# DRASIL

## A Knowledge-Based Approach to Scientific Software Development

Henry M, Aaron M, Maryyam N, Nicholas R, Dan S

McMaster University

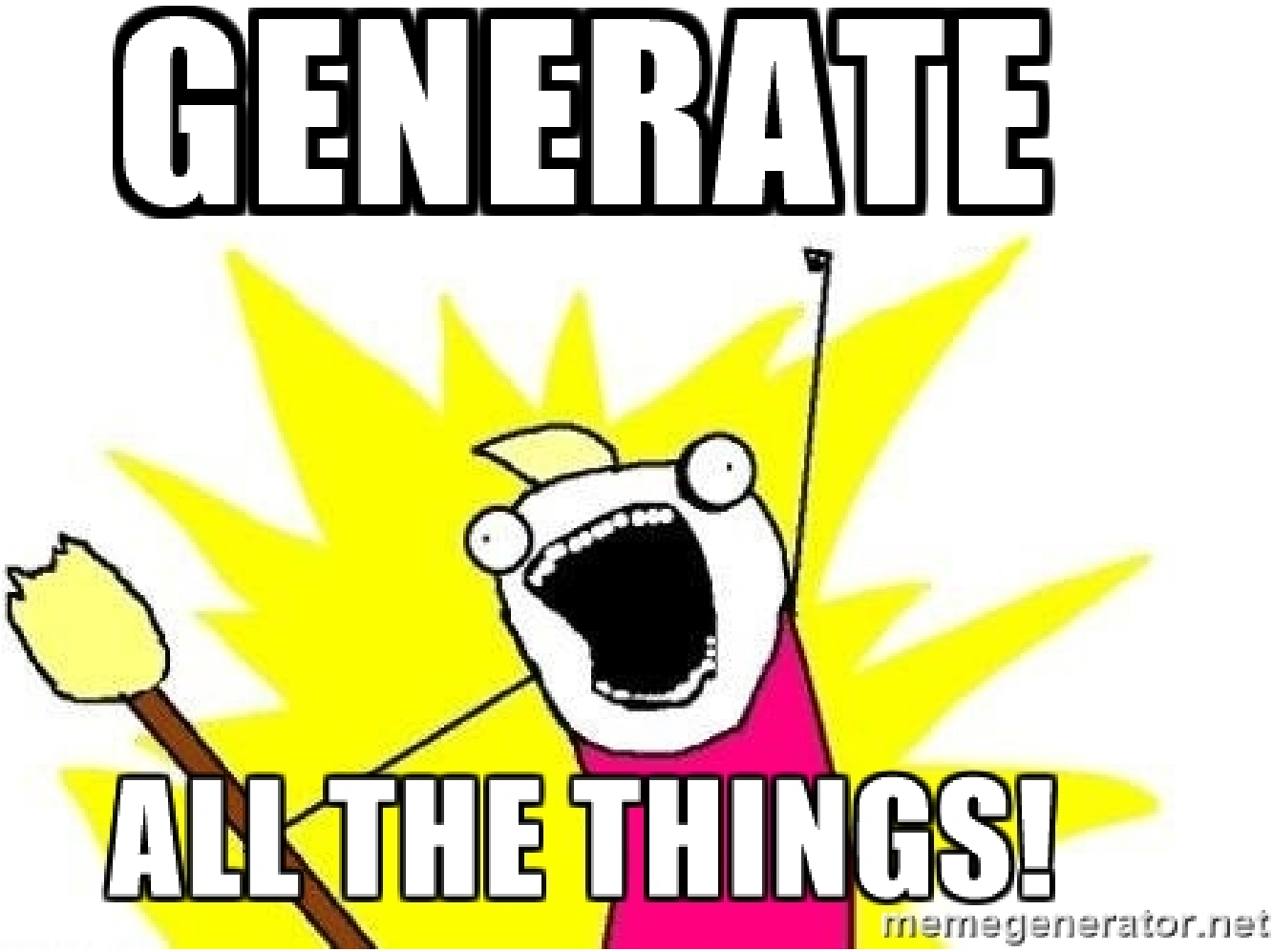
Literate Scientific Software Group, July 25, 2017

# Background Context

- $\exists$ problems $\in$ D where
- $D = \{$ scientific computing, engineering computing $\}$
- Problems $= [$
  - Inconsistent Software Requirement Specifications (SRS) across D
  - Inconsistency between code and documentation
  - Documentation is annoying to make and maintain
  - Hard to reuse code for different applications

