Combining Literate Programming and Code Generation to Improve the Reproducibility and Sustainability of Scientific Computing Software

Spencer Smith, Jacques Carette

Computing and Software Department
Faculty of Engineering
McMaster University

UGA-McMaster University Joint Workshop, 2022

Outline

- Sustainable and Reproducible Research Software
- Pain Points
- Treatment Options
  - Literate Programming
  - Code Generation
  - Holistic Approach
- Concluding Remarks

Health Goals

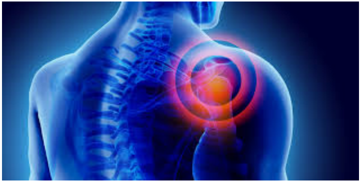
- **Sustainable** software satisfies, for a reasonable amount of effort, the software *requirements* for the present (like *correctness*), while also being maintainable, reusable, and *reproducible* for the future.
- **Reproducible** research includes all data, code, and documentation so that the computations can be *repeated in the future with identical results*.

Requires design, documentation, and verification (testing)

Problems with Achieving Goals: Pain Points

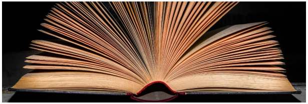
From Developer Interviews:

- Lack of time
- Lack of software development experience
- Lack of technology experience
- Frequency of change

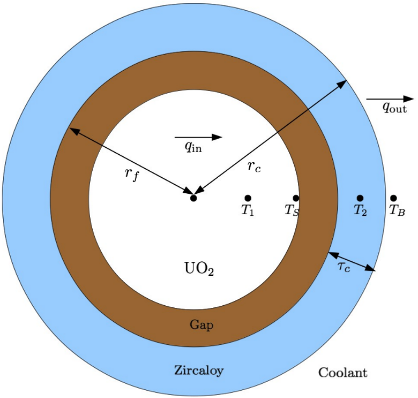


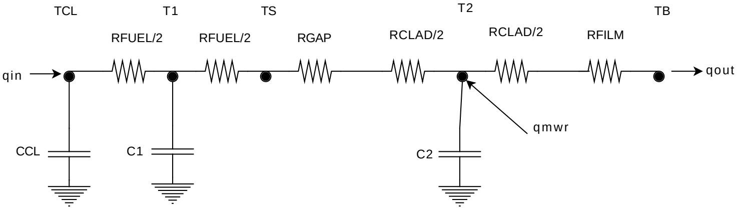
Treatment 1: Literate Programming

- “instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do” (Knuth, 1984, pg. 99)
- Interconnected “web” of pieces of code, or *chunks*
- Tangle extracts code
- Weave extracts docs (as LaTeX, html, pdf, text, etc.)
- CWEB, Sweave (R), Jupyter, emacs org mode, Maple worksheets, etc.



Health Goals
Literate
Code Gen
Holistic
Conclusion
References





Example

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 ($q'_{\text{N}_{\text{max}}}$) as given by DD25 of the SRS.

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

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

Health Goals
Literate
Code Gen
Holistic
Conclusion
References

LP Treatment Evaluation

- Uncovered 27 issues with previous docs
- Documentation improves reproducibility
- Pain point score:
  - Lack of time: ✓
  - Lack of dev exp: —
  - Lack of technology exp: ✗
  - Freq of change: ✓ and ✗
- Problems with literate programming
  - Does not scale well (best for small examples, lessons)
  - Difficult to refactor
  - *Manually* repeat information in text and code
  - *Manually* create traceability and structure

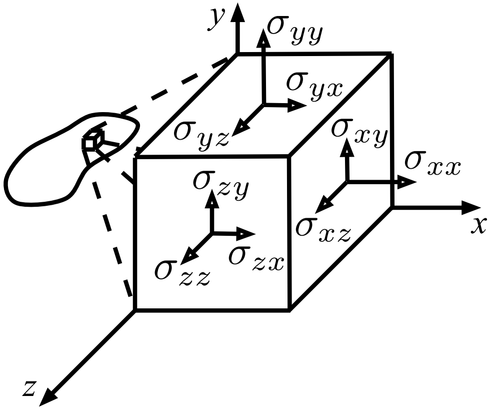
Health Goals
Literate
Code Gen
Holistic
Conclusion
References

Treatment 2: Code Generation

Health Goals
Literate
Code Gen
Holistic
Conclusion
References

A Virtual Material Testing Laboratory

Given the deformation history of a material particle, determine the internal stress within the material particle.



