Pragmatic Programmer Tip: Don’t Gather Requirements – Dig for them

Requirements rarely lie on the surface. They’re buried deep beneath layers of assumptions, misconceptions, and politics.
Resources

- "Rapid Development", Steve McConnell
  - Chapters 10, 14 (required)
- "Pragmatic Programmer", Hunt, Thomas
  - Chapter 7 (recommended)
- "Software Project Survival Guide", Steve McConnell
  - Chapter 8 (optional)
- Software Requirements Specification Template, Use case papers, (on class web, Resources link)
Outline

- What are requirements?
- Some interesting requirements facts
- How can we gather requirements?
- How can we specify requirements?
- Beware of scope creep
What are requirements?

- % def requirement
  n, something wanted or needed: necessity

- Requirements are *features*
  necessary to deliver with the product

- Requirements are necessary *attributes* of the product
Brainstorm!

What types of software project requirements can you think of? General categories …

Examples requirements types:
• Feature set
• GUI
• Performance
• Reliability
• Expansibility (ie. support plug ins)
• Environment operates in (ie. HW, OS, browsers)
• Schedule
How do we gather requirements?

Let’s start with two facts:

- Standish group survey of over 8000 projects, the number one reason that projects succeed is user involvement.
- Easy access to end users is one of three critical success factors in rapid-development projects (McConnell).
How do we gather requirements?

Is the answer obvious?

Why work with customers?
- Good relations improve development speed
- Improves perceived development speed
- They don’t always know what they want
- They do know what they want, and it changes over time
The most difficult part of requirements gathering is not the act of recording what the users want; it is the exploratory, development activity of helping users figure out what they want.

McConnell, SG
Words of Wisdom 2

Work with a User to Think Like a User – it’s the best way to get insight on how the system is easily used

Pragmatic Programmer Tip
How can we work with our customers?

What can we do during the lifecycle stages of:

- **Planning**
  - select lifecycle
  - identify real customer
  - establish interaction method

- **Requirements Analysis**
  - help customer determine what they want (ie.prototypes)
  - videotape customers operating
  - surveys, meetings, focus groups, discussions

- **Design**
  - design for change

- **Construction**
  - implement to allow change
  - show customer tangible signs of progress, phased delivery allowing feedback
And the results we expect?

- Improved efficiency
- Less Rework
- Reduced Risk
- Lack of friction

Why?
Throughout your travels with the customer, be sure to *set reasonable customer expectations*

Why is this important?
Outline

- What are requirements?
- Some interesting requirements facts
- How can we gather requirements?
- How can we specify requirements?
- Beware of scope creep
How can we specify requirements?

So… we’ve worked with the customer to understand their needs, how do we capture these requirements?

Possibilities include:
- Prototype
- Requirements Specification Document
  - Use Cases
  - Feature List

Ideas?
Prototype

- Build a prototype to capture requirements
- Extend until it demonstrates all the functional areas of the system. Broad but shallow.
- Develop a style guide that codifies the proto’s look and feel
- Proto is a baseline spec OR
  Can write detailed end-user doc based on proto, which becomes software spec
Use Cases

- Capture a particular use of the system
- Describe how external agents (actors) interact with the system (use case)
- Diagrams or textual description

Online ordering system

Use cases
- Browse Catalog and Select Items
- Call Sales Person
- Give Shipping Info
- Give Payment Info
- Get Confirmation #
Another use-case example

*http://odl-skopje.etf.ukim.edu.mk/uml-help/html/02day12.html#ovde*
Figure 7.1. Cockburn’s use case template

A. CHARACTERISTIC INFORMATION
   - Goal in context
   - Scope
   - Level
   - Preconditions
   - Success end condition
   - Failed end condition
   - Primary actor
   - Trigger

B. MAIN SUCCESS SCENARIO

C. EXTENSIONS

D. VARIATIONS

E. RELATED INFORMATION
   - Priority
   - Performance target
   - Frequency
   - Superordinate use case
   - Subordinate use cases
   - Channel to primary actor
   - Secondary actors
   - Channel to secondary actors

F. SCHEDULE

G. OPEN ISSUES
### Example buy goods use case

<table>
<thead>
<tr>
<th>Characteristic Info</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Goal</td>
<td>Buyer issues request to buy product, expects delivery</td>
</tr>
<tr>
<td>- Preconditions</td>
<td>We know buyer address</td>
</tr>
<tr>
<td>- Success end condition</td>
<td>Buyer has goods, we have $</td>
</tr>
<tr>
<td>- Failed end condition</td>
<td>No goods to buyer, no $ to us</td>
</tr>
<tr>
<td>- Primary actor</td>
<td>Buyer</td>
</tr>
<tr>
<td>- Trigger</td>
<td>Purchase request comes in</td>
</tr>
<tr>
<td>Main success scenario</td>
<td>...</td>
</tr>
</tbody>
</table>
Feature List

List of features together with a brief description of their function

WikiMedia Index to Feature List

1 Look and feel
2 Multimedia and extensions
3 Keeping track of edits
4 Structures and syntax
5 Editing
6 Discussions
7 Multilanguage support
8 Backend
9 Permissions
10 Search and Queries
11 Misc.
12 Empty set of help pages
13 Coming soon
Pulling it all together

How much is enough?

What are problems with over specifying?
What are problems with underspecifying?

You have to find a balance

• comprehensible vs. detailed correctness
• graphics vs. explicit wording and tables
• short and timely vs. complete and late
Organize your specification by viewpoint or category of requirements

Example:

- Administrative functions
  - New account
  - Change password
- Customer functions
  - Retrieve data
  - Edit data
  - Publish
  - Collaborative
- Performance
- Reliability
Words of Wisdom 5

After you create a specification, go over it to:

- Eliminate all requirements not absolutely necessary
- Simplify those that are more complicated than necessary
- Substitute cheaper options when available
- Move non essentials to future releases
Outline

- What are requirements?
- Some interesting requirements facts
- How can we gather requirements?
- How can we specify requirements?
- Beware of scope creep
Scope Creep

“The software was late and far over budget; in fact, it almost didn’t make it out the door. And it bore little resemblance to their original plans… Most software-development stinks”

Wall Street Journal

Our analysis found that the average requirements overrun on our projects is about 40%

Construx
How can you manage this?

Two strategies:
- Scope change document (nothing is free)
  - Analyze cost, impact, make tradeoffs
- Change control board

Your thoughts?
DILBERT

by

Scott Adams

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?

© Scott Adams, Inc./Dist. by UFS, Inc.