  $]$

# The Goal



# Purpose of Drasil

- Solve the four problems
- Promote
  - Reusability
    - Examples have fully documented code
    - Data base to build new examples
  - Maintainability
    - Make changes in one place, gets updated everywhere

# What is Drasil?

- Knowledge Capture (Data.Drasil)

# What is Drasil?

- Knowledge Capture (Data.Drasil)
- Language and Rendering (Language.Drasil)
  - Code Generation: transition from Drasil to working code
  - Documentation Generation: transition from Drasil to human readable documentation

# What is Drasil?

- Knowledge Capture (Data.Drasil)
- Language and Rendering (Language.Drasil)
  - Code Generation: transition from Drasil to working code
  - Documentation Generation: transition from Drasil to human readable documentation
- Case Studies (Example.Drasil)
  - This part is where you would input equations, requirements, and output code and documentation

# Introduction

- Scientific and engineering computing has the potential to lead other fields of software with its solid knowledge base

# Introduction

- Scientific and engineering computing has the potential to lead other fields of software with its solid knowledge base
- Drasil is intended to simplify the generation of documentation and code for scientific software

# Introduction

- Scientific and engineering computing has the potential to lead other fields of software with its solid knowledge base
- Drasil is intended to simplify the generation of documentation and code for scientific software
- Facilitate desirable software qualities such as traceability, verifiability, and reproducibility

# Introduction

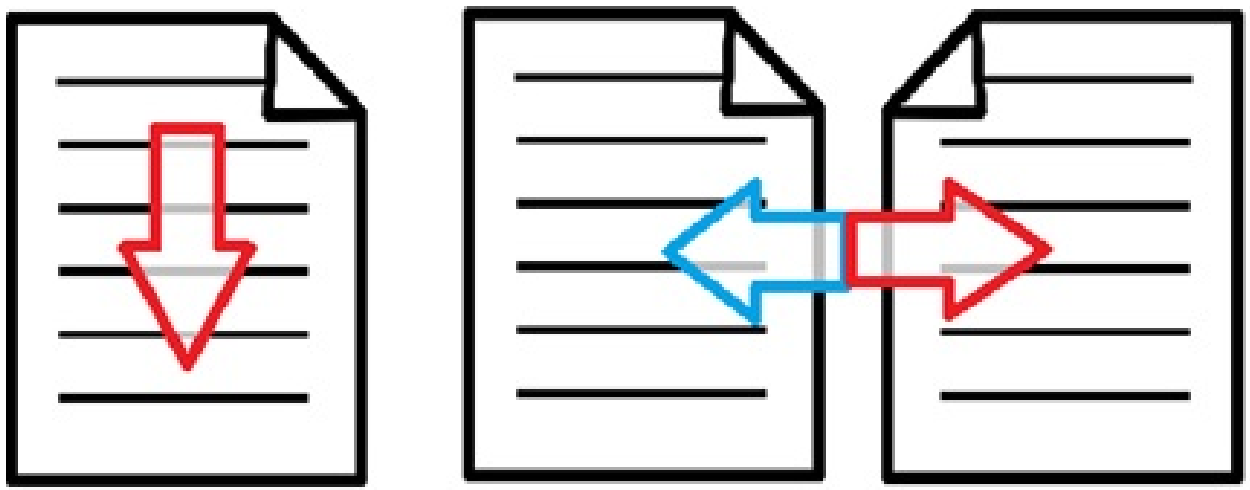
- Scientific and engineering computing has the potential to lead other fields of software with its solid knowledge base
- Drasil is intended to simplify the generation of documentation and code for scientific software
- Facilitate desirable software qualities such as traceability, verifiability, and reproducibility
- Case studies from which structural patterns and implicit relationships can be extracted, data can be captured, and core systems can be tested and implemented