Calculations

Given $F, Q, \kappa, \phi, \gamma$ calculate:

$$\mathbf{K} = \int_V \mathbf{B}^T \mathbf{D}^{vp} \mathbf{B} dV; \mathbf{F} = \mathbf{R}_i - \int_V \mathbf{B}^T \boldsymbol{\sigma}_i dV + \int_V \mathbf{B}^T \Delta \boldsymbol{\sigma}^{vp} dV \tag{1}$$

with

$$\mathbf{D}_{vp} = \mathbf{D} \left[\mathbf{I} - \Delta t C_1 \lambda' \frac{\partial Q}{\partial \boldsymbol{\sigma}} \left(\frac{\partial F}{\partial \boldsymbol{\sigma}} \right)^T \mathbf{D} \right], \lambda' = \frac{d\lambda}{dF} \tag{2}$$

$$\Delta \boldsymbol{\sigma}^{vp} = \Delta t C_1 \lambda \mathbf{D} \frac{\partial Q}{\partial \boldsymbol{\sigma}} \tag{3}$$

$$C_1 = [1 + \lambda' \Delta t (H_e + H_p)]^{-1} \tag{4}$$

$$H_e = \left(\frac{\partial F}{\partial \boldsymbol{\sigma}} \right)^T \mathbf{D} \left(\frac{\partial Q}{\partial \boldsymbol{\sigma}} \right) \tag{5}$$

$$H_p = -\frac{\partial F}{\partial \kappa} \left(\frac{\partial \kappa}{\partial \boldsymbol{\epsilon}^{vp}} \right)^T \frac{\partial Q}{\partial \boldsymbol{\sigma}} \tag{6}$$

Code Generation

- Specify variabilities: $F, Q, \kappa, \phi, \gamma$
- Symbolically calculate terms, including $\frac{\partial Q}{\partial \boldsymbol{\sigma}}$, $\frac{\partial F}{\partial \boldsymbol{\sigma}}$, etc.
- Symbolic processing avoids tedious and error-prone hand calculations
  - Reduces workload
  - Allows non-experts to deal with new problems
  - Increases reliability
- Use Maple Computer Algebra System

Knowledge Capture and Code Generation

Code generation works by codifying additional knowledge:

- Maple – symbolic math
- org mode – simple document structure
- lex and yacc – regular expressions and grammars
- ATLAS – hardware knowledge (Whaley et al., 2001)
- Spiral – FFT knowledge (Ofenbeck et al., 2017)
- Dolphin – Finite elem variational forms (Logg, 2006)
- Doxygen – API information

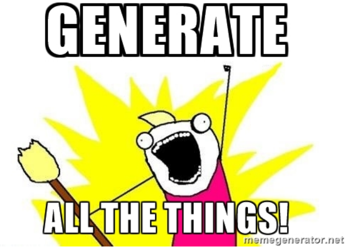
Code Generation Evaluation

- Domain level programming
- Pain point scores:
  - Lack of time: ✓
  - Lack of dev exp: ✓
  - Lack of technology exp: ✗
  - Freq of change: ✓
- Problems
  - Focus is generally only on the code
  - Code generation does not help with reproducibility

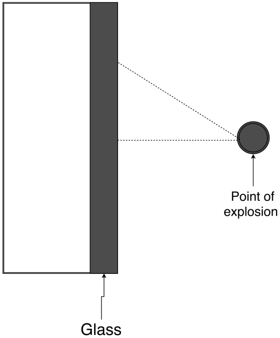
Health Goals
Literate
Code Gen
Holistic
Conclusion
References

Holistic Approach

- Combine
  - Lit programming emphasis on documentation
  - Code gen, but for everything
- Codify more knowledge
  - Physics knowledge
  - Computing knowledge
  - Document knowledge
  - Design knowledge
  - Traceability knowledge
  - Technology knowledge



GlassBR



Given

- dimensions of 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

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

...

html

ssBR

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

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

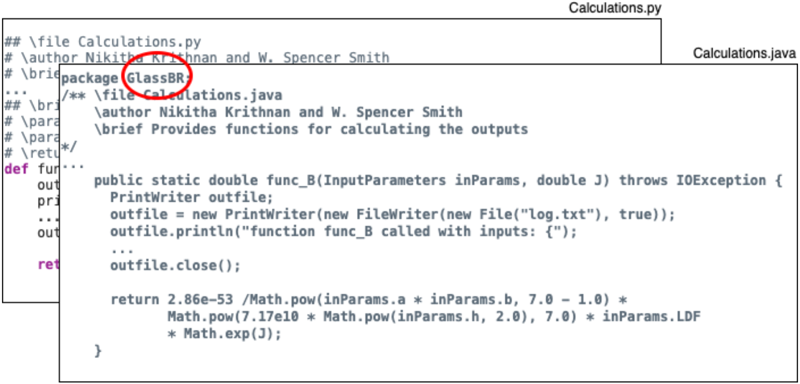
```
...

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
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:sdfTol**
Label	Stress distribution factor (Function) based on Pbtol
Symbol	J_{tol}
Units	Unitless
Equation	$$J_{\text{tol}} = \ln\left(\ln\left(\frac{1}{1-P_{\text{btol}}}\right)\frac{\left(\frac{a}{1000}\frac{b}{1000}\right)^{m-1}}{k\left(E\cdot 1000\left(\frac{h}{1000}\right)^2\right)^m LDF}\right)$$
Description	J_{tol} is the stress distribution factor (Function) based on Pbtol (Unitless) P_{btol} is the tolerable probability of breakage (Unitless) 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 minimum thickness (m) LDF is the load duration factor (Unitless)

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 * (7.17
        e10 * 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)))
```

