CAS 741 (Development of Scientific Computing Software)

Winter 2024

Artifact Generation

Dr. Spencer Smith

Faculty of Engineering, McMaster University

March 22, 2024

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

Administrative Details: Presentations

Unit VnV/Implement Week 12 Week of Apr 3

- Specific schedule depends on final class registration
- Informal presentations with the goal of improving everyone's written deliverables
- Domain experts and secondary reviewers (and others) will ask questions

Presentation Schedule

Presentation Sched Cont'd

- Implementation Present (15 min each)
  - **Mar 26: Reyhaneh, Waqar, Al, Tanya, Atiyeh**
  - **Apr 2: Nada, Phil, Xinyu, Fasil, Yi-Leng**
  - Apr 5: Gaofeng, Morteza, Valerie, Hunter, Ali
  - Apr 9: Cynthia, Adrian, Yiding, Kim Ying

Presentation Schedule

- 3 presentations each
  - SRS everyone
  - VnV and POC subset of class
  - Design subset of class
  - Implementation everyone
- If you will miss a presentation, please trade with someone
- Implementation presentation could be used to run a code review, or code walkthrough

Questions?

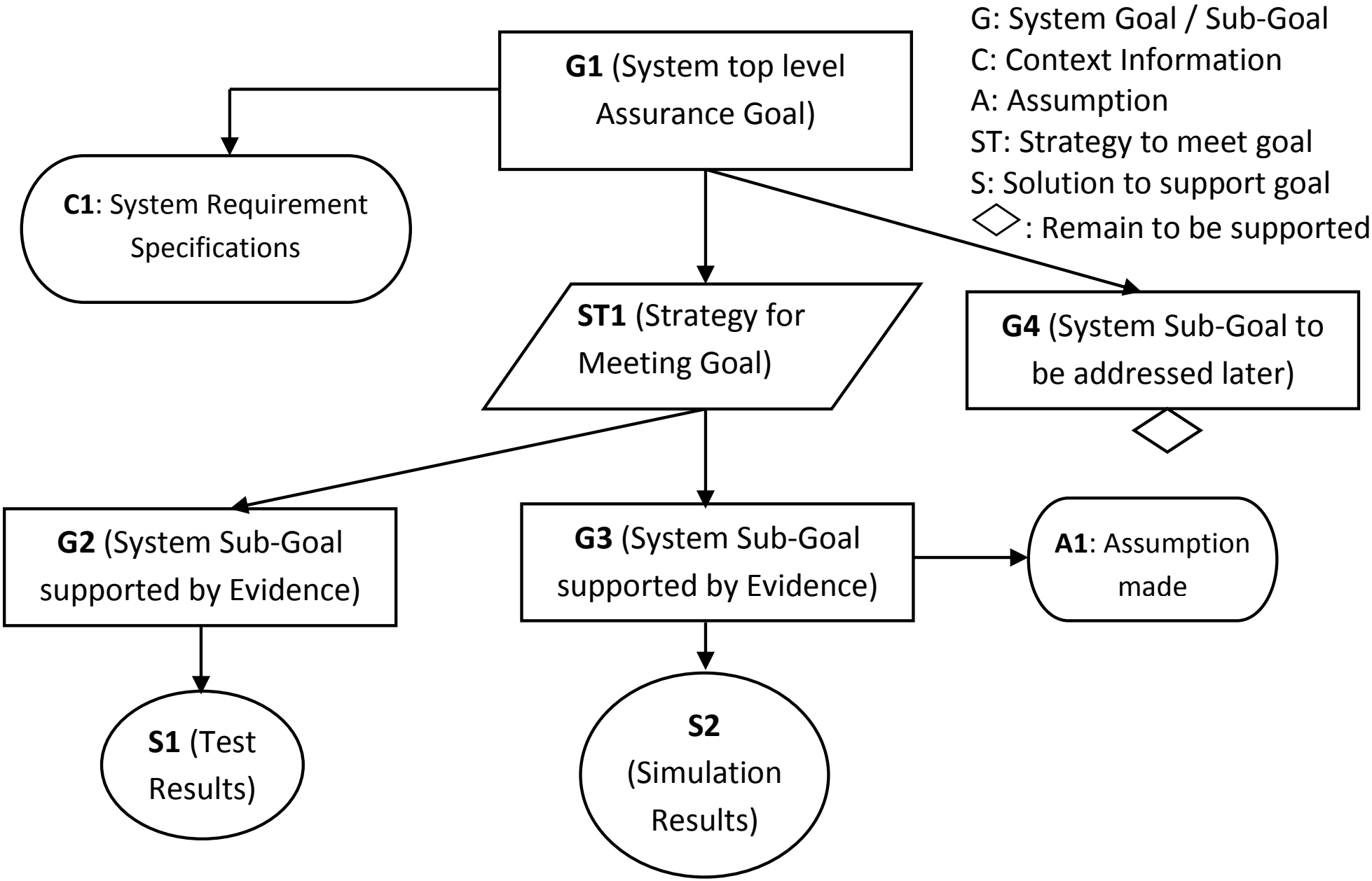
- Questions on administrative details?
- Questions on final documentation?
- Questions on reflection document?
- Questions on final implementation?
- Questions on VnV report?
- Other questions?

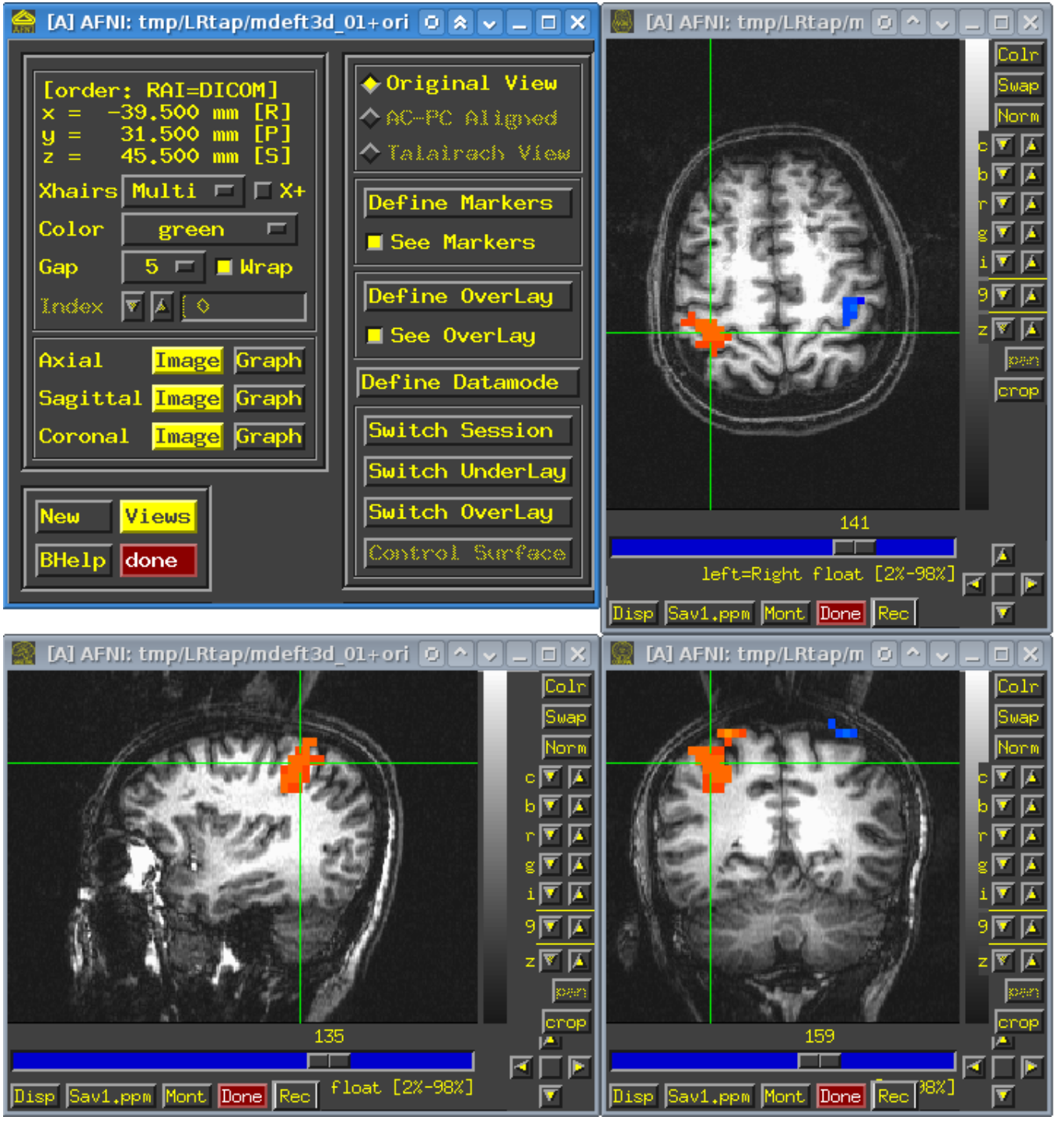
Assurance Cases in Scientific Computing [14, 13]

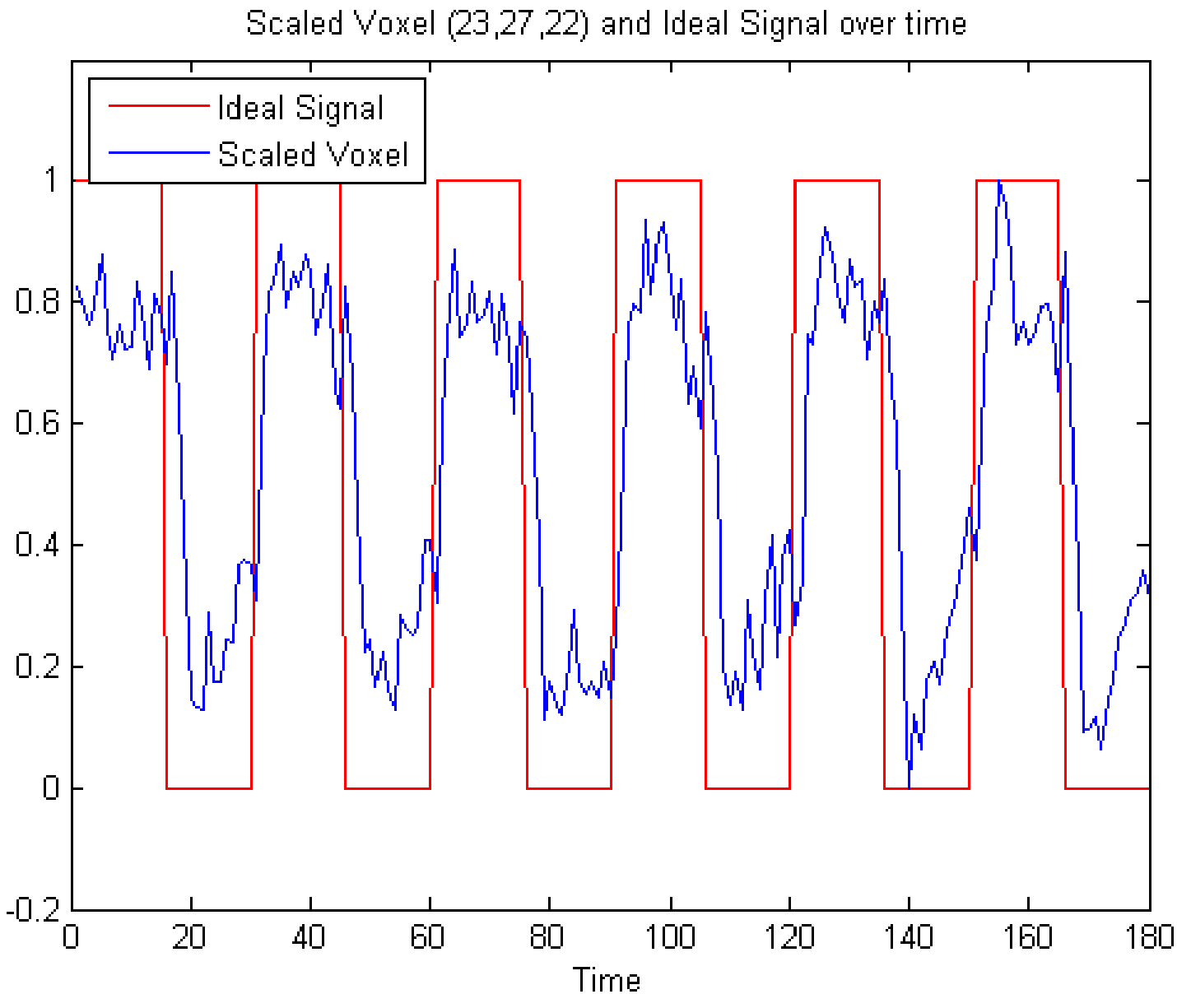
- Assurance cases
  - Organized and explicit argument for correctness
  - Successfully used for safety critical systems
- Advantages for SC
  - Engaging domain experts
  - Producing necessary and relevant documentation
  - Evidence that can be verified/replicated by a third party
- Example of 3dfim+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