# Daily Tasks

- Finding patterns within examples $\Rightarrow$ sentence combinators

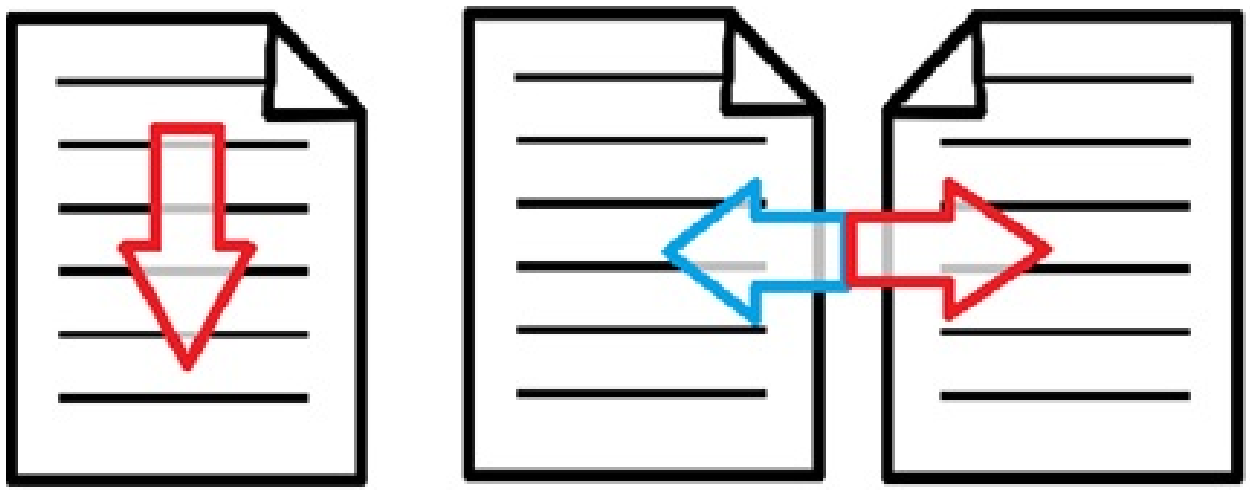
# Daily Tasks

- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts



# Daily Tasks

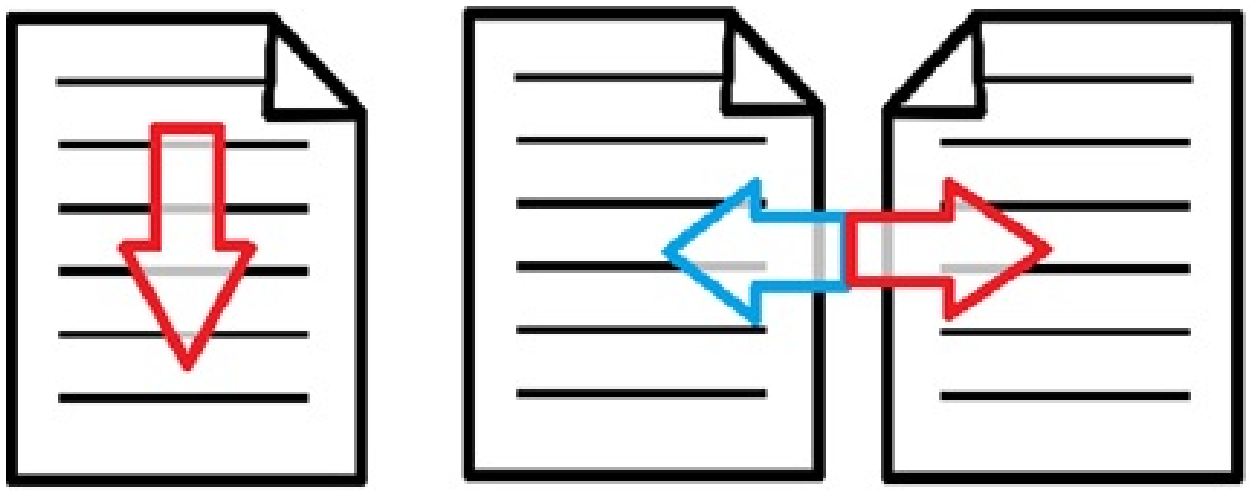
- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts



- Knowledge extraction

# Daily Tasks

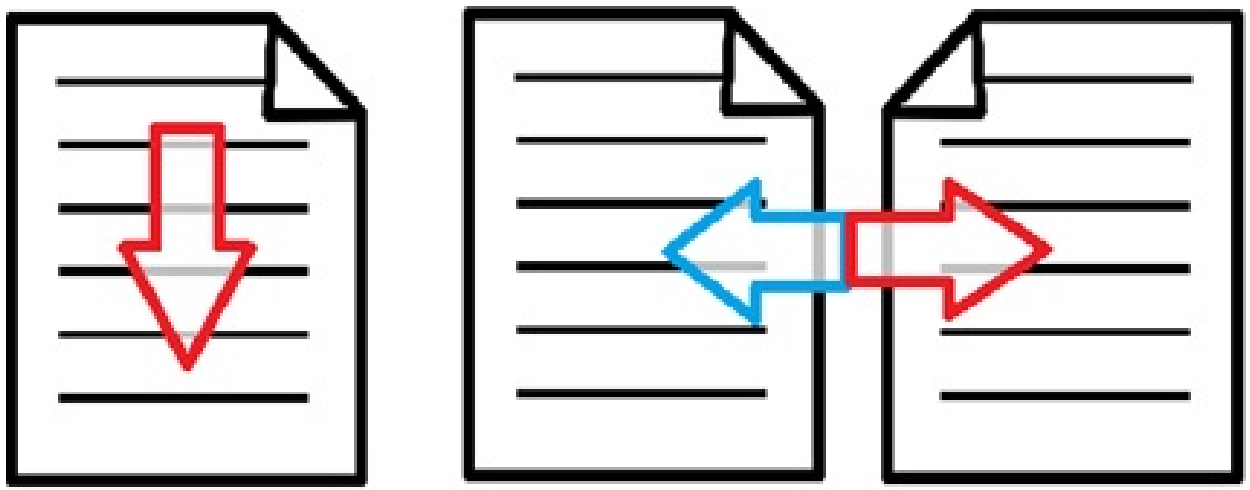
- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts



- Knowledge extraction
- Reduce duplication
  - Function efficiency
  - Building chunks off of each other

# Daily Tasks

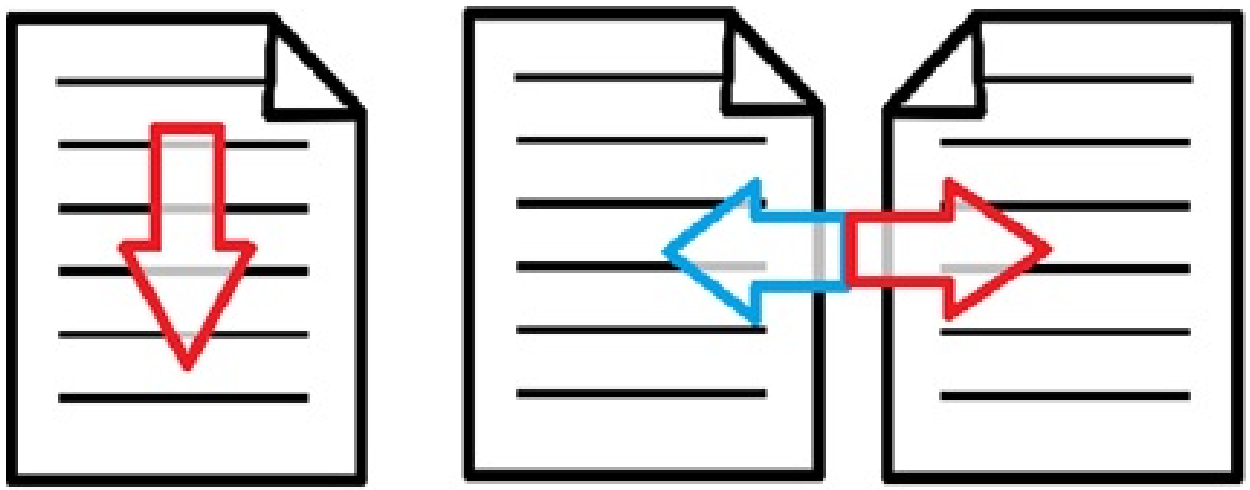
- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts



- Knowledge extraction
- Reduce duplication
  - Function efficiency
  - Building chunks off of each other
- Implement new functions/types created by supervisors

