Root Causes of Agile Project Failure

Jeffery Payne
Chief Executive Officer
Coveros, Inc.
jeff.payne@coveros.com
www.coveros.com

About Coveros

- Coveros helps organizations accelerate the delivery of secure, reliable software

- Our consulting services:
  - Agile software development
  - Application security
  - Software quality assurance

- Agile services
  - Agility assessments
  - Process improvement
  - Hands-on agile software development
  - Agile project management
  - Agile testing and automation
  - Agile training by role

Corporate Partners

Pop Quiz: Agile Development Means …

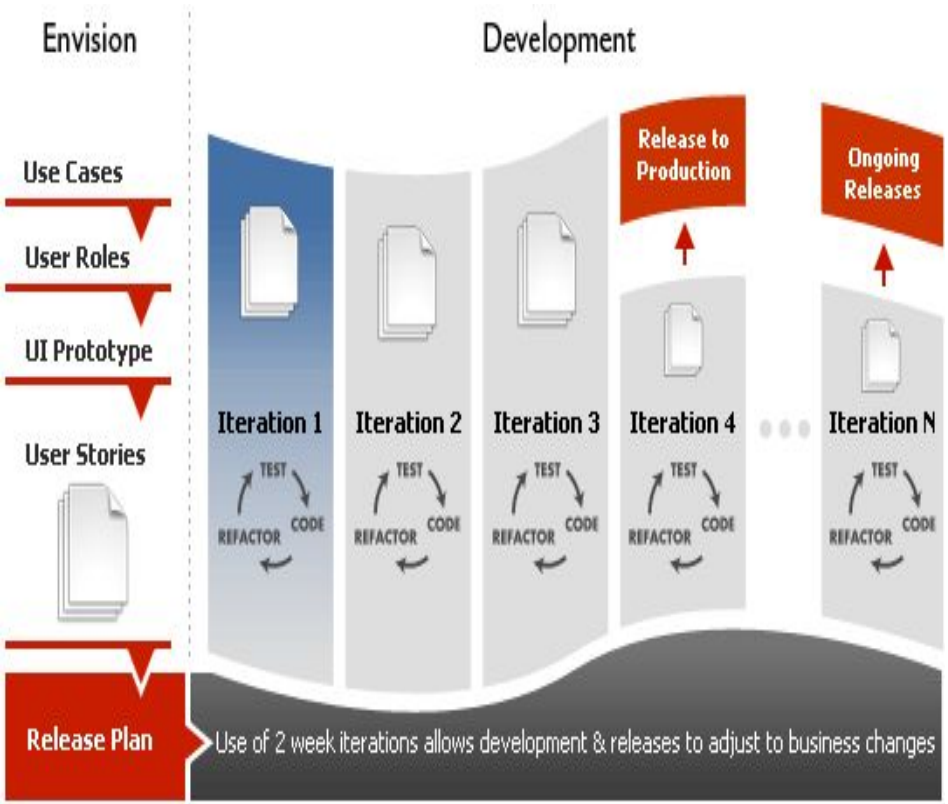
- No documentation. We don't need to write anything down!
- No process. We can do it any way we want!
- No overtime. We can go home at 5!
- No management. We decide when to deliver!
- No testers. Who needs them anyway!

What Agile Actually Is

- An approach to software development that recognizes that building software is much more a design process than a construction process
  - Adaptive over Predictive
  - People over Process
  - Visibility into the Process

- Agile Methodologies
  - Extreme Programming
  - SCRUM
  - Lean Development
  - Crystal
  - Agile RUP