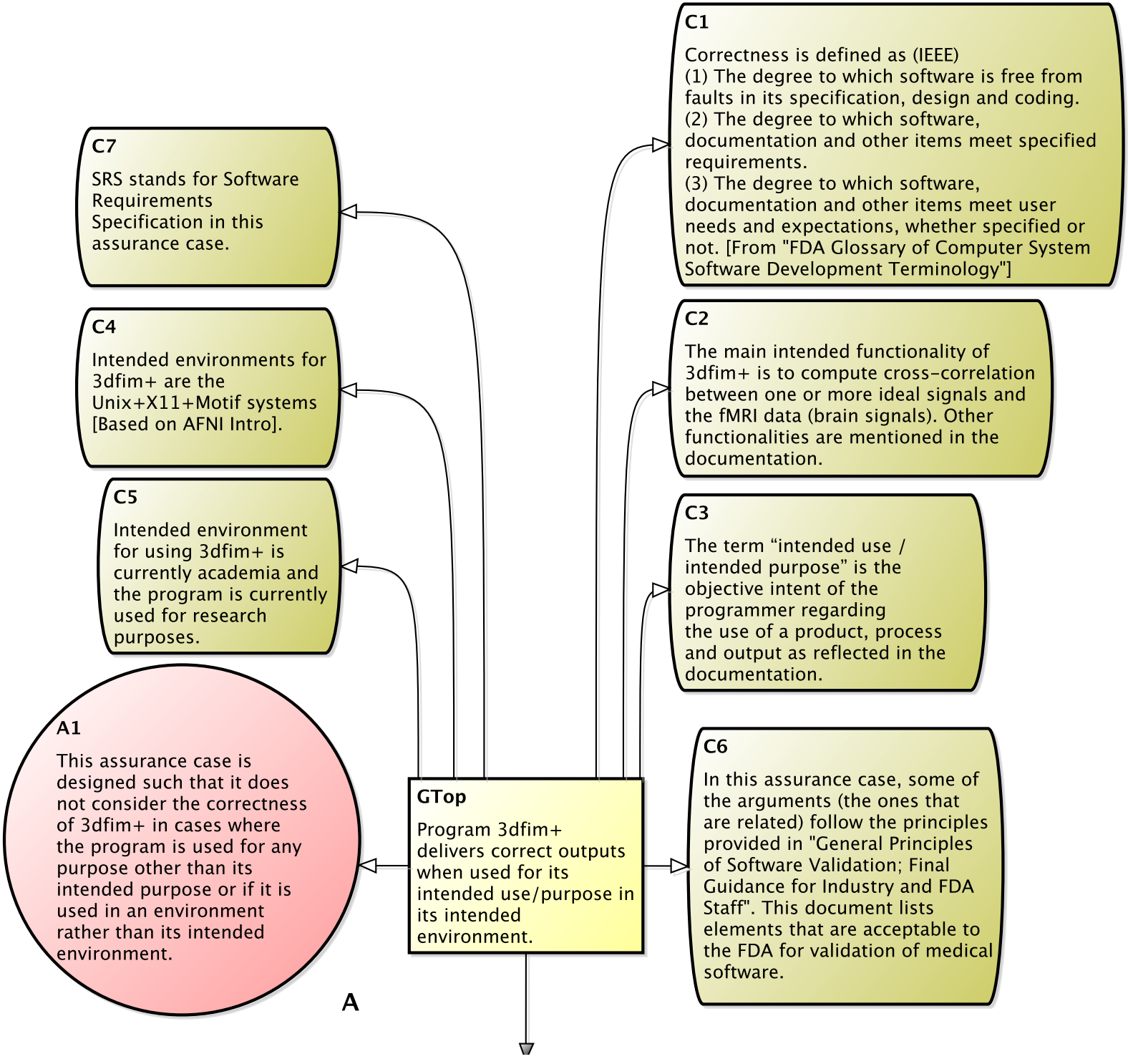
Assurance Cases in SC Motivation

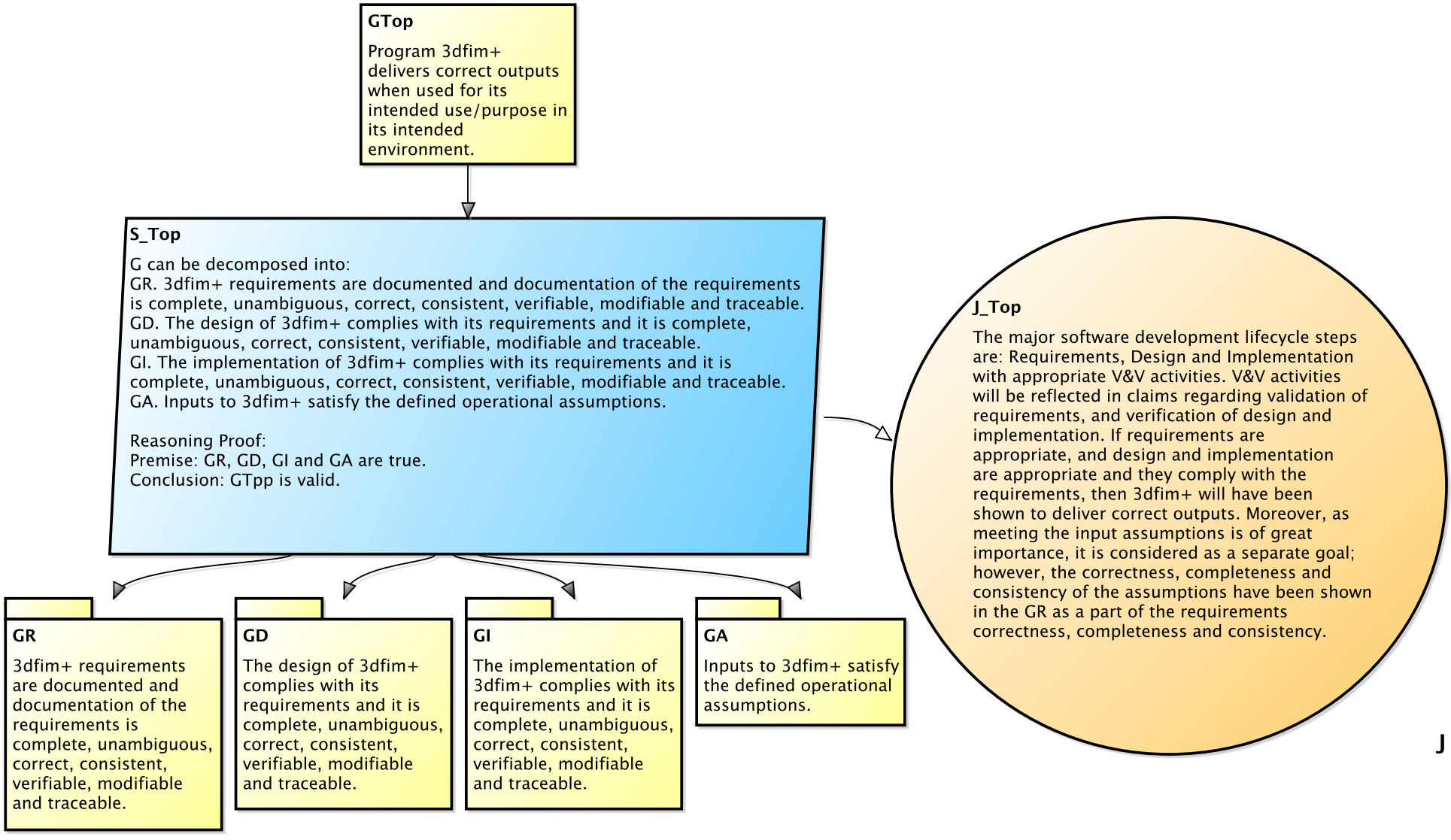
- Do we put too much trust in the quality of SCS?
- Are enough checks and balances in place, especially for safety related software?
- Problems with imposing external requirements for certification
  - External body does not have expertise
  - SCS developers dislike documentation
- Solution – Assurance 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

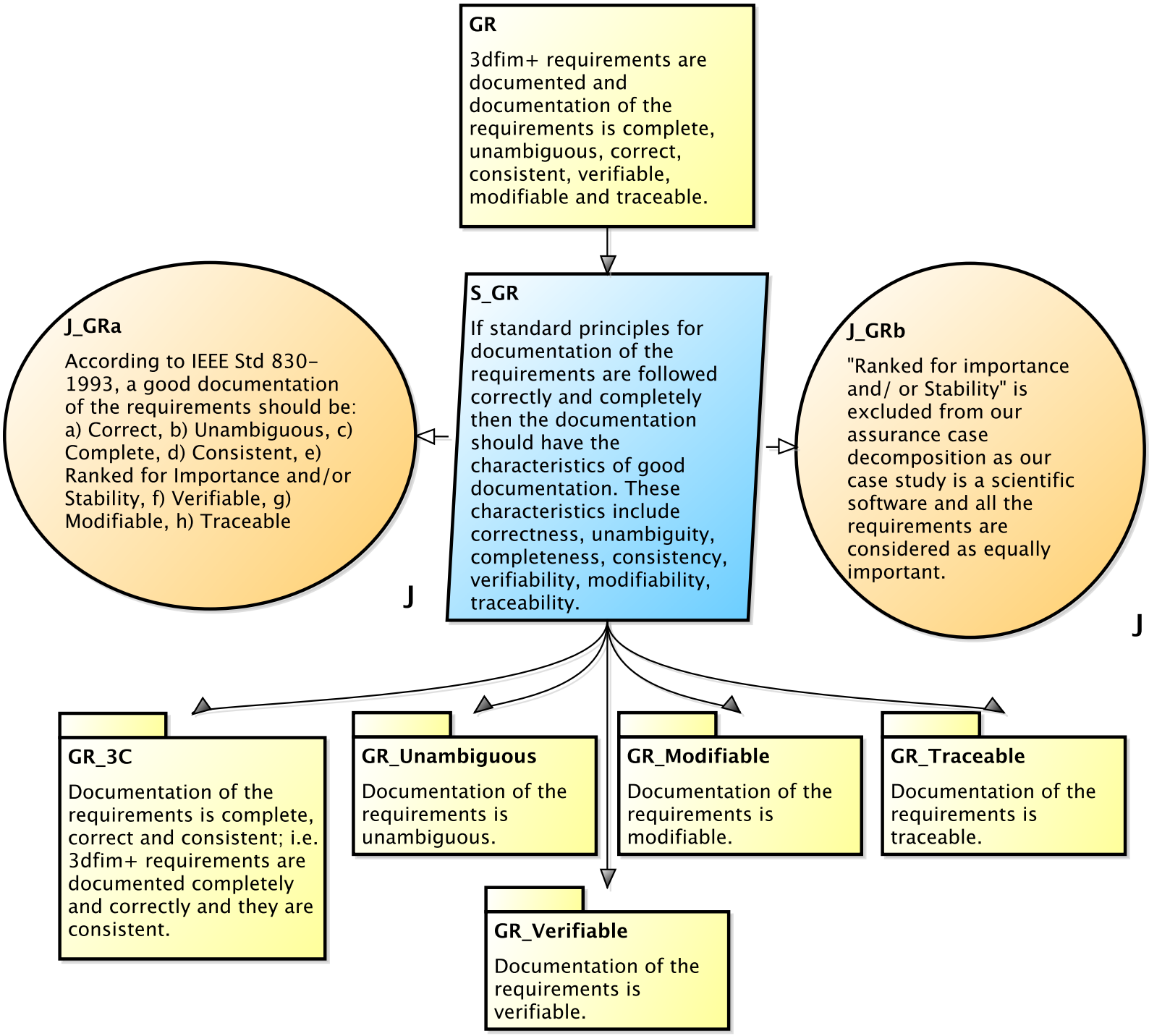

G: System Goal / Sub-Goal
C: Context Information
A: Assumption
ST: Strategy to meet goal
S: Solution to support goal
: Remain to be supported
G1 (System top level Assurance Goal)
C1: System Requirement Specifications
ST1 (Strategy for Meeting Goal)
G4 (System Sub-Goal to be addressed later)
G2 (System Sub-Goal supported by Evidence)
G3 (System Sub-Goal supported by Evidence)
A1: Assumption made
S1 (Test Results)
S2 (Simulation Results)

