Chapter 30

Agile Requirements Methods

Mitigating Requirements Risk

- The entire requirements discipline within the software lifecycle exists for only one reason:
  - To mitigate the risk that requirements-related issues will prevent a successful project outcome.

- If there were no such risks, then it would be far more efficient to go straight to code and eliminate the overhead of requirements-related activities.

Requirements Techniques and the Specific Project Risks They Address

Technique	Risk Addressed
Interviewing	• The development team might not understand who the real stakeholders are. • The team might not understand the basic needs of one or more stakeholders.
Requirements workshops	• The system might not appropriately address classes of specific user needs. • Lack of consensus among key stakeholders might prevent convergence on a set of requirements.
Brainstorming and idea reduction	• The team might not discover key needs or prospective innovative features. • Priorities are not well established, and a plethora of features obscures the fundamental "must haves."
Storyboards	• The prospective implementation misses the mark. • The approach is too hard to use or understand, or the operation's business purpose is lost in the planned implementation.

Requirements Techniques and the Specific Project Risks They Address

Use cases	• Users might not feel they have a stake in the implementation process. • Implementation fails to fulfill basic users needs in some way because some features are missing or because of poor usability, poor error and exception handling, and so on.
Vision document	• The development team members do not really understand what system they are trying to build, or what user needs or industry problem it addresses. • Lack of longer-term vision causes poor planning and poor architecture and design decisions.
Whole product plan	• The solution might lack the commercial elements necessary for successful adoption.
Scoping activities	• The project scope exceeds the time and resources available.
Supplementary specification	• The development team might not understand nonfunctional requirements: platforms, reliability, standards, and so on.

Requirements Techniques and the Specific Project Risks They Address

Tracing use cases to implementation	• Use cases might be described but not fully implemented in the system.
Tracing use cases to test cases	• Some use cases might not be tested, or alternative and exception conditions might not be understood, implemented, and tested.
Requirements traceability	• Critical requirements might be overlooked in the implementation. • The implementation might introduce requirements or features not called for in the original requirements. • A change in requirements might impact other parts of the system in unforeseen ways.
Change management	• New system requirements might be introduced in an uncontrolled fashion. The team might underestimate the negative impact of a change.

Documenting Requirements

- Most requirements artifacts, Vision documents, use cases, and so forth—and indeed software development artifacts in general, including the code—require documentation of some kind.

Documenting Requirements

- Do we really need to write this document at all? "Yes" only if one or more of these four criteria apply.
  1. The document communicates an important understanding or agreement for instances in which simpler verbal communication is either impractical.
  2. The documentation allows new team members to come up to speed more quickly and therefore renders both current and new team members more efficient.
  3. Investment in the document has an obvious long-term payoff because it will evolve, be maintained, and persist as an ongoing part of the development, testing, or maintenance activity.
  4. Some company, customer, or regulatory standard imposes a requirement for the document.

Agile Requirements Methods

- Extreme
- Agile
- Robust

Extreme Programming Principles/Characteristics

- The scope of the application or component permits coding by a team of three to ten programmers working at one location.
- One or more customers are on site to provide constant requirements input.
- Development occurs in frequent builds or iterations, each of which is releasable and delivers incremental user functionality.
- The unit of requirements gathering is the *user story*, a chunk of functionality that provides value to the user. User stories are written by the customers on site.

Extreme Programming Principles/Characteristics

- Programmers work in pairs and follow strict coding standards. They do their own unit testing and are supposed to routinely re-factor the code to keep the design simple.
- Since little attempt is made to understand or document future requirements, the code is constantly refactored (redesigned) to address changing user needs.

Applying Extreme Programming Principles to Requirements Risk Mitigation

Extreme Programming Principle	Mitigated Requirements Risk
Application or component scope is such that three to ten programmers at one location can do the coding.	Constant informal communication can minimize or eliminate much requirements documentation.
One or more customers are on site to provide constant requirements input.	Constant customer input and feedback dramatically reduces requirements-related risk.
Development occurs in frequent builds or iterations, each of which is releasable and delivers incremental user functionality.	Customer value feedback is almost immediate; this ship can't go too far off course.
The unit of requirements gathering is the user story, a bite of functionality that provides value to the user. Customers on site write user stories.	A use case describes sequences of events that deliver value to a user, as written by the developer with user input. User stories are often short descriptions of a path or scenario of a use case. Each captures the same basic intent—how the user interacts with the system to get something done.

Extreme Programming Principles/Characteristics

- Concepts: user elaboration
- Vision: verbal
- Requirements: use-case model
- Tool Support: Defect tracking, desktop tools

Agile Requirements Methods

- Concepts: user elaboration, interviews, workshops
- Vision: verbal, Delta vision document, Whole product plan
- Requirements: use-case model, use-case specifications, supplementary specifications
- Tool Support: Defect tracking, desktop tools, version control, requirements management tools

Robust Requirements Methods

- Concepts: user elaboration, interviews, workshops, storyboards prototypes
- Vision: verbal, Delta, vision document, Whole product plan, fully documented
- Requirements: use-case model, use-case specifications, supplementary specifications, technical methods as necessary
- Tool Support: Defect tracking, desktop tools, version control, requirements management tools, requirements traceability, analysis and design tools,
- Project control: requirements management plan, change control board, full configuration management, requirement analysis impact assessment

Key Points

- The purpose of the software development method is to mitigate the risks inherent in the project.
- The purpose of the requirements management method is to mitigate requirements-related risks on the project.
- No one method fits all projects, therefore the method must be tailored to the particular project.