Agile Risk-based Testing

Lightweight Use of an Established Best Practice

RBCS

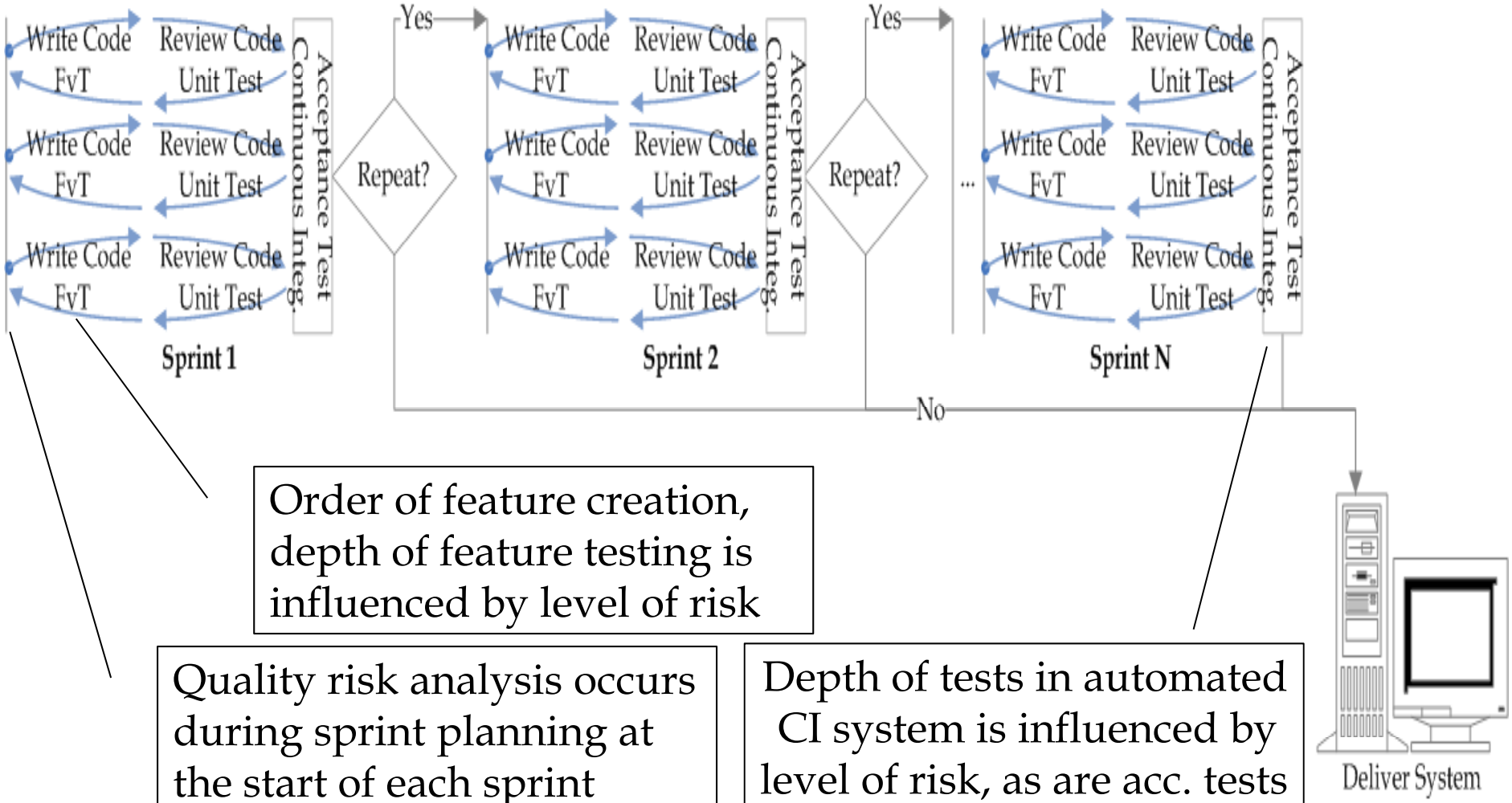
TIME TESTED.
TESTING IMPROVED.

www.RBCS-US.com

Introduction

- Risk-based testing is a proven best practice for test analysis, planning, estimation, design, execution, and results reporting
- Can you apply this best practice during agile projects?
- Sure, with a few minor modifications (and, of course, iteration)
- Let's see how this works…

Agile Lifecycle and Risk-based Testing



Order of feature creation, depth of feature testing is influenced by level of risk

Quality risk analysis occurs during sprint planning at the start of each sprint

Depth of tests in automated CI system is influenced by level of risk, as are acc. tests

