

DRASIL

A Knowledge-Based Approach to Scientific Software Development

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

McMaster University

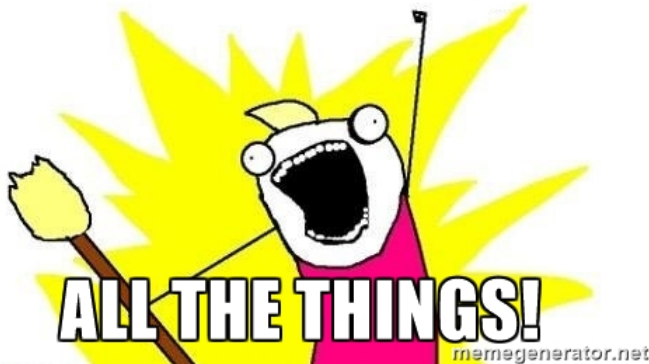
July 24, 2017

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]

The Goal

GENERATE



Purpose of Drasil

- ▶ Solve the four issues
- ▶ 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
- ▶ 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
- ▶ 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

next page

- ▶ Patterns within examples \Rightarrow sentence combinators

next page

- ▶ Patterns within examples \Rightarrow sentence combinators
- ▶ Patterns between examples \Rightarrow extraction of common sections, contents, and concepts

next page

- ▶ Patterns within examples \Rightarrow sentence combinators
- ▶ Patterns between examples \Rightarrow extraction of common sections, contents, and concepts
- ▶ Knowledge extraction

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

- ▶ Patterns within examples \Rightarrow sentence combinators
- ▶ 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

- ▶ Patterns within examples \Rightarrow sentence combinators
- ▶ 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
- ▶ Bug fixing

next page

- ▶ Patterns within examples \Rightarrow sentence combinators
- ▶ 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
- ▶ Bug fixing
- ▶ Opening/closing issues

Case Study Contributions

- ▶ SWHS
- ▶ NoPCM
- ▶ GlassBR
- ▶ HGHC
- ▶ SSP
- ▶ GamePhysics

end page

put content here