Introduction
Bob Galen

- Somewhere ‘north’ of 30 years experience 😊
- Various lifecycles – Waterfall variants, RUP, Agile, Chaos…
- Various domains – SaaS, Medical, Financial Services, Computer & Storage Systems, eCommerce, and Telecommunications
- Developer first, then Project Management / Leadership, then Testing
- Leveraged ‘pieces’ of Scrum in late 90’s; before ‘agile’ was ‘Agile’
- Agility @ Lucent in 2000 – 2001 using Extreme Programming
- Formally using Scrum since 2000
- Currently an independent Agile Coach (CSC – Certified Scrum Coach, one of 50 world-wide; 20+ in North America)
  - at RGCG, LLC and Director of Agile Solutions at Zenergy Technologies
- From Cary, North Carolina
- Connect w/me via LinkedIn and Twitter if you wish…

Bias Disclaimer:
Agile is THE BEST Methodology for Software Development...
However, NOT a Silver Bullet!
Outline

Introduction
1. Agile Goals & Criteria
2. Team Level: Done-Ness
3. Sprint Level: Story & Acceptance
4. Release Level: Freeze, Entry & Exit Criteria, and Release Criteria
5. Release Train / Influence Points
6. Q&A

The SCRUM Framework
Establishing Goals & Criteria
Why They Are Crucial?

- Agile teams are essentially self-directed, so plans don’t drive behavior or success...
- **People & Teams do and Goals drive each Team!**

  The teams then *swarm* around the goals, using their creativity and teamwork to figure out:
  - What’s most important
  - How to achieve it
  - Always looking for simple & creative—20% solutions
  - With focused and consistent quality practices & activities

Lean Principles Applied

- Most important aspects first
- KISS principle; no Gold-plating
- Small deliverables; worked on serially
- Deliver End-to-End behavior
  - In thin *Slices*
  - UI to Backend DB
- Driven to done, into *inventory* as completed components, as soon as possible!
  - Avoiding 90% done syndrome
  - Post-Sprint integration collaboration
- Rarely re-visited; mindset is to ruthlessly minimize rework
Product Owners – Hierarchical Goals

- The PO organization must coordinate
  - A hierarchical set of goals that drive the execution of the teams
  - Include release-centric goals
  - Include quality-centric goals
  - Include timing & workflow-centric goals

In conjunction with technology and project leadership

While often interfacing to operations and customer-facing organizations

Notions of Done-Ness

- Need to define “Done” from team members perspectives
- If you’re a developer, what does “I’m done with that story” mean?
  - Code complete; Code reviewed (paired)
  - Checked in – build successful; Unit tests developed – passed
  - Integration; QA collaboration
  - Run by the Product Owner; signed-off
- Every type of work should have a definition of Done-Ness! How else could you estimate the work?
Notions of Done-Ness
Testing

If you’re a tester, what does “I’m done with that story” mean?

- Test cases designed w/a broad view to test cases (unit, functional, acceptance, regression)
- Test cases pair-reviewed with development & test team members
- Test cases - checked into repository
- Ran test cases successfully; no issues
- Ran Acceptance Tests with the Product Owner
- Automated the Acceptance Test cases
- Connected the automation to the Continuous Integration environment
- Validated independent execution

Example: Salesforce.com
Done-Ness criteria, circa – 2010 Agile Conference

- **User Stories**
  - All defined Acceptance Criteria for a user story have been met.

- **Code**
  - Code implementing the user story functionality is checked in and follows department standards. This includes QE-reviewed automated tests checked in with all feature code.
  - No open regressions (you break it, you own it), with automated tests written for all regressions, and reviewed by QE.
  - No open P1 & P2 bugs for the implemented functionality in the sprint.
Example: Salesforce.com
Done-Ness criteria, circa – 2010 Agile Conference

- **Quality**
  - 70% of all test cases are automated and adhere to our automation coverage principles and standards.
  - Code Coverage of 70% (unless a different % discussed and agreed to by team).
  - Test plan, cases and execution for sprint functionality, and regression and cross functional test cases related to sprint functionality, need to be reviewed and entered into test repository with 100% of test cases in repository executed, and all P1/P2 cases passing.
  - All resolved bugs have been verified and closed for the sprint functionality.

- **Security**
  - Features adhere to security principles and standards with all critical issues resolved.
  - All high risk features have been Threat Modeled with the Product Security team. In depth security testing scheduled, if necessary, during the release.

