Software Engineering
(CS550)

Requirement Engineering

Jongmoon Baik

Session Objectives

- Overview of requirements from the management perspective
- Definitions
  - – requirements, constraints, and quality attributes
- Requirements Elicitation
- Role of Analysis

Overview

Requirements engineering remains one of the most problematic aspects of software-intensive systems development.

"So what's all the hub-bub,.... bub?"

Bugs Bunny (1940 -)

US Cartoon Character

Difficulties of Requirements

What is so hard about requirements?

- Everything!!!
  - Finding requirements
  - Writing down requirements
  - Measuring compliance
    - Verification
    - Validation

The Problem

"The hardest single part of building a software system is deciding what to build….No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."

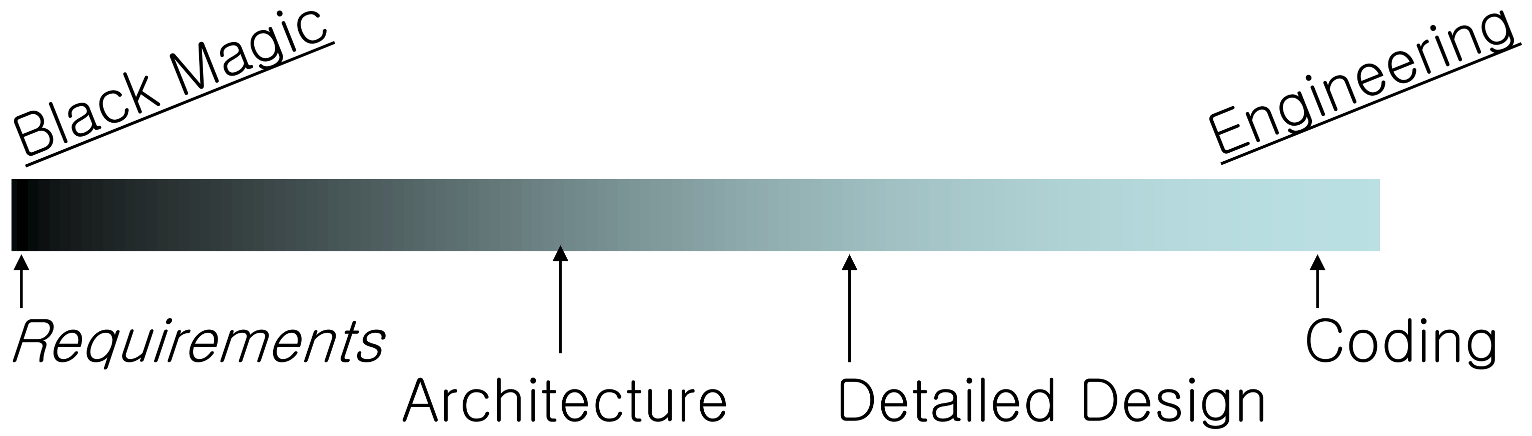
Frederick P. Brooks:
The Mythical Man-Month (2nd Edition), Addison-Wesley, 2001.

Difficulties of Requirements

- "You don't know what you don't know."
- Imprecise and tactile, as a result
  - difficult or impossible to codify some requirements
  - only weak methods available for requirements elicitation and analysis
- Requires excellent "people skills" as well as technical and programmatic expertise.
  - It's often hard to find one person and teams with these characteristics.

Observation #1

There is a d*egree of "magic" in some aspects of software development*



Observation #2

- *Requirements work is viewed as no fun*
  - *not technical enough for the "techies"*
  - *boring for those with people skills*
  - *demands endurance from the participants*
  - *there is little or no appreciation for the efforts of requirements teams*
  - *not interesting topic for researchers*

Perfect Requirements Team

- Management and Support Expertise
  - leadership, administrative, planners, financial, legal, database
- Technical Expertise
  - design, operational, domain
- Customer Representative
  - users, operations, buyers, management

- *Everyone would get along, everyone would stick it out through the entire requirements process…*

It Never Happens…

- The real system stakeholders are not identified and/or are not deemed "worthy" to be part of the team
- Team turnover: team members leave and new members have different ideas
- Management representation from provider and consumer is not involved
- Customer reps won't attend meetings

Definitions

Terms associated with the garnering, analysis, and management of requirements are terribly overused... let’s establish a baseline for conversation.