Agile Development Process

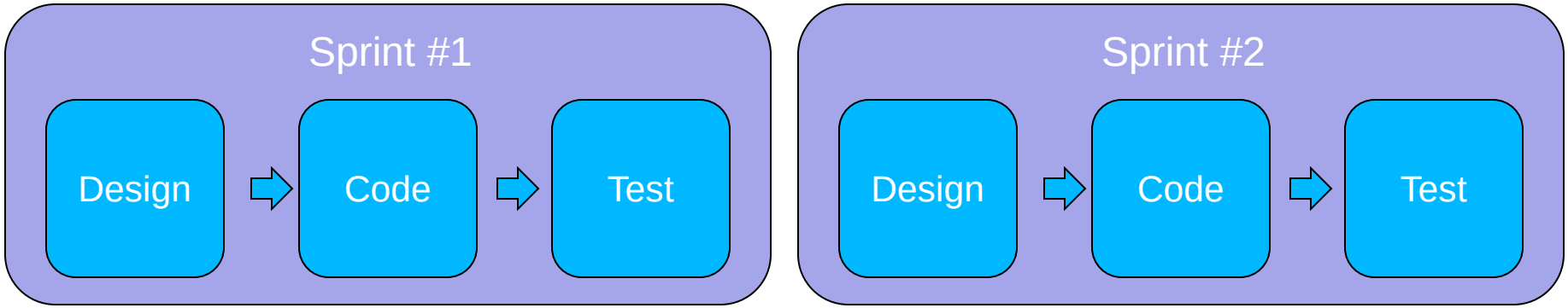


- Just in time requirements definition
- Integrated design, development, test
- Automated build, test, deployment process
- Disciplined iteration scope control
- Intimate customer involvement throughout entire process
- Continuous improvement

Root Causes of Agile Project Failure

Root Cause #1 – Scrummerfall

- Scrummerfall: Waterfall development inside Scrum



- Challenges with Scrummerfall
  - Increases need for unnecessary coordination and documentation
  - Decreases team velocity
  - Difficult to fit into short sprints

Scrummerfall – early warning signs

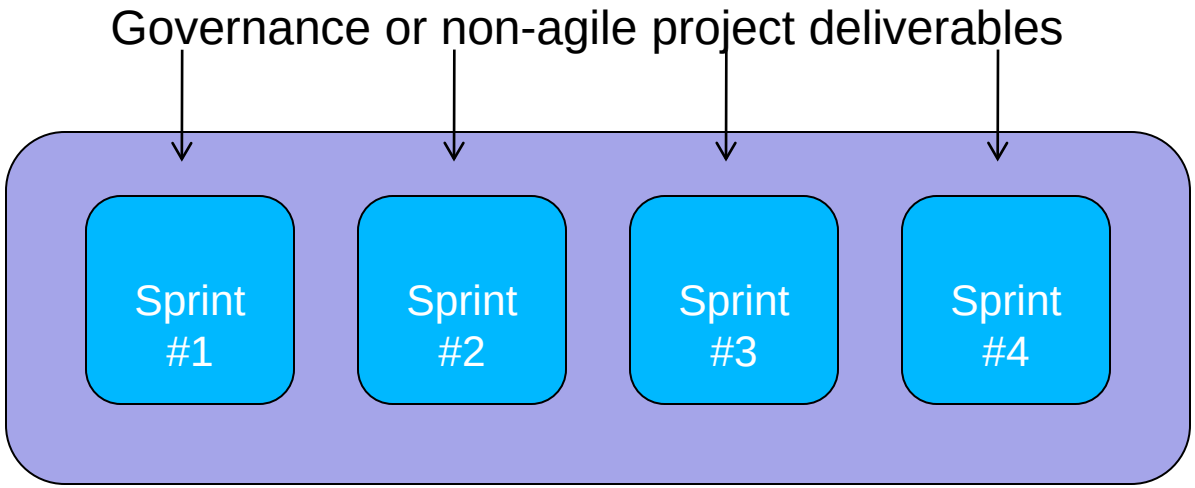
- Entire development team is not working together day-to-day on the same Stories
- Testing is often not completed by the end of a Sprint
- Testing of previous Sprint Stories is done in parallel with design / development of new Sprint Stories
- Moving toward one 9-12 month “Sprint”

Scrummerfall – What you should do

- Pair a developer and a tester to work together on specific Stories
  - Tester helps developer think through unit and integration tests for Story
  - Developer builds functionality and automates unit and integration tests
  - Tester reviews / validates unit and integration tests
  - Tester adds to system level test plan and creates additional tests
  - Developer reviews additions to test plan
- Stories are not marked as complete until all testing has been performed and defects are fixed

Root Cause #2 – Waterscrum done wrong

- Co-dependent Waterfall and Scrum projects



- Challenges with Waterscrum
  - Temptation is to lapse into a Waterfall process to align with the rest of the organization
  - Team shirks it's responsibility to the rest of the organization and agile is disbanded
  - Waterfall schedule slips and agile team does not adjust

Waterscrum – early warning signs

- Development team pushes back on providing necessary documentation
- Agile team misses deliverable date(s) associated with Waterfall schedule
- No visibility into progress on Waterfall process
- Team does not factor external delivery dates into Story priorities and schedule

