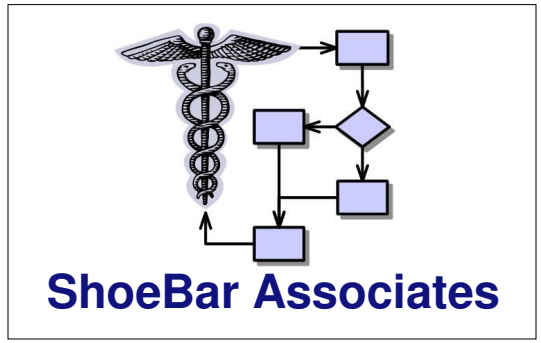
How Do We Capture Software Requirements?

Brian Shoemaker, Ph.D.



Disclaimer

I am not a professional software developer. The material I present is drawn from my own experience and observations, from reading, and from presentations I have attended. I make no claim that what I present is everything you need to know about requirements gathering – but I hope to present enough material that some part of it will be interesting, informative, or useful to you.

How Do We Capture Software Requirements?

What are requirements, and why write them?

What characterizes ***good*** requirements?

How do we find out the ***right*** requirements?

How do we translate requirements into tests?

Do requirements tools help?

Where can we turn for help on developing requirements?

Consider:

Have you ever tried driving to a new destination without a map?

“If you don’t know where you’re going, how will you know you got there?”

“A sailor without a destination cannot hope for a favorable wind.”
- Leon Tec, M.D.

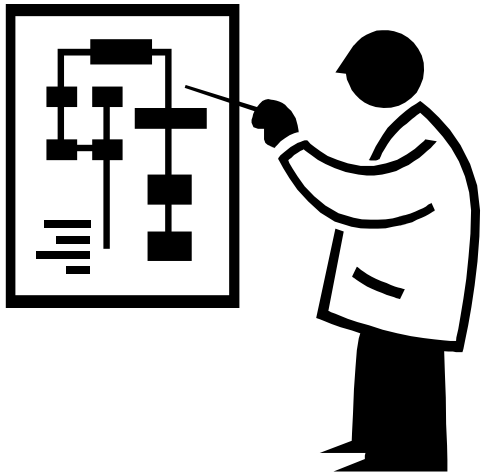
Why write requirements?

- Does everyone involved agree on what the product should do?
- We may have different purposes for requirements:
  Developing software
  Providing input to select OTS software
  Establishing basis for validating existing software
- Requirements fulfill needs from different sources:
  contractual
  marketing
  regulatory

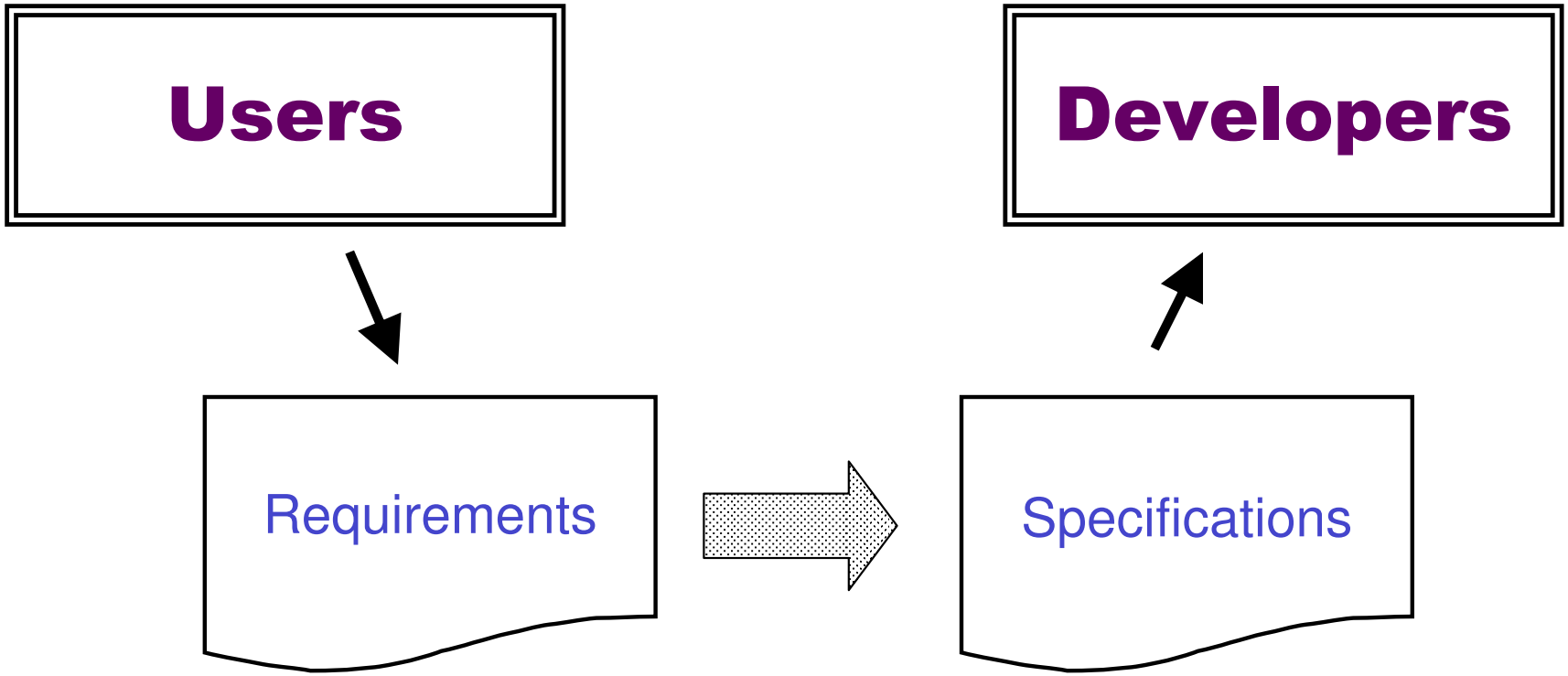
Careful of the Word “Requirements”

