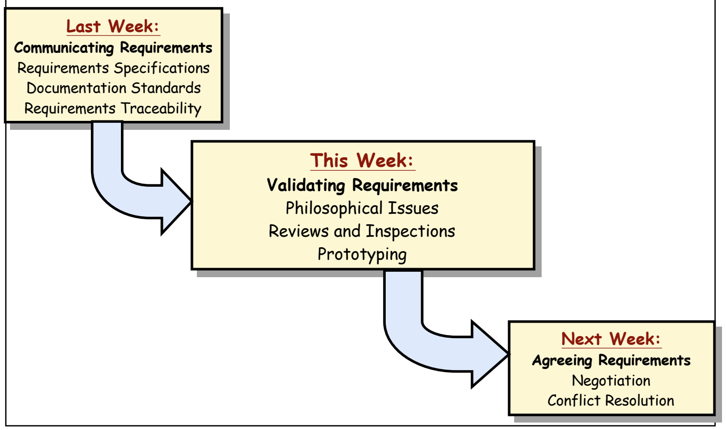
Lecture 10: Validating Requirements



Overview

- Two key problems for getting agreement:
  1) the problem of validation
     What is "truth" and what is "knowable"?
  2) the problem of negotiation
     How do you reconcile conflicting goals in a complex socio-cognitive setting?
- Validating Requirements
  - Inspections and Reviews
  - Prototyping
- Negotiating Requirements (next week)
  - Conflict and Conflict Resolution
  - Requirements Negotiation Techniques
    - Argumentation approaches
    - Knowledge-based approaches
  - Requirements Prioritization

The problem of validation

- logical positivist view:
    - "there is an objective world that can be modeled by building a consistent body of knowledge grounded in empirical observation"
  - In RE, assumes there is an objective problem that exists in the world
    - Build a consistent model; make sufficient empirical observations to check validity
    - Use tools that test consistency and completeness of the model
    - Use reviews, prototyping, etc to demonstrate the model is "valid"
- Popper's modification to logical positivism:
    - "theories can't be proven correct, they can only be refuted by finding exceptions"
  - In RE, design your requirements models to be refutable
    - Look for evidence that the model is wrong
    - E.g. collect scenarios and check the model supports them
- post-modernist view:
    - "there is no privileged viewpoint; all observation is value-laden; scientific investigation is culturally embedded"
    - E.g. Kuhn: science moves through paradigms
    - E.g. Toulmin: scientific theories are judged with respect to a *weltanschauung*
  - In RE, validation is always subjective and contextualised
    - Use stakeholder involvement so that they 'own' the requirements models
    - Use ethnographic techniques to understand the weltanschauungen

Reviews, Inspections, Walkthroughs…

Source: *Adapted from Blum, 1992, pp369-373*

- Note: these terms are not widely agreed
  - formality
    - **informal**: from meetings over coffee, to team get-togethers
    - **formal**: scheduled meetings, prepared participants, defined agenda, specific format, documented output
  - "Management reviews"
    - E.g. preliminary design review (PDR), critical design review (CDR), …
    - Used to provide confidence that the design is sound
    - Attended by management and sponsors (customers)
    - Usually a "dog-and-pony show"
  - "Walkthroughs"
    - developer technique (usually informal)
    - used by development teams to improve quality of product
    - focus is on finding defects
  - "(Fagan) Inspections"
    - a process management tool (always formal)
    - used to improve quality of the development process
    - collect defect data to analyze the quality of the process
    - written output is important
    - major role in training junior staff and transferring expertise

Benefits of formal inspection

Source: *Adapted from Blum, 1992, pp369-373 & Freedman and Weinberg, 1990.*

- **Formal inspection works well for programming:**
  - For applications programming:
    - more effective than testing
    - most reviewed programs run correctly first time
    - compare: 10-50 attempts for test/debug approach
  - Data from large projects
    - error reduction by a factor of 5; (10 in some reported cases)
    - improvement in productivity: 14% to 25%
    - percentage of errors found by inspection: 58% to 82%
    - cost reduction of 50%-80% for V&V (even including cost of inspection)
  - Effects on staff competence:
    - increased morale, reduced turnover
    - better estimation and scheduling (more knowledge about defect profiles)
    - better management recognition of staff ability
- **These benefits also apply to requirements inspections**
  - E.g. See study by Porter et. al.