- **Performance/Scalability**
  - Performance/Scalability impact of sprint functionality understood and quantified, and system testing scheduled
Example: Salesforce.com
Done-Ness criteria, circa – 2010 Agile Conference

- User Experience
  - UE has reviewed any new features or significant changes in the UI, and critical feedback has been incorporated, with all resulting P1 and P2 UI issues fixed.
  - Usability testing has been completed (unless deemed unnecessary), with all resulting P1 and P2 UI issues fixed.
  - Code and the UI have been reviewed to ensure 508 compliance - see the compliance checklist features that cannot be made compliant have been brought to the attention of the UE team

- Localization
  - All UI components have labels ready for localization vendors.

Example: Salesforce.com
Done-Ness criteria, circa – 2010 Agile Conference

- Documentation
  - User documentation that describes all aspects of the sprint functionality is complete and checked in.

- Product Metrics
  - Metrics to concretely measure customer usage (value delivery) of the sprint functionality have been defined.
Impact of Done-Ness Elsewhere

- It's not just an exit criteria!

- It's a heuristic for teams to check themselves as they collaborate in performing “acceptable” work
  - An consistently aligned ‘bar’ for professional engineering work

- It's also incredibly important in sizing the Product Backlog elements and in determining Release & Sprint plans
  - Including architecture & design and in working research spikes

Story Acceptance

- Each User Story should have acceptance criteria as part of the card

- They should focus on the verifiable behavior, core business logic, key requirements for the story

- Typically, they are crafted by the Product Owner
  - Leveraging skills of Business Analysts and Testers

- Story acceptance tests are normally automated and run as part of feature acceptance AND regression
  - FitNesse and Cucumber are among the Open Source tools enabling this
User Story Examples

As a dog owner, I want to sign-up for a kennel reservation over Christmas so that I get a confirmed spot

Verify individual as a registered pet owner
Verify that preferred members get 15% discount on basic service
Verify that preferred members get 25% discount on extended services and reservation priority over other members
Verify that past Christmas customers get reservation priority
Verify that declines get email with discount coupon for future services

Story – Sprint (Execution) Readiness

Prevents teams from taking on stories that are ill groomed or defined

Increases sprint success

✓ The story is well-written; and has a minimum of 5 Acceptance Tests defined
✓ The story has been sized to fit the teams velocity & sprint length: 1-13 points
✓ The team has vetted the story in several grooming sessions—it’s scope & nature is well understood
✓ If required, the story had a research-spike to explore (and refine) it’s architecture and design implications
✓ The team understands how to approach the testing of the stories’ functional and non-functional aspects
✓ Any dependencies to other stories and/or teams have been “connected” so that the story is synchronized and deliverable
✓ The story aligns with the Sprints’ Goal and is demonstrable
Code Freeze Dynamics still Apply!

- Try and front-load major features and high priority work

- Develop milestones for coalescing your code towards a freeze point
  - Enter your hardening Sprints with a specific freeze target
  - During hardening have a coding halt target

- May need layered freezes
  - UI versus Back-end Database layers
  - Database deployment script development

- Still may need to triage bugs to see what ‘fits’…usually in daily release Scrum of Scrums

Microsoft - Code Complete Model

- Microsoft employs a “code complete” strategy as defined below –

```
<table>
<thead>
<tr>
<th>Various project milestones</th>
<th>Visual Freeze</th>
<th>Feature Complete</th>
<th>Code Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release to manufacturing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stabilization leading to a zero bug fix release</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Typical Activities:
- Pilots
- Alpha & Beta testing
- Time buffer
- Iterative testing and rework
```
Schoor – Example of Repair Code Freeze

- Bruce Schoor has an extension to the Microsoft “code complete” model as defined below –

<table>
<thead>
<tr>
<th>Code Complete Milestone</th>
<th>UBF Unlimited Bug Fix</th>
<th>LBF Limited Bug Fix</th>
<th>RC Release Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Testing Focus</strong></td>
<td>Systematic Test Passes</td>
<td>Production &amp; Regression Tests</td>
<td>Acceptance Tests</td>
</tr>
<tr>
<td><strong>Release Goals</strong></td>
<td>Reduce # of High Priority Defects</td>
<td>Drive to Zero Defects</td>
<td>No More Repairs &amp; Release!</td>
</tr>
</tbody>
</table>

UBF - LBF Gate | LBF - RC Gate

Iteration / Goal Acceptance

- Each Scrum Sprint has a Product Owner determined goal
- Usually sprint success is not determined by the exact number of completed stories or tasks
- Instead, what most important is meeting the spirit of the goal

**Deliver a 6 minute demonstration of the software that demonstrates our most compelling value features and achieves venture capital investment interest**
Traditional
Entry & Exit Criteria