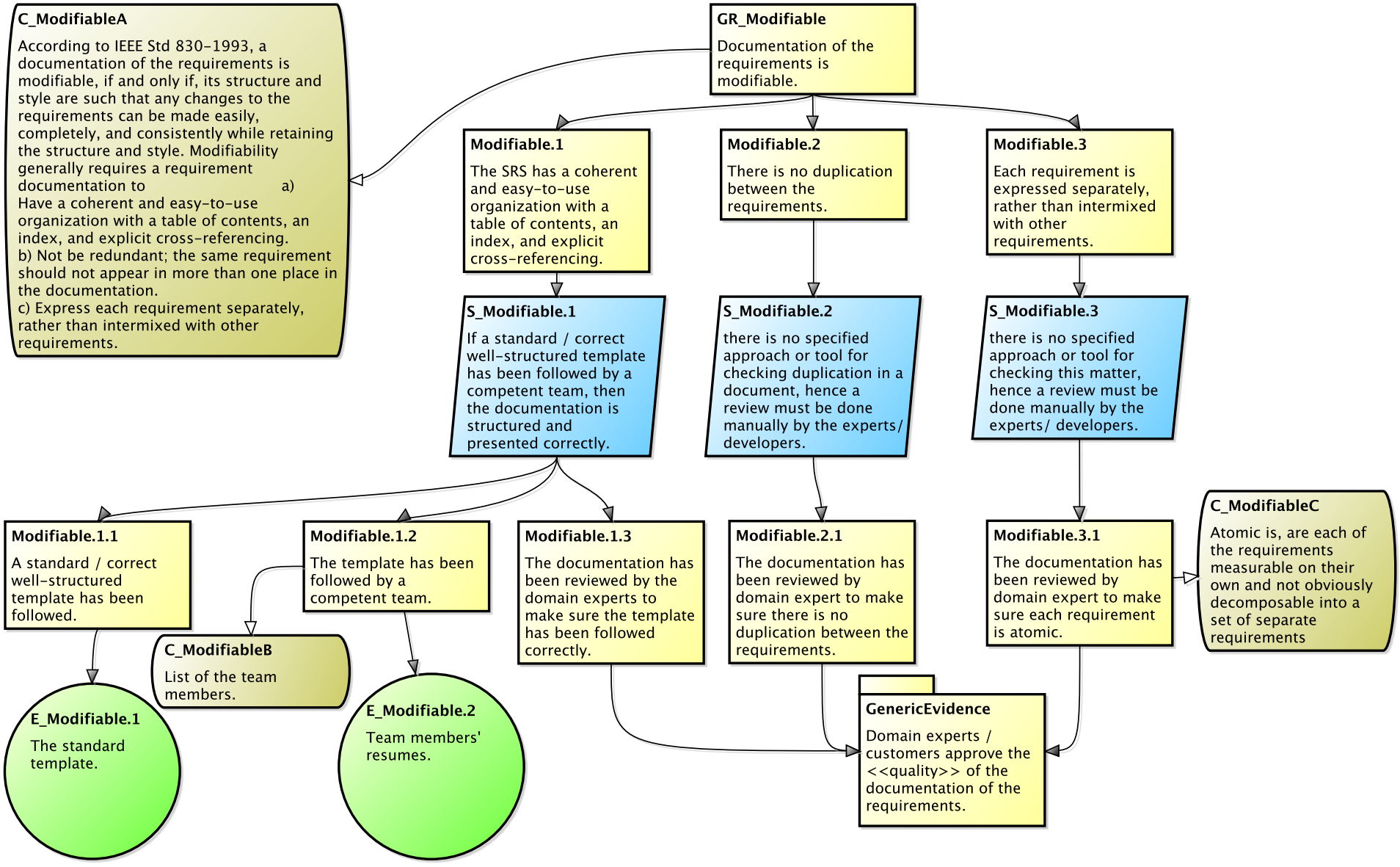

[A] AFNI: tmp/LRtap/mdeft3d_01+ori
[order: RAI=DICOM]
x = -39.500 mm [R]
y = 31.500 mm [P]
z = 45.500 mm [S]
Xhairs Multi X+
Color green
Gap 5 Wrap
Index 0
Axial Image Graph
Sagittal Image Graph
Coronal Image Graph
New Views
BHelp done
Original View
AC-PC Aligned
Talairach View
Define Markers
See Markers
Define OverLay
See OverLay
Define Datamode
Switch Session
Switch UnderLay
Switch OverLay
Control Surface
[A] AFNI: tmp/LRtap/m
Colr
Swap
Norm
pan
crop
141
left=Right float [2%-98%]
Disp Sav1.ppm Mont Done Rec
[A] AFNI: tmp/LRtap/mdeft3d_01+ori
135
float [2%-98%]
Disp Sav1.ppm Mont Done Rec
[A] AFNI: tmp/LRtap/m
159
Disp Sav1.ppm Mont Done Rec

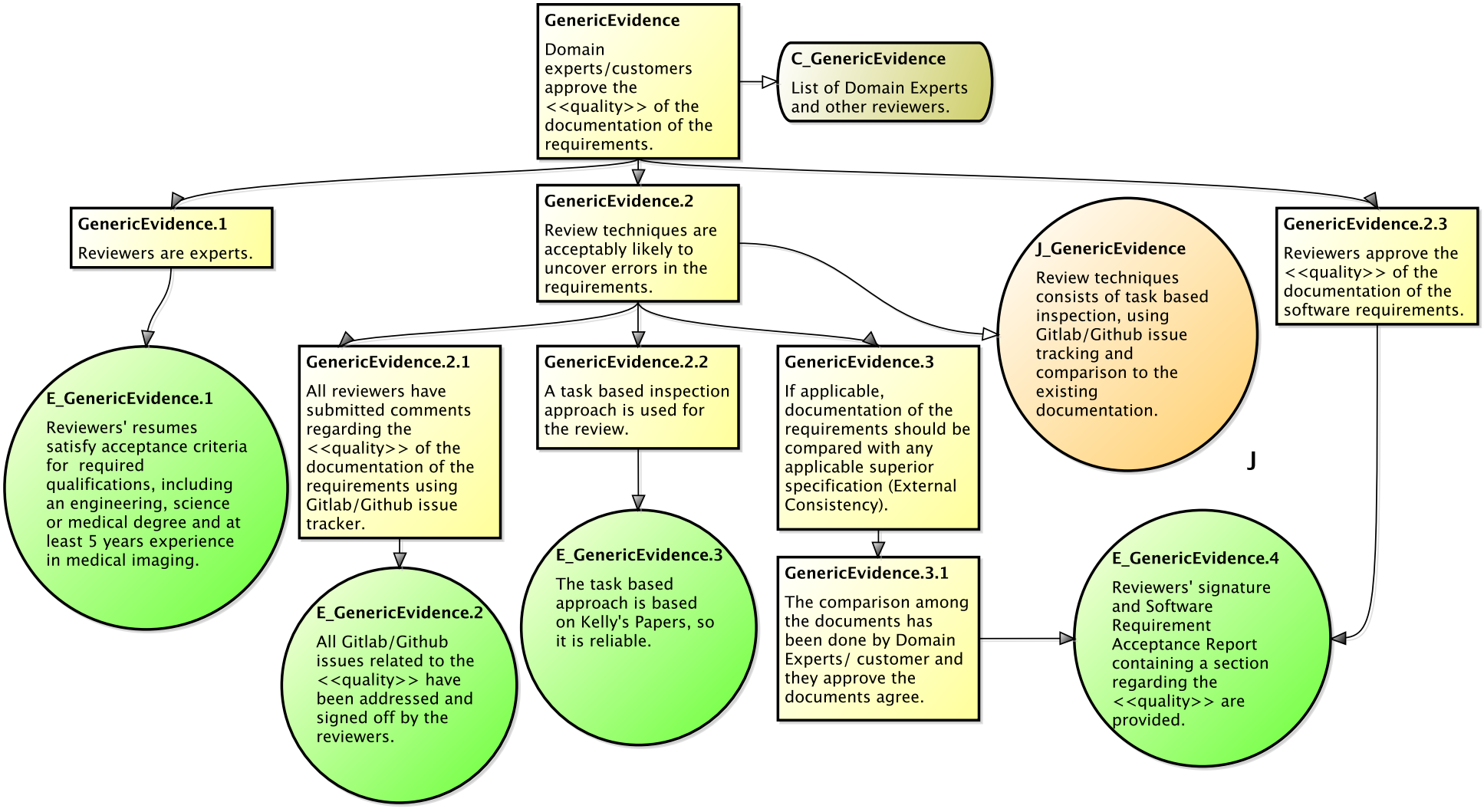

Scaled Voxel (23,27,22) and Ideal Signal over time
Ideal Signal
Scaled Voxel
1
0.8
0.6
0.4
0.2
0
-0.2
0
20
40
60
80
100
120
140
160
180
Time

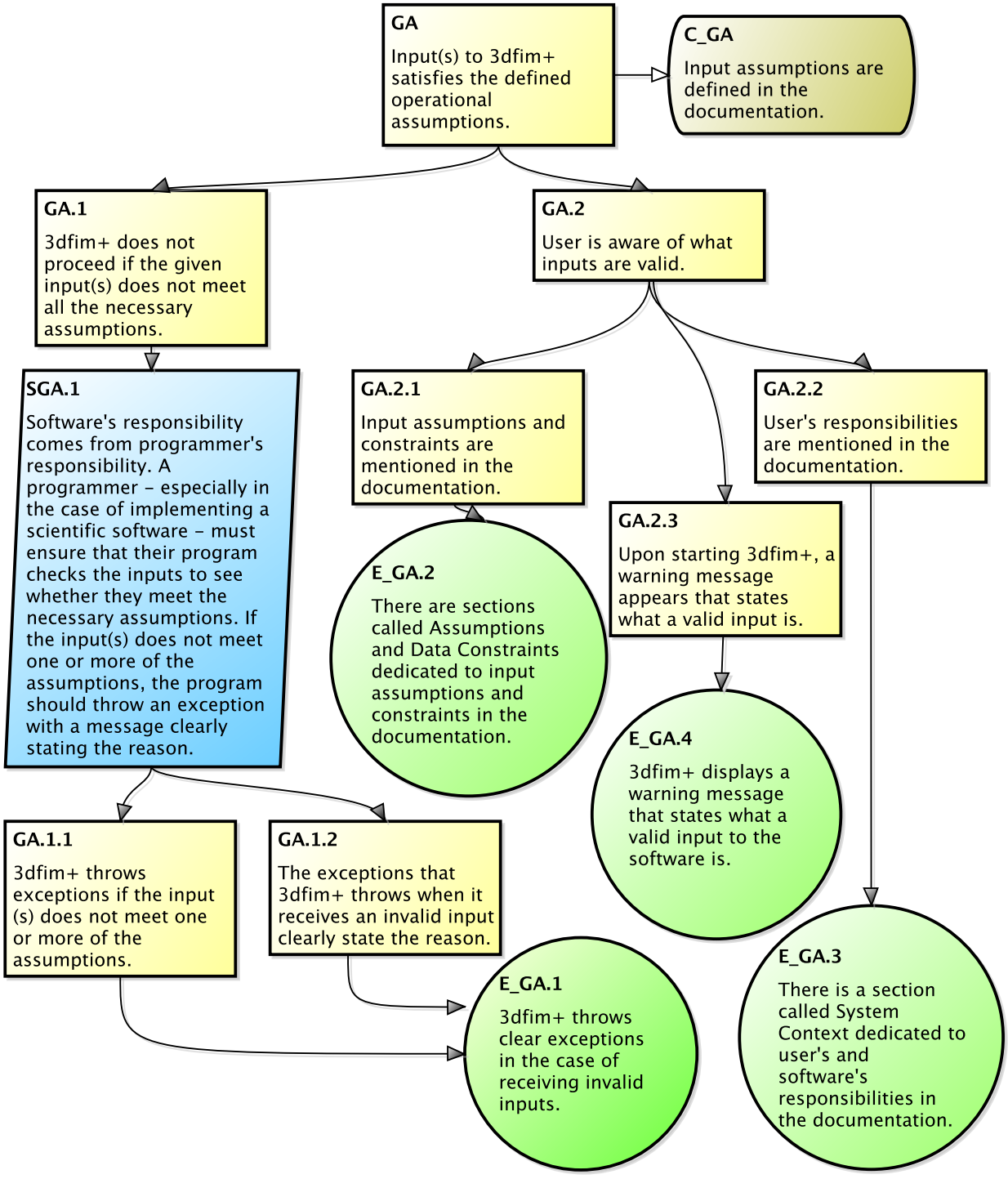

C1
Correctness is defined as (IEEE)
(1) The degree to which software is free from faults in its specification, design and coding.
(2) The degree to which software, documentation and other items meet specified requirements.
(3) The degree to which software, documentation and other items meet user needs and expectations, whether specified or not. [From "FDA Glossary of Computer System Software Development Terminology"]
C7
SRS stands for Software Requirements Specification in this assurance case.
C2
The main intended functionality of 3dfim+ is to compute cross-correlation between one or more ideal signals and the fMRI data (brain signals). Other functionalities are mentioned in the documentation.
C4
Intended environments for 3dfim+ are the Unix+X11+Motif systems [Based on AFNI Intro].
C5
Intended environment for using 3dfim+ is currently academia and the program is currently used for research purposes.
C3
The term “intended use / intended purpose” is the objective intent of the programmer regarding the use of a product, process and output as reflected in the documentation.
A1
This assurance case is designed such that it does not consider the correctness of 3dfim+ in cases where the program is used for any purpose other than its intended purpose or if it is used in an environment rather than its intended environment.
GTop
Program 3dfim+ delivers correct outputs when used for its intended use/purpose in its intended environment.
C6
In this assurance case, some of the arguments (the ones that are related) follow the principles provided in "General Principles of Software Validation; Final Guidance for Industry and FDA Staff". This document lists elements that are acceptable to the FDA for validation of medical software.
A


GTop
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: GTpp is valid.
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.
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


GR
3dfim+ requirements are documented and documentation of the requirements is complete, unambiguous, correct, consistent, verifiable, modifiable and traceable.
J_GRa
According to IEEE Std 830-1993, a good documentation of the requirements should be: a) Correct, b) Unambiguous, c) Complete, d) Consistent, e) Ranked for Importance and/or Stability, f) Verifiable, g) Modifiable, h) Traceable
S_GR
If standard principles for documentation of the requirements are followed correctly and completely then the documentation should have the characteristics of good documentation. These characteristics include correctness, unambiguity, completeness, consistency, verifiability, modifiability, traceability.
J_GRb
"Ranked for importance and/ or Stability" is excluded from our assurance case decomposition as our case study is a scientific software and all the requirements are considered as equally important.
J
J
GR_3C
Documentation of the requirements is complete, correct and consistent; i.e. 3dfim+ requirements are documented completely and correctly and they are consistent.
GR_Unambiguous
Documentation of the requirements is unambiguous.
GR_Modifiable
Documentation of the requirements is modifiable.
GR_Traceable
Documentation of the requirements is traceable.
GR_Verifiable
Documentation of the requirements is verifiable.


