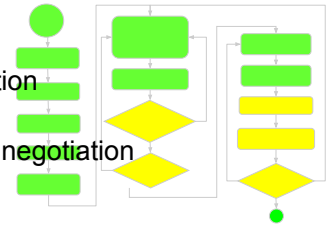
Requirements

- 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

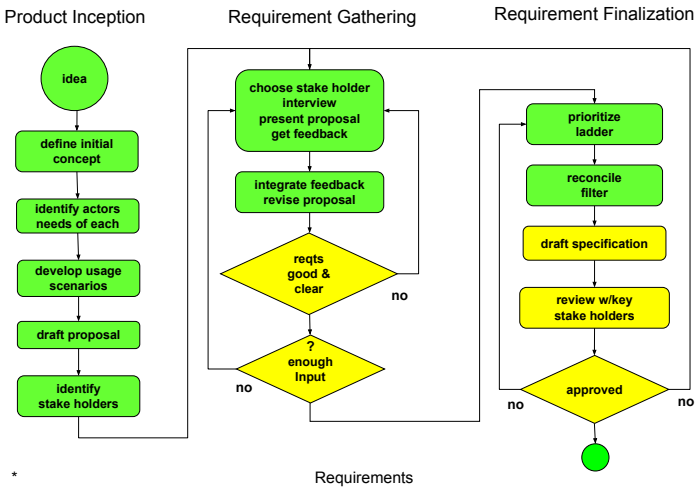
Usable Requirements

- 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 (in large/formal projects)

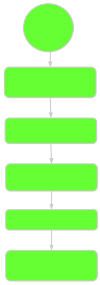
- Changes are managed carefully
  - there is a change control process
  - it may involve notifications and approvals
  - implications of changes must be understood
- Requirements are versioned
  - we all know what version we are using
- We track dependencies
  - of requirements on other requirements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

Requirements Development



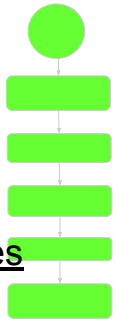
Process – Inception

- 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)



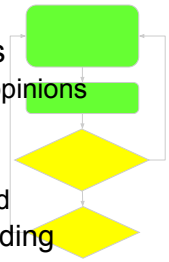
Process – Concept Development

- 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



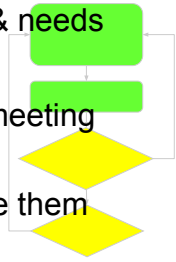
Process – Requirements Gathering

- 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
  - understand issues, don't try to resolve them



The Real Thing

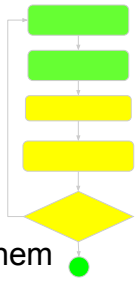
- In-class Requirements Elicitation Session
  - demonstration and discussion
  - so you can all observe and learn process
  - I need a team to volunteer for this
- Benefits of volunteering
  - score: better of 100% or +25%
- Costs
  - preparation must be completed sooner
  - your panel must be available 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

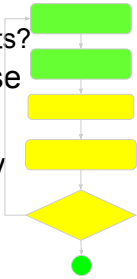
Process – Requirements Evaluation

- Assess quality of each requirement
  - vague, poorly substantiated, untestable
  - 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

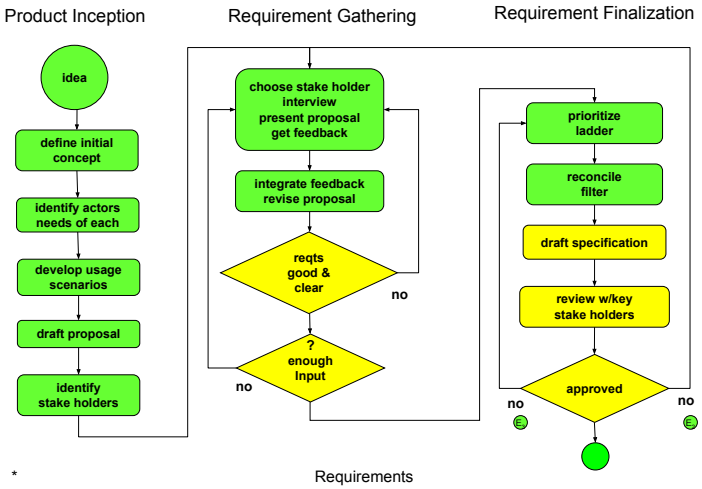


Process – Integration/Validation

- 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" instantiate 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"

