and it is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.

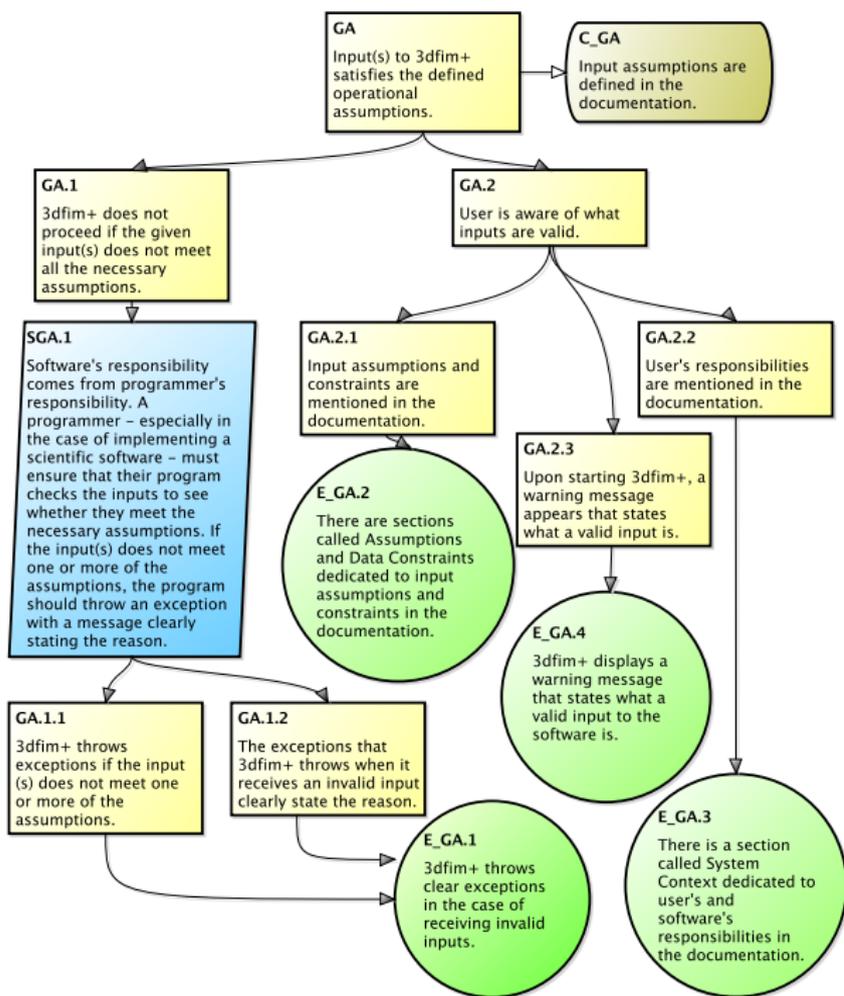
GA
Inputs to 3dfim+ satisfy the defined operational assumptions.

J_{Top}
The major software development lifecycle steps are: Requirements, Design and Implementation with appropriate V&V activities. V&V activities will be reflected in claims regarding validation of requirements, and verification of design and implementation. If requirements are appropriate, and design and implementation are appropriate and they comply with the requirements, then 3dfim+ will have been shown to deliver correct outputs. Moreover, as meeting the input assumptions is of great importance, it is considered as a separate goal; however, the correctness, completeness and consistency of the assumptions have been shown in the GR as a part of the requirements correctness, completeness and consistency.









Proposed Changes to 3dfim+

- No mistakes found in calculations
- Goal of original software was not certification
- Problems found
 - ▶ GR goal not satisfied
 - ▶ Not complete, verifiable, modifiable or traceable
 - ▶ Coordinate system information missing
 - ▶ Ambiguous rank function
 - ▶ Inputs not checked in code
 - ▶ User not informed of their responsibility to use tool with correct statistical model

Concluding Remarks

- Hopefully motivated assurance cases for SC
- Quality is improved by looking at a problem from different perspectives, assurance cases provide a systematic and rigorous way to introduce a new perspective
- An assurance cases will likely use the same documentation and ideas used in CAS 741
- However, an assurance case can focus and direct efforts right from the start of the project

Abstract for Artifact Generation Talk

- **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



PRODUCT SPECIFICATIONS

The appearance and specifications listed in this manual may vary due to constant product improvements.

Electrical requirements: 115 V, 60 Hz

Min. / Max. water pressure: 20 - 120 psi (138 - 827 kPa)

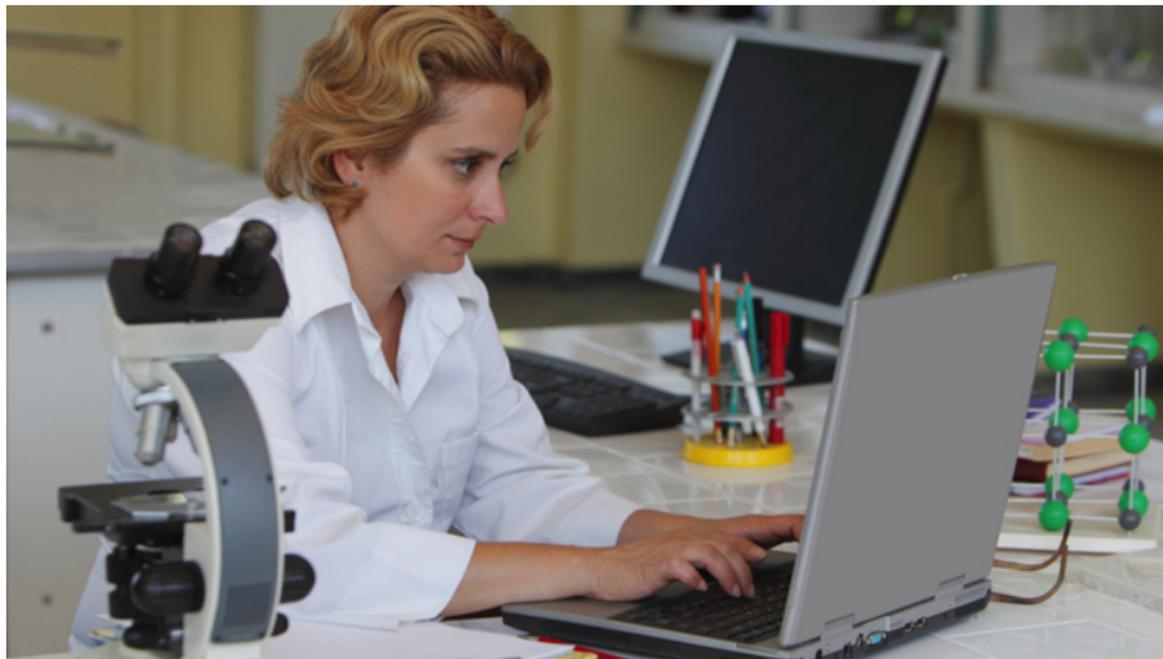
Model	LFCC22426*
Description	Counter-depth, French door refrigerator, bottom freezer
Net weight	243 lb (110 kg)

