



QAI /QAAM 2011 Conference
"Proven Practices For Managing
and Testing IT Projects"



QA Integration as Part of Agile Adoption

Sivakumar
Thekkenaduvath

10th Annual Mid-Atlantic
Software Quality and Program
Management Conference

09/27/2011

Agenda

Introduction

The Agile Solution

Agile Manifesto

SCRUM Framework

Stories and Estimation

Enterprise SCRUM adoption

Shea's Work Systems Model for Change

Agile Release Train and QA

Continuous Integration and QA

Metrics

SCRUM Metrics



Agenda

Distributed SCRUM Team Structure

QA Integration in Agile Framework: Proposed Model

Agile/SCRUM Career Path

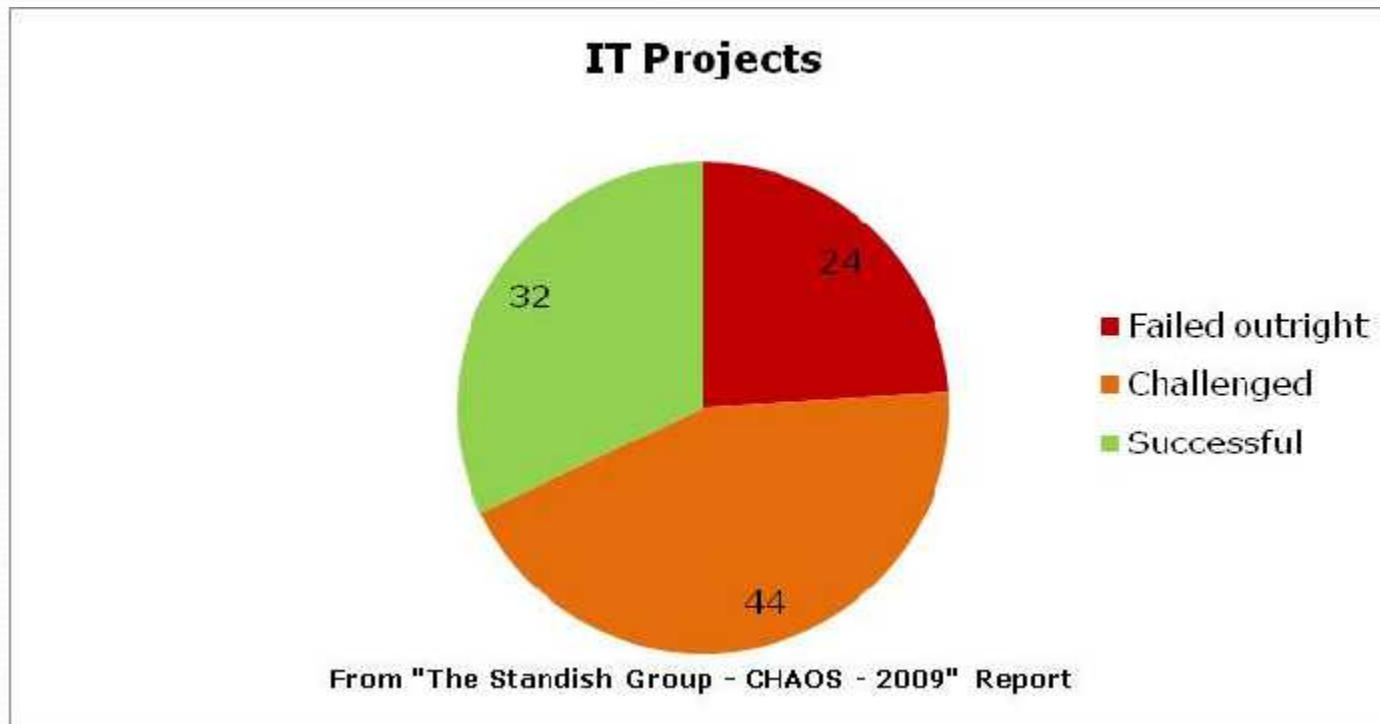
Distributed SCRUM and SoS for QA Integration