Details vs. Purpose

"There is no great art to devising a good plan of operations. The entire difficulty lies in this:

to remain faithful in action to the principles we have laid down for ourselves."

Carl von Clausewitz - "On War"

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 XP (agile) expression of 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?

Discussion Slides

Deeper

- Get the requirements right before you start! Is this a refutation of Ambler's BRUF?

  Ambler said it is futile to collect complete requirements before you start, and that requirements refinement is best achieved by real user feedback on an evolving product.

Deeper

- Why are requirements errors so expensive to fix later?

  Because they are the basis for the architecture, design, specifications, and code. Changing the requirements could necessitate redoing all of that other work.

- Beyond cost to fix, why are they more important than other types of errors?

  Because they are almost sure to translate into customer satisfaction issues.

Deeper

- Is it possible to meaningfully explore customer requirements if the general business requirements are not yet clear?

  It is hard to develop user requirements if you don't know who the users are or what problem you are solving.

  It is certainly possible, however, to explore user requirements before we settle on a price point, or understand who our partners might be.

Deeper

- Should all lower level requirements be traceable to higher level requirements?

  Lower level requirements are mostly expansions of higher level requirements. We should not "make up" new requirements as we elaborate our plans and designs.

  But – many requirements do not come from the product concept (e.g. legal and standards compliance), and so may not have been mentioned in user level requirements gathering activities.

Deeper

- Why might an "interface specification" be considered a requirement, rather than a specification or design detail?

  Internal interfaces usually are considered to be design details.

  External interfaces affect interoperability with other products and services, and interoperability constraints are clearly requirements.

  There may be legal/standards considerations.

Deeper

- Give examples of ambiguous or unbounded requirements.
  - *Response time to trivial request must be good.*
  - *System must be scalable to large loads.*
  - *System must be robust to user errors.*
  - *All code to be thoroughly tested.*
- How can we fix them?
  - *Often, by turning them into (non-arbitrary) quantitative statements.*

Deeper

- Why must requirements be traceable to an underlying problem?
  - *Requirements are often misunderstood. Being able to trace it back to the problem it is supposed to solve will help to ensure that we do solve the problem.*

Why traceable to a particular person?

- *If we need to verify or seek clarification on a requirement, we need to know who to ask.*

Deeper

- Why must requirements be prioritized?
  - *When we are trying to reconcile conflicting goals, or impossible schedules, we need to be able to balance them against one another or to order our requirements so that we can still satisfy the most important ones.*

Deeper

- Why do we say that requirements should say <u>what</u>, but not <u>how</u>?
  - *This might over-constrain the problem, making it more difficult to reconcile competing requirements.*
  - *Product design is driven by many factors other than functional requirements.*
  - *Most users do not understand the implementation technology and framework and so are not competent to design a viable implementation.*

Deeper

- Why might it make sense to specify design or implementation details in a set of design requirements?
  - *The use (or reuse) of particular components or technology may be part of a strategic initiative or significantly lower engineering costs.*
  - *Organizational realities (like multi-site or out-sourced) development) could necessitate independence of selected components.*
  - *Such considerations often impose requirements on system and component designs.*

Deeper

- Why should all requirements be testable?
  - *If it is a requirement, we should test it as part of our product validation. If it is not testable, how will we know if we have satisfied it?*
  - *In most cases, requirements are non-testable because they are vague. It is often difficult to clarify these issues, but the reward is a much clearer understanding of our goal, and much greater chance of achieving it.*