- Entry Criteria
  - Conditions that must be met prior to SQA / Testers beginning their testing efforts
  - Usually some sort of change log, content position
  - Smoke Testing (manual and/or automated) is a form of entry criteria – tied to execution / passing of focused testing

- Exit Criteria
  - Conditions that must be met prior to testing SQA / Testers completing testing on a specific deliverable or iteration
  - Normally includes coverage (test cases run, features completed)
  - Also includes quality attributes (pass rates, acceptable defect levels)

Traditional Smoke Testing
Automated “Entry” Criteria

- A set of tests that are run prior to SQA “accepting” a release for testing
  - Typically automated and “connected” to the build system
  - Intended to prevent wasted effort by SQA on broken releases (basic operations and core features)
  - Focus (tests) can / should change release over release
  - Programmatic form of release criteria
  - Usually defined collaboratively with and owned by the development team
Release Criteria

- Goals and objectives for the entire project release
- Usually they are multi-faceted, defining a broad set of conditions
  - Required artifacts & levels of detail
  - Testing activities or coverage levels
  - Quality or allowed defect levels
  - Results or performance metrics achievement levels
  - Collaboration with other groups – dependency management
  - Compliance levels
- That **IF MET** would mean the release could occur.

Release Criteria – Johanna Rothman

- Johanna Rothman – 5 Steps for definition
  1. Define Success
  2. Learn What’s Important for This Project
  3. Draft Release Criteria
  4. Make the Release Criteria SMART
     - Specific, Measurable, Attainable, Realistic and Trackable
  5. Gain Consensus on the Release Criteria
- Binary interpretation – (pass or fail) (go or no-go) intended to drive release decision-making
- Changed infrequently – holding to your initial goals!
Release Criteria – Rothman Example

Example Release Criteria

1. The code must support both Windows Vista and Windows XP
2. All priority P0, P1 and P6 defects will be repaired / addressed
3. For all documented bugs – on-line help, release notes and formal documentation will contain relevant updates
4. All QA tests run at 100% of expected coverage
5. No new P0 – P3 defects found within the last 3 weeks

To the left is a sample set of release criteria intended to guide activity for a given release. They speak to –

✓ Platforms
✓ Defects allowed
✓ Defect actions
✓ Coverage
✓ Maturity
✓ Performance
✓ Release date

Release Criteria – Another Format

An alternative approach is High / Low or Do / Don’t Do Contrasting Charts

High Priority / Focus

- Existing functionality can not be affected by new changes - (functional regression testing)
- Existing performance may not be degraded by new changes - (performance regression testing, we also lacked a performance benchmark)
- New functionality must work
- Component interoperability w/o performance regression

Low Priority / Focus

- Interfaces beyond 10/100/1000 Ethernet and ATM are lower priority
- Existing performance may not be degraded by new changes - even when running multiple components, don’t get hung up on improvement
- New functionality must work, except new reports that do not map to older reports
- Component interoperability across all permutations, we can identify (n) key
### Release Criteria – Another Format

#### Will Cover (In Bounds)
- Will run 100% of the existing regression suite prior to final release, must have > 95% pass rate of tests
- Will test all new features associated with higher performance network interfaces (GE 10, 100, 1000)
- Of the reporting options, will test the customer highest priority (most used) 50 for correctness.
- Performance will be tested at the 10Gb., 50% throughput level, with filtered recording for 10 hours.
- Will work with customer to demonstrate & run 100% of the acceptance tests.

#### Will Not Cover (Out-of-Bounds)
- Regression will not contain features presented in the last iteration and these will only be tested on a risk (< 20% of the tests) basis.
- Will not regress nor feature test on slower interfaces – less than 10 Gb.
- Of the remaining 250 reporting options, will only sample a few via exploratory testing.
- All other interface performance will not be tested. Also, the requirement is for 24 hours of record time and, at best, we can extrapolate from 10.
- There will be no usability testing nor support of the planned pilot releases.

---

### Product Owner Planning Levels in Large-Scale Agile Projects

<table>
<thead>
<tr>
<th>Level</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Vision</td>
<td>Yearly by Product Owner(s)</td>
</tr>
<tr>
<td>Product Roadmap</td>
<td>Semi-Annually by the Product Owner(s)</td>
</tr>
<tr>
<td>Release Plans</td>
<td>Quarterly by the Product Owner &amp; Team(s)</td>
</tr>
<tr>
<td>Iteration Plans</td>
<td>Monthly or bi-weekly by the Team(s)</td>
</tr>
<tr>
<td>Daily Plans</td>
<td>Daily by the team members</td>
</tr>
</tbody>
</table>
Release Train Management

