

Twelve Life Lessons from Software Engineering

1. Design
2. Testing
3. Reuse

Dan Ellis

Dept. Electrical Engineering, Columbia University

dpwe@ee.columbia.edu <http://www.ee.columbia.edu/~dpwe/e4896/>

“Software Engineering”

- Software **DIY**
 - installing shelves at home



- Software **Engineering**
 - building a bridge or an airplane



- Projects involving many people need **management**

My Life as a Programmer

- 1981 (High School): BASIC, 8-bit assembler
- 1985 (College internship): C on a VT-100
- 1988 (Job): APL on a GUI
- 1990 (Grad school): C on a GUI (NeXT)
- 1992 (Grad school): MATLAB
Unix, Make
Tcl/Tk
- 1997 (Post-doc): C++ modules, libraries
Autoconf, RCS
- 2001 (Faculty): MATLAB again
- 2013 (Faculty): Python, Git

I. Writing Code is Writing

- Software is **expressing** an idea
- There are **many ways** to do the same thing
 - differences are “second order”
- **Aesthetic** differences
 - know when you’re setting traps
 - minimal commenting



2. Think About the Future

- “DIY” is all about **quick hacks**
 - something you need for now
 - usually discarded tomorrow
 - but not always
- Many tools have grown **far beyond** original vision
 - you never know when this will happen
- Worth anticipating
 - even if you are the only user

```
function [B,A,T,BW,FC] = bpfiltbank(SR, FMIN, BPO, BANDS, TYPE, T, BW, FC, VERB)
% [B,A,T,BW,FC] = bpfiltbank(SR,FMIN,BPO,BANDS,TYPE,T,BW,FC,VERB)
% Returns matrices B and A where B is the filter coefficients and A is the
% definitions for an IIR constant-bandwidth filterbank. The filterbank
% Their center frequencies range from FMIN to FMAX. The filterbank
% FMIN with BPO bands per octave. The filterbank is designed by iirbpfilt.m to have
% is designed by iirbpfilt.m to have a bandwidth of BW and an order 2*N. If TYPE is 0, use
% and an order 2*N. If TYPE is 1, use Slaney/Patterson filters. If TYPE is 2, use Slaney
% filters. If TYPE is 2, use Slaney/Patterson filters, else use the 'twoptwoz' filters.
% the 'twoptwoz' filters, else use the 'twoptwoz' filters.
% TYPE=4 is modified Slaney/Patterson filters.
% T is a vector of 'group delay' in samples.
% BW is a vector of bandwidths of the filters.
% FC is a vector of the actual center frequencies.
% VERB=1 for messages
% dpwe 1994jun21. Uses iirbpfilt.m

if nargin < 7; TYPE=0; end
if nargin < 8; VERBOSE=0; end

FMAX = FMIN*exp(log(2)*BANDS/BPO);

if (FMIN <= 0) || (FMAX <= FMIN)
    error('bpfiltbank: must be 0 < FMIN < FMAX')
end
```

3. What Can Go Wrong

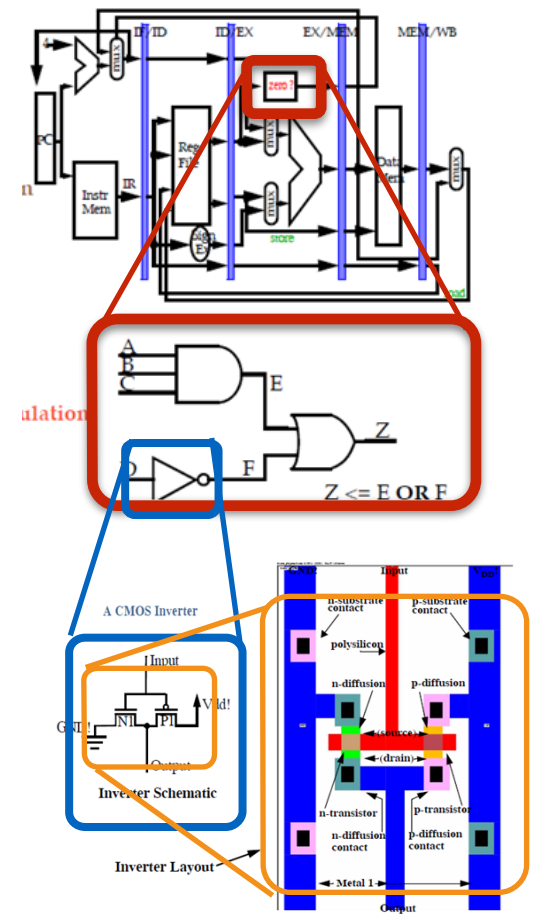
- We don't know how to solve the problem
- Program is too slow
- Program doesn't apply

