Lecture 12: Software Bricolage

Engineering Design with Embedded Systems

Patrick Lam
University of Waterloo

January 29, 2013

Last Time

Version control: copy, modify, merge.

Merge conflicts!

Software Bricolage

Software Bricolage

Today: Assignments versus real-world programming.

Most of your work for school is not open-ended until FYDP.

Some of your work in co-op will be open-ended. We'll see techniques for doing this work.

Schoolwork

Ideally:

- well-specified
- we provide tools and libraries you'll need (e.g. `LineGraphView`).

More on schoolwork

We have educational goals, so you get:

- Lots of template code—you fill in the blanks.
- Problems you can solve cleanly in a single language (Java).

e.g. Assignment 4: less than 50 lines; my Lab 1: 110 lines.

By the way, if you want to do an open-ended project for Labs 3 and 4, talk to me.

Real-world Programming

Often: check out large codebase,
fix something.

Sometimes: start a project from scratch.
(But not really from scratch).

Getting Started

You have a goal.

Need to formalize the goal:

- even high-quality software is no good unless it meets requirements.

ECE451 is all about requirements.

Steps to Build Software (per Philip Guo)

1. Forage
2. Tinker
3. Weld
4. Grow
5. Doubt
6. Refactor

Step 1: Foraging

(P. Lam collection)

Look for suitable components/libraries.
(maybe yours, maybe others').
Know what's out there.

It may be documented (if you're lucky).

Components may be in different languages.

Step 2: Tinker

(P. Lam collection)

- What can your code actually do?
- Experiment with the software!
- Give it test inputs.
- Instrument the code. Modify it.

This is very much like debugging.

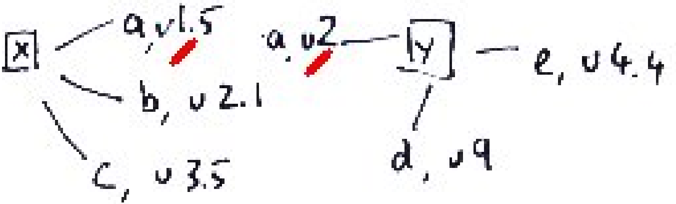
Repeat foraging and tinkering as needed.

Step 3: Weld

Two potential problems:

- dependencies;
- impedence mismatches.

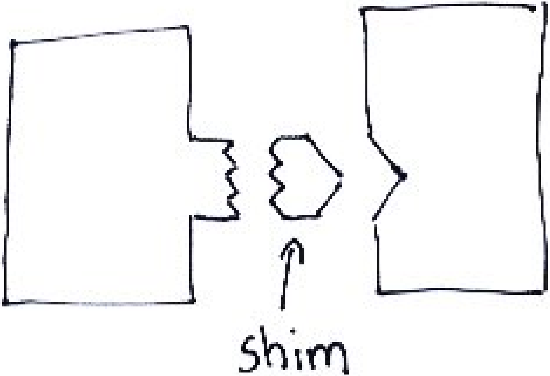
Dependencies



Sometimes you can’t get version you need.

Sometimes required versions conflict.

Impedence mismatches



You may have to build a shim
(e.g. XML output, CSV input!)

Step 4: Grow

Start building code.

Begin with simple examples, concrete code.

What's the simplest thing that can work?

Challenge: fix bad welds.

Step 5: Doubt

- Don’t reinvent the wheel.
  Know what’s in libraries.
  Ask the authors.
  Contribute to the library.

Step 6: Refactor

Clean your code, make it more general.

Improve interactions between your code and others.

Iterate

Iterate steps 4–6 as needed.
Grow, doubt, refactor.

Using the Web for Programming

Beware: Don’t indiscriminately copy code from the Internet.
Policy 71, and lawsuits (in industry).

Highly useful when used properly.

Three Main Ways

- Learn concepts.
- Clarify existing knowledge.
- Remind of details.

Learning concepts

Read tutorials.
Slow; hard to find good ones.
Gives an understanding of how things work.

Experiment with sample code.

Clarify existing knowledge

- Have some existing knowledge.
- Not quite sure about it.
- Also look up error messages (stackoverflow).

Remind of details

- especially syntax: not that important.

General tip: refine your queries iteratively.