PhD Committee Meeting #1

Steven Palmer

Computing and Software Department
Faculty of Engineering
McMaster University

June 28, 2017

Overview

1. Introduction
2. Current Progress
3. Work to Date
4. Future Plans

Education History

- Ph.D. Computer Science
  - McMaster University
  - Started May 2016.
- B.A.Sc. Computer Science
  - McMaster University 2016
- M.A.Sc. Chemical Engineering
  - University of Toronto
  - Withdrew ABD
- B.A.Sc. Chemical Engineering
  - University of Toronto 2011

Requirements

- Courses:
  - CAS 701 – Logic & Discrete Math (A+)
  - CAS 781 – Category Theory (A)
  - MATLS 711 – Advanced Thermodynamics (transfer)
- Comprehensive Exams:
  - Computing Fundamentals – Completed May 2017
  - Computer Science – Expected Nov 2017

Research Project

Drasil: A Framework for Literate Scientific Software

- Started in 2014 (Dan Szymczak, Ph.D. work)
- Overview:
  - Drasil is a tool for generating documentation and code for scientific software
  - Captures expert knowledge in reusable "chunks"
  - Uses "recipes" to organize chunks into documents
- Benefits:
  - Improve the qualities of verifiability, maintainability and reusability for scientific software
  - Simplify the certification/re-certification process by providing traceability across all documents and code

Dan’s Work

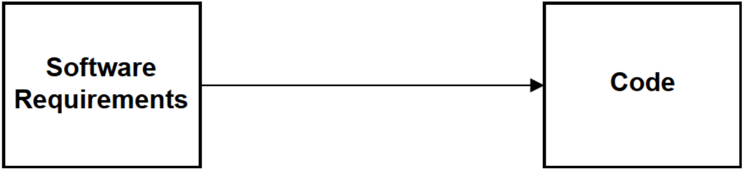
Dan’s work has focused on:

- Developing the Drasil infrastructure
- Designing the chunk system
- Designing recipes to produce a software requirements specification document from chunks

My Contribution

My work will focus on the code generation side of Drasil.

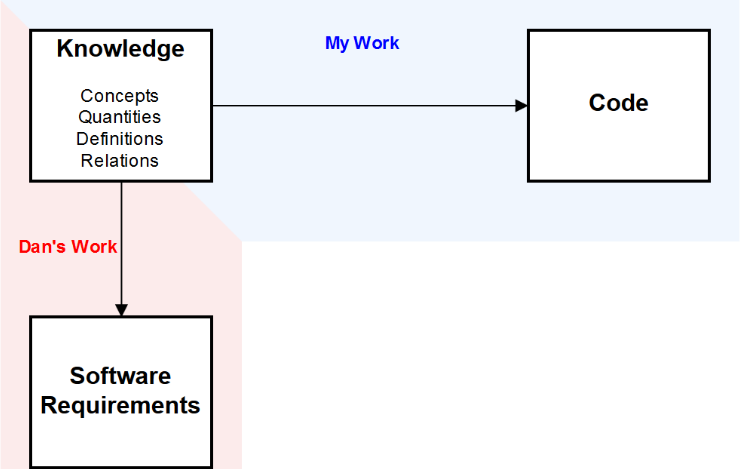
How do we get from requirements to an implementation?



My Contribution

Using Dan’s work as a starting point, this question can be restated:

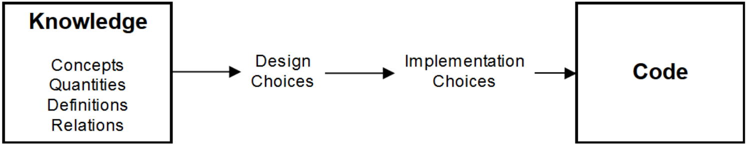
> How do we get from **a collection of knowledge** to an implementation?



My Contribution

In addition to the captured knowledge, there will need to be a way to accommodate choices:

- Design Choices:
  - What algorithms do we want to use?
  - Structure of input and output data?
  - ...
- Implementation Choices:
  - What language do we want to generate?
  - What optimizations do we want to make?
  - ...



Knowledge

Concepts
Quantities
Definitions
Relations

Design Choices

Implementation Choices

Code

Case Studies

Bottom-up approach: working backwards from an implementation to develop the code generation mechanism

We currently have six case studies for use as models to be replicated using Drasil:

- GlassBR (risk assessment software for glass plates subjected to blast loading)
- Straight-forward example where code consists of inputs loaded from a file, a series of calculation function calls, and outputs saved to a file

Implementation Language

GOOL

First step: Language Rendering

Goal: Provide rendering option for as many languages as we can!

GOOL integrated into Drasil for use as primary Implementation Language:

- ”Generic Object-Oriented Language”
- Developed by Jason Costabile
- Haskell DSL that allows rendering to different OO languages
- Currently supports C++, C#, Objective C, Java, Python, and Lua (can be extended to more)

GOOL

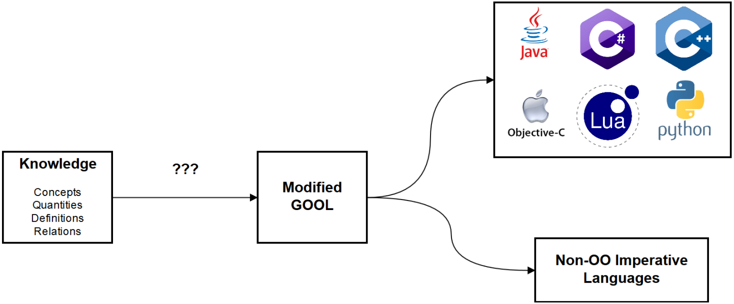
Second step: GOOL implementation of GlassBR (translated from Python)

This required some modifications to GOOL:

- GOOL was extended to allow non-OO code rendering (function and variable declarations outside of classes)
- Support was added for common console and file IO routines
- Other minor changes like library/module imports

GOOL

- The extension for non-OO code should allow for rendering of non-OO imperative languages.
- In the future we should be able to generate any imperative language!



Knowledge

Concepts
Quantities
Definitions
Relations

???

Modified GOOL

Non-OO Imperative Languages

Design Language

Third step: Bridging the gap between knowledge chunks and GOOL code (currently here)

- First attempt: pulling out patterns in the GOOL implementation of GlassBR
- Generation of input, output, and calculations modules achieved
- Code related to linear interpolation less obvious

Future Plans

Requirements

- Complete course requirements:
  - 3/8 complete, 5 more required
  - May look at taking some math courses outside of department
  - 4 this year, 1 next year vs. 5 this year?
- Complete Computer Science comprehensive exam in November.

Future Plans

Project

- Develop a first revision of the design language:
  - Finish code generation for GlassBR example to produce fully working code.
  - Extend code generation to other five examples.

- Implement C generation in GOOL
- Work on test case generation

Thank You!