Wiegers talks about three levels of requirements (http://www.processimpact.com/articles/reqtraps.html):

- *Business requirements* - high level objectives of the organization or customer
- ***User requirements*** - description of the tasks a user must be able to perform by use of the product
- *Functional requirements* - specific behaviors the software needs to exhibit



Requirements vs. Specification



May not apply if you purchase OTS software – until you customize it.

How Do We Capture Software Requirements?

What are requirements, and why write them?

What characterizes *good* requirements?

How do we find out the ***right*** requirements?

How do we translate requirements into tests?

Do requirements tools help?

Where can we turn for help on developing requirements?

Requirements gathering: learn from the experts

IEEE 830: “Recommended Practices for Software Requirements Specifications”

Characteristics of an SRS:

- Correct *(every item is one S/W shall meet)*
- Unambiguous
- Complete
- Consistent *(i.e. internally)*
- Ranked for importance / stability
- Verifiable
- Modifiable *(structured, not redundant)*
- Traceable

Test your requirements

- Measurable: Quality measure to gauge any solution
- Coherent: Definition of every essential subject matter term
- Consistent: Every use a defined term consistent with definition
- Complete: Wide enough context of the requirements;
  included conscious, unconscious, undreamed of requirements
- Relevant: Every requirement relevant to this system
- Solution Neutral: No solutions posturing as requirements
- Ranked: Stakeholder value defined for each requirement
- Traceable: Each requirement uniquely identifiable;
  Each requirement tagged to all parts of the system where used

Requirements How-Not-To's

You don't need:

- A web interface
- Username / password login
- Radio button to select age group

You *DO* need:

- Access for users outside the LAN
- Access only to authorized users
- Only one age group assigned each patient record

Another How-Not-To

- Customized tool for MedDRA coding; constraint = implemented via Oracle Forms
- Function: manage list of “words to remove”
- Requirement: “XYZ shall strip leading and trailing blanks from words to remove”
- QA review stalled; tests didn’t show that XYZ stripped leading blanks
- Leading blanks couldn’t be entered!

How Do We Capture Software Requirements?

What are requirements, and why write them?

What characterizes *good* requirements?

How do we find out the *right* requirements?

How do we translate requirements into tests?

Do requirements tools help?

Where can we turn for help on developing requirements?

Have we considered what's most important?

- Who are the "actors"?
- What is the system's boundary?
- OTS software: which features are actually used?
- Both normal and exception cases included?
- Everything from user / actor point of view?
- Do we understand the user's job?

