Context

- **Learning objectives:**
  - Learn how to write a useful, abstract specification of a software component
  - Learn how to do an effective technical review
- **Everyone is learning this for the first time**
- **This is only the first step**
  - Review and try to understand what is there
  - Discuss the specifications with the authors
  - Suggest improvements
  - Goal: converge on a common understanding of what should be built (the “contract”)
- **Focus on constructive critique:** how can it be better? Our goal is to help and build trust (not criticize).
**V&V Methods**

- Most applied V&V uses one of two methods
- Review: use of human skills to find defects
  - Pro: applies human understanding, skills. Good for detecting logical errors, problem misunderstanding
  - Con: poor at detecting inconsistent assumptions, details of consistency, completeness. Labor intensive
- Testing: use of machine execution
  - Pro: can be automated, repeated. Good at detecting detail errors, checking assumptions
  - Con: cannot establish correctness or quality
- Tend to reinforce each other

**Informal Reviews**

- *Informal* used with two meanings, usually both apply
  1. "Internal" — reviewers a team members (explicitly excludes management)
  2. "Unstructured"
     - No explicit process or recording of results
       - "Please read this for me" (requirements, design, etc.)
       - Could be several readers, selected by author
     - Author takes comments and makes revisions as he/she sees fit
Formal Reviews

- Includes people outside the team
- Explicit process, results recorded and tracked
- Standard types of industry reviews
  - Software peer review: technical review by author’s peers (our focus)
  - Software management review: management evaluation of project status
  - Software audit: external review for compliance with standards, regulations, contract, etc.

Example: Software Peer Review

- Definition: a form of technical review in which a software product is examined by peers of the product’s authors with the goal of finding defects
  - Also called “software inspection”
  - Most common type of technical review in industry
  - Often standardized part of milestone planning
- Formal Meeting held at a pre-defined time and place
  - Reviewers read artifact in advance
  - Facilitator leads discussion of artifact, often on line-by-line basis
  - Issues raised by discussion recorded
  - Author revises artifact after the meeting in response to issues
  - Revised artifact recirculated among reviewers for consensus
Peer Review Benefits

- Primary defect detection method where automation is not possible or practical
  - E.g. review for meaning, intent, goal satisfaction, human factors, etc.
  - Especially review of upstream artifacts (e.g. requirements, design)
- Very effective if done carefully, systematically
  - Analysis of 12,000 development projects showed defect detection rate of 60-65% for formal inspection 30% for testing
  - Bell-Northern found 1 hour code inspecting saves 2 to 4 hours code testing
  - Effect is magnified in earlier inspections (e.g., 30 times for requirements in one study)
Peer Review Issues

• Tendency for reviews to be incomplete and shallow
• Reviewers typically swamped with information, much of it irrelevant to the review purpose
• Reviewers lack clear individual responsibility
• Effectiveness depends on reviewers to initiate actions
• Large meeting size hampers effectiveness, increases cost
  – Makes detailed discussion difficult
  – Few present reviewers have expertise on any one issue
  – Wastes everyone else’s time and energy
• No way to cross-check unstated assumptions

Active Reviews

Improved Peer Review Method
Qualities of Effective Review

- Ensures adequate coverage of artifact in breadth and depth
- Reviewers review only issues on which they have expertise
- Individual responsibilities are clear and fulfilling them is evidence of a job well done.
  - Review process is active: i.e., performing the review produces visible output
  - Review process focuses on finding specific kinds of errors.
- Limit meetings to focused groups and purposes requiring common understanding or synergy
  - Permit detailed discussion of issues
  - Expose where assumptions differ

Active Review Process

Goal: Make the reviewer(s) think hard about what they are reviewing
1. Identify several types of reviews targeting different types of errors
2. Identify appropriate classes of reviewers for each type of review
3. Assign reviews to achieve coverage
   - Each applicable type of review is applied to each part of the specification
Active Reviews (2)

