# Functional Specs

CSE 403, Winter 2003

Software Engineering

http://www.cs.washington.edu/education/courses/403/03wi/

# Readings and References

- References
  - » *Painless Functional Specifications*, Joel Spolsky
    - http://www.joelonsoftware.com/printerFriendly/articles/fog0000000036.html
  - » *Anchoring the Software Process*, Barry Boehm, USC
    - http://citeseer.nj.nec.com/boehm95anchoring.html

# Elements of Lifecycle Objectives (LCO)

| | |
|---|---|
| • Operational Concepts | What is it? |
| • System Requirements | What does it do for us? |
| • System and software architecture | How? |
| • Lifecycle plan | Who wants it? Who'll support it? |
| • Feasibility Rationale | Is this really true? |

# System Requirements

- Essential features of the system
  - » defined at a level appropriate to the spin cycle
  - » capabilities, interfaces, reliability levels, appearance
  - » Easy to change early on, grows increasingly more difficult
- Customer's involvement very important
  - » they know the domain of interest far better than you do
  - » what fits with their daily work and life patterns
  - » what might the future bring
- Neither you nor the customer know everything
  - » try to build joint ownership of the process
  - » open communication can make change more acceptable

# System and Software Architecture

- Sufficient detail to support feasibility analysis
  - » multiple viable choices is great at this stage
  - » people lock on to a particular architecture *very* quickly and get attached to their perceived piece
- If you can’t define an architecture that seems to make sense, don’t ignore the problem
  - » Basic data flow or performance problems will kill a system, no matter how many features it has
  - » Rethink why and for whom you are doing this

# Risk Reduction

- “Failing to write a spec is the *single biggest unnecessary risk* you take in a software project”
  - » Joel Spolsky
- The act of writing the spec -- describing how the program works [from user perspective] in minute detail -- will force you to actually design the program
  - » you get a chance to see the potholes before you fall in
  - » you get a chance to back up and change your mind before you’ve written thousands of lines of code

# Once in motion, ideas stay in motion

- People get attached to their creations
  - » if it’s just a paragraph or two, it’s easy to change
  - » if it’s pages and pages, it’s hard to change
- Nobody wants to throw out hard work
  - » even if the problem it solves is now irrelevant!
  - » it feels like criticizing, instead of discussing
- Architects get blinders very quickly
  - » if we take that approach, then my group won’t be needed at all on this project ∴ that’s a bad approach

# Specs as Communication Support

- It’s always amazing how:
  - » people hear and remember some things
  - » people hear and don’t remember other things
  - » two people hear exactly the same thing and remember something completely different
- Write stuff down, then point to it when needed
  - » single source of information
  - » fantasy reduction benefits are huge
    - but remember, specs are a tool, not a magic elixer

# Face the problems early

- Writing an outline of program features makes you think about the high level areas of interest
  - » Can’t overlook major functional areas
- Writing the details makes you think about how you are going to do these things
  - » Can’t overlook major architectural defects
- While you’ve still got time, you can toss the early architecture and replace it completely

# What’s in the spec?

- An author
  - » Take responsibility for your work
- Scenarios
  - » Let the customers see these ideas in action
- Non-goals
  - » Eliminate the “implied” goals
- Overview
  - » Elevator pitch with a drawing or two

# What else is in the spec?

- Details of operation from user perspective
  - » what's it look like to the various users
  - » what happens during overload, weekends
  - » general performance parameters
  - » typical equipment requirements
- Open issues
  - » state them explicitly
- Side notes
  - » for different reader communities

# Spolsky's Rules for Writing

- Be funny
  - » be specific, people love it and will discuss it
- Be understandable
  - » a customer who understands will help you succeed
- Write as simply as possible
- Review and reread
- Templates considered harmful
  - » an entry to fix every oversight in the last 5 years