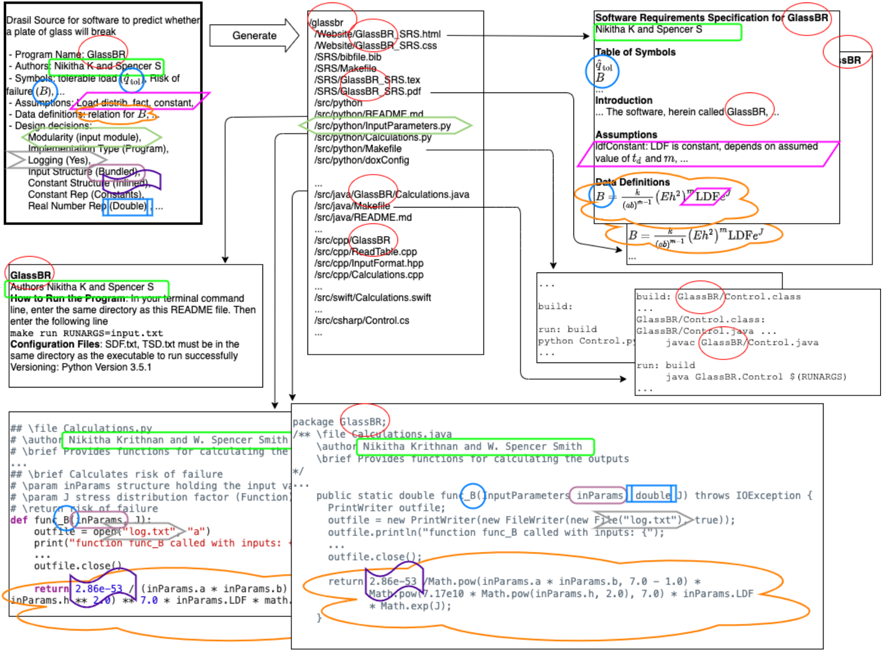
J_{tol} without Unit Conversion

```
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 (Carette et al., 2021)



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

Health Goals
Literate
Code Gen
Holistic
Conclusion
References

Holistic Treatment Evaluation

- Sustainable and reproducible
- Can generate literate documents, if desired
- Pain point scores:
  - Lack of time: ✓
  - Lack of dev exp: ✓
  - Lack of technology exp: ✓
  - Freq of change: ✓
- Treats all pain points, and no side effects, but expensive medicine!

Health Goals

Literate

Code Gen

Holistic

Conclusion

References

Concluding Remarks

- Documentation *does not have to be painful*
- Combine benefits of Literate Programming
  - Emphasis on documentation, reproducibility
  - Organize information for a human being
- with benefits of Code Generation
  - *Capture knowledge only once*
  - *Generate all things!*
  - *Refactoring by regenerating*
- *Codify as much knowledge as possible*
- *Domain experts work at domain expert level*
- *Consistent by construction*
- Can address additional pain points
- Can absorb other treatment options, like testing, CI
- Requires additional research and “clinical trials”

References I

Jacques Carette, Spencer Smith, Jason Balaci, Anthony Hunt, Ting-Yu Wu, Samuel Crawford, Dong Chen, Dan Szymczak, Brooks MacLachlan, Dan Scime, and Maryyam Niazi. Drasil, 2 2021. URL https://github.com/JacquesCarette/Drasil/tree/v0.1-alpha.

D. E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984. doi: 10.1093/comjnl/27.2.97. URL http://comjnl.oxfordjournals.org/content/27/2/97.abstract.

Anders Logg. Automating the finite element method. Technical Report Preprint 2006-01, Finite Element Centre, 2006. URL http://www.femcenter.org/preprints/.

References II

Georg Ofenbeck, Tiark Rompf, and Markus Püschel. Staging for generic programming in space and time. In *GPCE*, pages 15–28. ACM, 2017.

R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimization of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, 2001.

Image Credits

- Holistic Medicine: 6 Websites for Finding Natural Healing Advice
- Pain & Spine Center
- The Symptoms of a Rotator Cuff Injury and What You Should Do
- 16 Books Featuring Books on the Cover
- Difference Between Naturopathic and Holistic Medicine