“Just definitions prevent or put an end to a dispute”

Nathanial Emmons (1745-1841)

US Congregational Pastor

Requirements Defined:

IEEE '90

1. A condition or capability needed by a user to solve a problem or achieve an objective
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents
3. A documented representation of a condition or capability as in (1) or (2)

Requirement…

- A requirement is a clear description of
  - the purpose the software is to serve
  - what the software must do to serve that purpose
- Raw requirements are often written by and/or from the standpoint of users
  - function oriented
  - informal and incomplete
  - must be refined

Constraints

- Designers and implementers do not have complete freedom to create as they please.
  - pre-made design decisions
  - rules, standards
  - budget, schedule

- These are *constraints* on how designers and implementers may satisfy the requirement.
  - constraints limit choices

Remember Business Desires

- To make money by:
  - Providing a product, or service
  - Cutting Costs

- For the betterment of mankind…?

Types of Requirements

- Functional (what)
  - "Input X produces Y"
- ~~Non Functional~~ (Christel and Kang)
  - **Quality Attributes**
  - Interfaces
  - Design constraints
- Implied (Very Nebulous…)

KAIST 한국과학기술원 Korea Advanced Institute of Science and Technology

Quality Attributes – 1

- Long ago, software engineers realized that structure is important in software.
- While many structures can satisfy functional needs, only a few can satisfy the combined demands imposed by
  - performance, availability, modifiability,.. and many others,...
- These things are *quality attributes*.

Quality Attributes – 2

- Why use the term *quality attribute* rather than *non-functional* requirement?
  - *non-functional* is a misleading term
    - Implies requirements can be cleanly divided into two nice neat halves.
    - Quality attribute properties are bound up inescapably with the functionality of the system.
    - Try to describe an "-ility" without describing functionality.

Requirements Engineering

- Systematic way of getting from need to specification – *must be planned*
  - elicit need
  - analyze
    - validate and quantify functional, quality attribute requirements, and constraints
    - set completion criteria
    - establish working agreement (**Statement of Work**)
  - document

Requirements Elicitation

The first step in any project is to find out what the client wants – it is often a treacherous step. Academics often ignore the topic of requirements elicitation and practitioners get cold sweats fretting about it.

“Father Brown laid down his cigar and said carefully: ‘It isn't that they can't see the solution. It is that they can't see the problem.’”

G.K.Chesterton

The Point of a Pin in *The Father Brown Stories*.

Stakeholders – 1

- A stakeholder is someone who will be affected in some way by the thing you build.

☐ users	☐ designers
☐ customers	☐ operators
☐ marketers	☐ maintainers
☐ developers	☐ managers

Stakeholders – 2

- Customers ≠ Users
  - A customer
    - pays for the product
    - must be convinced of the value of the product determined by cost and benefit to their organization
  - A user
    - lives with the product
    - must perceive the utility of the product determined by the overall benefit to them

Requirements Elicitation:

- If you do not meet the needs of the true users, your product will fail.
- The first task of requirements elicitation is to identify those users.
- If you do not have access to the true users, you will not determine their needs.
- It is risky to do requirements elicitation at arm’s length or through intermediaries.

Identifying Users – 1

- A user is anyone who
  - will interact with the product regularly
  - will change the way they work because of the product
  - will have to create input or use output from the product

Identifying Users – 2

- Prioritize **ALL** stakeholders and their needs.