# Daily Tasks

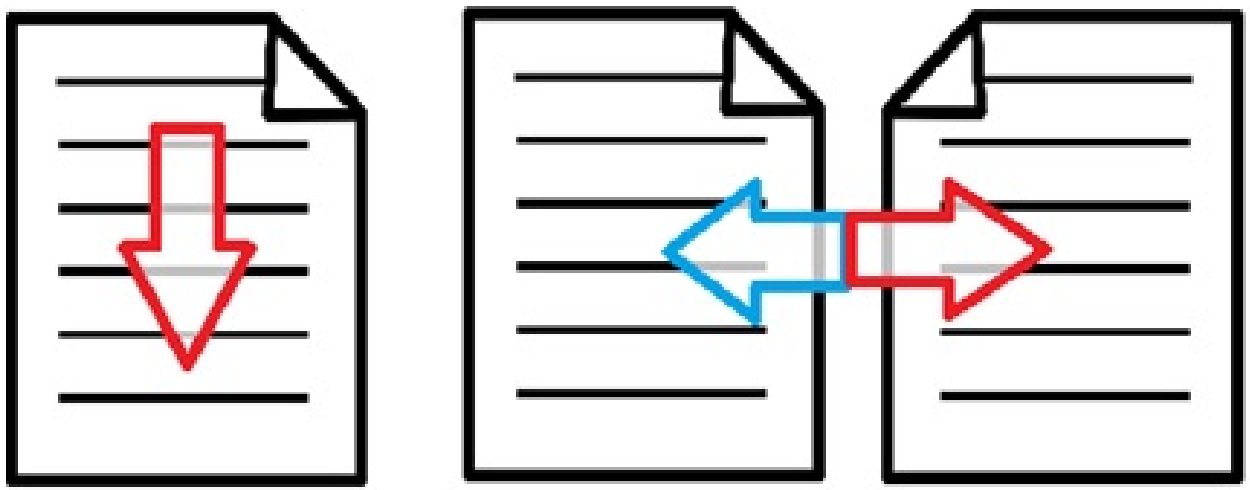
- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts



- Knowledge extraction
- Reduce duplication
  - Function efficiency
  - Building chunks off of each other
- Implement new functions/types created by supervisors
- Code cleanup and bug fixing

# Daily Tasks

- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts



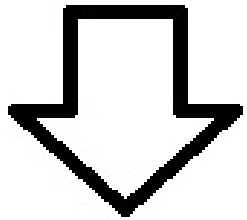
- Knowledge extraction
- Reduce duplication
  - Function efficiency
  - Building chunks off of each other
- Implement new functions/types created by supervisors
- Code cleanup and bug fixing
- Opening/closing issues

| Var | Physical Constraints | Software Constraints | Typical Value | Typical Uncertainty |
|---|---|---|---|---|
| $P_{btol}$ | $0.0 < P_{btol}$ and $P_{btol} < 1.0$ | None | 0.008 | 1.0e-3 |
| $TNT$ | $TNT > 0.0$ | None | 1 | 0.1 |
| $a$ | $a > 0.0$ and $\frac{a}{b} > 1.0$ | $d_{min} \leq a$, $a \leq d_{max}$, and $\frac{a}{b} < AR_{max}$ | 1500.0 m | 0.1 |
| $b$ | $b > 0.0$ and $b < a$ | $d_{min} \leq b$, $b \leq d_{max}$, and $\frac{a}{b} < AR_{max}$ | 1200.0 m | 0.1 |
| $w$ | $w \geq 0.0$ | $w_{max} \leq w$ and $w \leq w_{min}$ | 42.0 kg | 0.1 |
| $SD$ | $SD > 0.0$ | $SD_{min} < SD$ and $SD < SD_{max}$ | 45.0 m | 0.1 |

