# PhD Committee Meeting #4

Dan Szymczak

Computing and Software Department
Faculty of Engineering
McMaster University

June 28, 2018

# Overview

1. Current Progress
2. Example.
3. Next Steps.

# Research Topic Recap

## Motivation

- Too much duplication!

# Research Topic Recap

## Motivation

- Too much duplication!
- (Re-)Certification is expensive

# Research Topic Recap

## Motivation

- Too much duplication!
- (Re-)Certification is expensive
- Inter-/intra-artifact consistency issues

# Research Topic Recap

## Motivation

- Too much duplication!
- (Re-)Certification is expensive
- Inter-/intra-artifact consistency issues
- Promote reusability

# Research Topic Recap

## Motivation

- Too much duplication!
- (Re-)Certification is expensive
- Inter-/intra-artifact consistency issues
- Promote reusability
- Design for change

# Research Topic Recap

## KBSE & The Drasil Framework

A Knowledge-Based Software Engineering Approach

- Too much duplication!
- (Re-)Certification is expensive
- Inter-/intra-artifact consistency issues
- Promote reusability
- Design for change

# Research Topic Recap

## KBSE & The Drasil Framework

A Knowledge-Based Software Engineering Approach

- Single knowledge-base
- (Re-)Certification is expensive
- Inter-/intra-artifact consistency issues
- Promote reusability
- Design for change

# Research Topic Recap

## KBSE & The Drasil Framework

A Knowledge-Based Software Engineering Approach

- Single knowledge-base
- Generate artifacts
- Inter-/intra-artifact consistency issues
- Promote reusability
- Design for change

# Research Topic Recap

## KBSE & The Drasil Framework

A Knowledge-Based Software Engineering Approach

- Single knowledge-base
- Generate artifacts
- Guaranteed consistency
- Promote reusability
- Design for change

# Research Topic Recap

## KBSE & The Drasil Framework

A Knowledge-Based Software Engineering Approach

- Single knowledge-base
- Generate artifacts
- Guaranteed consistency
- Reusable across projects
- Design for change

# Research Topic Recap

## KBSE & The Drasil Framework

A Knowledge-Based Software Engineering Approach

- Single knowledge-base
- Generate artifacts
- Guaranteed consistency
- Reusable across projects
- Easy to mix and match

Drasil Framework – Generating Software Families

- One “source”, multiple views
  - Requirements
  - Design
  - Test Cases
  - Build instructions
  - ...

# Current Program Progress

- Completed all necessary graduate courses.
- Completed comprehensive examinations.
- Drasil framework proof-of-concept completed
    - Scoped-down due to nature of project
    - Generating SRS for six case studies & code for one

# Current Progress

## Cont'd

Currently Writing:

- Journal paper for ACM TOSEM
- Thesis

# Research Topic

## How?

Knowledge Capture!

- Advantages
  - Avoid duplication through chunk reuse.
  - Improve understandability, traceability and reproducibility.
  - Increased flexibility

# Example: $h_g$

A simple example taken from the SRS for FP

$h_g$ is a symbol which appears in several locations including:

- The Software Requirements Specification
- The Literate Programmer's Manual
- The Source Code

Let's take a look!

# Example: $h_g$

## SRS Definition for $h_g$ (original)

| **Number** | **DD1** |
|---|---|
| Label | $h_g$ |
| Units | $ML^0t^{-3}T^{-1}$ |
| SI | $\frac{\mathrm{kW}}{\mathrm{m}^2(^\circ\mathrm{C})}$ |
| Equation | $h_g = \frac{2k_c h_p}{2k_c + \tau_c h_p}$ |
| Description | $h_g$ is the gap conductance<br>$\tau_c$ is the clad thickness<br>$h_p$ is initial gap film conductance<br>$k_c$ is the clad conductivity<br>NOTE: Equation taken from the code |
| Sources | source code |

# Example: $h_g$

## LPM Definition for $h_g$ (original)

$$h_g = \frac{2k_c h_p}{2k_c + \tau_c h_p} \tag{1}$$

The corresponding C code is given by:

```
double calc_hg(double k_c,double h_b,double tau_c)
{
 return (2*(k_c)*(h_p)) / ((2*(k_c)) + (tau_c*(h_p)));
}
```