Model	LFCS27596*
Description	Standard-depth, Door-in-Door French door refrigerator, bottom freezer
Net weight	284 lb (129 kg)

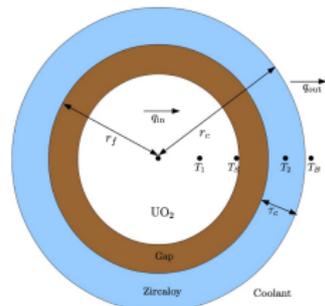
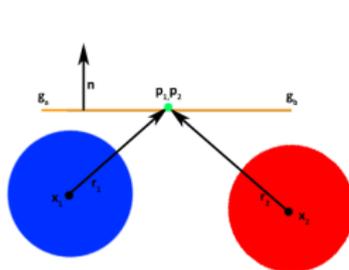
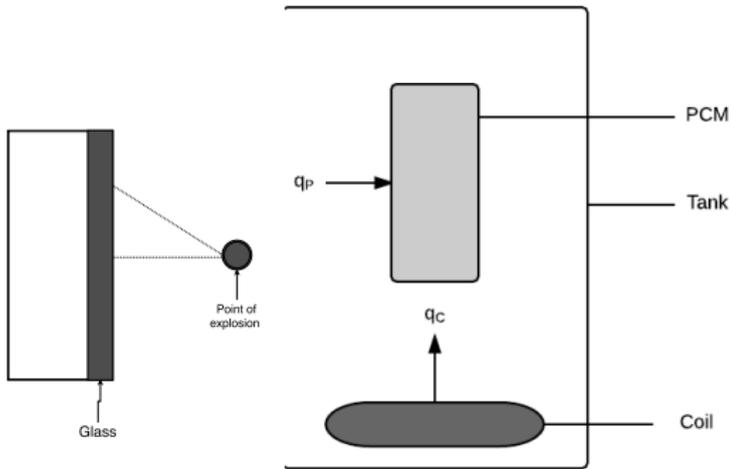
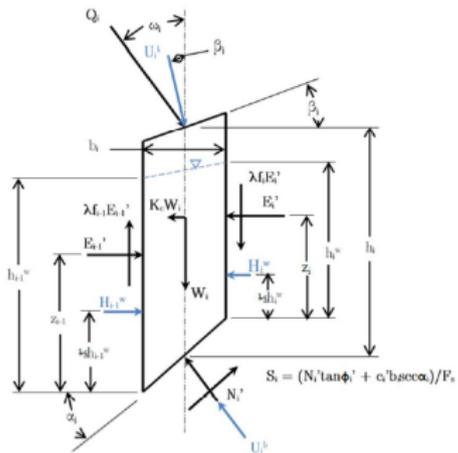
Model	LFCC23596*
Description	Counter-depth, Door-in-Door French door refrigerator, bottom freezer
Net weight	269 lb (122 kg)