**Input Data Constraints**

# insert title

```
s6_2_5_table1 = Table [S "Var", S "Physical Cons", S "Software Constraints", S "Typical Value",
  S "Uncertainty"] (mkTable [(\x -> x!!0), (\x -> x!!1), (\x -> x!!2), (\x -> x!!3),
   (\x -> x!!4)] [[(P $ plate_len ^. symbol), (P $ plate_len ^. symbol) +:+ S "> 0 and" +:+
   (P $ plate_len ^. symbol) :+: S "/" :+: (P $ plate_width ^. symbol) +:+ S "> 1",
   (P $ dim_min ^. symbol) +:+ S "<=" +:+ (P $ plate_len ^. symbol) +:+ S "<=" +:+
   (P $ dim_max ^. symbol) +:+ S "and" +:+ (P $ plate_len ^. symbol) :+: S "/" :+:
   (P $ plate_width ^. symbol) +:+ S "<" +:+ (P $ ar_max ^. symbol), S "1500" +:+
   Sy (unit_symb plate_len), S "10%"], [(P $ plate_width ^. symbol),
   (P $ (plate_width ^. symbol)) +:+ S "> 0 and" +:+ (P $ plate_width ^. symbol)
   +:+ S "<" +:+ (P $ plate_len ^. symbol), (P $ dim_min ^. symbol) +:+ S "<=" +:+
   (P $ plate_width ^. symbol) +:+ S "<=" +:+ (P $ dim_max ^.symbol) +:+ S "and" +:+
   (P $ plate_len ^. symbol) :+: S "/" :+: (P $ plate_width ^. symbol) +:+ S "<" +:+
   (P $ ar_max ^. symbol), S "1200" +:+ Sy (unit_symb plate_width), S "10%"],
   [(P $ pb_tol ^. symbol), S "0 <" +:+ (P $ pb_tol ^. symbol) +:+ S "< 1", S "-", S "0.008", S "0
   [(P $ char_weight ^. symbol), (P $ char_weight ^. symbol) +:+ S ">= 0", (P $ cWeightMin ^. symb
   +:+ S "<" +:+ (P $ char_weight ^. symbol) +:+ S "<" +:+ (P $ cWeightMax ^. symbol), S "42" +:+
   Sy (unit_symb char_weight), S "10%"],[(P $ tNT ^. symbol), (P $ tNT ^. symbol) :+:
   S " > 0", S "-", S "1", S "10%"], [(P $ standOffDist ^. symbol), (P $ standOffDist ^. symbol)
   +:+ S "> 0", (P $ sd_min ^. symbol) +:+ S "<" +:+ (P $ standOffDist ^. symbol) +:+ S "<" +:+
   (P $ sd_max ^. symbol), S "45" :+: Sy (unit_symb standOffDist), S "10%"]])
```



```
s6_2_5_table1 :: Contents
s6_2_5_table1 = inDataConstTbl (gbInputDataConstraints)
```

# insert title

```
plate_len :: UncertQ
plate_len = uqcND "plate_len" (nounPhraseSP "plate length (long dimension)")
  1A metre Real
  [ physc $ \c -> c :> (Dbl 0),
    physc $ \c -> (c / (C plate_width)) :> (Dbl 1),
    sfwrc $ \c -> (C dim_min) :<= c,
    sfwrc $ \c -> c :<= (C dim_max),
    sfwrc $ \c -> (c / (C plate_width)) :< (C ar_max) ] (Dbl 1500) defaultUncrt
```

Figure

```
-- Creates the input Data Constraints Table
inDataConstTbl :: (UncertainQuantity c, SymbolForm c, Constrained c) => [c] -> Contents
inDataConstTbl qlst = Table ([S "Var"] ++ (isPhys $ physC (head qlst) qlst) ++
  (isSfwr $ sfwrC (head qlst) qlst) ++ [S "Typical" +:+ titleize value] ++
  (isUnc $ typUnc (head qlst) qlst))
  (map (\x -> fmtInputConstr x qlst) qlst)
  (S "Input Data Constraints") True
  where isPhys [] = []
        isPhys _  = [titleize' physicalConstraint]
        isSfwr [] = []
        isSfwr _  = [titleize' softwareConstraint]
        isUnc  [] = []
        isUnc  _  = [S "Typical Uncertainty"]
```

Figure

# Ease of Use

- Input:

# Ease of Use

- Input:
  - Equations (DataDefs, Instance Models)

# Ease of Use

- Input:
  - Equations (DataDefs, Instance Models)
  - Requirements

# Ease of Use

- Input:
  - Equations (DataDefs, Instance Models)
  - Requirements
  - Assumptions

# Ease of Use

- Input:
  - Equations (DataDefs, Instance Models)
  - Requirements
  - Assumptions
- Output:

# Ease of Use

- Input:
  - Equations (DataDefs, Instance Models)
  - Requirements
  - Assumptions
- Output:
  - Code that fits the requirements and assumptions

# Ease of Use

- Input:
  - Equations (DataDefs, Instance Models)
  - Requirements
  - Assumptions
- Output:
  - Code that fits the requirements and assumptions
  - Documentation (Module Guide, Software Requirements Specification)

# Errors in Drasil?

- Catching and correcting errors in software:

# Errors in Drasil?

- Catching and correcting errors in software:
  - If there is an error, it will be everywhere

# Errors in Drasil?

- Catching and correcting errors in software:
  - If there is an error, it will be everywhere
  - Easy to spot