4. Design review questionnaires (key difference)
   – Define questions that the review must answer by using the specification
   – Target questions to bring out key issues
   – Phrase questions to require “active” answers (not just “yes”)

5. Review consists using the artifact to fill out questionnaire

6. Review process: overview, review, meet
   – One-on-one or small, group
   – Discuss issues identified in review
   – Track and respond to issues

Examples

• Active review asks a qualified reviewer to check a specific part of a work product for specific kinds of defects by answering specific questions, e.g.,
  – Ask a designer to check the functional completeness by showing the calls sequences sufficient to implement a set of use cases
  – Ask a systems analyst to check the ability to create required subsets by showing which modules would use which
  – For each access program in the module, what exceptions that can occur?
  – Ask a technical writer to check the SRS for grammatical errors
• Can be applied to any kind of artifact from requirements to code
Conventional vs. Active Questions

- Goal: Make the reviewer(s) think hard about what they are reviewing*
  - Define questions that the review must answer by using the specification
  - Target questions to bring out key issues
  - Phrase questions to require “active” answers (not just “yes”)

<table>
<thead>
<tr>
<th></th>
<th>Conventional Design Review Questions</th>
<th>Active Design Review Questions*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are exceptions defined for every program?</td>
<td>Are exceptions defined for every program?</td>
<td>For each access program in the module, what exceptions that can occur?</td>
</tr>
<tr>
<td>Are the right exceptions defined for every program?</td>
<td>Are the right exceptions defined for every program?</td>
<td>What is the the range or set of legal values?</td>
</tr>
<tr>
<td>Are the data types defined?</td>
<td>Are the data types defined?</td>
<td>For each data type, what are: • an expression for a literal value of that data type; • a declaration statement to declare a variable for that type; • the greatest and least values in the range of that data type?</td>
</tr>
<tr>
<td>Are the programs sufficient?</td>
<td>Are the programs sufficient?</td>
<td>Write a short pseudo-code program that uses the design to accomplish (some defined task).</td>
</tr>
</tbody>
</table>

Applying Use Cases

- Use cases or scenarios can be effectively used in active review
- Apply requirements scenarios to verify design against requirements
  - “Show the sequence of program calls that would implement use case C”
  - “Which modules would have to change to add feature F (a likely change)?”
- Conversely, can check properties ask the reviewer to construct scenarios
  - “What sequence of calls would result in an exception E?”
Why Active Reviews Work

- Focuses reviewer’s skills and energies where they have skills and where those skills are needed
- Largest part of review process (filling out questionnaires) is conducted independently and in parallel
- Reviewers actively use the artifact
- Downside: much more work for V&V

Summary

- Reviews are usually only practical method for reviewing
  - Early artifacts (requirements, etc.)
  - Defects in understanding, some qualities, etc.
- Effective method of defect detection
- Active reviews are more effective than standard inspections
How: suggested process

- Identify the set of important properties to review for
  - Every term used is well defined
  - Semantics: the effect of each possible service call is clearly specified
  - Exceptions: the result of any exception is defined...
- Divide your team up so different members review for different kinds of issues
- Create a set of questions and test cases that will exercise each of the properties you wish to review for
- Review: each reviewer
  - Uses the spec to answer a set of questions
  - Creates a list of issues identified in the review
  - Discuss issues with developer

How: creating questions

- Use the fact that the form and content of interface specifications is pretty standardized
- Consider generic kinds of questions that will exercise different parts of the spec:
  - E.g. For each exception in the Exception Dictionary, give a sequence of service calls (with parameter values if needed) that should throw the exception.
  - For each service, give an example of a legal call (including parameter values)
Next Steps

• In class: revise to create at least one question that a reviewer should answer
• Revise the set of questions
• Contact UNC students and ask them to explain the specification (desired by Prof. Carlos)
• For unanswered issues
  – Communicate open issues
  – Where possible, suggest changes to spec

Questions?