Scope: End User Developers

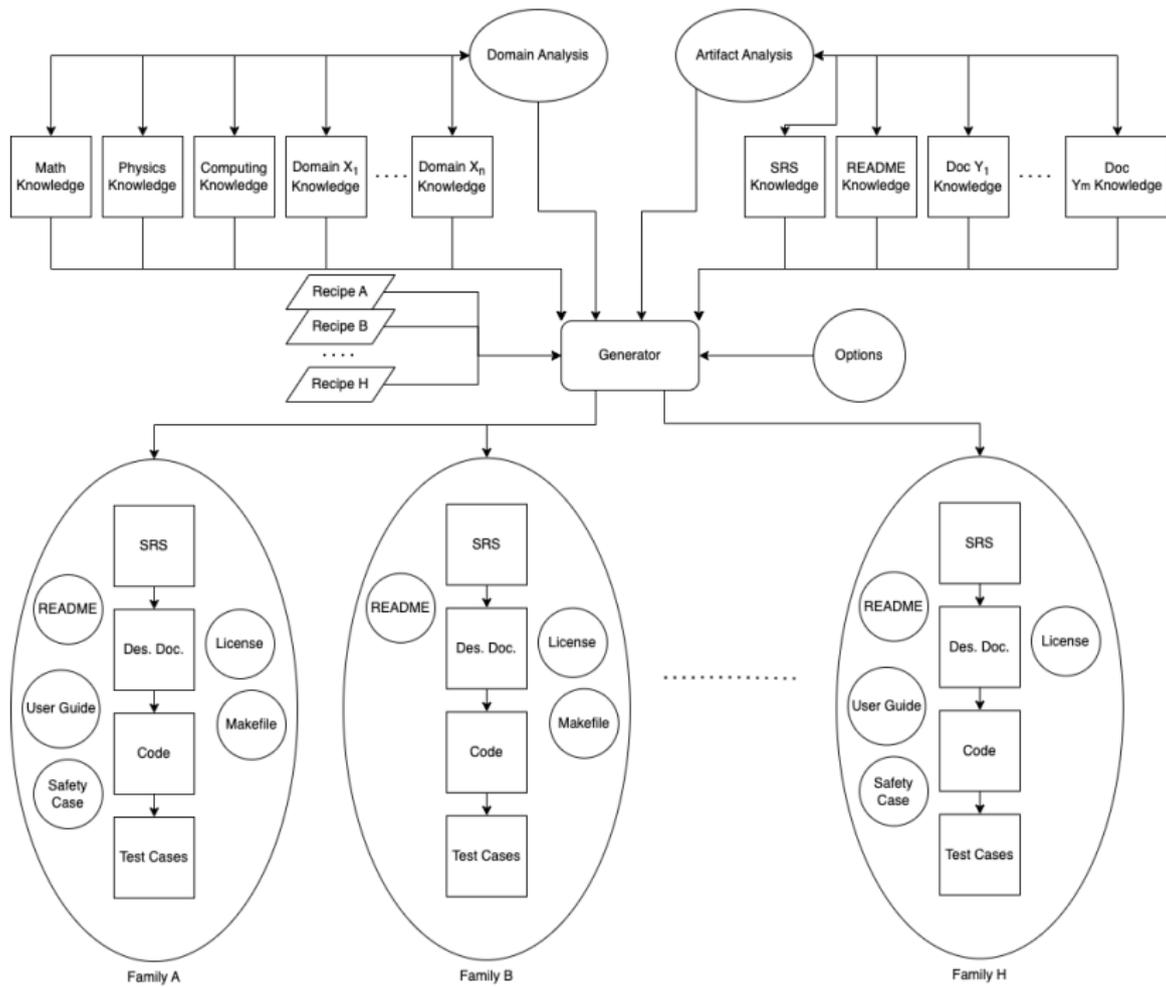


Scope: Physical Science



Build on Success of MDSE

- Codify (capture) code and non-code info together
 - ▶ Natural language (text)
 - ▶ Definitions
 - ▶ Assumptions
 - ▶ Rationale, Derivations
 - ▶ Abstract theory
 - ▶ Etc.
- Generate all artifacts from one framework
 - ▶ Requirements
 - ▶ User manuals
 - ▶ Build scripts, dev environment (CI etc)
 - ▶ Assurance case
 - ▶ Code (in different languages)
 - ▶ Test cases
 - ▶ etc.



$$\mathbf{a} = \frac{d\mathbf{v}}{dt} \text{ and } \mathbf{v} = \frac{d\mathbf{p}}{dt}$$

$$\mathbf{F} = m\mathbf{a}$$

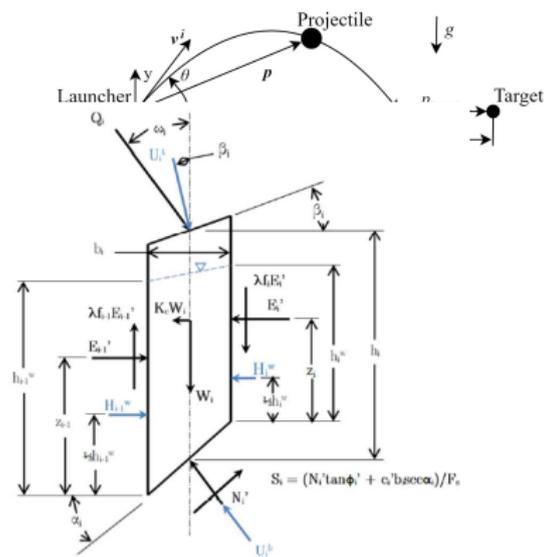
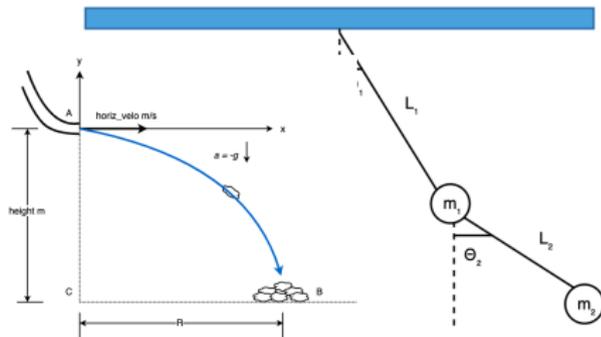
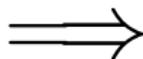
$$m \frac{d\mathbf{v}}{dt} = m\mathbf{g} - c\mathbf{v}$$

$$g = 9.8m/s^2 \text{ or } g = 32.2ft/s^2$$

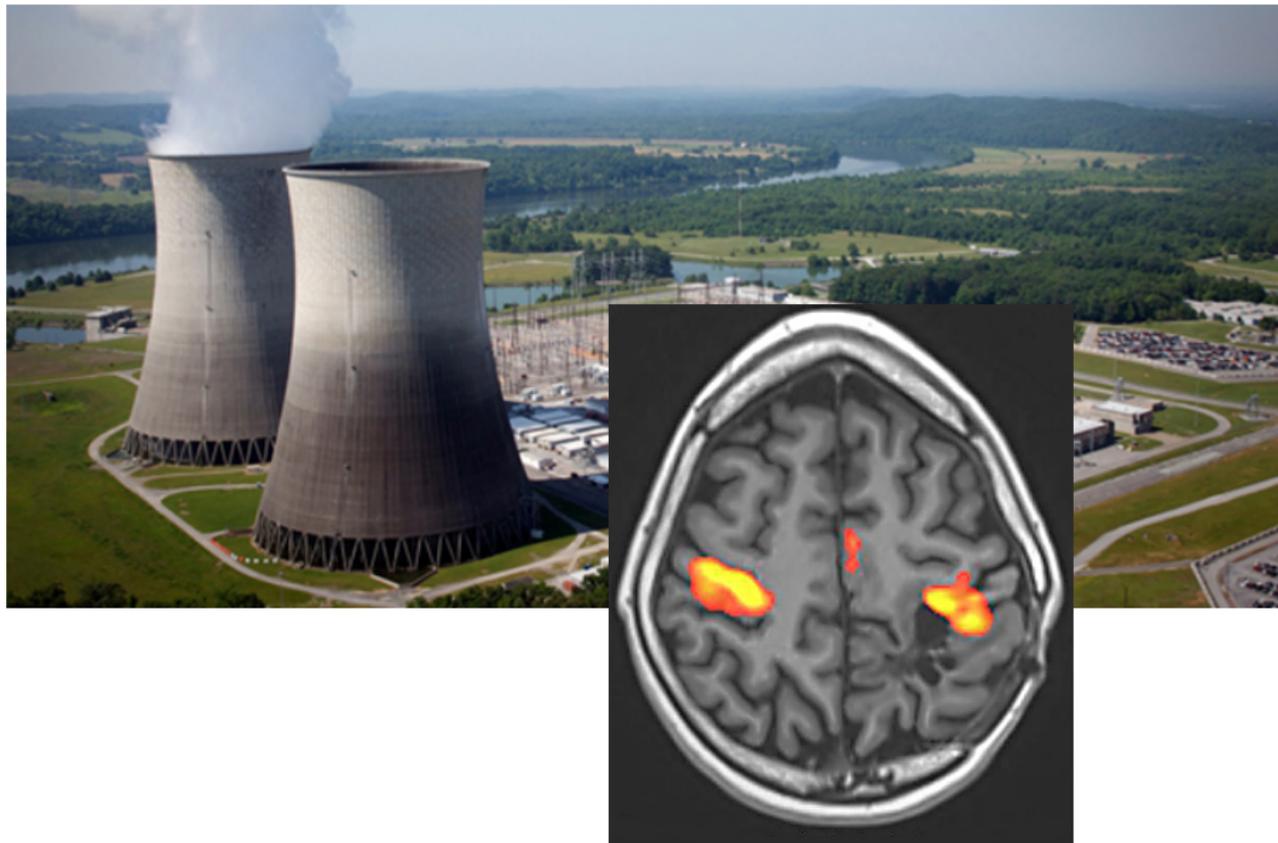
$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$$