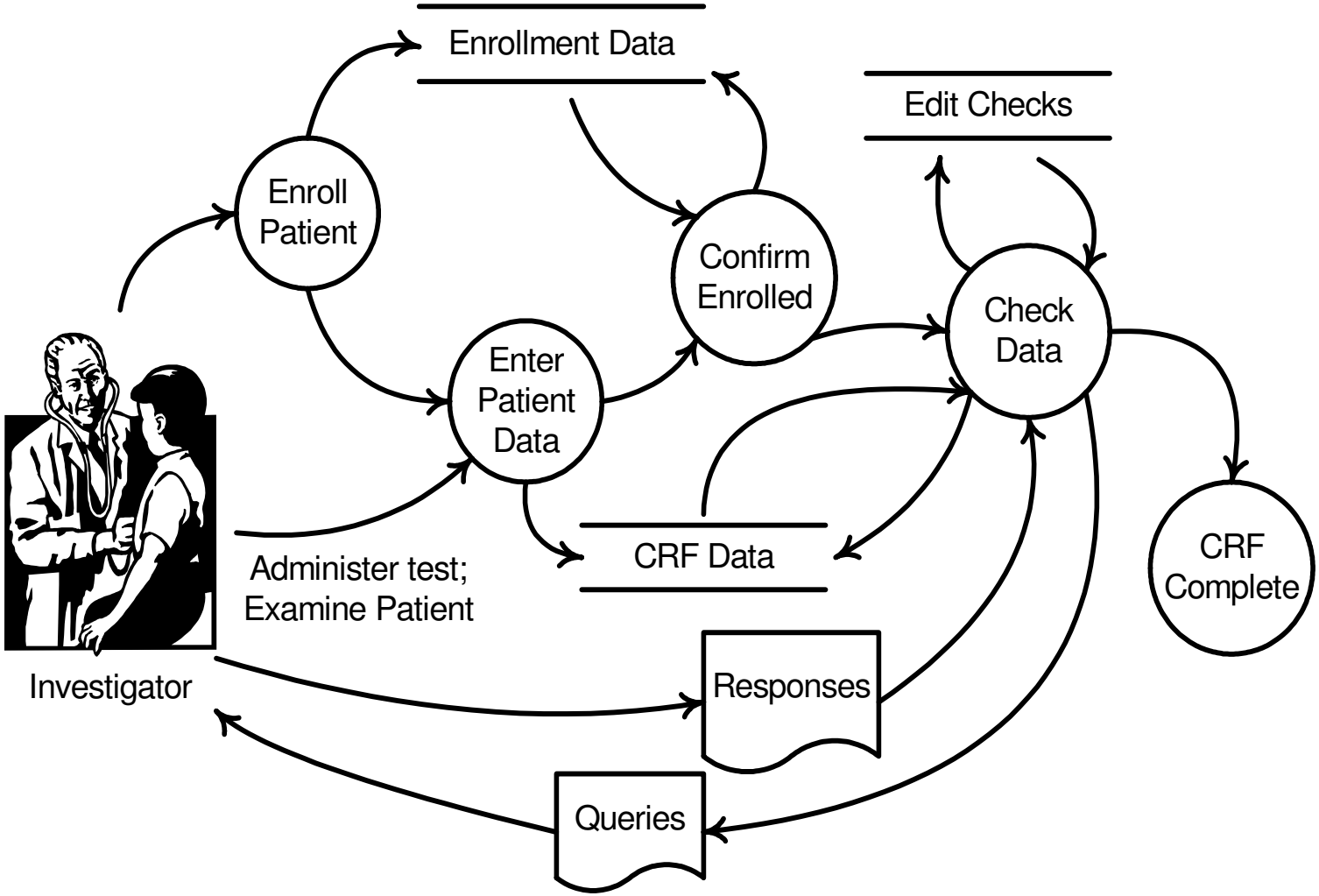
In Search of Requirements

Where to search for requirements?

Robertson and Robertson (software development) suggest:

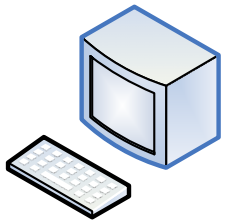
- Management
- Subject matter experts
- Developers
- **Inspectors**
- Market forces
- **Legal requirements**
- Opposition
- **Professional bodies**
- Public opinion
- **Government**
- Special interest groups
- Technical experts
- Cultural interests
- **Adjacent systems**

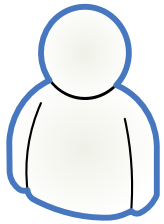
Review Process through Use Cases

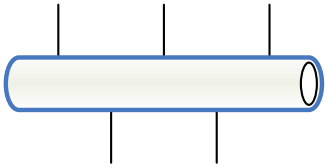


A Boundary Example

Pattern design software

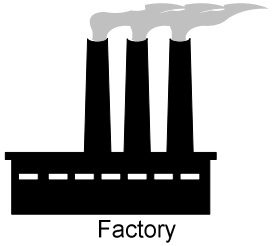


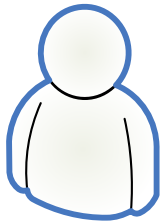




Network

Manufacturing equipment





What about prototyping?