# Errors in Drasil?

- Catching and correcting errors in software:
  - If there is an error, it will be everywhere
  - Easy to spot
  - Once it's fixed, it is also fixed everywhere else

# Case Study Contributions

- Each member assigned a case study as well as tasks and issues
- SWHS
  - Largest Example
  - ODEs

# Case Study Contributions

- Each member assigned a case study as well as tasks and issues
- SWHS
  - Largest Example
  - ODEs
- NoPCM
  - Builds off pre-existing SWHS example

# Case Study Contributions

- Each member assigned a case study as well as tasks and issues
- SWHS
  - Largest Example
  - ODEs
- NoPCM
  - Builds off pre-existing SWHS example
- GlassBR
  - General definition's omitted

# Case Study Contributions

- Each member assigned a case study as well as tasks and issues
- SWHS
  - Largest Example
  - ODEs
- NoPCM
  - Builds off pre-existing SWHS example
- GlassBR
  - General definition's omitted
- SSP
  - Indexing
  - Sophisticated math
  - Diversity of symbols

# Case Study Contributions

- Each member assigned a case study as well as tasks and issues
- SWHS
  - Largest Example
  - ODEs
- NoPCM
  - Builds off pre-existing SWHS example
- GlassBR
  - General definition's omitted
- SSP
  - Indexing
  - Sophisticated math
  - Diversity of symbols
- GamePhysics
  - Most ambiguous example
  - SRS for a game physics library

# Daily Tasks

- Finding patterns within examples $\Rightarrow$ sentence combinators

# Daily Tasks

- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts

# Daily Tasks

- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts
- Knowledge extraction

# Daily Tasks

- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts
- Knowledge extraction
- Reduce duplication
  - Function efficiency
  - Building chunks off of each other

# Daily Tasks

- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts
- Knowledge extraction
- Reduce duplication
  - Function efficiency
  - Building chunks off of each other
- Implement new functions/types created by supervisors

# Daily Tasks

- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts
- Knowledge extraction
- Reduce duplication
  - Function efficiency
  - Building chunks off of each other
- Implement new functions/types created by supervisors
- Code cleanup and bug fixing

# Daily Tasks

- Finding patterns within examples $\Rightarrow$ sentence combinators
- Finding patterns between examples $\Rightarrow$ extraction of common sections, contents, and concepts
- Knowledge extraction
- Reduce duplication
  - Function efficiency
  - Building chunks off of each other
- Implement new functions/types created by supervisors
- Code cleanup and bug fixing
- Opening/closing issues

# Drasil Logic

- Drasil is a knowledge capturing system that allows for the easy reuse of information

# Drasil Logic

- Drasil is a knowledge capturing system that allows for the easy reuse of information
- Knowledge capture is achieved through the use of data types called chunks

# Drasil Logic

- Drasil is a knowledge capturing system that allows for the easy reuse of information
- Knowledge capture is achieved through the use of data types called chunks
- Combination of chunks to grow information

# Drasil Logic

- Drasil is a knowledge capturing system that allows for the easy reuse of information
- Knowledge capture is achieved through the use of data types called chunks
- Combination of chunks to grow information
- Related information should stem from one source (reduces duplication)

# Collaborative Efforts

- Peer review of code

# Collaborative Efforts

- Peer review of code
- Discussion of all around issues (ex. cyclic imports, referencing problems)

# Collaborative Efforts

- Peer review of code
- Discussion of all around issues (ex. cyclic imports, referencing problems)
- A lot of collaboration through GitHub

# Collaboration via Github

GitHub

# Collaboration via Github

Git is a version control system, github is a git repository hosting service that is free.

- Git allows us to collaborate effectively, even when team members are not in the same location

# Collaboration via Github

Git is a version control system, github is a git repository hosting service that is free.

- Git allows us to collaborate effectively, even when team members are not in the same location
- Git combined with haskell, allows us to make large changes while easily maintaining a working version of Drasil

# Collaboration via Github

Git is a version control system, github is a git repository hosting service that is free.

- Git allows us to collaborate effectively, even when team members are not in the same location
- Git combined with haskell, allows us to make large changes while easily maintaining a working version of Drasil
- Git (when used properly) prevents catastrophic lose of work

# End

For more information about Drasil and LLS visit our github page:
https://github.com/JacquesCarette/literate-scientific-software
You can even build a working version yourself!