$$\sigma_{ij} = D_{ijkl}\epsilon_{kl}$$

coordinate system



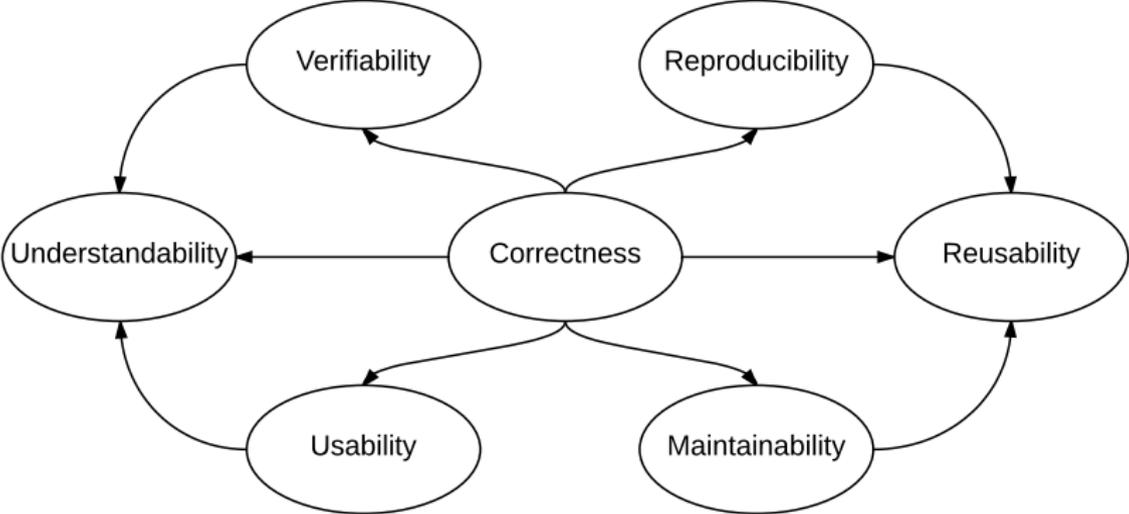
Motivation: Safety



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 [16]
- 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
SWHS	1000	F77	1	5	X	✓	X	X	X
Astro	5000	C	2	10	X	✓	X	X	X
Glass	1300	F90	1	<1	X	✓	X	X	X
Soil	800	M	1	5	✓	✓	✓	✓	X
Neuro	1000	M	1	5	✓	✓	X	✓	X
Acoust	200	M	4	2.5	X	✓	X	X	X

Perceived Advantages from Participants

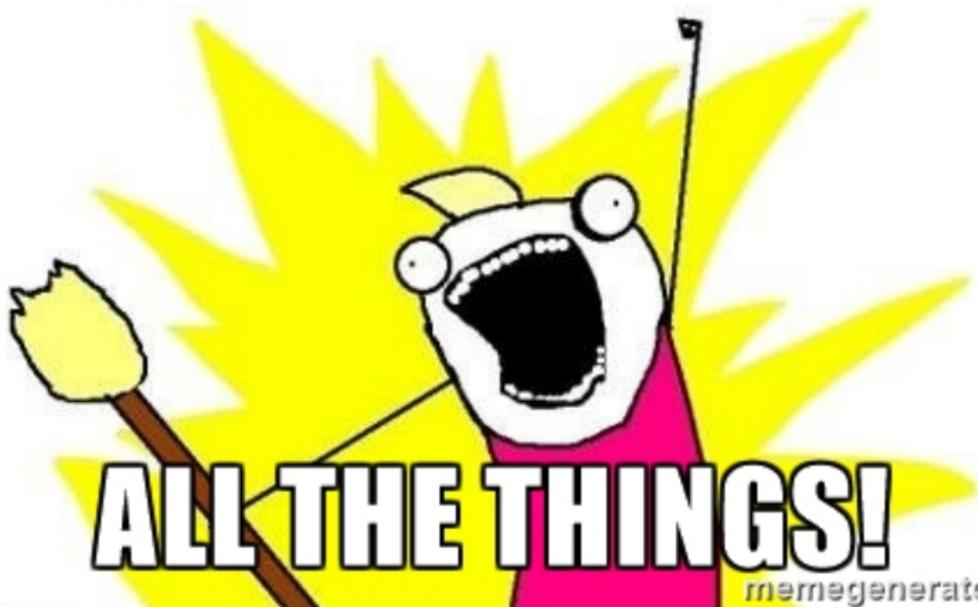
- Documentation of assumptions
- All variables have explicit units
- SRS helpful with new graduate students
- Modules result in more user friendly code
- Traceability between modules and requirements useful
- Better organized code
- Information sharing on design choices
- Detailed record of knowledge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

