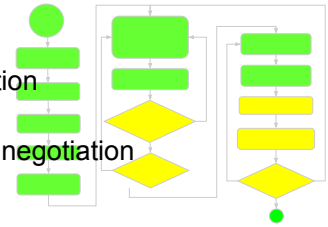
Requirements

- Overview
  - importance of getting requirements right
  - difficulty of getting requirements right
  - types and levels of requirements
  - characteristics of good requirements
- Requirements Development Process
  - inception
  - gathering, classification
  - evaluation and rationalization
  - prioritization
  - integration, reconciliation, negotiation
  - validation



Product Requirements

- before we can build anything ...
  - we must know what it is we are to build
- identify necessary conditions for success
  - bad requirements ensure product failure
  - no matter how well we do the rest of the job
- they are the basis for the product design
  - we design a product to meet the requirements
  - they guide most decisions, settle many arguments
- they are the basis for acceptance testing
  - if requirements are met, product is acceptable

Why Requirements are Difficult

- Marketing requirements are soft & vague
  - statistical results from general surveys
  - inferences from incomplete information
- Customers can't tell you what they want
  - they don't yet understand how they'll use it
  - opinions may be poorly formed or expressed
- Requirements aren't stable
  - the customer's business needs change
  - new stake-holders bring new requirements
  - technology and competition keep evolving

Get it Right ASAP

where found	where introduced		
	requirements	architecture design	construction
requirements	1x		
design	3x	1x	
construction	5-10x	10x	1x
system test	10x	15x	10x
post-ship	10-100x	25-100x	10-25x

Levels of S/W Requirements

- Requirements exist in levels
  - business requirements
    - markets to be addressed
    - business constraints and directions
  - user level functional requirements
    - supported capabilities
    - behavior in specified situations
  - component level requirements/specifications
- Lower levels are successive refinements
  - should be consistent with higher level goals

Types of S/W Requirements

- Functional Requirements
  - it must be able to X
  - when X happens it must/must-not Y
- Non-functional Requirements
  - aesthetic qualities (sound, graphics, narrative)
  - user facing (usability, familiarity, fun, challenge)
  - performance and RAS (speed, reliability, lifetime)
  - interface specifications (e.g. busses, protocols)
  - design constraints (e.g. technology, methodology)
  - support (e.g. services, materials, response times)
  - environmental (temperatures, radiation levels)

Good Requirements

- Clear
  - bounded and unambiguous
- Traceable
  - we know who gave it to us
  - we know what problem it addresses
- Confirmed
  - not arbitrary or a mere wish, but a real requirement
- Prioritized
  - we know how important it is (e.g. can, should, must)
- Within appropriate scope and level
  - reasonably falls within established project scope
  - specifies what it must do, not how it should do it

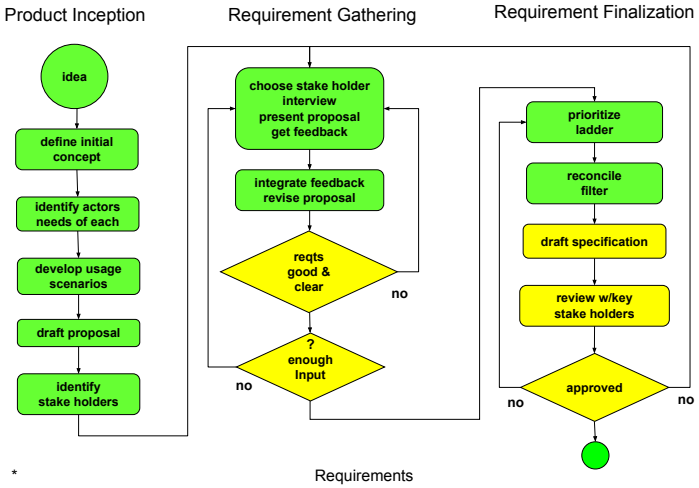
Usable Requirements

- Complete
  - no TBD details
- Testable
  - we can measure and confirm compliance
- Achievable
  - we believe we know how to do it
- Consistent
  - with higher level goals
  - no unresolved conflicts among requirements