- Iterative model with a release target
  - Product centric
  - Focused on a production push/release
- Synchronized Sprints across teams
  - Some teams are unsynchronized, but leads to less efficient cross-team (product) interactions
- Continuous Integration is the glue
  - Including automated unit and feature tests; partial regression
- Notion of a “Hardening Sprint”
  - Focused more on Integration & Regression testing
  - Assumption that it’s mostly automated
  - Environment promotion
- Define a final Hardening Sprint where the product is readied for release
  - Documentation, Support, Compliance, UAT, Training

Release Train Management

“Internal” Driving Forces

- Customer’s ability to “accept” the release
- Value being delivered in the release – purely scope
- Hardening Sprint “reality”
  - Time, Complexity, Automation, Size, Compliance, and Industry
- Internal team readiness
  - Customer support
  - Sales & Marketing readiness
  - Overall documentation & training
The Agile Release Train
Synchronized

Example: eCommerce / SaaS Model

10 days 10 days 5 + 2 days

Team 1 Team 2 Team 3 Team 4

Iterate Iterate Harden
Iterate Iterate Harden
Iterate Iterate Harden
Iterate Iterate Harden

Continuous Integration
Continuous Integration
Continuous Integration
Continuous Integration

Environment Evolution
Dev + QA Dev + QA QA -> Staging Production

Rinse & Repeat

External Release

Internal Release

X-team Harden Docs, Training

RGalen Consulting Group, LLC
Release Train Management
“Motivational” Driving Forces

- **Vision**
  - Schedule for delivery
  - Feature set(s); Minimal Marketable Features (MMF’s)
  - Goals for each team

- **Themes**
  - Which teams will be working on what components?
  - ‘Packages’ of User Stories
  - Prioritized by business value & need

- **End-to-End Use Cases**
  - Integration focused execution

Release Goal Setting
A Key for Coordination

- As you scale, each planning level should create criteria (Sprint Goals) that are –
  - Interrelated and cohesive
  - Focused towards the end product release and not simply on each teams deliverables
  - Identify dependencies and overall workflow

- The traditional notion of Chartering also applies at the higher levels, with Charters defined as:
  - Goals, Objectives & Scope
  - Clearly measurable view to “Done” – Release Criteria
  - Multi-faceted view towards quality (defects, coverage, non-functional requirements)
  - Allowing for team scope trade-offs
Levels of Criteria

<table>
<thead>
<tr>
<th>Activity</th>
<th>Criteria</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Team Work Products</td>
<td>Done’ness criteria</td>
<td>Pairing or pair inspections of code prior to check-in; or development, execution and passing of unit tests.</td>
</tr>
<tr>
<td>User Story or Theme Level</td>
<td>Acceptance Tests</td>
<td>Development of FitNesse based acceptance tests with the customer AND their successful execution and passing. Developed toward individual stories and/or themes for sets of stories.</td>
</tr>
<tr>
<td>Sprint or Iteration Level</td>
<td>Done’ness criteria</td>
<td>Defining a Sprint Goal that clarifies the feature development and all external dependencies associated with a sprint.</td>
</tr>
<tr>
<td>Release Level</td>
<td>Release criteria</td>
<td>Defining a broad set of conditions (artifacts, testing activities or coverage levels, results/metrics, collaboration with other groups, meeting compliance levels, etc.) that IF MET would mean the release could occur.</td>
</tr>
</tbody>
</table>

Exercise

- Gather in small groups; better that you are from the same company/organization OR those with similar characteristics
- In this section, we discussed the various levels of criteria that are important within agile teams.

1) Amongst your team, discuss how you’ve established goals & criteria in your Agile (or non-Agile) teams
   - What levels are the most important?
   - What levels don’t matter as much?

2) What part does the team play in definition? Should play?

3) Does defining what “done” means really matter? How?

4) If you had only one to pick, which would it be? And why?
Wrap-up

- Self-directed Agile teams are “directed” by their Goals. There are 4 Levels:
  1. Professional / Done-ness
  2. Feature / Story
  3. Iteration / Sprint
  4. Release Criteria

of goals driving Excellence in any agile team.

In Waterfall you get what you Plan for…
In Agile, you get what your Goals drive you towards…

Questions?

Thank you!
Contact Info

Bob Galen
Principal Consultant, RGalen Consulting Group, L.L.C.

Director of Agile Solutions, Zenergy Technologies,

Experience-driven agile focused training, coaching & consulting

Contact: (919) 272-0719
bob@rgalen.com
bob.galen@zenergytechnologies.com
www.rgalen.com

Blogs
Project Times - http://www.projecttimes.com/robert-galen/
My Podcast on all things ‘agile’ - http://www.meta-cast.com/

Copyright © 2013 RGCG, LLC