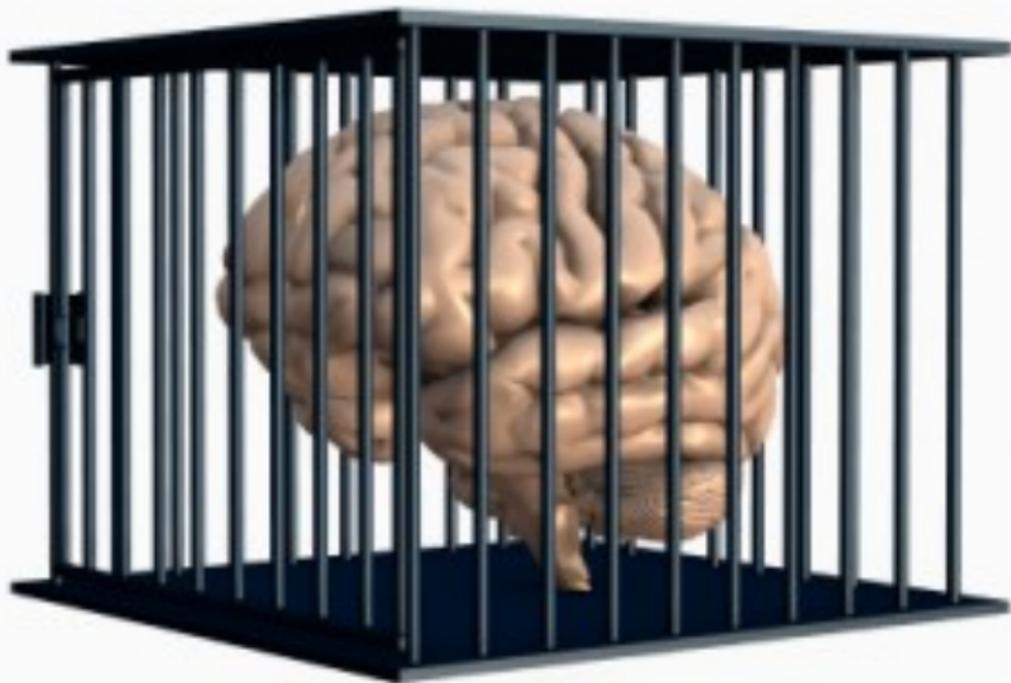
The Solution?

