

DRASIL

A Knowledge-Based Approach to Scientific Software Development

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

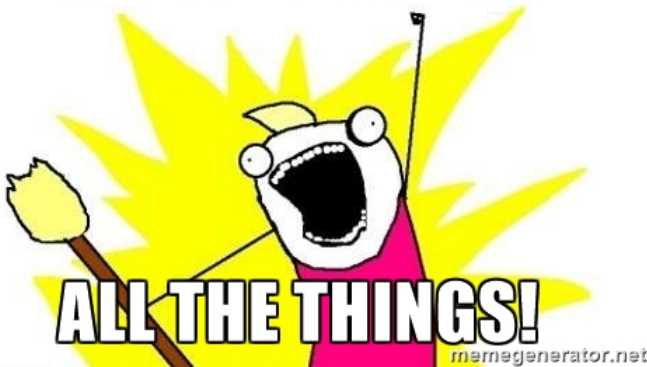
McMaster University

Literate Scientific Software Group, July 25, 2017

Background Context

- \exists problems $\in D$ where
- $D = \{ \text{scientific computing, engineering computing} \}$
- 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]

GENERATE



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
-
- Also to 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
- Also to facilitate desirable software qualities such as traceability, verifiability, and reproducibility
- Case study from which structural patterns and implicit relationships can be extracted, data can be captured, and core systems can be tested and implemented

- Finding patterns within examples \Rightarrow sentence combinators

- 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

- 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

- Input:

- Input:
 - Equations (DataDefs, Instance Models)

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

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

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

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

- 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

- Finding patterns within examples \Rightarrow sentence combinators

- 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

- 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

- 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

Collaboration via 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

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!