Sustainable Career

George Dinwiddie
Software Development Consultant and Coach
iDIA Computing, LLC
http://idiacomputing.com/
http://blog.gdinwiddie.com/

What makes a sustainable career as a software developer?

- To be able to satisfy the needs of those requesting the software
- To be competent at designing good solutions, even as implementation languages and deployment environments evolve.

Satisfying needs

- Understanding the needs
  - Listening well
  - Bridging communication gaps
- Splitting those needs into small pieces of functionality – User Stories
- Verifying those pieces of functionality
  - Against your own understanding
    - unit or programmer tests
  - Against the needs of those requesting
    - acceptance or customer tests

Competent at Good Solutions

- Much more than knowing programming language and frameworks!
- A sense of what makes a good design
  - Agile 2007 “Design Sense” session
    - by Michael Feathers and Emmanuel Gaillot
- Techniques that help you reliably produce the designs you know are good

Knowledge to Sustain your Career

- Continuous, Life-long Learning
- Principles over Specifics
- Heuristics over Recipes
- Learn more than one way to do things
  - The Law of the Hammer
  - The Rule of Three

What makes a good design?

- Many things can be derived from the twin concepts of Coupling and Cohesion
    - Larry Constantine – mid-1960s
- Cohesion
    - Each piece does **one thing** well
    - Extract Method, Extract Class, Move Method
- Lack of Coupling
    - Pieces are independent of each other
    - Dependency Inversion Principle

More good design ideas

- Layering by level of detail/abstraction
- Encapsulation
- Parallelism – Open and Close together
- Conceptual Integrity
  - metaphor, domain driven design
- Avoid code smells
  - Duplication
  - Insufficient Abstraction – Naked Primitives

Kent Beck's rules for Simple Design

1. Runs all the tests
2. Contains no duplicate code
3. Expresses all the ideas the author wants to express
4. Minimizes classes and methods

evaluated in this order

Advantages of Simple Design

- Robustness
  - "*There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies.*" - C. A. R. Hoare
- Accomodates Change
- Easier to Understand

“Simple” and “Easy” are different things

Two ways to Construct Software

- Design and then Implement it
  - Advantage for staffing:
    - A smart architect directs
    - A bunch of cheap, mediocre coders
- Grow it
  - Bit by bit
  - Verifying as you go
  - Evolving over time
  - Continuously refining the design

Tools for Growing Software

- Test Driven Development
- Refactoring
  - *Refactoring: Improving the Design of Existing Code* by Martin Fowler
- Design Patterns
  - *Refactoring to Patterns* by Joshua Kerievsky
- Design Principles
  - “Design Principles and Design Patterns” by Robert C. Martin (http://objectmentor.com/resources/articles/Principles_and_Patterns.pdf)

TDD of new code

- Many people wonder, “How can I write a test for code that doesn't exist?”
- I like to start with the story...

First Story:

Accept a credit card for payment of an order. A charge for the amount of the order should be sent to the merchant bank.

second story:

accept cash payment for exact amount of order.

- merchant bank should not be notified.
- change should be calculated as $0.00

3rd Story:

Accept payment greater than the order amount.

- Change should be calculated correctly.

Legacy Code

- It's much more common that greenfield projects
- I've see greenfield projects turn into legacy in a couple of months
- Better learn how to cope with it
- Book: *Working Effectively with Legacy Code* by Michael Feathers

Let's look at some code I found...

the bottom line...

Take Care of Your Career

It's got to last you the rest of your working life.

Further Reading

- Bridging Communication Gaps

  http://blog.gdinwiddie.com/2007/01/02/book-report-communication-gaps-and-how-to-close-them/

- Design Sense session

  http://beautifulcode.oreillynet.com/2007/08/design_sense.php

  http://www.agile2007.org/agile2007/index.php?page=sub/&id=816

Further Further Reading

- Design Principles

  http://objectmentor.com/resources/publishedArticles.html
- Refactoring

  http://refactoring.com/catalog/
- Design Patterns

  http://www.hillside.net/patterns/onlinepatterncatalog.htm