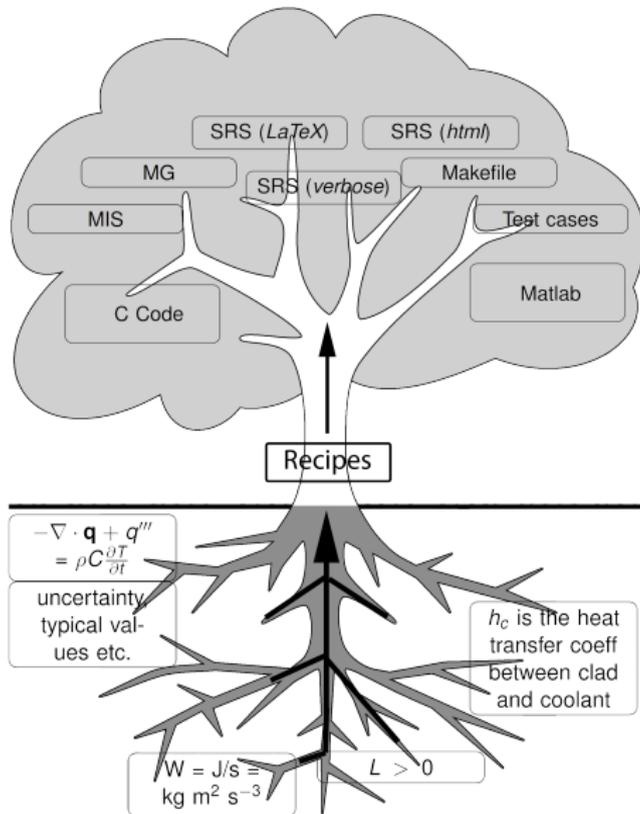
GENERATE

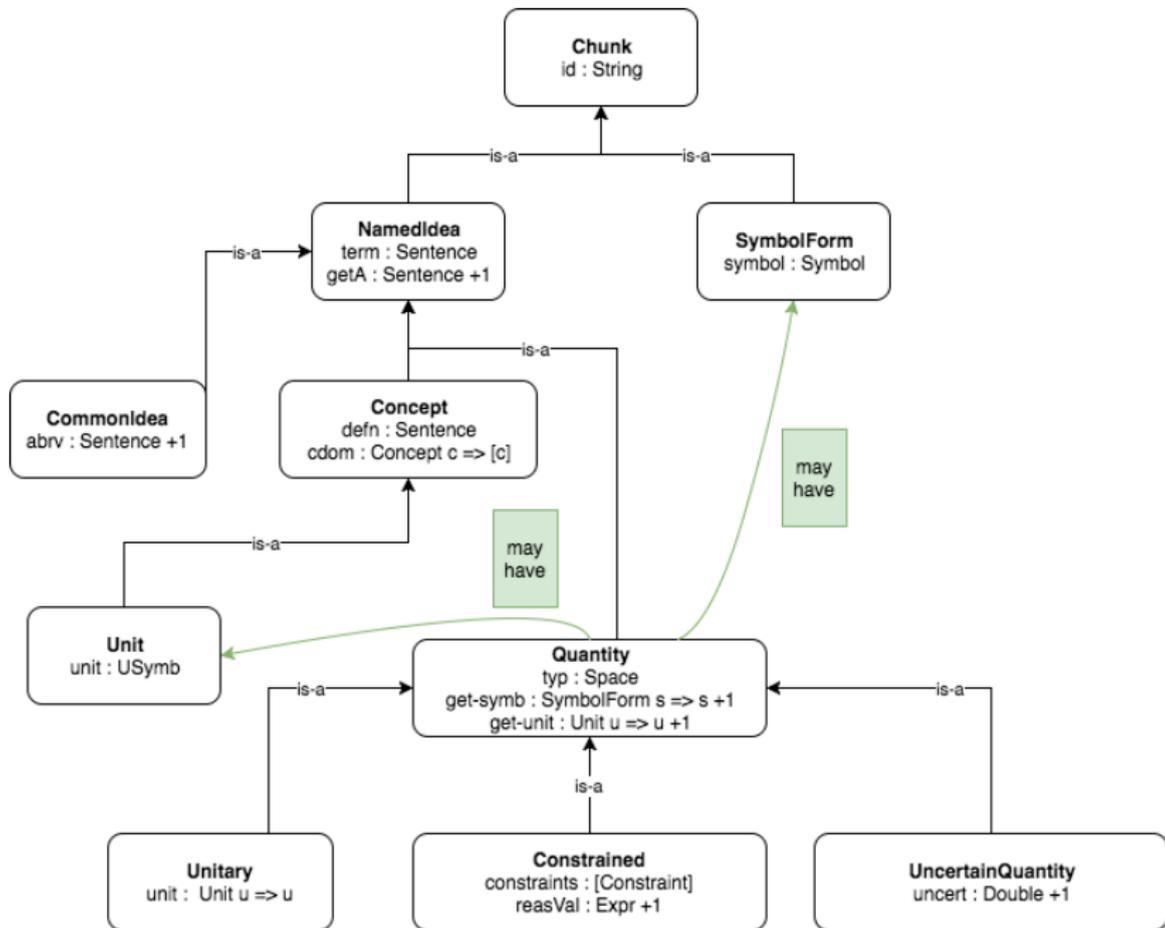


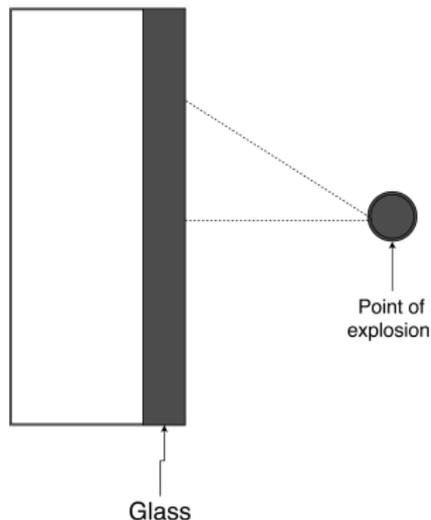
memegenerator.net

Knowledge Capture









Given

- dimensions of glass plane
- glass type
- explosion characteristics
- tolerable breakage probability

Predict whether the glass will withstand the explosion

Drasil Inputs:

- Program Name: GlassBR
- Authors: Nikitha K and Spencer S
- Symbols: tolerable load (\hat{q}_{tol}), Risk of failure (B), ...
- Assumptions: Load duration factor constant,
- Data definitions: relation for B , ...
- Design decisions:
 - Modularity (input module),
 - Implementation Type (Program),
 - Logging (Yes),
 - Input Structure (Bundled),
 - Constant Structure (Inlined),
 - Constant Rep (Constants),
 - Real Number Rep (Double),
 - ...

Drasil Inputs:

- Program Name: GlassBR
- Authors: Nikitha K and Spencer S
- Symbols: tolerable load (\hat{q}_{tol}), Risk of failure (B), ...
- Assumptions: Load duration factor constant,
- Data definitions: relation for B , ...
- Design decisions:
 - Modularity (input module),
 - Implementation Type (Program),
 - Logging (Yes),
 - Input Structure (Bundled),
 - Constant Structure (Inlined),
 - Constant Rep (Constants),
 - Real Number Rep (Double),
 - ...

```
/glassbr
/Website/GlassBR_SRS.html
/Website/GlassBR_SRS.css
/SRS/bibfile.bib
/SRS/Makefile
/SRS/GlassBR_SRS.tex
/SRS/GlassBR_SRS.pdf
/src/python
/src/python/README.md
/src/python/InputParameters.py
/src/python/Calculations.py
/src/python/Makefile
/src/python/doxConfig
...
```

```
...
/src/java/GlassBR/Calculations.java
/src/java/Makefile
/src/java/README.md
...
/src/cpp/GlassBR
/src/cpp/ReadTable.cpp
/src/cpp/InputFormat.hpp
/src/cpp/Calculations.cpp
...
/src/swift/Calculations.swift
...
/src/csharp/Control.cs
...
```

/glassbr
/Website/GlassBR_SRS.html
/Website/GlassBR_SRS.css
/SRS/bibfile.bib
/SRS/Makefile
/SRS/GlassBR_SRS.tex
/SRS/GlassBR_SRS.pdf
/src/python
/src/python/README.md
/src/python/InputParameters.py
/src/python/Calculations.py
/src/python/Makefile
/src/python/doxConfig
...

...
/src/java/GlassBR/Calculations.java
/src/java/Makefile
/src/java/README.md
...
/src/cpp/GlassBR
/src/cpp/ReadTable.cpp
/src/cpp/InputFormat.hpp
/src/cpp/Calculations.cpp
...
/src/swift/Calculations.swift
...
/src/csharp/Control.cs
...

Software Requirements Specification for GlassBR

Nikitha K and Spencer S

Table of Symbols

\hat{q}_{tol}

B

...

Introduction

... The software, herein called **GlassBR**, ...

Assumptions

ldfConstant: LDF is constant, depends on assumed value of t_d and m , ...

Data Definitions

$$B = \frac{k}{(ab)^{m-1}} (Eh^2)^m \text{LDF} e^J$$

...

html

sBR

$$B = \frac{k}{(ab)^{m-1}} (Eh^2)^m \text{LDF} e^J$$

GlassBR

Authors: Nikitha K and Spencer S

How to Run the Program: In your terminal command line, enter the same directory as this README file. Then enter the following line

```
make run RUNARGS=input.txt
```

Configuration Files: SDF.txt, TSD.txt must be in the same directory as the executable to run successfully

Versioning: Python Version 3.5.1

```
...
```

```
build:
```

```
run: build
```

```
python Control.py
```

```
...
```

```
build: GlassBR/Control.class
```

```
...
```

```
GlassBR/Control.class:
```

```
GlassBR/Control.java ...
```

```
javac GlassBR/Control.java
```

```
run: build
```

```
java GlassBR.Control $(RUNARGS)
```

```
...
```

Calculations.py

```
## \file Calculations.py  
# \author Nikitha Krithnan and W. Spencer Smith
```

Calculations.java

```
# \brief package GlassBR  
...  
## \file Calculations.java  
## \author Nikitha Krithnan and W. Spencer Smith  
# \para \brief Provides functions for calculating the outputs  
# \para  
# \return */  
def func...  
    out public static double func_B(InputParameters inParams, double J) throws IOException {  
    pri     PrintWriter outfile;  
    ...     outfile = new PrintWriter(new FileWriter(new File("log.txt"), true));  
    out     outfile.println("function func_B called with inputs: {}");  
           ...  
    ret     outfile.close();  
  
           return 2.86e-53 /Math.pow(inParams.a * inParams.b, 7.0 - 1.0) *  
                Math.pow(7.17e10 * Math.pow(inParams.h, 2.0), 7.0) * inParams.LDF  
                * Math.exp(J);  
    }
```