- Users are more important if
  - there are more of them
  - they use the system more intensively
  - they use it under stress
  - their mistakes are more costly

Eliciting the Needs

- Eliciting the user is the first step, but also the most problematic, because mistakes
  - – propagate indefinitely
  - – are very hard to detect
  - – are very hard to repair

Especially quality attribute oriented mistakes!

- Critical issues
  - – what do the users need to do (function)
  - – how would they like to do it (quality attribute)

Problems

- Some major issues encountered when you elicit requirements include
  - inarticulateness
  - terminology
  - hidden assumptions
  - preconceived solutions

“You don’t know what you don’t know.”

Inarticulateness – 1

- Many stakeholders (especially users) cannot explain what they do or what they need. They
  - remember the exception, and forget the routine
  - underemphasize the prominence of simple stuff
  - focus on what doesn't work, not what does work

- One articulate user can mislead other stakeholders and build a false consensus.

Inarticulateness – 2

- Quality attribute requirements are the most difficult for stakeholders to articulate.
  - One word descriptions such as "modifiable" are meaningless.
    - How do you measure modifiability?
    - How important is modifiability with respect to other quality attributes?
  - One stakeholder's modifiability is another stakeholders scalability.

Guidelines

- Technical/business feasibility
- Stakeholders
- Environment
  - Architecture / constraints
- Elicitation models
- Identify candidates for prototyping
- Use scenarios

Sommerville and Sawyer '97

Universal Starting Points

- Solution without a problem to solve
  - The "why"
- Solution looking for a problem
- Simile – Team focus
  - Metaphor
  - Project name
- Norm
  - Mockup – Front end prototype

Gause and Weinberg '89

Starting point:

Research the Company and Problem

- Company
  - Focus
  - Vision
  - Business drivers
  - Core competency
- Past problems
  - Specification
- In house expertise

Potential Solutions – 1

- Observation
  - Watch users work (covertly and overtly)
- Interviews with key stakeholders
  - Research interviewee?
- Logging
  - Have users write down what they do as they do it
  - Have them log their time on task
- Develop use case and quality attribute scenarios
  - One describes function one describe quality attribute characteristics

Potential Solutions – 2

- Use cases describe required *functionality*.
- Quality attribute scenarios describe required *quality attribute properties*.
- Which modifiability requirement is more meaningful?
  - *"The system shall be modifiable."*
  - *"Modify the system to utilize a different COTS discrete event generation package in 12 staff months."*

Terminology

- Stakeholders (especially users) have a different vocabulary from that of designers and developers.
  - Special cultures result from special terms used in special ways.
  - Developers must understand terms in the context of the stakeholders and their work
    - critical to understanding stakeholder needs

Potential Solutions

- Domain expert
  - Enlist a domain expert that also has a knowledge of software engineering
- Domain dictionary
  - Build a dictionary of key technical terms and their definition before you elicit requirements
- Domain training
  - Train software engineers in the domain or vice-versa

Hidden Assumptions (Implied)

- The stuff "everybody knows" often goes unstated.
  - The obvious may not be obvious to those lacking domain expertise.
  - A system that violates critical assumptions will fail.

- No matter how obvious, critical assumptions must be explicitly stated and recorded.

Potential Solutions

- Observation (again watch them at work)
- Use Cases
- Role playing
  - have non-experts walk through key use cases and test them for completeness
- Prototypes
- Formal analysis
  - cast requirements into a formal specification that can be rigorously checked for completeness

Preconceived Solutions

- Some stakeholders think they know the answers to their problems.
  - Sometimes they describe their idea of a solution, not the problem
    - *"Just write the code, after all its only pictures,…"*
    - *"I need a Pentium with,…"*
- Sometimes they do, but these answers may not be the best.
  - Their ideas may be incomplete, out of date, or wrong

Potential Solutions

- Brainstorming
  - just get everything out on the table; distill raw data – separate problems from solutions
- Causal analysis
  - find out why users want each feature and quality attribute characteristic
- Fantasy
  - invite stakeholders to describe the perfect solution
- Prototypes

Refinement: Analysis

Look for:

- Consistency with objective
- Abstraction vs. detail
- Categorization (triage)
- Bounded and unambiguous
- Specific source (person)
- Conflicts with others
- Achievable
- Testable

Refinement: Negotiation

- Do customers want more than possible (cost, time, scope, quality)
- Prioritize by value and cost
  - Value to the customer
  - Value to other stakeholders?
  - Difficulty to achieve (do the hard first?)

