Literate Software Development

Dan Szymczak

Computing and Software Department
Faculty of Engineering
McMaster University

Ernie Mileta Visit, Dec. 11, 2014

Overview

1. Literate Software Development.
2. Example.
3. Next Steps.

Literate Software Development for Scientific Software

- Motivation
  - Improve the qualities of verifiability, maintainability and reusability.
  - Save money and time when managing change.
- One "source," multiple views
  - Requirements, including or excluding derivations.
  - Design
  - Test Cases
  - Build instructions
  - ...
- Advantages
  - Reuse chunks and transformed chunks to avoid duplication.
  - Improve understandability, traceability and reproducibility.
  - Increased flexibility

Example: h_g and h_c

A simple example taken from the SRS for FP

h_g and h_c are symbols which appear in several locations including:

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

Let's take a look!

Example: h_g and h_c

SRS Definition for h_g

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

Example: h_g and h_c

LPM Definition for h_g

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

The corresponding C code is given by:

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

Example: h_g and h_c

A simple example taken from the SRS for FP

Modifying h_g or h_c to reflect changes in requirements is not a simple matter. It involves the following steps (at the very least):

- 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 and h_c

Simplifying the process

What if we could maintain everything in one source?

Example: h_g and h_c

Example h_g chunk

```
@Type
Data Definition
@Number
1

@Symbol
h_g

@Description
h_g is the gap conductance
!{tau_c}.Description
!{h_p}.Description
!{k_c}.Description

@Equation
...
```

Next Steps

What next?

- Generate a document from our chunks
- Add options for different document ”views”
  - Ex. SRS with or without derivations
- Add options for additional document types
- Generate the source code from the equations!

Thank You!