J_{tol} in SRS.pdf

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

J_{tol} in SRS.html

```
...
<th>Equation</th>
<td>
\[{J_{\text{tol}}}] = \ln \left( \ln \left( \frac{1}{1 - \{P_{\text{tol}}\}} \right) \frac{\left( \frac{a}{1000} \frac{b}{1000} \right)^{m-1} \{k \left( E \cdot 1000 \left( \frac{h}{1000} \right)^2 \right)^m \text{LDF}} \right) \right)
</td>
...
```

J_{tol} in Python

```
## \brief Calculates stress distribution factor (Function)
      based on Pbtol
# \param inParams structure holding the input values
# \return stress distribution factor (Function) based on
      Pbtol
def func_J_tol(inParams):
    outfile = open("log.txt", "a")
    print("function func_J_tol called with inputs: {",
          file=outfile)
    print("  inParams = ", end="", file=outfile)
    print("Instance of InputParameters object", file=
          outfile)
    print("  }", file=outfile)
    outfile.close()

    return math.log(math.log(1.0 / (1.0 - inParams.P_btoll)
        ) * ((inParams.a / 1000.0 * (inParams.b / 1000.0))
            ** (7.0 - 1.0) / (2.86e-53 * (7.17e10 * 1000.0 *
                (inParams.h / 1000.0) ** 2.0) ** 7.0 * inParams.
                LDF)))
```

J_{tol} in Java

```
/** \brief Calculates stress distribution factor (
    Function) based on Pbtol
    \param inParams structure holding the input values
    \return stress distribution factor (Function)
        based on Pbtol
*/
public static double func_J_tol(InputParameters
    inParams) throws IOException {
    PrintWriter outfile;
    outfile = new PrintWriter(new FileWriter(new File(
        "log.txt"), true));
    ...
    return Math.log(Math.log(1.0 / (1.0 - inParams.
        P_btol)) * (Math.pow(inParams.a / 1000.0 * (
        inParams.b / 1000.0), 7.0 - 1.0) / (2.86e-53 *
        Math.pow(7.17e10 * 1000.0 * Math.pow(inParams
        .h / 1000.0, 2.0), 7.0) * inParams.LDF)));
}
```

J_{tol} in Drasil (Haskell)

```
tolStrDisFacEq :: Expr
tolStrDisFacEq = ln (ln (recip_ (exactDbl 1 $- sy pbTol))
  'mulRe' (((sy plateLen $/ exactDbl 1000) 'mulRe' (sy
    plateWidth $/ exactDbl 1000))) $^ (sy sflawParamM $-
    exactDbl 1) $/
  (sy sflawParamK 'mulRe' ((sy modElas 'mulRe' exactDbl
    1000 'mulRe'
    square (sy minThick $/ exactDbl 1000)) $^ sy
    sflawParamM) 'mulRe' sy lDurFac)))
```

J_{tol} without Unit Conversion

```
tolStrDisFacEq :: Expr
tolStrDisFacEq = ln (ln (recip_ (exactDbl 1 $- sy pbTol))
  'mulRe' ((sy plateLen 'mulRe' sy plateWidth) $^ (sy
    sflawParamM $- exactDbl 1) $/
    (sy sflawParamK 'mulRe' ((sy modElas 'mulRe'
      square (sy minThick)) $^ sy sflawParamM) 'mulRe' sy
      lDurFac)))
```

Drasil Inputs:

- Program Name: GlassBR
- Authors: Nikitha K and Spencer S
- Symbols: tolerable load (\hat{q}_{tol}), Risk of failure (B), ...
- Assumptions: Load duration factor constant,
- Data definitions: relation for B , ...
- Design decisions:
 - Modularity (input module),
 - Implementation Type (Program),
 - Logging (Yes),
 - Input Structure (Bundled),
 - Constant Structure (Inlined),
 - Constant Rep (Constants),
 - Real Number Rep (Double),
 - ...

Drasil Source for software to predict whether a plate of glass will break

- Program Name: GlassBR
- Authors: Nikitha K and Spencer S
- Symbols: tolerable load (q_{tol}), Risk of failure (B), ...
- Assumptions: Load distrib. fact. constant,
- Data definitions: relation for B ,
- Design decisions:
 - Modularity (input module),
 - Implementation Type (Program),
 - Logging (Yes)
 - Input Structure (Bundled)
 - Constant Structure (mixed)
 - Constant Rep (Constants)
 - Real Number Rep (Double) ...

Generate

```

/glassbr
/Website/GlassBR_SRS.html
/Website/GlassBR_SRS.css
/SRS/bibfile.bib
/SRS/Makefile
/SRS/GlassBR_SRS.tex
/SRS/GlassBR_SRS.pdf
/src/python
/src/python/README.md
/src/python/InputParameters.py
/src/python/Calculations.py
/src/python/Makefile
/src/python/dochConfig
...
/src/java/GlassBR/Calculations.java
/src/java/Makefile
/src/java/README.md
...
/src/cpp/GlassBR
/src/cpp/ReadTable.cpp
/src/cpp/InputFormat.hpp
/src/cpp/Calculations.cpp
...
/src/swift/Calculations.swift
...
/src/csharp/Control.cs
...
    
```

Software Requirements Specification for GlassBR
Nikitha K and Spencer S

Table of Symbols

q_{tol}
 B

Introduction
... The software, herein called GlassBR, ...

Assumptions
LDF is constant, depends on assumed value of q_{tol} and m , ...

Data Definitions

$$B = \frac{k}{(ab)^{m-1}} (Eh^2)^m LDF e^J$$

$$B = \frac{k}{(ab)^{m-1}} (Eh^2)^m LDF e^J$$

GlassBR
Authors Nikitha K and Spencer S
How to Run the Program: In your terminal command line, enter the same directory as this README file. Then enter the following line
make run RUNARGS=input.txt
Configuration Files: SDF.txt, TSD.txt must be in the same directory as the executable to run successfully
Versioning: Python Version 3.5.1

```

...
build:
python Control.py
...
build: GlassBR/Control.class
...
GlassBR/Control.class:
GlassBR/Control.java ...
run: build
python Control.py
...
run: build
java GlassBR.Control $(RUNARGS)
...
    
```