Role of Analysis – 1

- Raw requirements tend to describe a desired product from an unstructured operational perspective such as
  - who will use it
  - what the user would like to have
  - in what context(s) it will be used
  - function and quality attribute necessities

- Unstructured wants and needs must be refined into a *requirements specification*.

Role of Analysis – 2

- *Requirements elicitation* is a *divergent* process that gathers more and more data.

- *Requirements Analysis* is a *convergent* process that
  - refines data rather than gathers it
  - structures information
  - prioritizes needs

Role of Analysis – 2

- Each functional requirement, quality attribute, and constraint must be
  - clarified – understandable by all stakeholders
  - quantified – measurable, testable
  - Prioritized
    - According to importance (to which stakeholder)
    - Consideration of difficulty to implement

Clarification – 1

- Each raw requirement must be refined to articulate the need and capture all that is relevant to designers and implementers
  - What is needed?
  - When is it needed?
  - How much of it is needed?
  - How badly is it needed?
  - For how long is it needed?
  - How likely is the need to change over time?

Clarification – 2

- Clarifying and refining requirements may feel a little like elicitation. Clarification
  - requires iteration with the stakeholders
  - may be slow, but should converge

- If you generate lots of new requirements, then you may need to revisit elicitation
  - Do you have the right/same stakeholders?
  - Have any environmental, technological, organizational, or personnel changes occurred?

Quantification

- Raw requirements tend to be unspecific and qualitative
  – Must be able to prove that a product satisfies a requirements

- Requirements specifications must say how big, how much, how fast often, and so forth.
  – If not, they you are setting the stage for failure and disappointment.

Example: Clarification

- Raw Requirement: *"The system shall be intuitively easy to use."*
  - This is un-testable!
- The system interface shall
  - be learnable to 90% proficiency in 2 weeks
  - have an avg. user error rate of less than 2%
  - score at least 85% on a user satisfaction test

Example: Clarification

- Raw Requirement:

  "The system shall be modifiable"

  (You will always loose with this requirement!)

- The system shall accommodate
  - changes in the user interface without impact to other elements of the system
  - changes to element X in Y staff hours

Priorities

- Some requirements are more important than other requirements
  - some functionality is urgently needed
  - some quality attributes are essential
  - some requirements are hard to achieve

- Prioritization in the specification is essential for
  - setting expectations, reasoning about technical tradeoffs, planning the work

Priorities - Example

- Involve stakeholders in prioritization
  - Quality Attribute Workshop (QAW)