Inspection Constraints

Source: *Adapted from Blum, 1992, pp369-373 & Freedman and Weinberg, 1990.*

- **Size**
  - "enough people so that all the relevant expertise is available"
  - min: 3 (4 if author is present)
  - max: 7 (less if leader is inexperienced)
- **Duration**
  - never more than 2 hours
    - concentration will flag if longer
- **Outputs**
  - all reviewers must agree on the result
    - accept; re-work; re-inspect;
  - all findings should be documented
    - summary report (for management)
    - detailed list of issues
- **Scope**
  - focus on small part of a design, not the whole thing
- **Timing**
  - Examines a product once its author has finished it
  - not too soon
    - product not ready - find problems the author is already aware of
  - not too late
    - product in use - errors are now very costly to fix
- **Purpose**
  - Remember the biggest gains come from fixing the process
    - collect data to help you not to make the same errors next time

Inspection Guidelines

Source: *Adapted from Freedman and Weinberg, 1990.*

- **Prior to the review**
  - schedule Formal Reviews into the project planning
  - train all reviewers
  - ensure all attendees prepare in advance
- **During the review**
  - review the product, not its author
    - keep comments constructive, professional and task-focussed
  - stick to the agenda
    - leader must prevent drift
  - limit debate and rebuttal
    - record issues for later discussion/resolution
  - identify problems but don't try to solve them
  - take written notes
- **After the review**
  - review the review process

Choosing Reviewers

Source: *Adapted from Freedman and Weinberg, 1990.*

- **Possibilities**
  - specialists in reviewing (e.g. QA people)
  - people from the same team as the author
  - people invited for specialist expertise
  - people with an interest in the product
  - visitors who have something to contribute
  - people from other parts of the organization
- **Exclude**
  - anyone responsible for reviewing the author
    - i.e. line manager, appraiser, etc.
  - anyone with known personality clashes with other reviewers
  - anyone who is not qualified to contribute
  - all management
  - anyone whose presence creates a conflict of interest

Structuring the inspection

Source: Adapted from Porter, Votta and Basili, 1995

→ Can structure the review in different ways

- Ad-hoc
  - Rely on expertise of the reviewers
- Checklist
  - uses a checklist of questions/issues
  - checklists tailored to the kind of document (Porter et. al. have examples)
- active reviews (perspective based reading)
  - each reviewer reads for a specific purpose, using specialized questionnaires
  - effectively different reviewers take different perspectives

→ The differences may matter

- E.g. Porter et. al. study indicates that:
  - active reviews find more faults than ad hoc or checklist methods
  - no effective different between ad hoc and checklist methods
  - the inspection meeting might be superfluous!

Prototyping

→ Definitions

- "A software prototype is a partial implementation constructed primarily to enable customers, users, or developers to learn more about a problem or its solution." [Davis 1990]
- "Prototyping is the process of building a working model of the system" [Agresti 1986]

→ Approaches to prototyping

- Explanatory
  - explain, demonstrate and inform – then throw away
  - e.g. a 'presentation prototype used at the initiation of the project
- Exploratory
  - used to determine problems, elicit needs, clarify goals, compare design options
  - informal, unstructured and thrown away.
- Experimental
  - evaluate technical issues and model behaviour; test suitability of a technology
  - detailed, throw away (or possibly) enhance as product.
- Evolutionary (e.g. "operational prototypes", "pilot systems"):
  - development seen as continuous process of adapting the system
  - prototype is an early deliverable, to be continually improved.

Throwaway or Evolve?

→ Throwaway Prototyping

- Purpose:
  - to learn more about the problem or its solution…
  - hence discard after the desired knowledge is gained.
- Use:
  - early or late
- Approach:
  - horizontal - build only one layer (e.g. UI)
  - "quick and dirty"
- Advantages:
  - Learning medium for better convergence
  - Early delivery □ early testing □ less cost
  - Successful even if it fails!
- Disadvantages:
  - Wasted effort if requirements change rapidly
  - Often replaces proper documentation of the requirements
  - May set customers' expectations too high
  - Can get developed into final product

→ Evolutionary Prototyping

- Purpose
  - to learn more about the problem or its solution…
  - …and to reduce risk by building parts of the system early
- Use:
  - incremental; evolutionary
- Approach:
  - vertical - partial implementation of all layers;
  - designed to be extended/adapted
- Advantages:
  - Requirements not frozen
  - Return to last increment if error is found
  - Flexible(?)
- Disadvantages:
  - Can end up with complex, unstructured system which is hard to maintain
  - early architectural choice may be poor
  - Optimal solutions not guaranteed
  - Lacks control and direction

Brooks: "Plan to throw one away - you will anyway!"