C_ModifiableA
According to IEEE Std 830-1993, a documentation of the requirements is modifiable, if and only if, its structure and style are such that any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style. Modifiability generally requires a requirement documentation to a) Have a coherent and easy-to-use organization with a table of contents, an index, and explicit cross-referencing. b) Not be redundant; the same requirement should not appear in more than one place in the documentation. c) Express each requirement separately, rather than intermixed with other requirements.
GR_Modifiable
Documentation of the requirements is modifiable.
Modifiable.1
The SRS has a coherent and easy-to-use organization with a table of contents, an index, and explicit cross-referencing.
Modifiable.2
There is no duplication between the requirements.
Modifiable.3
Each requirement is expressed separately, rather than intermixed with other requirements.
S_Modifiable.1
If a standard / correct well-structured template has been followed by a competent team, then the documentation is structured and presented correctly.
S_Modifiable.2
there is no specified approach or tool for checking duplication in a document, hence a review must be done manually by the experts/ developers.
S_Modifiable.3
there is no specified approach or tool for checking this matter, hence a review must be done manually by the experts/ developers.
Modifiable.1.1
A standard / correct well-structured template has been followed.
Modifiable.1.2
The template has been followed by a competent team.
Modifiable.1.3
The documentation has been reviewed by the domain experts to make sure the template has been followed correctly.
Modifiable.2.1
The documentation has been reviewed by domain expert to make sure there is no duplication between the requirements.
Modifiable.3.1
The documentation has been reviewed by domain expert to make sure each requirement is atomic.
C_ModifiableC
Atomic is, are each of the requirements measurable on their own and not obviously decomposable into a set of separate requirements
C_ModifiableB
List of the team members.
E_Modifiable.1
The standard template.
E_Modifiable.2
Team members' resumes.
GenericEvidence
Domain experts / customers approve the <<quality>> of the documentation of the requirements.


GenericEvidence
Domain experts/customers approve the <<quality>> of the documentation of the requirements.
C_GenericEvidence
List of Domain Experts and other reviewers.
GenericEvidence.1
Reviewers are experts.
GenericEvidence.2
Review techniques are acceptably likely to uncover errors in the requirements.
J_GenericEvidence
Review techniques consists of task based inspection, using Gitlab/Github issue tracking and comparison to the existing documentation.
GenericEvidence.2.3
Reviewers approve the <<quality>> of the documentation of the software requirements.
E_GenericEvidence.1
Reviewers' resumes satisfy acceptance criteria for required qualifications, including an engineering, science or medical degree and at least 5 years experience in medical imaging.
GenericEvidence.2.1
All reviewers have submitted comments regarding the <<quality>> of the documentation of the requirements using Gitlab/Github issue tracker.
GenericEvidence.2.2
A task based inspection approach is used for the review.
GenericEvidence.3
If applicable, documentation of the requirements should be compared with any applicable superior specification (External Consistency).
J
E_GenericEvidence.2
All Gitlab/Github issues related to the <<quality>> have been addressed and signed off by the reviewers.
E_GenericEvidence.3
The task based approach is based on Kelly's Papers, so it is reliable.
GenericEvidence.3.1
The comparison among the documents has been done by Domain Experts/ customer and they approve the documents agree.
E_GenericEvidence.4
Reviewers' signature and Software Requirement Acceptance Report containing a section regarding the <<quality>> are provided.


GA
Input(s) to 3dfim+ satisfies the defined operational assumptions.
C_GA
Input assumptions are defined in the documentation.
GA.1
3dfim+ does not proceed if the given input(s) does not meet all the necessary assumptions.
GA.2
User is aware of what inputs are valid.
SGA.1
Software's responsibility comes from programmer's responsibility. A programmer – especially in the case of implementing a scientific software – must ensure that their program checks the inputs to see whether they meet the necessary assumptions. If the input(s) does not meet one or more of the assumptions, the program should throw an exception with a message clearly stating the reason.
GA.2.1
Input assumptions and constraints are mentioned in the documentation.
GA.2.2
User's responsibilities are mentioned in the documentation.
GA.2.3
Upon starting 3dfim+, a warning message appears that states what a valid input is.
E_GA.2
There are sections called Assumptions and Data Constraints dedicated to input assumptions and constraints in the documentation.
E_GA.4
3dfim+ displays a warning message that states what a valid input to the software is.
GA.1.1
3dfim+ throws exceptions if the input (s) does not meet one or more of the assumptions.
GA.1.2
The exceptions that 3dfim+ throws when it receives an invalid input clearly state the reason.
E_GA.1
3dfim+ throws clear exceptions in the case of receiving invalid inputs.
E_GA.3
There is a section called System Context dedicated to user's and software's responsibilities in the documentation.

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

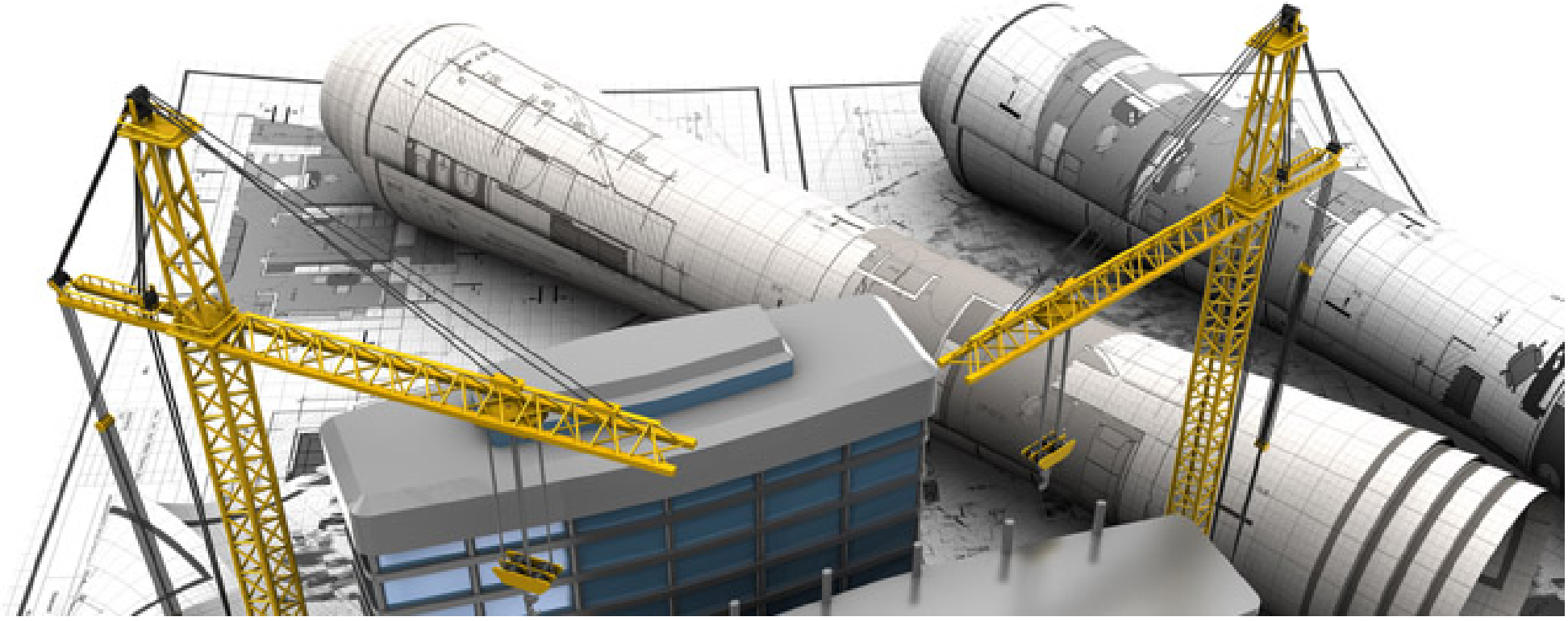
Concluding Remarks

- 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 Lines in User Manual

PRODUCT SPECIFICATIONS

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

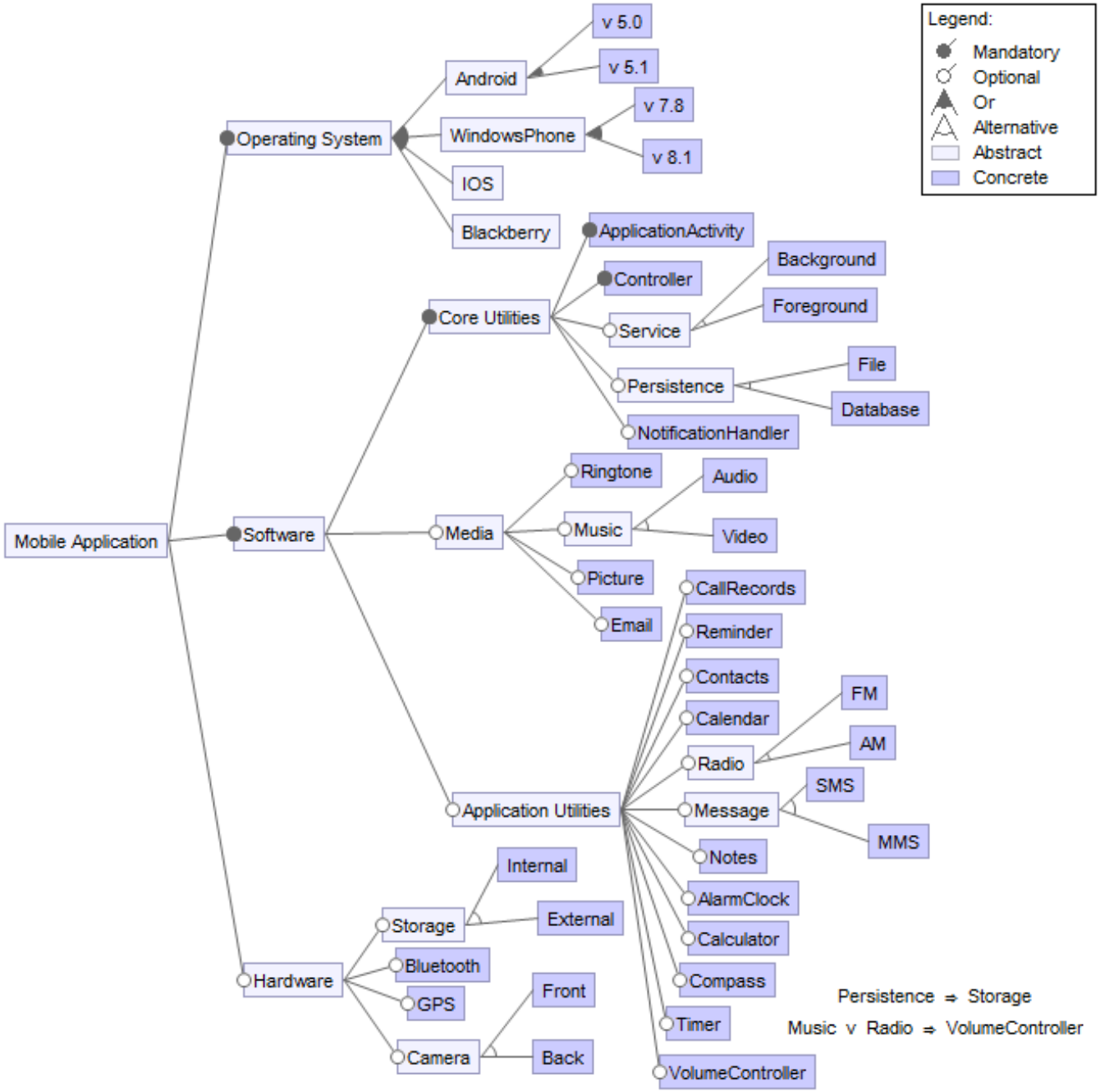
Electrical requirements: 115 V, 60 Hz

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)

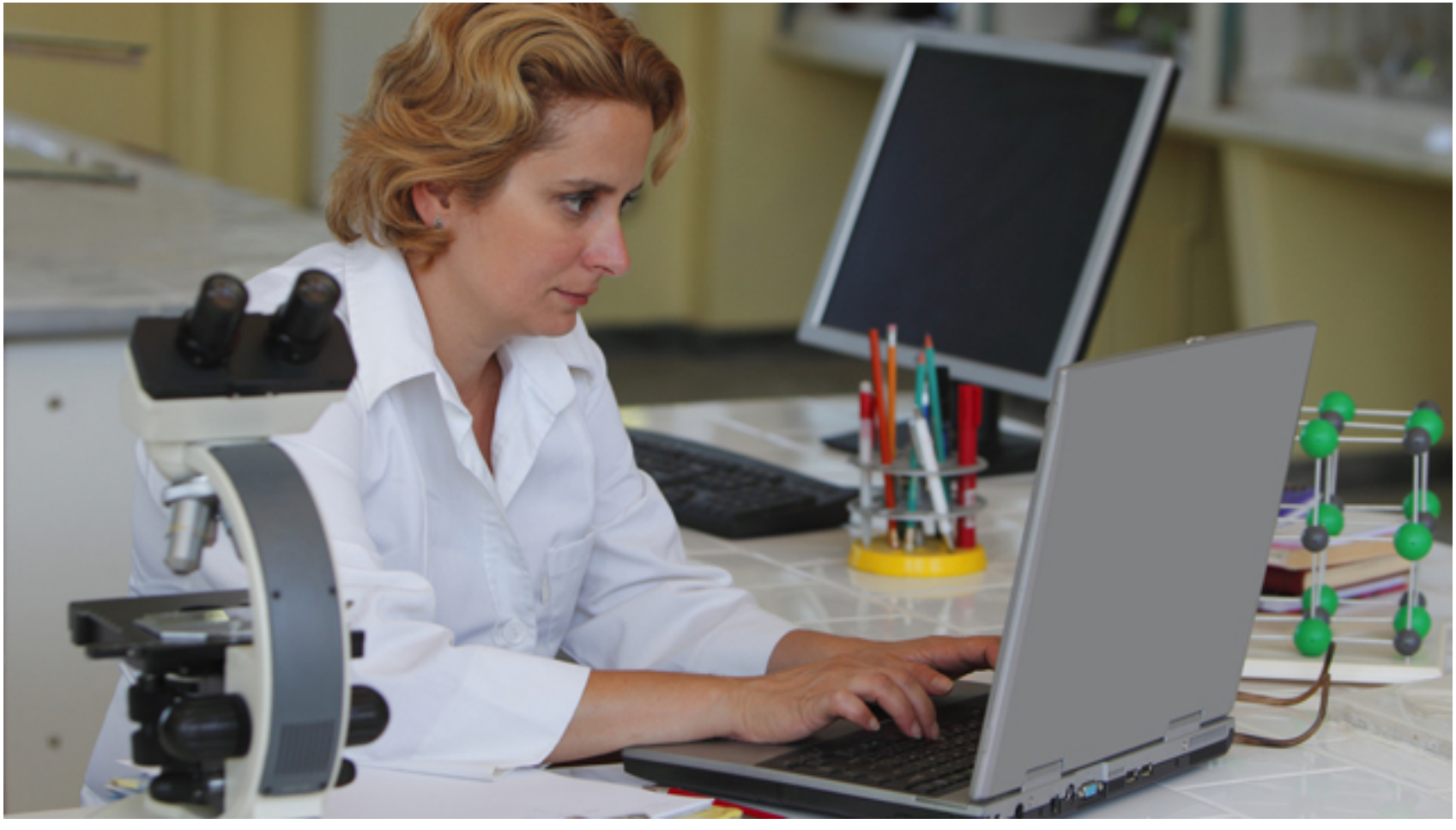
Model	LFCS27596*
Description	Standard-depth, Door-in-Door French door refrigerator, bottom freezer
Net weight	284 lb (129 kg)