- Program has **bugs**
 - programs are complicated machines
 - sometimes we get in too deep
 - we layer complexity until it fails



4. Modularity

- **Decomposing** a problem is the genius of engineering
 - Software > Language > OS > Machine Code > Microcode > VLSI > Transistors > Physics
- Decomposing a problem can make the problem disappear
 - threshold of triviality
- Modules help **shape your thinking**
- Modules offer re-use
- Top-down code composition
 - http://software-carpentry.org/4_0/invperc/assembly.html



5. Interfaces and Data Structures

- Program design has several parts
 - modules
 - data structures
 - interfaces / APIs
- The right **representation** can make all the difference
 - frames the function of the modules
- Opportunity to increase **generality**, future applications
- Opens door to **existing modules...**

6. Use Libraries / Tools

- We are **not** working in a vacuum
 - there are (probably) other people facing similar problems
 - some of them have made huge **investments** in tools
 - well-used tools/components are debugged
- Using a **library** involves a learning curve
 - it could be faster to write it yourself...
 - ... but it might still be better to use a tool
- A judgement call

7. Create Libraries / Tools

- Be on the lookout for **recurrent idioms**
- If you don't find a library, it's an opportunity
 - to help the community (fame and glory)
 - to increase your future **productivity**
- **Same issues as any sharing of code**
 - big investment
 - but: code review, beta testers
- **Design becomes important**
 - but design is always good

8. Publish Your Code

- What is needed for a **stand-alone presentation of this code?**
 - minimal documentation
 - sometimes have a target in mind
 - but worthwhile even without
 - your **future self** as the audience
- E.g. Matlab “**publish**”
 - combination of narrative & execution
 - examples of execution
 - (also, an implicit test case)

9. Version Control

- If other people are using your code, you can't just **change** it
 - edits may introduce bugs
 - users may rely on parts you consider unimportant
- Keep backups
- Make it possible for people to quickly **identify** which version they're using
- Maintain changelogs

10. Include Explicit Tests

- Often want to go back and **tweak** code
 - danger! you think you know what's going on
 - “no need to check this...”
- **Automated** tests
 - in Make file
 - as part of release process
- Just the **obvious** cases
- http://software-carpentry.org/4_0/test/index.html

I 0b. Will bugs be observable?

- Beware of cases where you don't know what to expect
 - you can't tell if it's doing what you think it was doing



11. Optimization

- One “second-order aspect” is execution **speed**
 - factors governing speed are frequently mysterious: cache size, compiler optimizations, parallelism
- Execution time is frequently **dominated** by one or two pieces - the “long pole”
 - **profiling** to identify + prioritize
- There’s usually low-hanging fruit

Profile Summary
Generated 12-Feb-2014 15:31:42 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
SAcC	1	2.304 s	0.025 s	
SAcC_main	4	2.171 s	0.058 s	
SAcC_pitchtrack	4	1.492 s	0.041 s	
cal_ac	4	1.309 s	0.058 s	
cal_ac>sub_cal_ac	4	0.937 s	0.265 s	
autocorrelogram	4	0.497 s	0.166 s	
autocorr (MEX-file)	96	0.332 s	0.332 s	
audioread	4	0.249 s	0.025 s	

12. Diminishing Returns

- **Some people love programming**
 - your own **private universe** - “castles in the sky”
- **Be critical & aware**
 - the **balance** between programming for the future and getting the job done
 - you can always fix it later

