Keep it simple stupid and other key ways of delivering a successful software project

Dr David Glance
Director UWA Centre for Software Practice
The Reality of Software Engineering
Alternatively...

A programmer is walking along a beach and finds a lamp. He rubs the lamp, and a genie appears. *I am the most powerful genie in the world. I can grant you any wish, but only one wish.*

The programmer pulls out a map, points to it and says, *I want peace in the Middle East.*

The genie responds, *Gee, I don’t know. Those people have been fighting for millennia. I can do just about anything, but this is likely beyond my limits.*

The programmer then says, *Well, I am a programmer, and my programs have lots of users. Please make all my users satisfied with my software and let them ask for sensible changes.*

At which point the genie responds, *Um, let me see that map again.*

http://www.cs.uni.edu/~mccormic/humor.html
Finally...

I'll need to know your requirements before I start to design the software.

First of all, what are you trying to accomplish?

I'm trying to make you design my software.

I mean what are you trying to accomplish with the software?

I won't know what I can accomplish until you tell me what the software can do.

Try to get this concept through your thick skull: the software can do whatever I design it to do!

Can you design it to tell you my requirements?
A software engineer is:
- Trained
- Understands the way in which software is developed from inception to maintenance i.e. the software lifecycle, the role of users, requirements, testing, deployment, more testing, management and refinement
- Understands and holds by a code of ethics and practice

Programmers simply code

You will hear about a range of other things that software engineering consists of *but* the basic principles are:
Basic Principles of Software Engineering

- Building software is hard, delivering useful software that is used is harder still.
- Software needs to actually be useful, otherwise it will never be used.
- Users may not know what they want or how to tell you.
  - The assumption that the user knows what they are doing is often wrong.
- Building a system that implements a bad process will end up being a bad system.
- Building a system that implements an analogue system is often no better than the existing process.
  - “Don’t pave the cow path.”
- Finding innovation in business process is rare (one reason for just using Common Off-The Shelf Systems (COTSS).
Different Types of Software Development

- Three types:
  - Bespoke: Users commission specific software
  - General: You write the software and then make it available to people to use
  - Bespoke $\rightarrow$ General

- Different issues involved
  - Bespoke – users have more control and you may end up simply doing what they want even if it isn’t right
  - General – you have to lead users in giving them something that they might not have known they needed

- Still hard in either case
Ways of mitigating complexity

- KISS (Keep it simple stupid)
  - Focus on the essential elements first before implementing bells and whistles
- Users often don’t know what they want until they find out so
  - Provide mock-ups
  - Deliver early and often and iterate (agile)
- Under promise and over-deliver
  - But watch out because if you set expectations you will be expected to always fulfill them
  - Be realistic about estimates
- Partition the problem into discrete apps
- Don’t try for perfection, you will never achieve it
Software Engineering Approaches

- Development teams have tried a range of approaches
- Classic Waterfall is still the predominant approach despite people pretending that they are doing Agile
  - Gather Requirements
  - Analysis and Modelling
    - Develop
    - Test
    - Deploy
- Agile involves faster iterations with rapid prototyping and deployment
  - Testing is done on users “good enough” software
Mobile Apps

- Mobile apps work because they have simplified the functionality of their desktop counterparts.
- They decompose the problem down into different parts.
  - E.g.
  - Facebook, Messenger, Groups, Instagram
  - Music, Podcast, Messages, News, etc.
- Key feature is that they made the software fun to use.
Build it and they will come?

- Self eHealth: Health and Activity application used for running UWA Self eHealth Challenge
- 330 active users
Web use
Mobile Use

All Time:
- 184 App Store Views
- 145 App Units
- $0 Sales
- 1,009 Sessions

Last 30 Days:
- 42 App Store Views, +31%
- 18 App Units, -14%
- 0 In-App Purchases
- 0 (Daily Average) Paying Users
- $0 Sales
- 149 Sessions, -63%
- 22 Active Devices, -12%
- 0 Crashes

App Units by Territory:
- Australia: 12
- United States: 2
- China: 1
The life of an app developer

- App development is a long tail – a few make money, the majority don’t
- Users have expectations about price and what should be free
- To pay 1 developer salary of $60,000 a year, you need about $120,000 of income
- A $1 app would need to be downloaded 182,000 times (Apple and Google take a cut) a year to cover that salary
- Since most apps don’t get used (16% of people try an app more than 2 times) ([http://www.digitaltrends.com/mobile/16-percent-of-mobile-userstry-out-a-buggy-app-more-than-twice/](http://www.digitaltrends.com/mobile/16-percent-of-mobile-userstry-out-a-buggy-app-more-than-twice/))
- Not much chance of revenue through advertising
So you are about to do a project

- Listen to your customer
- Remember that they are always right – even when they aren’t – they are the ones paying you
- Strive for simplicity and elegance – so suggest doing a few things well is better than doing lots of things badly
- Don’t pave the cow path
- Create prototypes or mockups that users can see early
- Try and test although you won’t be good at this
- Be professional – appearances do count