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

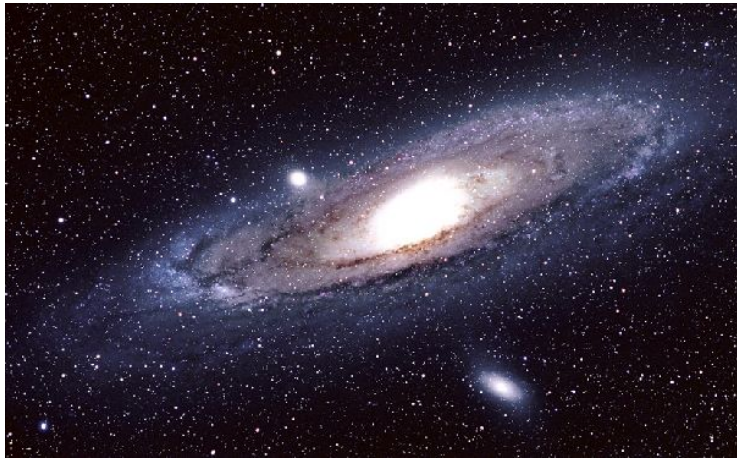

Legend:
Mandatory
Optional
Or
Alternative
Abstract
Concrete
Mobile Application
Software
Hardware
Operating System
Android
v 5.0
v 5.1
WindowsPhone
v 7.8
v 8.1
IOS
Blackberry
Core Utilities
ApplicationActivity
Controller
Service
Background
Foreground
Persistence
File
Database
NotificationHandler
Media
Ringtone
Music
Audio
Video
Picture
Email
Application Utilities
CallRecords
Reminder
Contacts
Calendar
Radio
FM
AM
Message
SMS
MMS
Notes
AlarmClock
Calculator
Compass
Timer
VolumeController
Storage
Internal
External
Bluetooth
GPS
Camera
Front
Back
Persistence ⇒ Storage
Music ∨ Radio ⇒ VolumeController

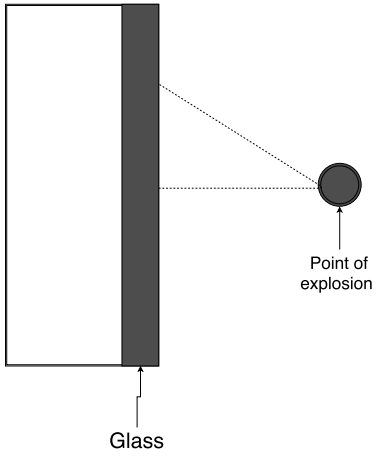
[15]

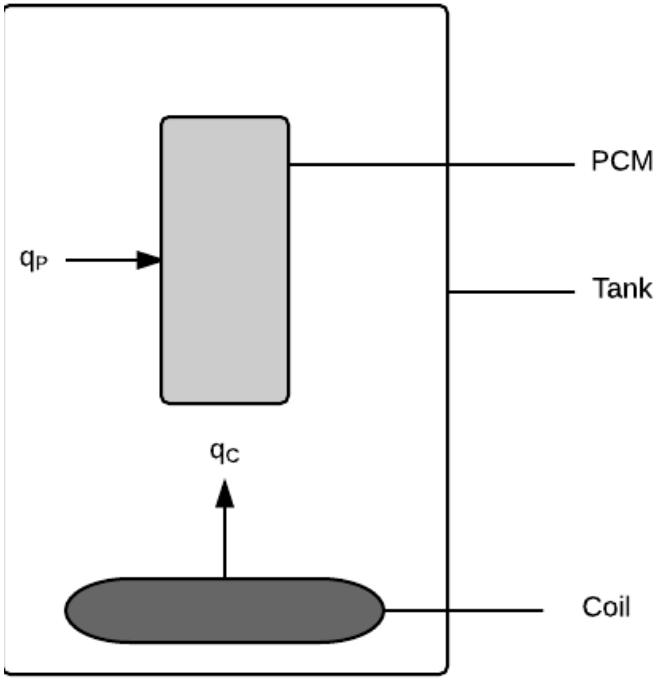
Scope: End User Developers

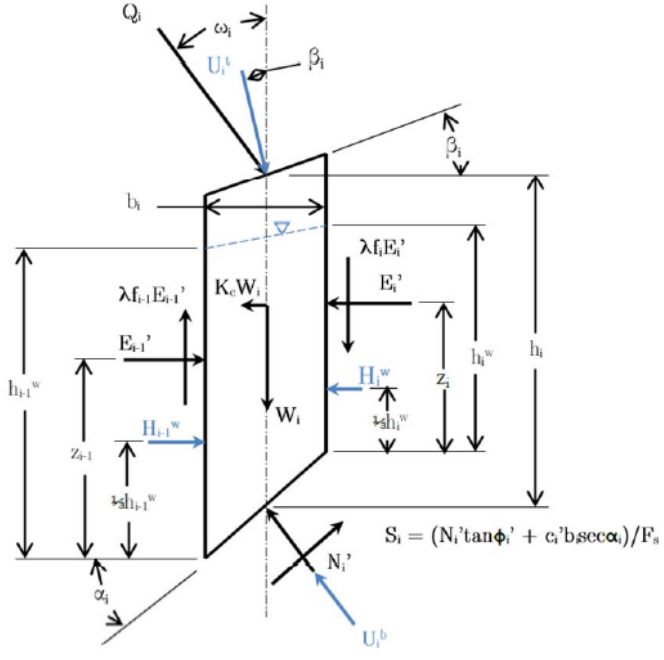


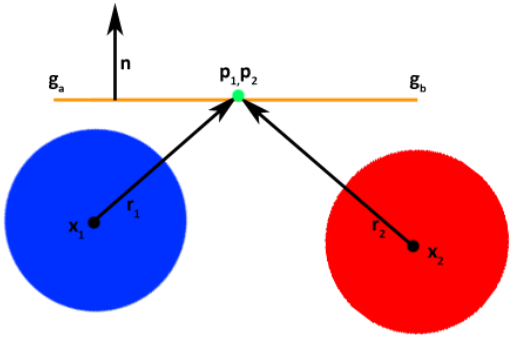
Scope: Physical Science

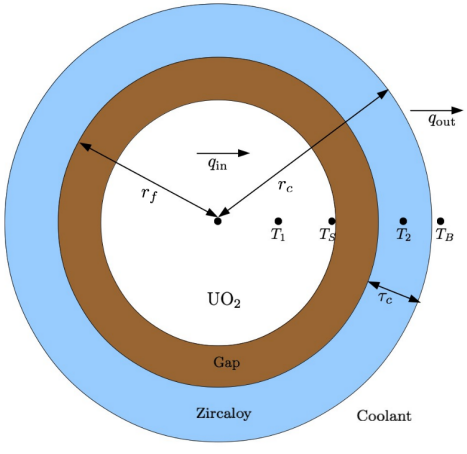






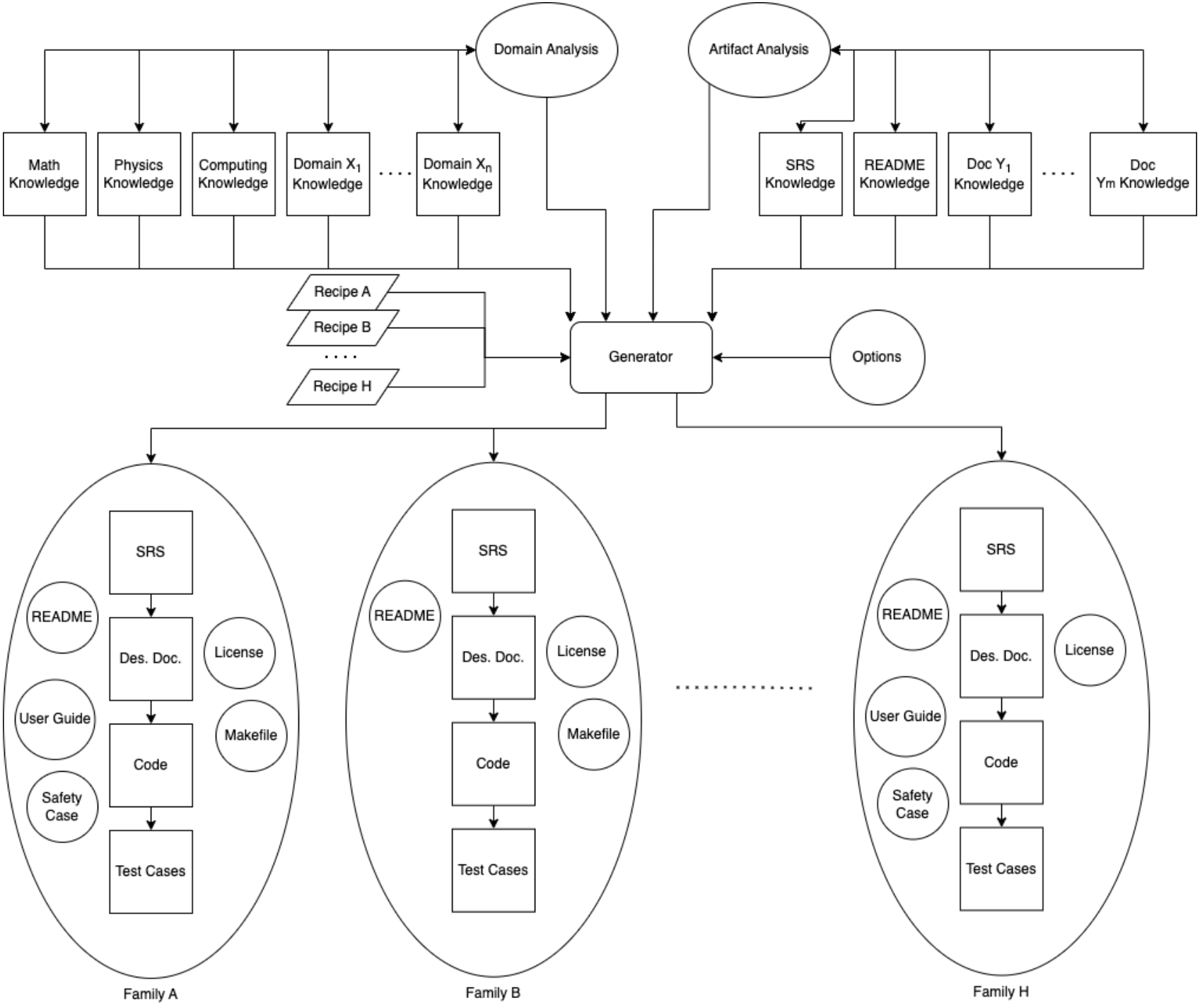






Build on Success of MDSE

- 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.