Well Managed Requirements

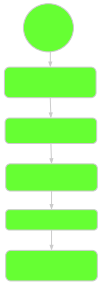
- Changes are managed carefully
  - there is a change control process
  - it may involve notifications and approvals
  - implications of changes must be understood
- Requirements are versioned
  - we all know what version we are using
- We track dependencies
  - of requirements on other requirements
  - of specifications on requirements

Requirements Development



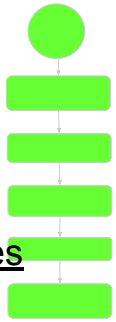
Process – Inception

- Start by identifying the problem to be solved
  - a pressing problem that can be solved or improved
  - engineers often start with the solution ☺
- Put a fence around the product
  - what will it do?
  - in what operational context will it work?
- Gather background information
  - existing products, relevant technology
- Identify stake-holders
  - potential customers (of various types)
  - potential advisors (sales, marketing, partners)
  - Collaborators (Q/A, support, management, legal)



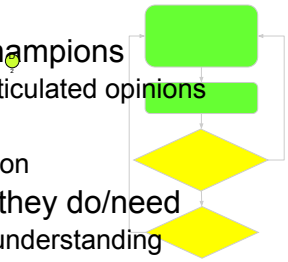
Process – Concept Development

- what kinds of people face this problem?
  - identify distinct sub-classes of users
- Understand the needs of each user
  - what must the product to do for them?
  - how would they use this product?
  - what would make it well suited for them?
- Represent this information in use cases
  - each use case is one simple story
  - how a typical operation would be performed



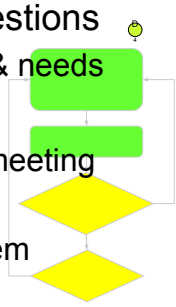
Process – Requirements Gathering

- Initial use cases are often brain-stormed
  - part of the process of developing the concept
  - resulting scenarios are hypothetical examples
- Requirements Elicitation
  - domain experts and product champions
    - they often have clear and well articulated opinions
  - representative users
    - a potential gold-mine of information
  - gather information about what they do/need
    - ask questions to ensure correct understanding
    - distill into requirements, classify by level/type



Keys to Successful Elicitation

- You're there to learn
  - not to sell or defend a proposal
  - let the customer do most of the talking
- Start with general, open-ended questions
  - understand what the customer does & needs
- Keep the meeting moving on track
  - have an experienced facilitator lead meeting
  - finish a topic, and then move on
  - identify issues, don't try to resolve them



The Real Thing

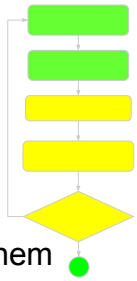
- In-class Requirements Elicitation Session
  - demonstration and discussion
  - I need a team to volunteer for this
- Benefits of volunteering
  - score: better of 100% or +25%
- Cost
  - preparation must be completed sooner
  - panel must be available at that time
  - public evaluation of performance

Surveys vs. Interviews

- Advantages
  - quickly reach very large audiences
- Disadvantages
  - questions may be confusing, leading, biased
  - answers are easily misinterpreted
  - respondents are self-selected
  - conversation enables better understanding
- Combinations
  - surveys to collect basic statistical data
  - interviews to develop understanding
  - surveys may identify interview candidates

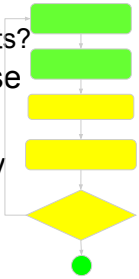
Process – Requirements Evaluation

- Assess quality of each requirement
  - vague, untestable, poorly substantiated
  - figure out how to fix poor requirements
- Ensure each requirement is rated
  - for value to the success of the product
  - for feasibility, risk and difficulty
- Prioritize the requirements
  - assign an priority to each, and ladder them