Assessing Quality Risks on Agile Projects

- Select, allocate, and prioritize test conditions to maximize effectiveness and efficiency
- Quality risk analysis supports this process
  - Risk: a possible negative outcome
  - Level of risk: based on likelihood and impact
  - Quality risks: potential problems with product quality
  - Project risks: potential problems for project success
- Agile quality risk analysis occurs:
  - At a high level during release planning by business stakeholders
  - At a detailed level during iteration planning by the whole team
- In each iteration, the tester designs, implements, and executes tests for the risks

Quality Risks

- Quality risks include all features and attributes that can affect customer, user, stakeholder satisfaction
  - Incorrect calculations (functional)
  - Slow response time (non-functional performance risk)
  - Confusing interface (non-functional usability risk)
- Risk analysis prioritizes tasks and guides the sizing of the tasks
  - High risks require extensive testing, come earlier, and involve more story points
  - Low risks receive cursory testing, come later, and involve fewer story points
- Risk-based prioritization also includes release and iteration backlog items

Process of Quality Risk Analysis

- Agile quality risk analysis process (iteration planning)
  - Gather the agile team
  - List iteration backlog items
  - Identify functional, non-functional quality risks for each item
  - Assess identified risks: categorize each risk, determine risk level
  - Build consensus and ensure a good distribution of risk ratings
  - Use level of risk to choose extent of testing
  - Select appropriate test techniques for each risk item
- Adjustments may occur during an iteration
- Risk analysis may detect opportunities for early defect removal (e.g., problems in user stories)

Quality risks are potential system problems which could reduce user satisfaction

Risk priority number: Aggregate measure of problem risk

Impact: business, customer, or operational risk of problem

Likelihood: technical risk of problem

Tracing information to requirements, user story, or other risk bases

Quality Risk	Likeli-hood	Impact	Risk Pri. #	Extent of Testing	Tracing
Risk Category 1					
Risk 1					
Risk 2					
Risk *n*					

A hierarchy of risk categories can help organize the list and jog your memory.

1 = Very high
2 = High
3 = Medium
4 = Low
5 = Very low

The product of technical and business risk, from 1-25.

1-5 = Extensive
6-10 = Broad
11-15 = Cursory
16-20 = Opportunity
21-25 = Report bugs

Estimating Testing Effort

- During iteration planning, user stories are estimated
- Story size give implementation effort
- Risk level should influence story size
- Techniques such as planning poker can be used to reach consensus, involve whole team, and avoid missing anything
- Reliable estimation, including testing, is necessary for smooth work pace and meaningful velocity

Example: Allocating Test Effort

- Extensive: run large number of tests, both broad and deep, combine and vary interesting conditions, use all relevant techniques with strong coverage criteria
- Broad: run medium number of tests, exercise many different interesting conditions, use most relevant techniques with medium coverage criteria
- Cursory: run small number of tests, sample most interesting conditions, use efficient techniques with weak coverage criteria
- Opportunity: leverage other tests or activities to test 1-2 interesting conditions, investing very little time and effort, using reactive techniques especially
- Report bugs only: allocate only a small amount of extra time to report and manage these accidental bugs

Collaborative User Story Creation

- Developers, testers, and business stakeholders collaborate to capture requirements in user stories
- User stories include:
  - Functional and non-functional elements
  - Acceptance criteria for each element
- Testers bring a unique perspective to this process
  - Identify missing elements
  - Ask open-ended questions
  - Identify quality and project risks
  - Suggest tests for the user story
  - Confirm the acceptance criteria
- Acceptance criteria clarify the feature and establish clear completion measures