Domain Analysis
Artifact Analysis
Math Knowledge
Physics Knowledge
Computing Knowledge
Domain X_1 Knowledge
Domain X_n Knowledge
SRS Knowledge
README Knowledge
Doc Y_1 Knowledge
Doc Y_m Knowledge
Recipe A
Recipe B
Recipe H
Generator
Options
SRS
README
Des. Doc.
License
User Guide
Makefile
Code
Safety Case
Test Cases
Family A
Family B
Family H

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

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

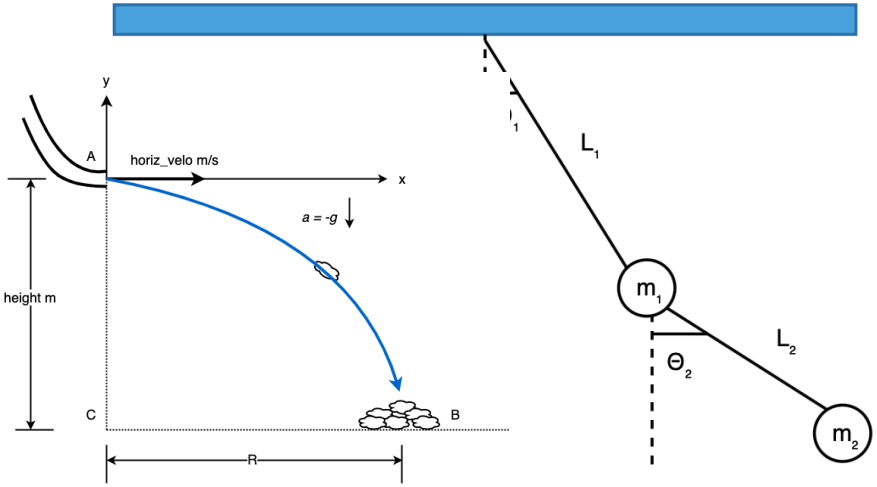
$$m\frac{d\mathbf{v}}{dt} = m\mathbf{g} - c\mathbf{v}$$

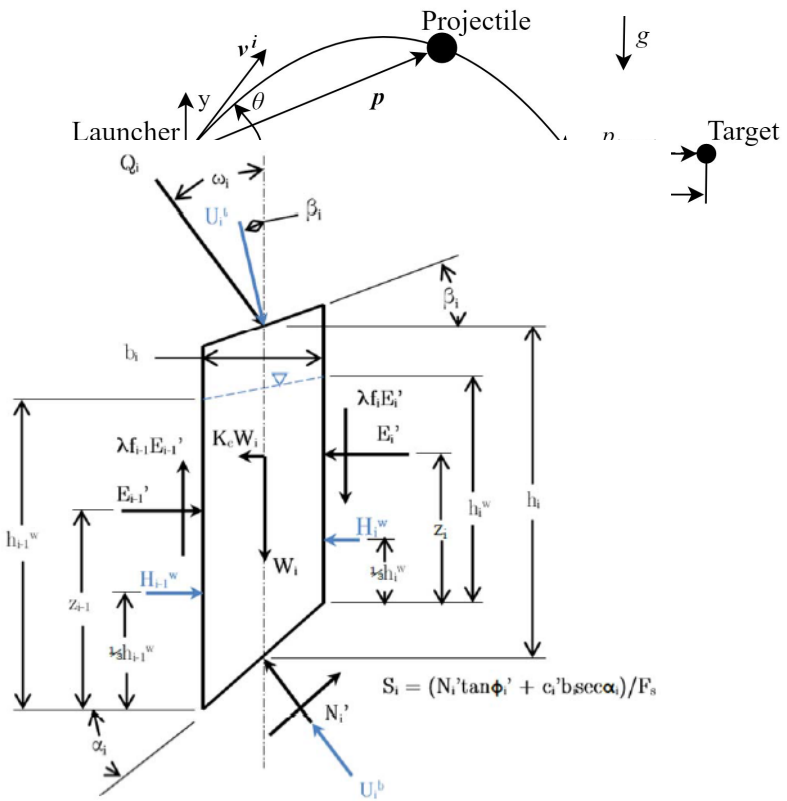
$$g = 9.8m/s^2 \text{ or } g = 32.2ft/s^2$$

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

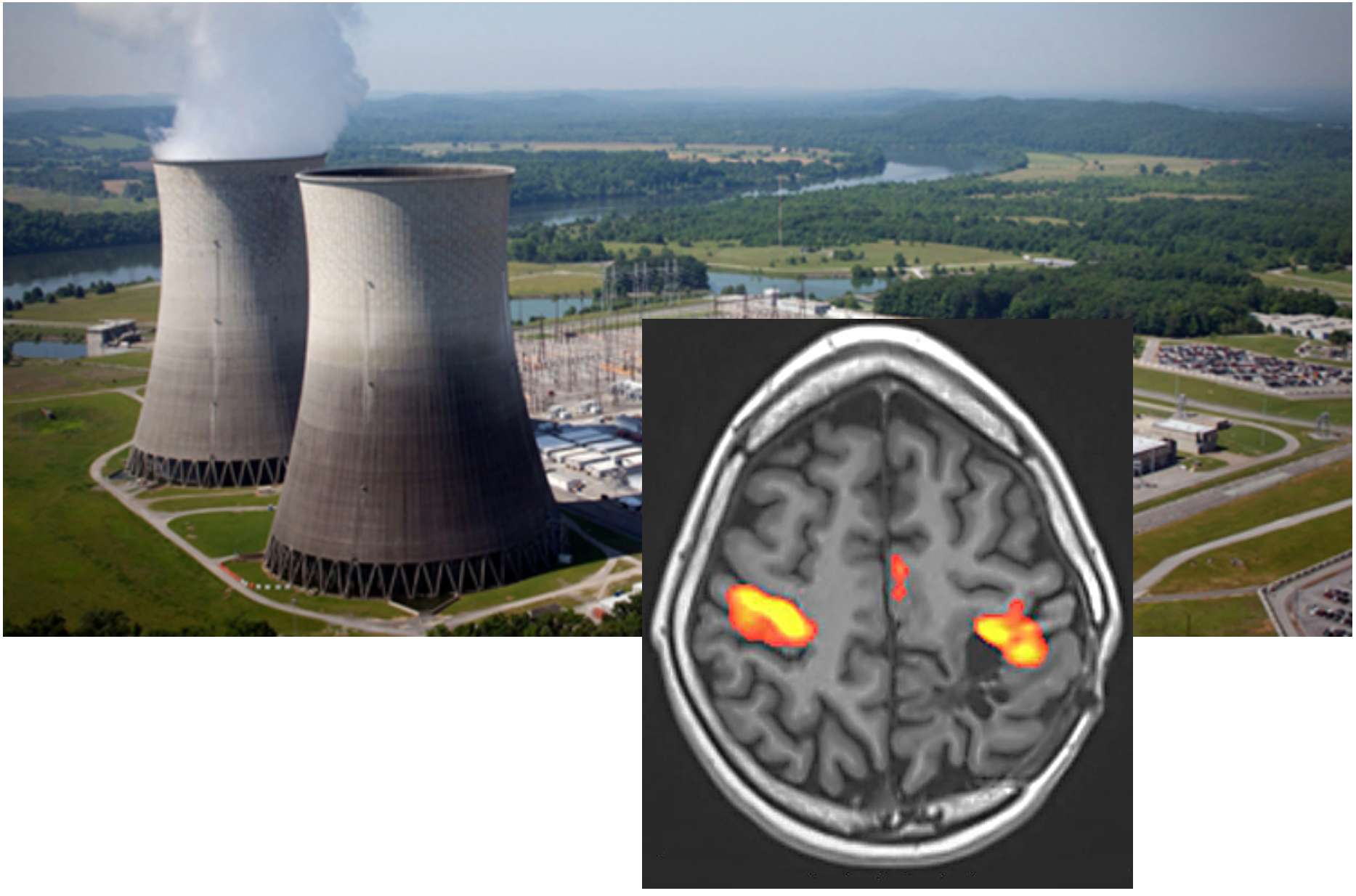
$$\sigma_{ij} = D_{ijkl}\epsilon_{kl}$$

coordinate system



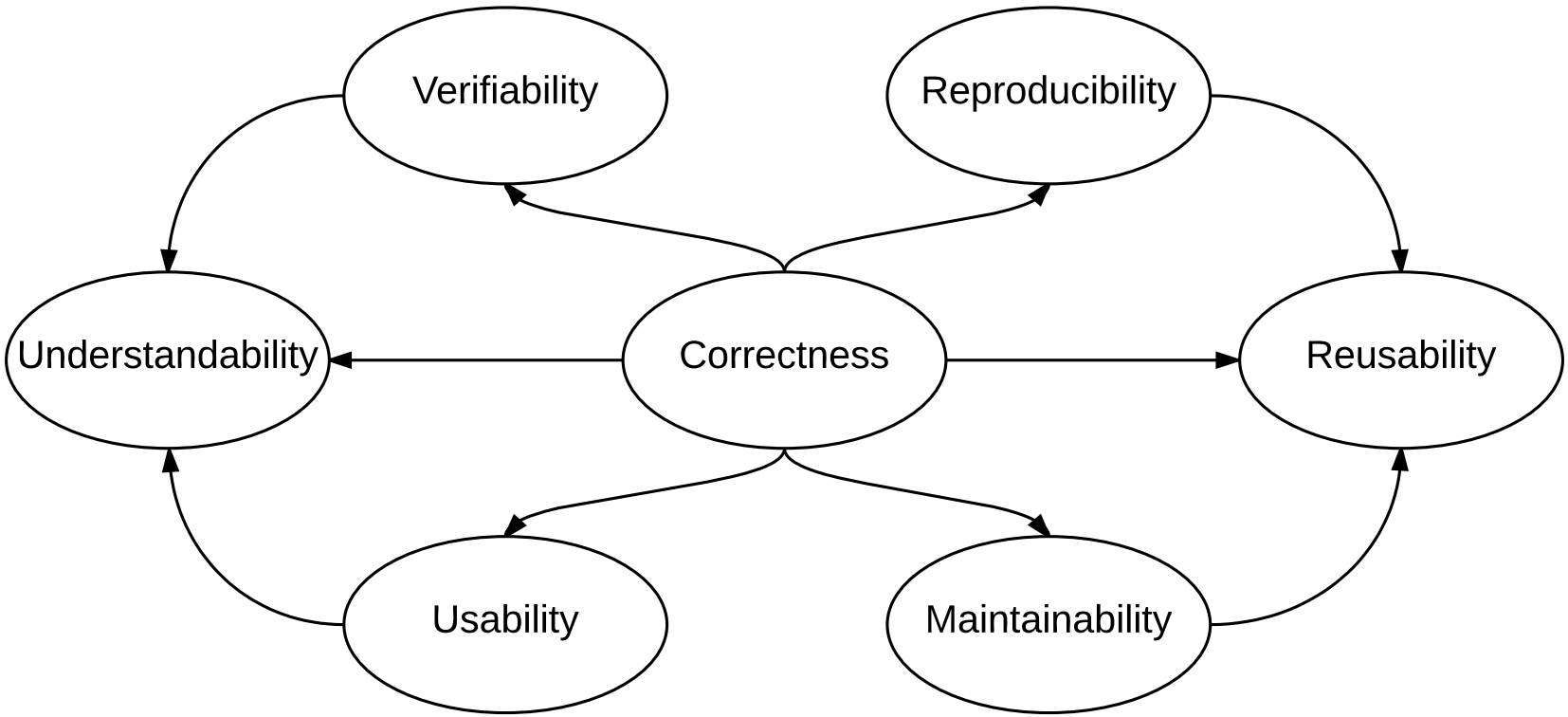


Motivation: Safety



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	✗	✓	✗	✗	✗
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	✗	✓	✗	✗	✗

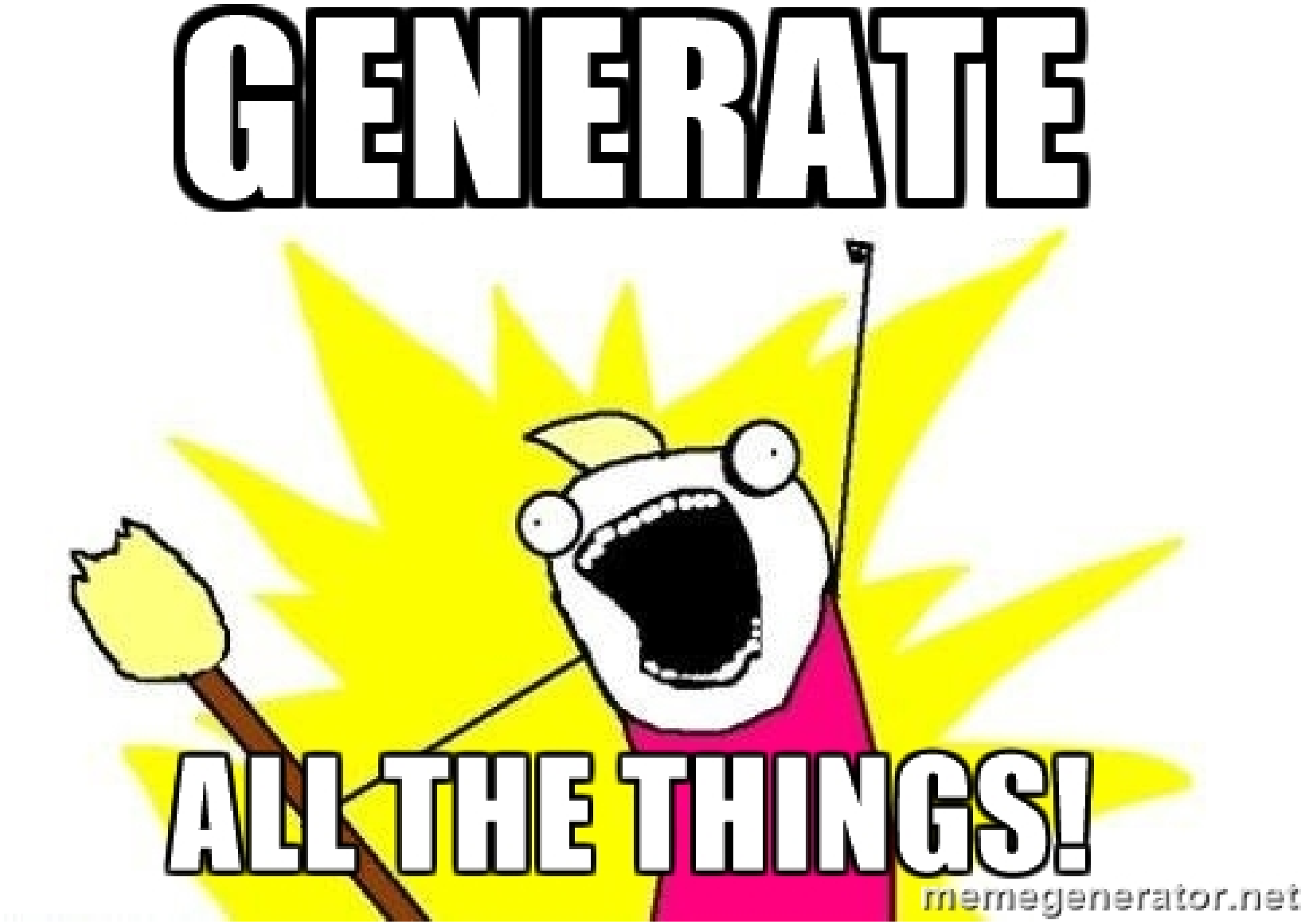
Perceived Advantages from Participants

- 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)