Waterscrum – What you should do

- Designate a team representative to communicate / coordinate with the rest of the organization on at least a weekly basis
  - Should be the responsibility of the project manager or product manager
- Assign a “customer” to the project from the Waterfall initiative to assure agile deliverables meet organizational needs
- Define documentation or functionality constrained by Waterfall process in terms of Stories and place them in the appropriate Sprints to meet Waterfall schedule

Root Cause #3 – Ad-hoc development

- Team characterizes its development process as Agile to justify poor programming practices

- Poor development practices
  - Little or no structured unit testing
  - Little or no test automation or continuous integration
  - Little or no necessary product documentation
  - Handoffs between development and testing

- Challenges with Ad-hoc development
  - Ad-hoc development *has never worked* within any software development process except perhaps when prototyping something
  - Significant quality and maintainability issues will exist
  - Not a rigorous process

Ad-hoc development troubles – early warning signs

- Lack of evidence that adequate unit testing has been done
  - No automation infrastructure to support unit testing
  - Limited code coverage
- Lack of evidence that architecture / design has been thought through at a high level
- Individual developers working on multiple Stories at the same time
- Lack of testers on the team
- Builds break and are not fixed within a day

Ad-hoc development – What you should do

- Incorporate unit testing infrastructure (ex. jUnit, nUnit) and code coverage (ex. Cobertura, Quilt) into continuous integration
- Incorporate design activity into Sprint kickoff meeting
- Incorporate test planning activity into Sprint planning and Sprint kickoff meeting
- Pair developers and testers during Sprints

Root Cause #4 – Non-existent customers

- Customer's are not involved throughout entire development process
  - Requirement definition / priority
  - Functional clarifications
  - User acceptance testing

- Challenges with Non-existent customers
  - Significantly reduces the business value of Agile as requirements are not validated
  - Increases in rework due to changing requirements
  - Reductions in Sprint velocity and productivity

Non-existent customers – early warning signs

- Customer is not intimately involved in initial planning phase before Sprints
- Customer is not available to answer questions regarding Stories / requirements during Sprints
- Customer is not part of User Acceptance Testing or UAT is pre-scripted by development team

Non-existent customers – What you should do

- Proactively seek out at least one customer to be involved in project
  - Talk to customer support to identify the most “vocal” client you have
  - Don’t assume you already know what customers want
- If customer simply cannot be involved day-to-day or even week-to-week, work with customer to identify appropriate “proxy” to act on their behalf
- Do initial User Acceptance Testing at the client’s site to ease them into the process

Root Cause #5 – Frozen requirements

- Complete requirements list created up front that feeds into Agile Sprints
- Often occurs when development group has moved toward Agile but business side has not
- Challenges with Frozen requirements:
  - 80% of requirements will typically not be useful to customers when implemented
  - Does not allow Agile team to adapt to changes in business circumstances
  - Prioritization of requirements across Sprints will not be accurate

Frozen requirements – early warning signs

- SRS requirements document has been produced for the project
- Contract exists that specifies fixed requirements to be delivered within a particular timeframe
- Customer is not available / willing to discuss Story priorities during iterative planning process
- Business resists changes to upcoming Sprints based upon customer feedback

Frozen requirements – What you should do

- Prioritize existing requirements so highest priority requirements are satisfied first
- Discuss priorities with customer during UAT and highlight differences between upfront plan and their needs
  - May result in modifications to priorities that can help drive a more effective process
  - May result in agreement to change requirements once they better understand the impact
- If business resists requirements change, pull customer into discussion with business and get agreement

Root Cause #6 – Fixed scope … with a deadline

- Traditional project management fixes scope and estimates schedule and resources necessary to complete project
  - Sometimes all three are fixed in size!
- Agile project management fixes schedule (Sprints) and resources (Staff available) and estimates scope
- Challenges with Fixed scope
  - We don't know what features have the most business value up front
  - Customers don't know what features have the most business value up front
  - The market changes constantly, necessitating change
  - Scope is the most difficult thing to predict up front

Fixed scope – early warning signs

- There is resistance to any type of changes in priority, initial scoping of a Sprint, or initial scoping of a release
- The word “time-box” gets introduced along with the assumption that a particular scope will be completed within the time-box
- Efforts to incorporate appropriate refactoring and rework into Sprints are resisted

Fixed scope – What you should do

- Push back as hard as you can on this trend
  - Explain how and why Agile works
  - Emphasize that this is not Agile
  - Emphasize the importance of building maintainable code and cost of rework

- RUN AWAY AND HIDE!

Questions?

Thank You