Writing a System Specification

• “Partner” with the customer in developing a system specification
  – let the customer guide this process
  – the customer sets the groundrules for success
  – non-computer-ese
  – problem domain
  – no loyalty to a solution

Writing a software requirements specification (SRS)

• Your contract with the customer
• “This is how we see that software can solve your problem”
• Do away with, or state, your assumptions
• Set the stage for financial concerns:
  – progress payments

Managing the Customer

• “We want the problem solved THIS way.”
• “We don’t like your design. We would prefer you do it this way…”
• “We don’t like the steps you took in your source code. We would prefer you do it this way…”
then when it didn’t work …

• “You’re the expert, why didn’t you talk us out of it?”
Managing the Customer

• “We’re adopted a corporate policy where we only pay 85% of our invoices. Sue us.”
• “Since you changed our network, our copier keeps getting “out of toner” errors.”
• “We don’t want you to finish the project, we only needed you for design. We’ll pay you for what you did, but you won’t be needing all that staff you hired.”

Managing the Customer

• “We need some technology. How much will it cost?”
• “Is your software “information superhighway” compatible?”
• “Is your software Y2K compliant?” (this was April, 2002)

Managing the Customer

• “Requirements spec? We don’t actually like to write anything down. We’re afraid of industrial espionage.”
• “Give us a quote for $24K just to get the purchase order signed, and you can add more later.”
• “We can’t agree to a contract to do the job until the job is completed.”
Managing the Customer

• “You’re not allowed to talk to the users. Only we can talk to the users.”
• “The users aren’t happy so we’re not paying you.”

Managing the Customer

• Never lie - about schedule, budget, capabilities; to win the job, keep the job, or enhance your standing.
• Keep the customer appraised relentlessly - every week, every change, every problem, every success.

Managing the Customer

• Decide - as an expert - where flexibility is needed and design it in. Do not design flexibility or generality where it is not needed. Remember, flexibility is expensive. Better to implement incrementally and absorb changes slowly.
Managing the Customer

• Educate your customer - about software development and turning his problem space into your solution space.
• Learn from your customer - about his problem space.
• Sympathize with your customer

Managing the Customer

• Never say no, just tell how much it will cost.
• Recognize these signs of customer manipulation:
  _ Refusal to develop the requirements combined with a refusal to budge from a fixed price.
  _ Refusal to budge from a fixed price in the face of a changing spec.

Managing the Customer

• Take extra care with the user and user interface. That is the only viewport into your system. Customers rarely view source code.
Example of requirements creep - structure chart and a one-paragraph description of level-two modules

A PC is to sit on an Ethernet with access to TCP/IP devices, and collect statistical data on devices within its domain. The devices are simple routers and servers, and each contain 20 parameters that can be queried by the PC. The PC has one screen: the user can specify a device (by TCP/IP address), and the PC will display the value of the 20 parameters, e.g. Number of retransmissions = 46.
requirements creep

After requirements, after design, during coding:

• Change 1: The customer forgot to tell you that the 20 parameters are in the device’s volatile RAM, so the PC must query each device every 15 minutes and keep the values in it’s own memory in the event the user will need them after the device is turned off.

• Change 2: the PC must keep a complete history of each device (all 20 parameters, logged every 15 minutes).

requirements creep

• Change 3: display trends (charts) for some of the parameters over time. Let the user request which ones.

• Change 4: parameters grow to 75.

• Change 5: some of the 75 parameters are run-time values that the user must be able to change.

requirements creep

• Change 6: command each device to reset.

• Change 7: peek and poke into every address of the device’s memory.
The Sensitive Issue of Money

• A Salary of $60K/year –
  – $30/hr. salary
  – $15/hr. overhead (burden)
  – 5% profit add $2.25/hr
  = $47.25/hr.
• x 2000 hrs/year = $94,500 per year just to keep you working and busy

The Sensitive Issue of Money

• Labor is the most expensive product component
• Companies go out of business when unprofitable
• Keep issues of money open and out front
• Your customer is trying to enhance his profit at your expense

The Sensitive Issue of Money

• Track and present costs as importantly as schedule.
• Work out a schedule of progress payments based on identifiable milestones.
• Stop work if payments stop.
• “Time & Materials” is always better than “Firm Fixed Price”
The System Spec must answer…

- What is the domain?
  - What is the deployment environment?
- What are the requirements?
  - can they be listed?
- Are there any Customer-Constraints
  - budget
  - schedule
  - technical
  - environmental

The SRS must demonstrate…

- Results of your analysis
- A proposed solution
  - System architecture in block diagram
- Budget/Schedule boundaries
- A proposed methodology
  - means to demonstrate and track progress (integration thread?)
- Known problem areas
- Measures of Success
- The “future” of the system:
  - enhancements, maturity