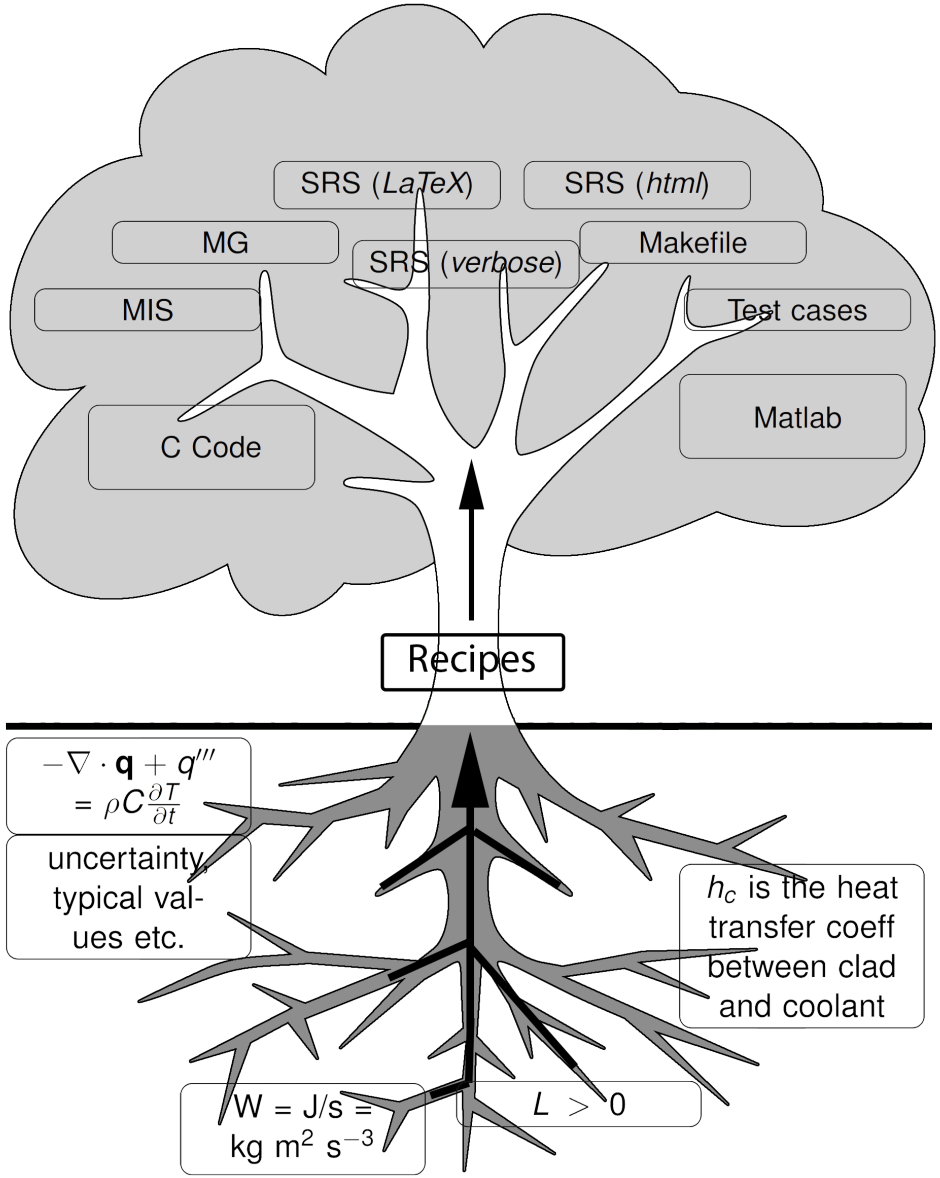
- SRS is too long
- SRS is not necessary
- DDD will not work in reality, since needs upfront requirements
- Too much SE jargon
- Difficult without a team of people
- Too difficult to maintain
- Not amenable to change
- Too tied to waterfall process
- Reports counterproductive [10]

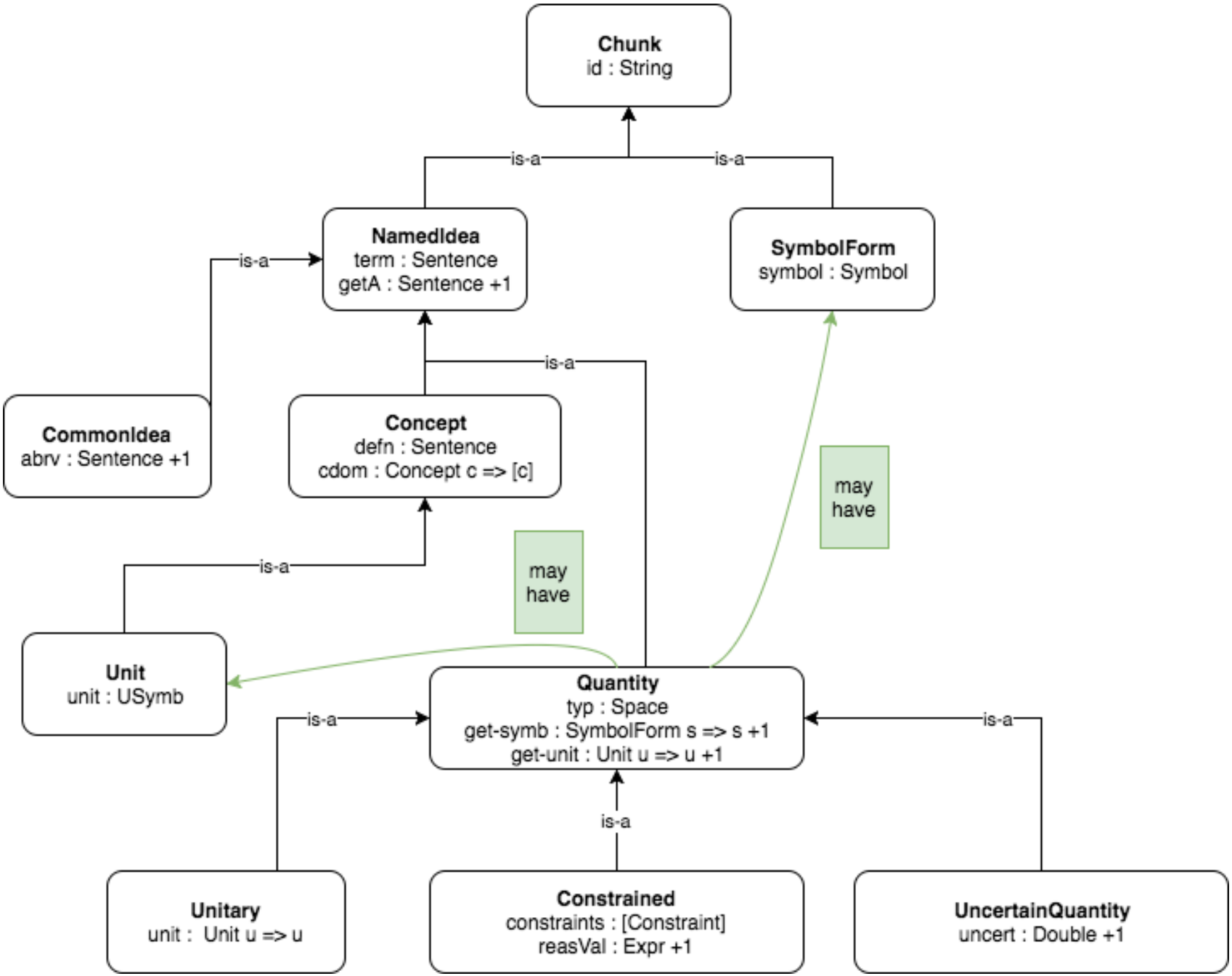
The Solution?


GENERATE
ALL THE THINGS!
memegenerator.net

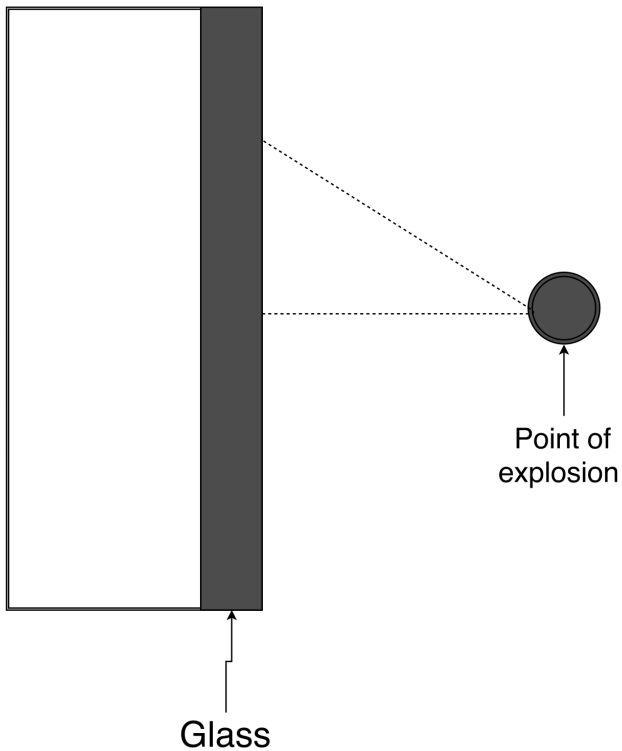
Knowledge Capture

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
CommonIdea
abrv : Sentence +1
Concept
defn : Sentence
cdom : Concept c => [c]
is-a
may have
may have
is-a
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

GlassBR



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 ($\hat{q}_{\text{tol}}$), 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 ($\hat{q}_{\text{tol}}$), 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
...

tex

Software Requirements Specification for GlassBR
Nikitha K and Spencer S

Table of Symbols
$\hat{q}_{\text{tol}}$
B
...

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

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

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

html

ssBR

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

README.md

GlassBR

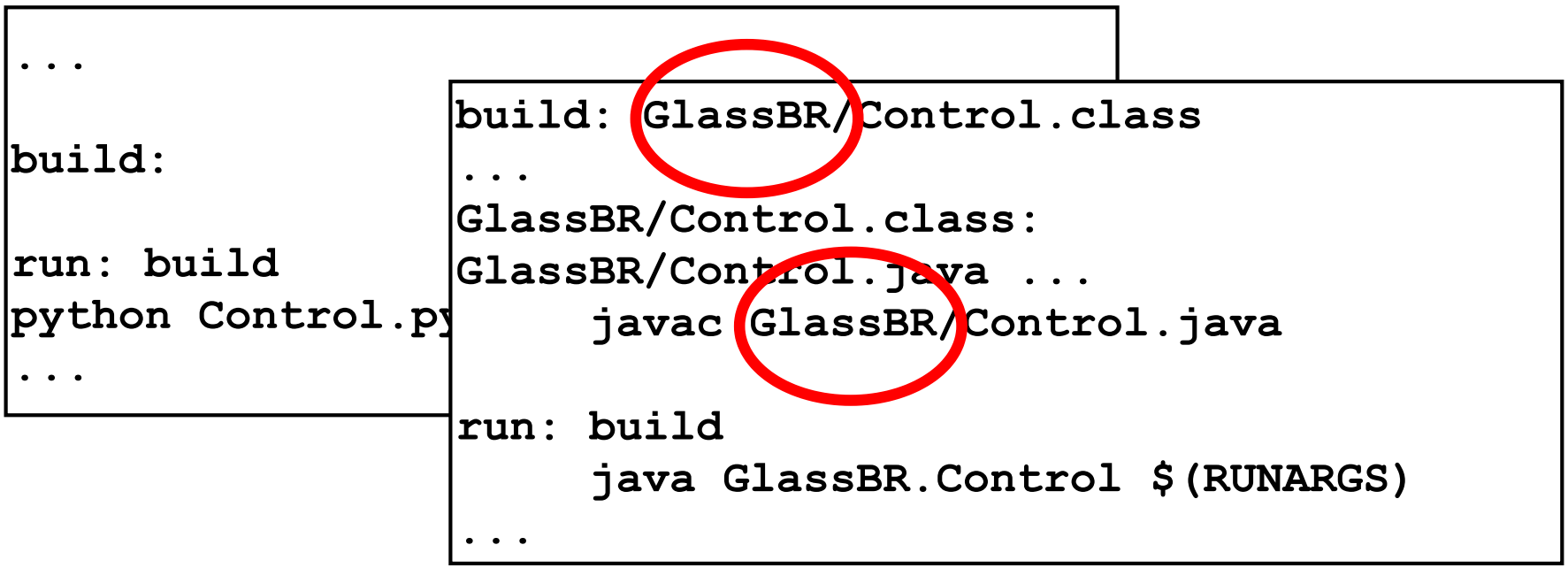
Authors Nikitha K and Spencer S

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
# \brie
...
## \bri
# \para
# \para
# \retu
def fur
    out
    pri
    ...
    out

    ret
```

Calculations.java

```
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(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	$J_{tol} = \log\left(\log\left(\frac{1}{1-P_{btol}}\right)\frac{\left(\frac{a}{1000}\frac{b}{1000}\right)^{m-1}}{k\left(\left(E\cdot 1000\left(\frac{h}{1000}\right)^2\right)\right)^m \cdot LDF}\right)$
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) (m) b is the plate width (short dimension) (m) m is the surface flaw parameter ($\frac{\mathrm{m}^{12}}{\mathrm{N}^7}$) k is the surface flaw parameter ($\frac{\mathrm{m}^{12}}{\mathrm{N}^7}$) E is the modulus of elasticity of glass (Pa) h is the actual thickness (m) LDF is the load duration factor