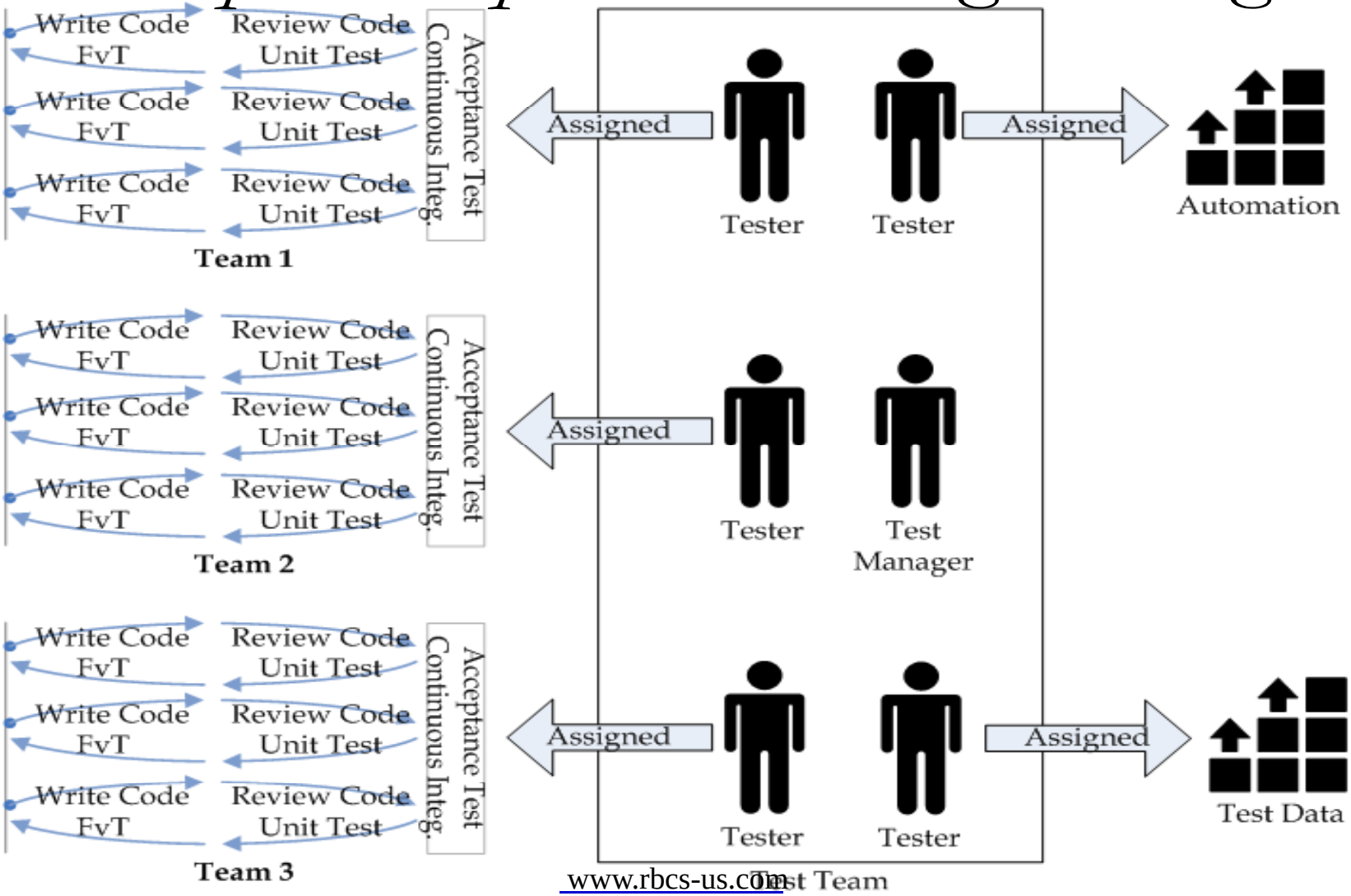
Creating User Stories

- INVEST technique
  - Independent
  - Negotiable
  - Valuable
  - Estimatable
  - Sized appropriately
  - Testable
- Collaborators can also brainstorm and mind map
- 3C elements:
  - Card: physical description of story and its benefits
  - Conversation: how the software will be used
  - Confirmation: checking of the acceptance criteria (positive and negative) by various participants
- User story docs: concise, sufficient, necessary

Example: Acceptance Criteria

- Assume you are testing a browser-based application
- Working with the product owner, you might define the following acceptance criteria
  - All screens display correctly on current and previous versions of Internet Explorer, Firefox, Safari, and Chrome
  - At 100% magnification, no horizontal scrolling is required to view any part of any screen

Example: Independent Testing and Agile



Status of Testing

- In agile projects, change happens, rapidly and often
- When the features change, so does product quality and quality risk!
- Change can make a mess of your status reporting processes if you're not careful
- Change also means that accurate test status is critical for smart team decisions
- Change can have a retroactive impact on features from previous iterations
- So, change often means existing tests must change and risks must be re-evaluated

