1. Define project specifics
2. Team goals and objectives
3. Background and strategic fit
4. Assumptions
5. User Stories or Use Cases
6. User Interaction and Design
7. Questions
8. What we’re NOT Doing

- Evolve the document over time, concurrently with development
Requirements checking

- **Validity.** Does the system provide the functions which best support the customer’s needs?
- **Consistency.** Are there any requirements conflicts?
- **Completeness.** Are all functions required by the customer included?
- **Realism.** Can the requirements be implemented given available budget and technology?
- **Verifiability.** Can the requirements be checked?
Requirements validation techniques

• Requirements reviews
  – Systematic manual analysis of the requirements.
  – Review/commit changes to repository as part of workflow
    • Multiple team members OK it before committing
    • All team members get notification when it's updated

• Prototyping
  – Using an executable model of the system to check requirements.

• Test-case generation
  – Developing tests for requirements to check testability.
Your Living Requirements Document: A Shared Google Doc

- Authors, Team, Project Title
- Intro – including problem, innovation, science, core technical advance (2-3 pages)
  - Define project specifics, team goals/objectives, background, and assumptions
- System architecture overview
  - High level diagram (1 page)
  - User interaction and design (1+ page)
- Requirements (functional and non-functional)
  - User stories or use cases (links)
  - Prototyping code, tests, metrics: github commits/issues
- System models: contexts, sequences, behavioral/UML, state
- Appendices
  - Technologies employed
PRDv1: Your Living Requirements Document: A Shared Google Doc

• Authors, Team, Project Title
• Intro – including problem, innovation, science, core technical advance (2-3 pages)
  – Define project specifics, team goals/objectives, background, and assumptions
• System architecture overview
  – High level diagram (1 page)
  – User interaction and design (1+ page)
• Requirements (functional and non-functional)
  – User stories or use cases (links) → 10 for PRDv1 prioritized
  – Prototyping code, tests, metrics (5+ user stories): github commits/issues
• System models: contexts, sequences, behavioral/UML, state
• Appendices
  – Technologies employed
PRDv2: Your Living Requirements Document: A Shared Google Doc

• Authors, Team, Project Title
• Intro: problem, innovation, science, core technical advance
  – Define project specifics, team goals/objectives, background, and assumptions
• System architecture overview
  – High level diagram (1 page)
  – User interaction and design (1 page)
• Requirements (functional and non-functional)
  – User stories or use cases (links) → 20+ for PRDv2 prioritized
  – Prototyping code, tests, metrics (10+ user stories): github commits/issues
• System models (1+ pages)
  – Contexts, interactions, structural, behavioral (UML)
  – Use cases, sequencing, event response, system state, classes/objects
• Appendices - Technologies employed