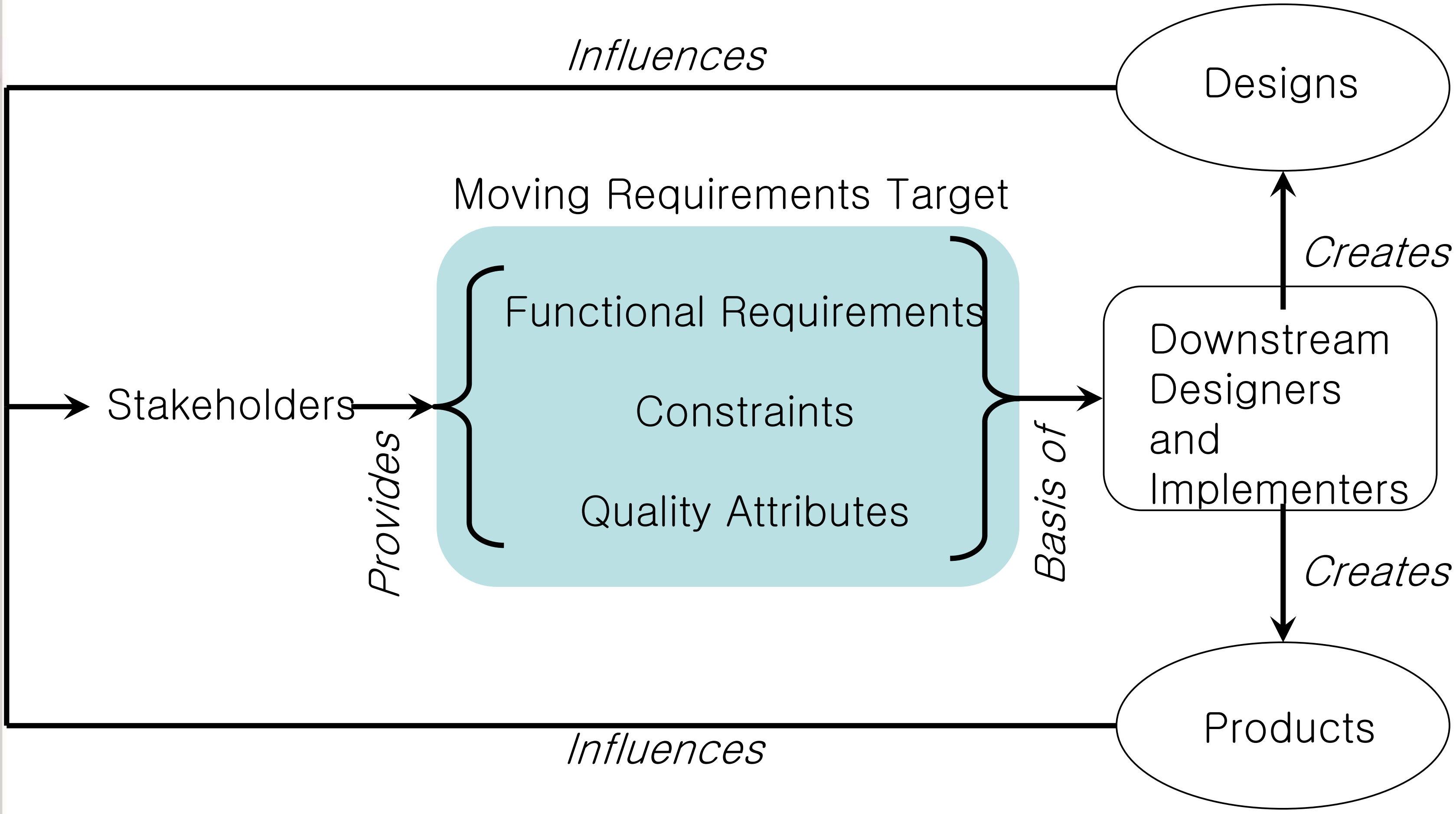
- Keep it simple…get creative…

"The system shall respond to external interrupts in 3ms."

"The system shall display warning messages in red."

Stakeholder votes	Critical	Important	Don't Care
Response time	20	10	2
Warning Messages	0	2	30
:	:	:	:

Moving Requirements



Summary

- Requirements elicitation is difficult and problematic.
  - elicitation is necessarily divergent
  - difficulty and importance is underestimated by developers and management alike
- Analysis is fundamentally about clarification and quantification.
  - analysis must be convergent
- Requirements activities must be planned.

References

- Davis, A. *Software Requirements: Objects, Functions, and States*. Upper Saddle River, NJ: Prentice Hall, 1993
- Gause, D.; Weinberg, G. *Exploring Requirements: Quality Before Design*, New York, NY: Dorset House Publishing
- Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; Wood, W. *Quality Attribute Workshops (QAWs), Third Edition*. Technical Report CMU/SEI-2003-TR-016: Software Engineering Institute, 2003.

Q & A