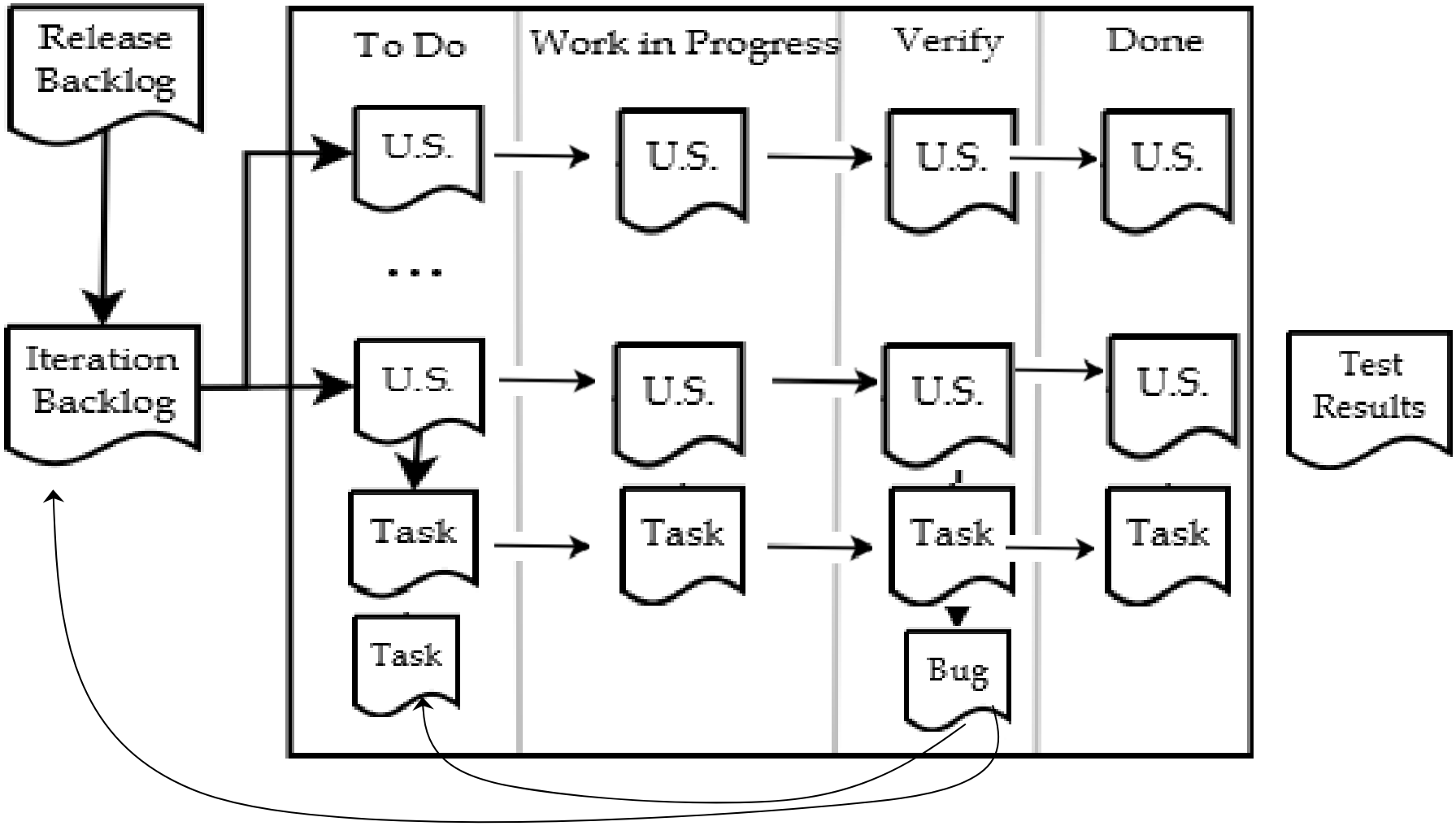
Communicating Test Results

- Test progress can be recorded using automated test results, agile task boards, and burndown charts
- Test status can be communicated via wikis, standard test management tools, and during stand-ups
- Project, product, and process metrics can be gathered (e.g., customer satisfaction, test pass/fail, defects found/fixed, test basis coverage, risks mitigated, etc.)
  - Metrics should be relevant and helpful
  - Metrics should never be misused
- Automating the gathering and reporting of status and metrics allows testers to focus on testing

Daily Stand-up Meetings

- In agile task boards, tasks move into columns (*to do*, *work in progress*, *verify*, and *done*)
- *Done* means all tests for the task pass and all risks associated with the task are sufficiently mitigated
- Task board status reviewed during stand-ups, which include testers and developers (whole team)
- Each attendee should address:
  - What have you completed since the last meeting?
  - What do you plan to complete by the next meeting?
  - What is getting in your way?
- Team discusses any blockages or delays for any task, and works collaboratively to resolve them

Example: Task Board



Conclusions

- Risk-based testing is a long-proven testing best practices
- In any lifecycle, proper risk-based testing requires complete integration into the lifecycle
- In Agile lifecycles, risk-based analysis, planning, estimation, design, execution, and results reporting permeate each iteration
- Risk awareness complements and heightens other Agile techniques (e.g., planning poker)
- Use risk-based testing for great agility