Deeper C_7

- What could reasonably give rise to unachievable requirements?
  - *Somebody could easily want something, that turned out to be beyond our technological capabilities.*
  - *Some managers confuse requirements with stretch goals. The former must be achieved. The latter, we would like to achieve (or at least get closer to).*

Deeper C_8

- Why is it important that the implications of each requirements change be studied and understood before it is accepted?
  - *An ill considered requirements change could greatly affect the amount of work involved, or even the feasibility of a project.*
  - *If the change is made late in the project, it could invalidate large amounts of work.*
  - *This doesn't mean no changes … but it does mean we should weigh each carefully.*

Deeper C_9

- Why is it important that we track the dependencies of requirements to specifications and other requirements?
  - *This cross referencing will help us to more quickly and completely understand the implications of proposed requirements changes.*
  - *When a requirement changes, we know what specifications must change (and which groups must be consulted)*

Deeper D_1

- Why is it important to identify the full range of likely stake-holders?
  - *Any stake holder may be a source of key requirements. If we fail to talk to a stake holder, we may miss key requirements that they could have contributed.*
  - *Many stake holders will be only briefly involved in a project … but they may still offer valuable input during that time.*

Deeper D_2

- Why is the difference between a domain expert and a potential user?
  - *Domain experts might include consultants, sales people, and managers who know the domain but wouldn't actually use the product. They are often more technically savvy.*
- Why is it critical to talk to potential users?
  - *These are the people we must ultimately satisfy.*

Deeper D_3

- Why do we want to describing our product in too much detail?
  - *We don't want to bias their thinking.*
- Why do we want to avoid "selling" our product?
  - *The purpose and value of this meeting is gathering input about what these users need. Other discussions would shift attention or take time away from that goal.*

Deeper D_4

- Why is it important to start an interview with general and open-ended questions about what they do and how, and what problems they face?
  1. *This is the context within which our product must operate.*
  2. *We are giving them room to tell us things that we didn't know to ask about, and to describe problems we were not aware of.*

Deeper E_1

- What can we do when we seem to have conflicting or contradictory requirements?
  - *Start by understanding them, the underlying issues may not truly be in conflict.*
  - *It may be possible to define a reasonable trade-off or to offer different options for different situations.*
  - *Assess the likelihood of the respective situations and the importance of the associated requirements, and let priorities guide your decision.*

Deeper E_2

- How do we decide when we have gathered "enough" input?
  - *Do we have input from all "key" stake-holders?*
  - *Do we have high quality input from representative samples of all major classes of stake-holders?*
  - *Is the input we have been receiving starting to converge?*

Deeper E_3

- How do we know if the correct approach is more input, or re-prioritization?
  - *Understand the reasons why the proposal was not accepted. Typical problems are:*
  1. *Differing perceptions of relative priorities.*
  2. *Incomplete understanding of project constraints (on one or both sides).*
  3. *A key stake-holder who did not have adequate input.*

Deeper E_4

- Why is value/cost/risk the right way to decide what to keep in a proposed release?
  - *Value: because we want to deliver the greatest possible customer value.*
  - *Cost: because price-performance helps us get the best return on invested effort.*
  - *Risk: because a product that might not be buildable, or a badly missed schedule is a lose for everyone.*

Deeper A_1

- Are developer-facing characteristics legitimately sources of requirements?
  - *It is a requirement that the software be built, supported and maintained. Factors that bear on the ease of development and maintenance can make or break a product.*
  - *It is common to require internal features that will enable future features or cost savings.*

Deeper

- How does XP manage requirments?

  They are maintained, as user stories, on 3x5 cards. These stories evolve over time as they are refined through prototyping and user feedback.

Deeper

- Which of these steps would or would not make sense in an XP project?
  - *Inception* *is still done*
  - *Concept development* *is still done*
  - *Gathering* *is done iteratively, feedback vs. elicitation*
  - *Evaluation/Prioritization* *is still done*
  - *Integration/Reconciliation* *is still done*
  - *Validation* *is by customer feedback*

The basic process does not greatly change.