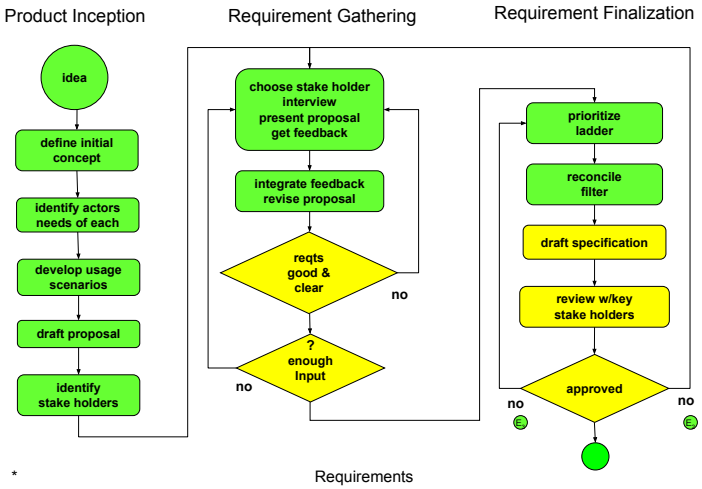


Process – Integration/Validation

- Integration
  - combine all of the requirements
  - reconcile and resolve any conflicts
  - assess completeness and stability
    Are we ready to go with these requirements?
  - decide which to address in this release
- Validation
  - final review of correctness and quality
  - ensure overall consistency with goals
  - obtain all required buy-ins



Requirements Development



Feature Phasing

- we can seldom satisfy all requirements
  - in a reasonable project budget or time frame
- some features can wait for next release
  - some features are merely desired
  - some features only become required later
- some features we cannot yet specify
  - need real data on how product will be used
  - need results of further prototyping/analysis
- use priority, cost, and risk to sort features
  - into this release, next release, and later

Requirements: the bottom line

- s/w products aren't "collections of features"
  - they are "tools", that do "things", for "people"
  - you must know **who those people are**
  - you must understand **what they need**
  - "requirements" help fill in these understandings
- your audience and purpose are critical
  - you should be able to concisely describe each
  - these are your fundamental requirements
  - hang them on the wall in front of your desk
- don't mistake "details" for "purpose"

For the next lecture

- Sisson: User Characterization
  - web-centric examination of approaches & problems
- Rouse: What Players Want
  - getting inside your customers' heads
- Wiegers: Developing Use Cases
  - a different approach to requirements development
- Wells: user story cards
  - the "*agile*" alternative to requirements
- UML:
  - introduction to a family of modeling languages
  - use-case, state and activity diagrams

Supplementary Slides

Eliciting Requirements

- Use a formal process
  - create, distribute and follow an agenda
  - ask prepared, open-ended questions
  - take detailed written minutes
  - prepare a written report on each meeting
- Desired results
  - a clearer understanding of the problem
  - identify additional stake-holders & needs
  - feedback on the proposal (value assessments, constraints, concerns, updated use cases)

Conflict Resolution

- Win-Win negotiation is a must
  - if key requirements aren't met, product fails
  - people will help you, if you help them
- Must have priority assessments
  - understand each group's key requirements
  - all requirements are not equally important
- Must be able to trace requirement origins
  - we may have misunderstood a requirement
- Divide and Conquer
  - de-couple problems to solve one-at-a-time

Validating Requirements

- Are these requirements good?
  - clear, well justified, and widely agreed to
  - traceable and prioritized
  - measurable and testable
  - do we believe we can satisfy them?
- Are these requirements complete?
  - have all open issues/conflicts been resolved
  - do we believe all requirements to be stable
- When these answers are yes, we're ready

Requirements Work Products

- user level specification
  - description used for requirements elicitation
  - user level requirements (typically use cases)
- requirements meeting reports
  - report from each elicitation or approval meeting
- system level specification
  - system model used for requirements analysis
    - may be more component oriented than user spec
  - system level requirements (typically in writing)
    - prioritized, cross-referenced to user requirements

Requirements Management

- a must for large and complex projects
  - requirements become contractual obligations
- each requirement should have
  - a clear and measurable statement
  - a unique identification number
  - a priority, flexibility, certainty assessment
  - a history of its source, and all changes
- this often entails a specialized database
  - and highly specified change processes
  - with designated approvers for all changes

Q: How much (reqts) process?

A: Enough to give us confidence

- How obvious is the problem?
- How many stake-holders are there?
- How clear are they on their needs?
- How complex are the use cases?
- How obvious is the feature set?
- How demanding are your customers?
- How good is your competition?
- What are the consequences of error?