Lecture 30 — Software Bricolage

Patrick Lam & Jeff Zarnett
p.lam@ece.uwaterloo.ca & jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

December 29, 2014

Part I

Software Bricolage

Software Bricolage

Today: Assignments vs 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

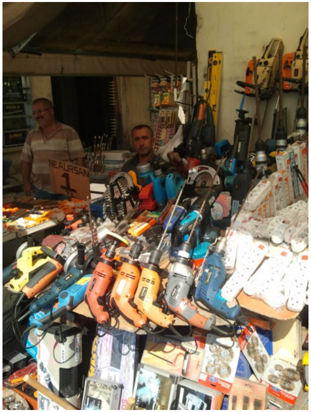
(P. Lam collection)

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.

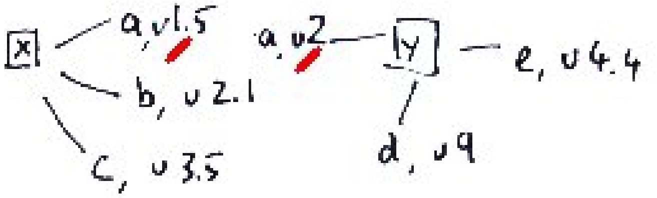
This is very much like debugging.

Repeat foraging and tinkering as needed.

Step 3: Weld

Two potential problems:

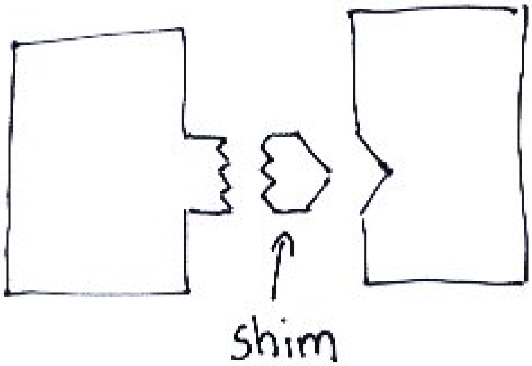
- dependencies;
- impedence mismatches.



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.

Part II

Licensing

Software Licenses

A Software License is a legal instrument that tells us how a piece of software may be used or distributed.

It grants you the rights to do things that would otherwise be an infringement (violation) of copyright law.

Software Licenses

Answers questions like:

- Where, how & how often can you install the program?
- Can you copy, modify, or redistribute it?
- Can you look at the underlying source code?

Software Licenses: Ignore Them?

This is boring. Who cares?! We do - can’t ignore it.

If you do not explicitly declare a license, you effectively get one anyway.

Code, like other works, is automatically copyrighted by default.

People can read code, but they have no legal right to use it.

Software Licenses: Proprietary

Impossible to generalize.

You get the rights specified in the license and that's it.

Software Licenses: GNU GPL

Most common example of an open-source license.

Referred to as “Copyleft”.

Ensures code can be used, copied, modified, and redistributed.

Code under this license can’t be used in proprietary programs.

Any changes you make must be licensed under GPL.

Software Licenses: GNU GPLv2 & GPLv3

More recent update to the GPL: version 3.

Intended to address a shortcoming called Tivoization.

Named after the Tivo digital video recorder.

Software Licenses: GNU GPLv2 & GPLv3

Tivo was based on Linux, open source software.

Added hardware protection to prevent modifying the software.

Software Licenses: GNU LGPL

The Lesser GPL is intended for software libraries.

Binary (compiled) code can be linked to proprietary programs.

Library code must follow GPL restrictions.

Software Licenses: BSD

A permissive open-source license.

Allows any use of code, even making it a proprietary product.

Disclaims all liability.

No requirement to share changes under any license.

Software Licenses: Mozilla

Open source license from the people who make Firefox.

Allows source code to be mixed - open-source + proprietary.

Any file licensed under MPL must be open source;
    But not all files must be open-sourced.

Hybrid between BSD and GPL licenses.

Software Licenses: Public Domain

No copyright owner - may be used by anyone for any purpose.

All copyrighted works will eventually enter the public domain.
	But this can take a long time. Life + 70 years.

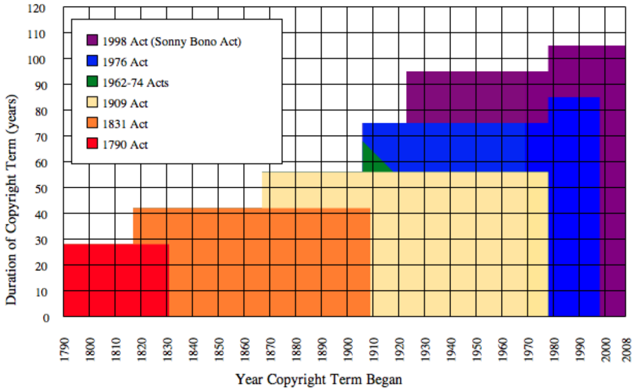
Authors can explicitly put code in the public domain.

Software Licenses: Public Domain

In the USA the amount of time before work enters the public domain keeps getting extended.

Disney is a big advocate: want to keep Steamboat Willie (first Mickey Mouse animation) from entering the public domain.

Software Licenses: Public Domain



You may safely assume a work will never enter the public domain unless the author chooses to put it there.

License Violations

License violations are taken seriously and lead to legal action.

Importing open source code into your software can get you into a lot of trouble.

License Violation Example

Landgericht Hamburg found a company, FANTEC, guilty of violating the GPL in their media player.

They distributed their firmware, containing some software licensed under the GPL (`iptables`).

They did not distribute the code as the GPL requires.

The court required FANTEC to pay penalty fee & legal costs.