J_{tol} in SRS.tex

```
...
Label & Stress distribution factor (Function) based on
    Pbtol

\\ \midrule \\
Symbol & ${J_{\text{tol}}}$

\\ \midrule \\
Units & Unitless

\\ \midrule \\
Equation & \begin{displaymath}
           {J_{\text{tol}}}=\ln\left(\ln\left(\frac{1}{1-{
               P_{\text{b}\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} LDF}\right)
           \end{displaymath}
\\ \midrule \\
Description & ...
```

J_{tol} in SRS.html

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

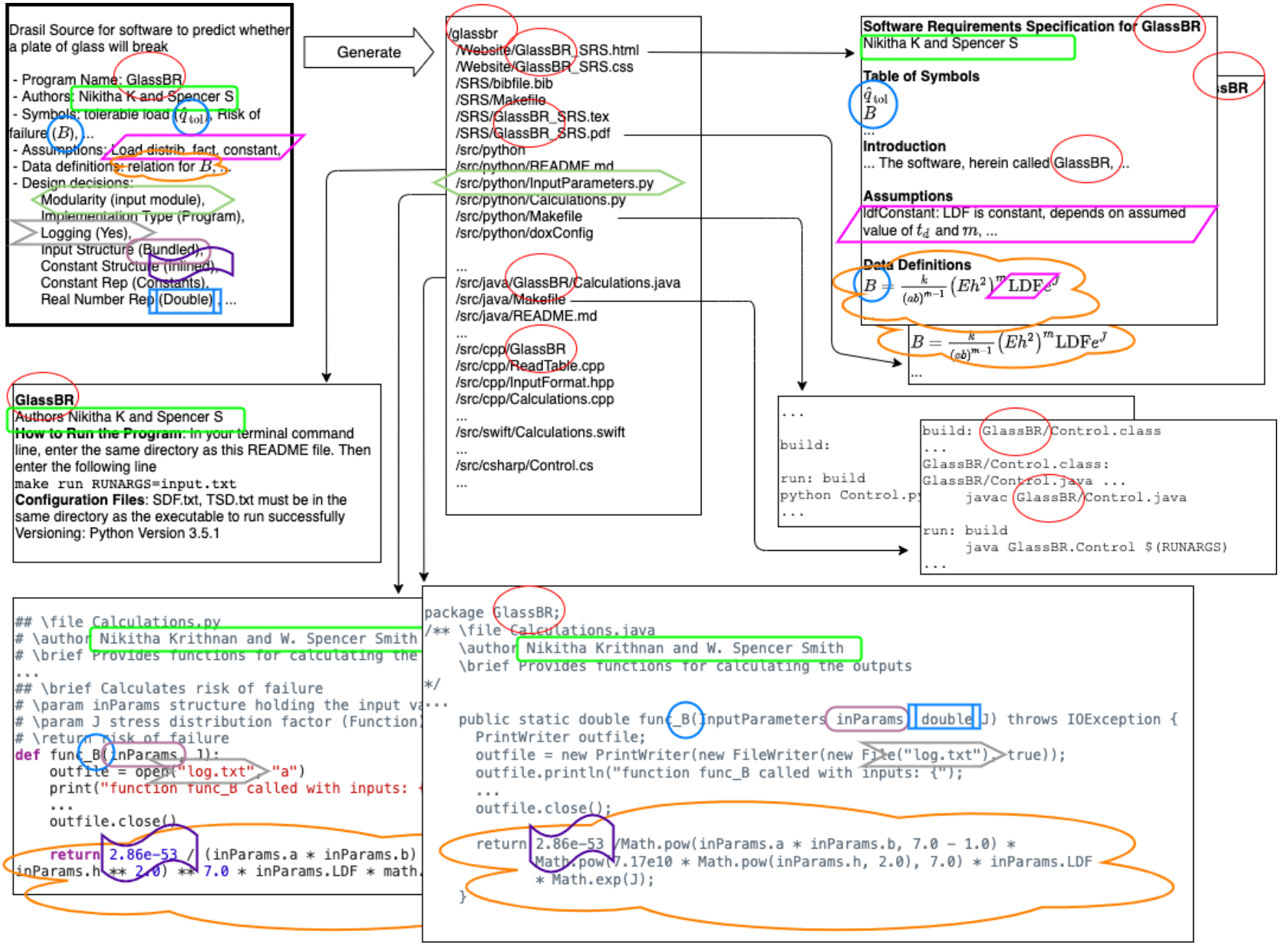
```
tolStrDisFacEq :: Expr
tolStrDisFacEq = ln (ln (recip_ (exactDbl 1 $- sy pbTol))
  `mulRe` (((sy plateLen $/ exactDbl 1000) `mulRe` (sy
      plateWidth $/ exactDbl 1000)) $^ (sy sflawParamM $-
      exactDbl 1) $/
    (sy sflawParamK `mulRe` ((sy modElas `mulRe` exactDbl
       1000 `mulRe`
   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

J_{tol} without Unit Conversion

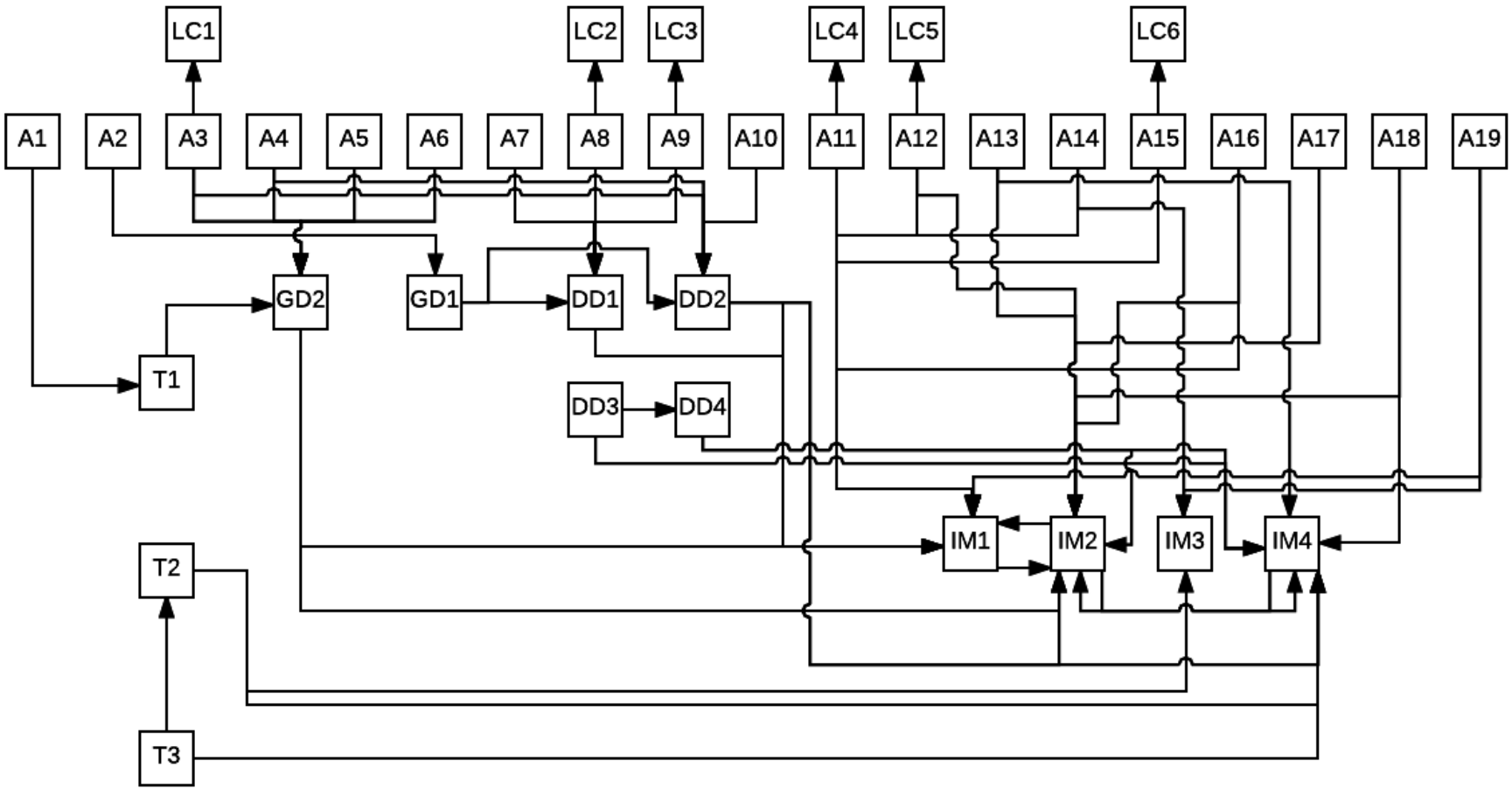
```
tolStrDisFacEq :: Expr
tolStrDisFacEq = ln (ln (recip_ (exactDbl 1 $- sy pbTol))
  `mulRe` ((sy plateLen `mulRe` sy plateWidth) $^ (sy
     sflawParamM $- exactDbl 1) $/
    (sy sflawParamK `mulRe` ((sy modElas `mulRe`
    square (sy minThick)) $^ sy sflawParamM) `mulRe` sy
       lDurFac)))
```

Drasil Inputs:

- Program Name: GlassBR
- Authors: Nikitha K and Spencer S
- Symbols: tolerable load ($\hat{q}_{\text{tol}}$), 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 ($\hat{q}_{tol}$), 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 (Inlined),
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/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
$B = \frac{k}{(ab)^{m-1}} \left(Eh^2\right)^m \mathrm{LDF} e^J$
$B = \frac{k}{(ab)^{m-1}} \left(Eh^2\right)^m \mathrm{LDF} e^J$
...
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)
...
\file Calculations.py
\author Nikitha Krithnan and W. Spencer Smith
\brief Provides functions for calculating the
...
\brief Calculates risk of failure
# \param inParams structure holding the input va
# \param J stress distribution factor (Function)
# \return risk of failure
def func_B(inParams, J):
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(7.17e10 * Math.pow(inParams.h, 2.0), 7.0) * 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]

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

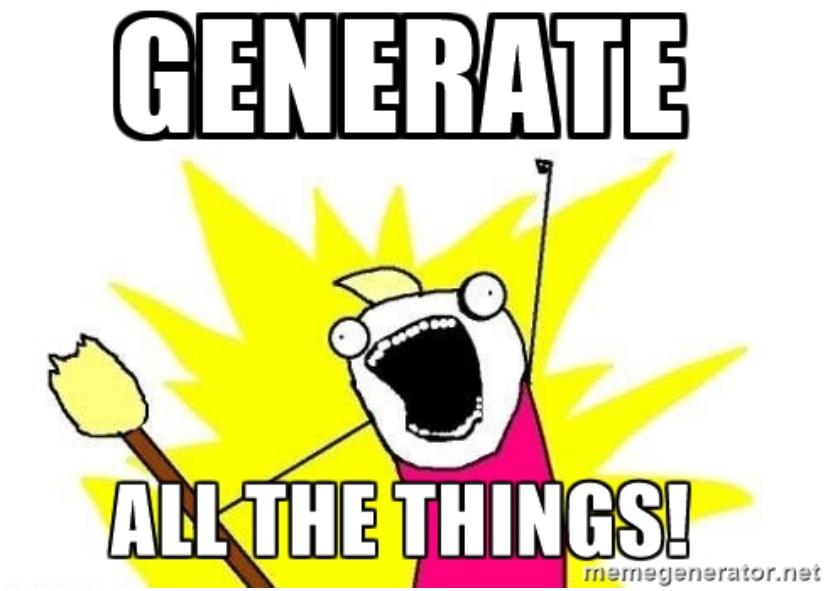
$$R_1^{\text{manual}} = \frac{f}{8\pi k_{\text{AV}}} + \frac{1}{2\pi r_f h_g} + \frac{\tau_c}{4\pi r_f k_c} \tag{2}$$

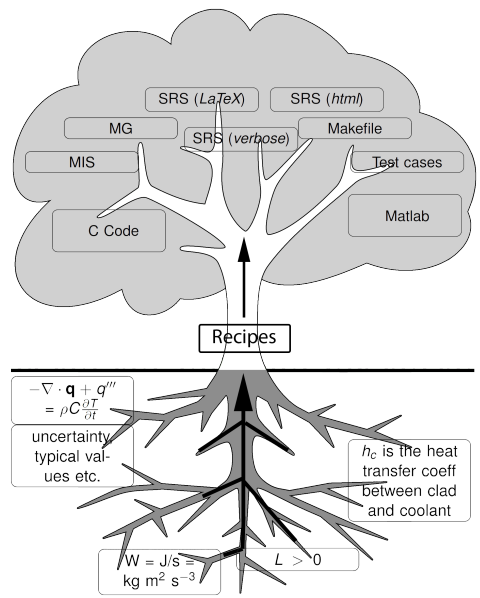
- Uncovered 27 issues with the previous documentation
  - Incompleteness (R_{gap})
  - Inconsistency(r, r_0, h_g)
  - Verifiability problems (R_1)
  - Lack of traceability (circuit analogy)
- Advantages of proposed approach
  - Abstract to concrete
  - Separation of concerns
  - Every equation, assumption,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

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 Wassyng.
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 Wassyng.
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.