# Example: $h_g$

A simple example taken from the SRS for FP

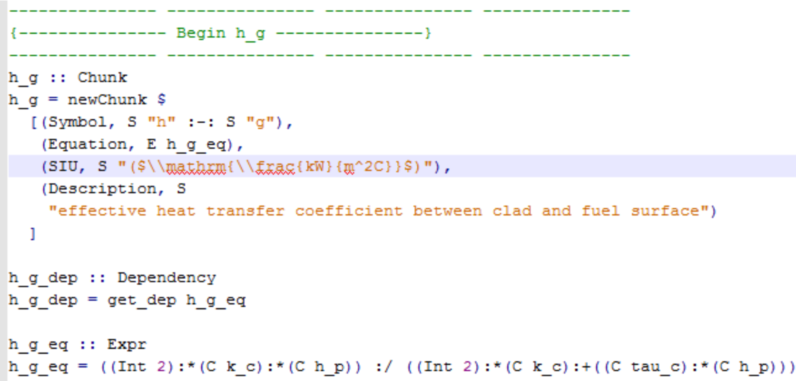
Modifying $h_g$ to reflect changes in requirements is not a simple matter. It involves, at the very least, the following steps:

- Update the definition in the SRS, LPM, and all other documents which reference the symbol
- Modify the source code to reflect the new requirements
- Trace all dependencies
- Modify dependents to accomodate the change
- Ensure each of the documents is now up to date and consistent

# Example: $h_g$

## Simplifying the process

Here is an example of a “chunk” for $h_g$:



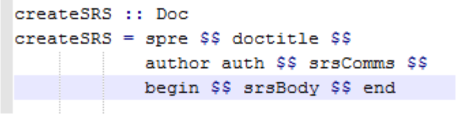
```
--------------- --------------- --------------- ---------------
{--------------- Begin h_g ---------------}
--------------- --------------- --------------- ---------------
h_g :: Chunk
h_g = newChunk $
  [(Symbol, S "h" :-: S "g"),
   (Equation, E h_g_eq),
   (SIU, S "($\\mathrm{\\frac{kW}{m^2C}}$)"),
   (Description, S
    "effective heat transfer coefficient between clad and fuel surface")
  ]

h_g_dep :: Dependency
h_g_dep = get_dep h_g_eq

h_g_eq :: Expr
h_g_eq = ((Int 2):*(C k_c):*(C h_p)) :/ ((Int 2):*(C k_c):+((C tau_c):*(C h_p)))
```

# Example: $h_g$

## How do we generate?

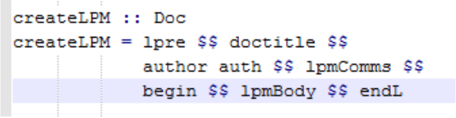
What do we do with the “chunk”?
That depends on the “recipe”!

- To create our SRS we use the following recipe:



```
createSRS :: Doc
createSRS = spre $$ doctitle $$
            author auth $$ srsComms $$
            begin $$ srsBody $$ end
```

- To create our LPM we use the following recipe:



```
createLPM :: Doc
createLPM = lpre $$ doctitle $$
            author auth $$ lpmComms $$
            begin $$ lpmBody $$ endL
```

# Example: $h_g$

## Generated SRS Output

| **Number** | **DD2** |
|---|---|
| Label | $h_g$ |
| Units | $ML^0t^{-3}T^{-1}$ |
| SI | $\frac{\text{kW}}{\text{m}^{2\circ}\text{C}}$ |
| Equation | $h_g = \frac{2k_ch_p}{2k_c+\tau_ch_p}$ |
| Description | $h_g$ is the effective heat transfer coefficient between clad and fuel surface<br>$k_c$ is the clad conductivity<br>$h_p$ is the initial gap film conductance<br>$\tau_c$ is the clad thickness<br>NOTE: Equation taken from the code |
| Sources | source code |

# Example: $h_g$

## Generated LPM Output

$$h_g = \frac{2k_c h_p}{2k_c + \tau_c h_p} \tag{2}$$

The corresponding C code is given by:

```
double calc_h_g(double k_c, double h_p, double tau_c)
{
        return 2*k_c*h_p/(2*k_c+tau_c*h_p);
}
```

# Next Steps

## Broad Strokes

### What next?

- Comprehensive examination part two.
- Complete final graduate level course.
  - Looking for a category theory course, but open to suggestions.
- Complete thesis.

Thank You!