Summary

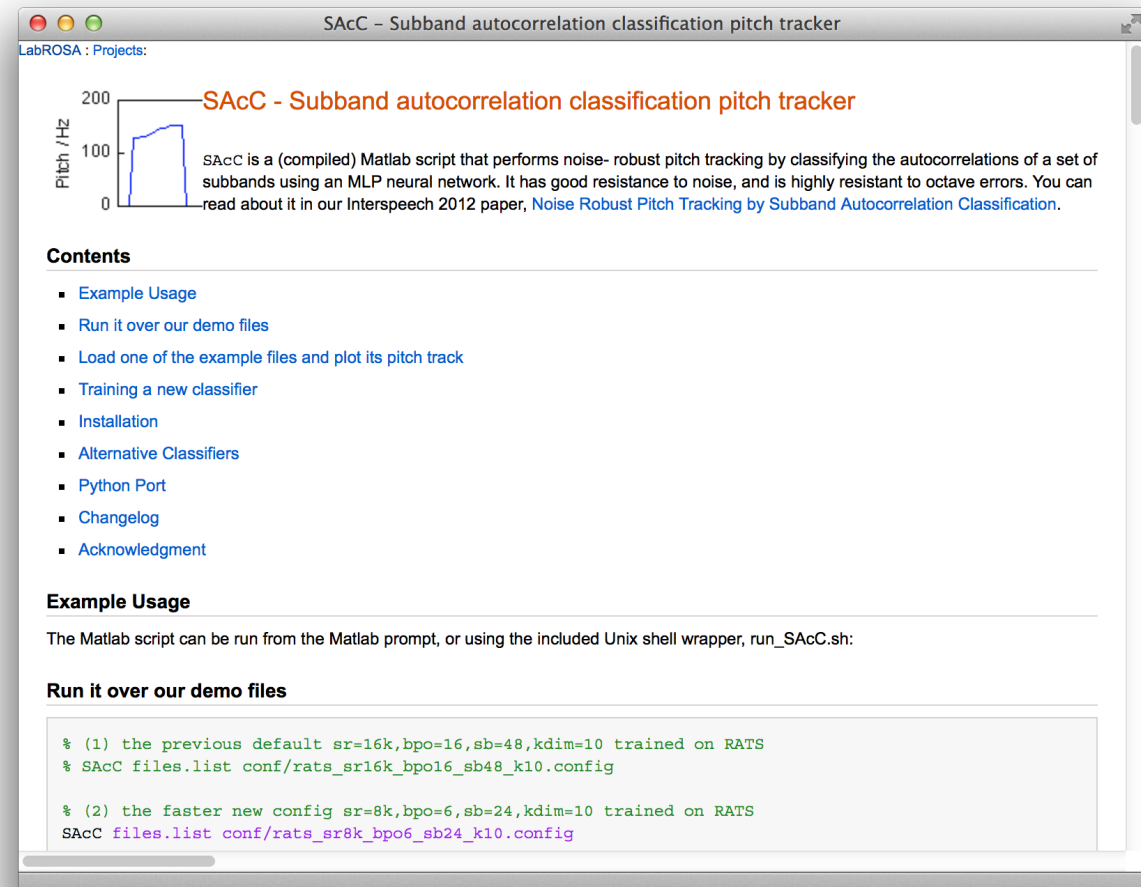
- Programming is **serious**
 - it can take much, much longer than necessary
 - getting hit by bugs is better than not noticing them
- Try to emulate a **professional**
 - even if you never plan to program professionally
- Learn by **doing**
 - i.e., the hard way

SAcC

- 2006: Student A re-implements a C-based system in Matlab
- 2010: Student B re-uses code to develop a new feature
- 2012: Feature is incorporated into DARPA program system
 - Industrial research lab expects consistent releases
 - Pressure to improve performance

SAcC

- Source release
- Version tracking
- Automated releases
- Automated tests
- Compiled target
- Python port



The screenshot shows a web browser window titled "SAcC - Subband autocorrelation classification pitch tracker". The page content includes:

- A plot of Pitch / Hz (0 to 200) showing a blue line representing a pitch track.
- A description: "SAcC is a (compiled) Matlab script that performs noise-robust pitch tracking by classifying the autocorrelations of a set of subbands using an MLP neural network. It has good resistance to noise, and is highly resistant to octave errors. You can read about it in our Interspeech 2012 paper, [Noise Robust Pitch Tracking by Subband Autocorrelation Classification](#)."
- A "Contents" section with a list of links: Example Usage, Run it over our demo files, Load one of the example files and plot its pitch track, Training a new classifier, Installation, Alternative Classifiers, Python Port, Changelog, and Acknowledgment.
- An "Example Usage" section stating: "The Matlab script can be run from the Matlab prompt, or using the included Unix shell wrapper, run_SAcC.sh:"
- A "Run it over our demo files" section with a code block showing two example configurations:

```
% (1) the previous default sr=16k,bpo=16,sb=48,kdim=10 trained on RATS
SAcC files.list conf/rats_sr16k_bpo16_sb48_k10.config

% (2) the faster new config sr=8k,bpo=6,sb=24,kdim=10 trained on RATS
SAcC files.list conf/rats_sr8k_bpo6_sb24_k10.config
```