Introduction

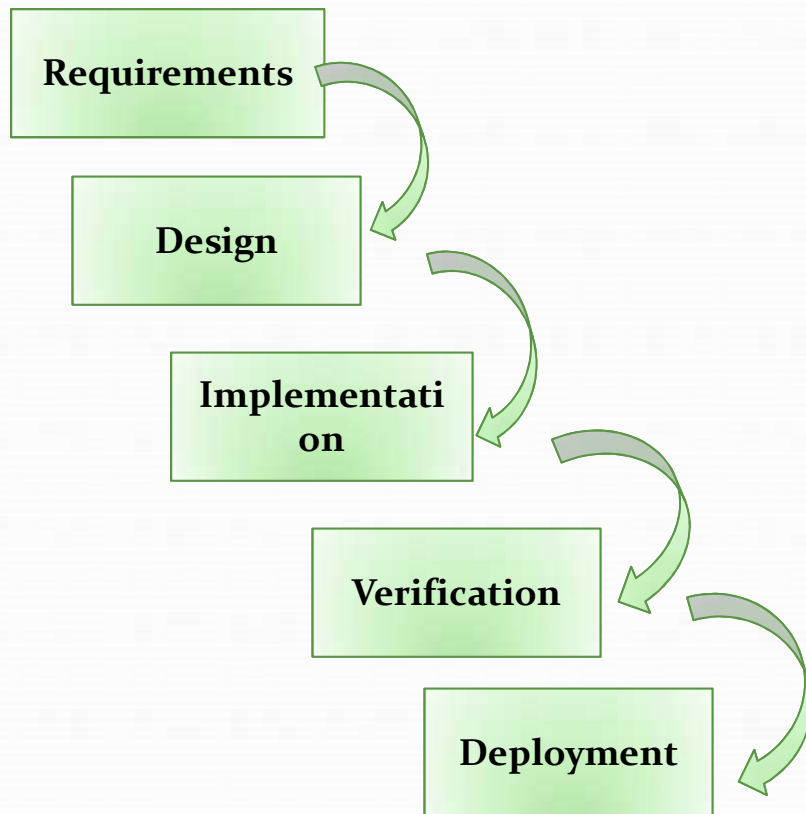
"If you always do what you always did, you always get what you always got"

- Anonymous



Traditional Software Development Cycle - Issues

Waterfall Methodology



The Problems

- Lengthy Development cycle dominated by Requirements and Design Phase
- The final product does not meet business goals because requirements change by the time development is completed
- Up to 45% of the features developed are not used
- End User sees product very late. Changes in direction are costly
- Methodology calls for extensive documentation, that must be changed repeatedly to stay current.
- IT departments struggle to adopt methodologies developed for other industries
- Business feels that IT is not an accelerator or a driver of value

The Agile Solution

Objective: Obtaining the smallest workable piece of functionality to **deliver business value early continually improving it**, adding further functionality throughout the life of the project.

Breaking up tasks

- Agile methods break tasks into small increments with optimal planning
- Agile methods produce completely developed and tested features every few weeks
- If a project being delivered under the waterfall method is cancelled at any point up to the end, there is nothing to show for it beyond a huge resources bill
- With the Agile method, being cancelled at any point will still leave the customer with some worthwhile code, that has likely already been put into live operation



Task broken down & Smaller work pieces delivered throughout the course of the project.