- Low-fidelity (paper) vs. high-fidelity
- Similar approach: storyboarding
- Remember the purpose of a prototype

Requirements: go beyond the regulations

Consider the relative frequency of paragraphs containing certain terms in each regulation:

Term	GLP (Pt 58)	GMP (Pt 211)	QSR (Pt 820)
Document or *Documentation*:	19	8	50
Record:	22	50	26
Sign or *Signature*:	2	6	9
Approve or *Approval*:	9	21	18
Computer:	1	3	2
Software:	0	0	6

(Distinct occurrences only, not including subpart titles or multiple uses in the same paragraph.)

S/W Requirements in Pharma GMP (Part 210/211)?

- 211.68: "Computer" classed as equipment; controls mandated to assure integrity of data
  - changed only by authorized personnel
  - input/output checked for accuracy
  - backups to be maintained
- 211.80: Records accessible through a computer, though stored at some remote location, are considered accessible to an FDA inspector

S/W Requirements in Device QSR (Part 820)?

- 820.3 (Definitions): Software / firmware included in a device considered a "component;" S/W version can distinguish a lot or batch
- 820.30 (Design Controls): Class I devices automated through computer software subject to design controls; design validation includes software validation
- 820.70 (Production and Process Controls): Where computer systems used in production or quality system, S/W shall be validated for intended use, S/W changes shall be validated, validations shall be documented.
- 820.181: Software specifications included in Device Master Record

How Do We Capture Software Requirements?

What are requirements, and why write them?

What characterizes good requirements?

How do we find out the right requirements?

How do we translate requirements into tests?

Do requirements tools help?

Where can we turn for help on developing requirements?

Requirements to tests

- Empirical:
  Review the requirements for those that flow together.
  Make lists; write test sequences around them.
- Use Case based: turn each use case into a test
- Remember that each requirement is not only an action but an expected result.

How Do We Capture Software Requirements?

What are requirements, and why write them?

What characterizes good requirements?

How do we find out the right requirements?

How do we translate requirements into tests?

Do requirements tools help?

Where can we turn for help on developing requirements?

What would we need in a tool?

This is a requirements task in itself!

- Requirements text entry
- Tracking of requirements (as modified)
- Traceability (auto numbering)
- Configuration management

Also desirable:

- Direct linkage to tests
- Direct linkage to code
- Ability to store related materials (e.g. diagrams)

Tool lists I found

- Volere (sister site to Atlantic Systems Guild)
  http://www.volere.co.uk/tools.htm
- Tigris.org: open source s/w engineering tools
  http://requirements.tigris.org/
- Ian Alexander's web site
  http://easyweb.easynet.co.uk/~iany/other/vendors.htm

A thought to bear in mind

From http://www.Softwareprojects.org :
"There are several tools available that will make your life easier by automating the requirement management process. In the end, they can't do the job for you - as seen, the tasks are complex and require high level of abstraction and analytical thinking. But these tools can keep track of things where human memory fails, and can perform the steps that need to be repeated."

How Do We Capture Software Requirements?

What are requirements, and why write them?

What characterizes good requirements?

How do we find out the right requirements?

How do we translate requirements into tests?

Do requirements tools help?

Where can we turn for help on developing requirements?

My Own Picks

- Robertson, James and Suzanne, *Mastering the Requirements Process*, Harlow (England), Addison-Wesley / ACM Press, 1999.
- Robertson, Suzanne, "An Early Start to Testing: How to Test Requirements," http://systemsguild.com/GuildSite/SQR/Testreqs.html .
- Institute of Electrical and Electronics Engineers, "Recommended Practice for Software Requirements Specifications" (IEEE 830-1998), 01-May-1998.
- Schneider, Geri, and Jason P. Winters, *Applying Use Cases: A Practical Guide*, Harlow (England), Addison-Wesley / ACM Press, 1998.

Others

I have not read these, but located them in an online search.

- Requirements by Collaboration: Workshops for Defining Needs, by Ellen Gottesdiener (Addison-Wesley, 2002).
- Customer-Centered Products, by Ivy F. Hooks and Kristin A. Farry (AMACOM, 2001).
- Use Cases: Requirements in Context, 2nd Edition, by Daryl Kulak and Eamonn Guiney (Addison-Wesley, 2003).
- Requirements Engineering: A Good Practice Guide, by Ian Sommerville and Pete Sawyer (John Wiley & Sons, 1997).
- More About Software Requirements: Thorny Issues and Practical Advice, by Karl E. Wiegers (Microsoft Press, 2005).
- Software Requirements, 2nd Edition, by Karl E. Wiegers (Microsoft Press, 2003).

Of course, an Amazon search on "software requirements" nets many pages of results - one could spend weeks just reading their summaries.