Chapters 5-6-7

Supplementary Notes
Requirements Engineering

• **Systems vs Software Requirements**
  – Systems Requirements cover computing operational needs
    • Hardware Specs: Operating system, RAM, HD, Net, Peripherals, etc.
    • Environment/Operational Specs: Browser, Web Server, JVM, etc.
    • Auxiliary support applications: Word, PDF reader, Flash, etc.
  – Software Requirements cover implementation specifications
    • All of the inputs, transformations and outputs of the product
    • Tools, data structures, algorithms, etc.
    • Real examples of each, provided by the customer = functional requirements

• **Requirements in general include**
  – Everything you need to know to deliver a working product
  – Enough information to develop a product that passes the customer’s acceptance test
  – Enough information to create tests that prove you met customer’s goals
Joel Spolsky: Painless Functional Specs

• Two-part review of importance and value of functional specs
  – Part 1: Why Bother?
    • http://www.joelonsoftware.com/articles/fog0000000036.html
  – Part 2: What’s A Spec?
    • http://www.joelonsoftware.com/articles/fog0000000035.html

*Much of the content from next few slides is based on these two articles*
Why Bother?

“failing to write a spec is the single biggest unnecessary risk you take in a software project”

• Key Benefits
  • Designing programs ahead of time saves time, improves quality
  • Improves communication and saves rewrite time
  • Enables realistic scheduling
  • Let’s you know when you are done!

• Ways and Means
  • Use plain language
  • Everyone, customer and programmer, should know what it means
  • If there is more than one way to interpret the sentence, it needs to be rewritten
  • As comprehensive as possible

• Example
  • Frankie’s GUI for the Pentagon
  • Medical software written in Java for embedded systems that had no JVMs
  • CD Baby: 2 years for Jeremy/Rails 2 months for Derek/PhP (but it’s not the story you think it is):
What’s A Spec?

• Technical Specifications
  – Tech Specs are more like the systems & software specs on slide 2
  – Often covers things like dev tools, data structures, algorithms, etc.

• Functional Specifications
  – What we are mainly concerned with for our projects
  – Specifies how a product will work
  – Lists screens, menus, inputs, outputs, etc.
    • Simplest description possible
    • No fancy words or complex explanations. Compare these two “specs”:
      – Assume a function AddressOf(x) which is defined as the mapping from a user x, to the
        RFC-822 compliant email address of that user, an ANSI string. Let us assume user A and
        user B, where A wants to send an email to user B. So user A initiates a new message using
        any (but not all) of the techniques defined elsewhere, and types AddressOf(B) in the To:
        editbox.
        – Miss Piggy wants to go to lunch, so she starts a new email and types Kermit’s address in
          the "To:" box. {Technical note: the address must be a standard Internet address (RFC-822
          compliant.)}
  – Review the example Spolsky gives:
    \[\text{http://www.joelonsoftware.com/articles/WhatTimeIsIt.html}\]
Fact/Fallacy Tidbit

• Fact 25

  Missing requirements are the hardest requirements errors to correct

• Discussion
  – Requirements come from people-to-people communication
  – Therefore naturally error-prone
  – Missed requirements = missed logic, potentially affecting all aspects of the delivered product
    • Easy to find an error that is in existing code
    • Hard to find an error in code that doesn’t exist!

From Robert Glass, “Facts & Fallacies of Software Engineering”