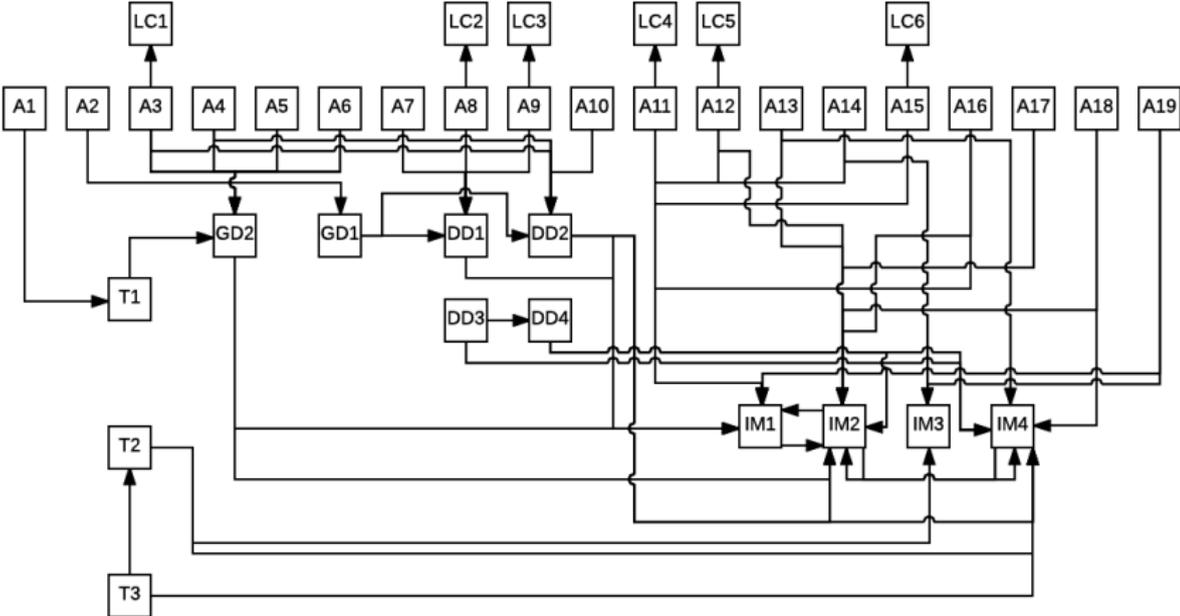
```

#!/usr/bin/env python
# \author Nikitha Krithnan and W. Spencer Smith
# \brief Provides functions for calculating the
...
# \brief Calculates risk of failure
# \param inParams structure holding the input values
# \param J stress distribution factor (Function)
# \return risk of failure
def func_B(inParams):
    outfile = open("log.txt", "a")
    print("function func_B called with inputs: ")
    ...
    outfile.close()
    return 2.86e-53 / ((inParams.a * inParams.b)
inParams.h ** 2.0) ** 7.0 * inParams.LDF * math
    
```

```

package GlassBR;
/** \file Calculations.java
 \author Nikitha Krithnan and W. Spencer Smith
 \brief Provides functions for calculating the outputs
 */
public static double func_B(InputParameters inParams, double J) throws IOException {
    PrintWriter outfile;
    outfile = new PrintWriter(new FileWriter(new File("log.txt"), true));
    outfile.println("function func_B called with inputs: {}");
    ...
    outfile.close();
    return 2.86e-53 / Math.pow(inParams.a * inParams.b, 7.0 - 1.0) *
    Math.pow(inParams.h, 2.0) * inParams.LDF
    * Math.exp(J);
}
    
```

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%
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 for verification of change

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.

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]

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

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

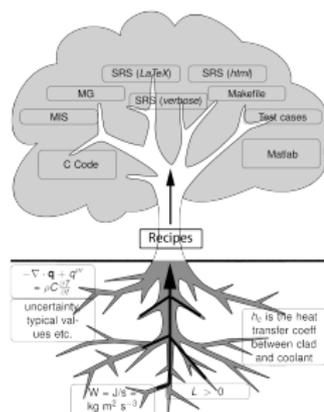
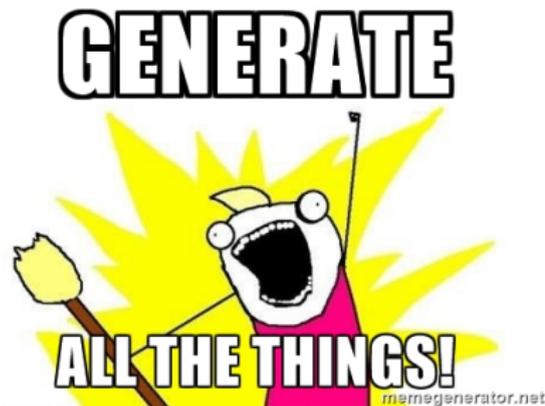
- Uncovered 27 issues with the previous documentation
 - ▶ Incompleteness (R_{gap})
 - ▶ Inconsistency(r, r_0, h_g)
 - ▶ Verifiability problems (R_1)
 - ▶ Lack of traceability (circuit analogy)
- Advantages of proposed approach
 - ▶ Abstract to concrete
 - ▶ Separation of concerns
 - ▶ Every equation,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

NO



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](#)



Drasil Links

- [Drasil on GitHub](#)
- [Design Language for Code Variabilities in Chapter 6 of Brook's thesis](#)
- [Drasil Generated Examples](#)
- [Drasil Haddock Documentation](#)
- [Package Dependency Graph \(at the bottom of the page\)](#)

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

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.



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.



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.

References III



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.



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.

References IV



Patrick J. Roache.

Verification and Validation in Computational Science and Engineering.

Hermosa Publishers, Albuquerque, New Mexico, 1998.



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). In conjunction with SC16: The International Conference for High Performance Computing, Networking, Storage and Analysis, November 2016.

8 pp.

References V



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.



W. Spencer Smith, Mojdeh Sayari Nejad, and Alan Wassying.

Assurance cases for scientific computing software (poster).

In *ICSE 2018 Proceedings of the 40th International Conference on Software Engineering*, May 2018.

2 pp.



W. Spencer Smith, Mojdeh Sayari Nejad, and Alan Wassying.

Raising the bar: Assurance cases for scientific computing software.

Computing in Science and Engineering, 23(1):47–57, February 2020.

References VI



Muhammad Usman, Muhammad Zohaib Iqbal, and Muhammad Uzair Khan.

A product-line model-driven engineering approach for generating feature-based mobile applications.

Journal of Systems and Software, 123:1–32, 01 2017.



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.