The Agile Manifesto



- 1. Individuals and interactions**
over processes and tools



- 3. Customer collaboration**
over contract negotiation



- 2. Working software**
over comprehensive
documentation



- 4. Responding to change**
over following a plan

Paths to Agile



The Grassroots Movement

- IT dev teams find Waterfall lifecycle too rigid and try out Agile



A Process Maturity Journey

- Agile chosen in preference to Waterfall by IT mgmnt it is seen as light-weight



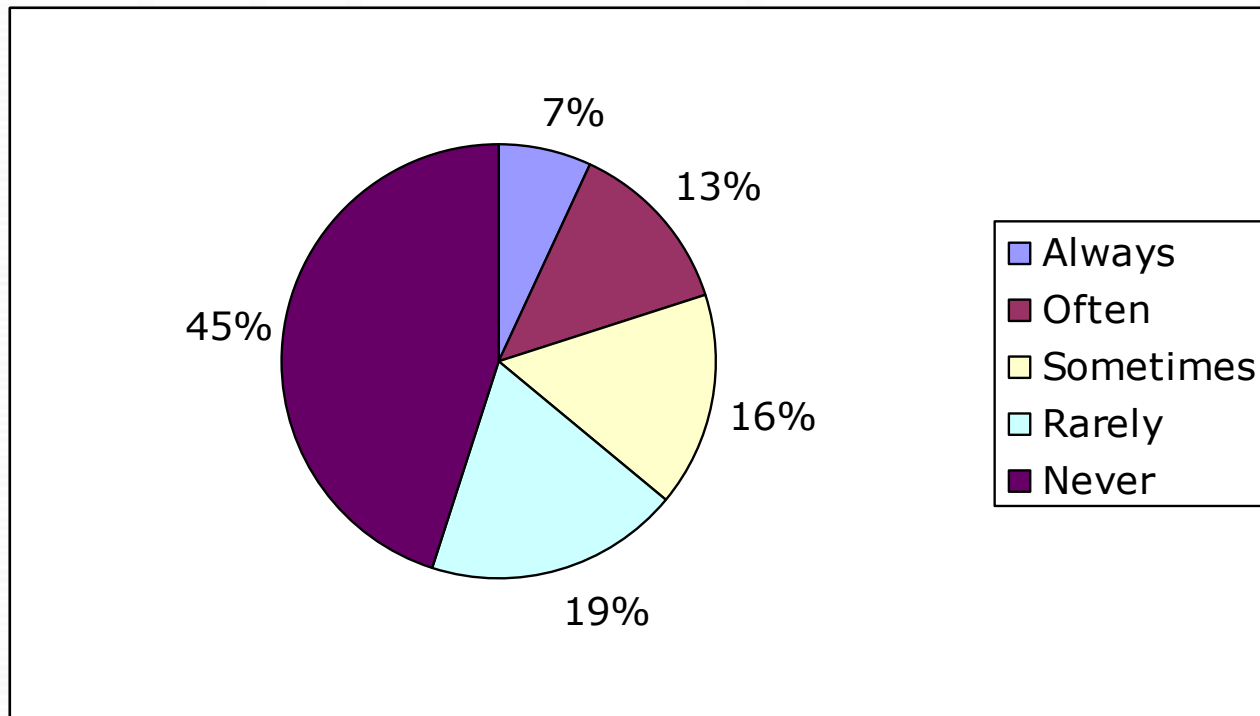
Business Driven

- Business asks IT for faster time to market

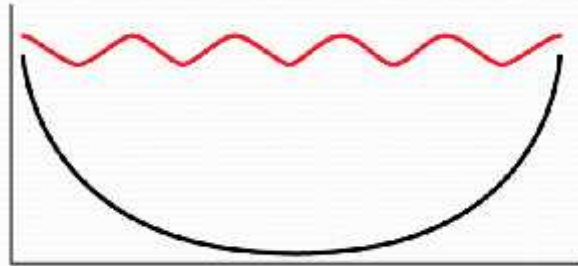
AGILE

Used features

- 7% of the features are Always Used
- 13% of the features are Often Used
- 16% of the features are Sometimes Used
- 19% of the features are Rarely Used
- 45% of the features are Never Used

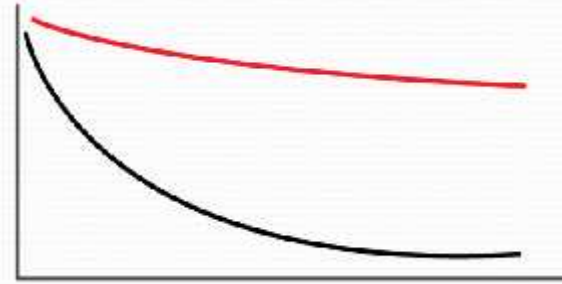


Agile Development Value Proposition



Visibility

- Business user intervention
- Continuous progress evaluation



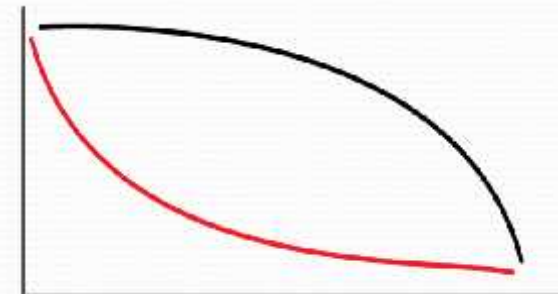
Adaptability

- Easily adapting to changing requirements
- Delivering software satisfying desired Business Needs



Business Value

- Accelerated initial delivery of business value
- Maximized value by continuous planning and feedback



Risk

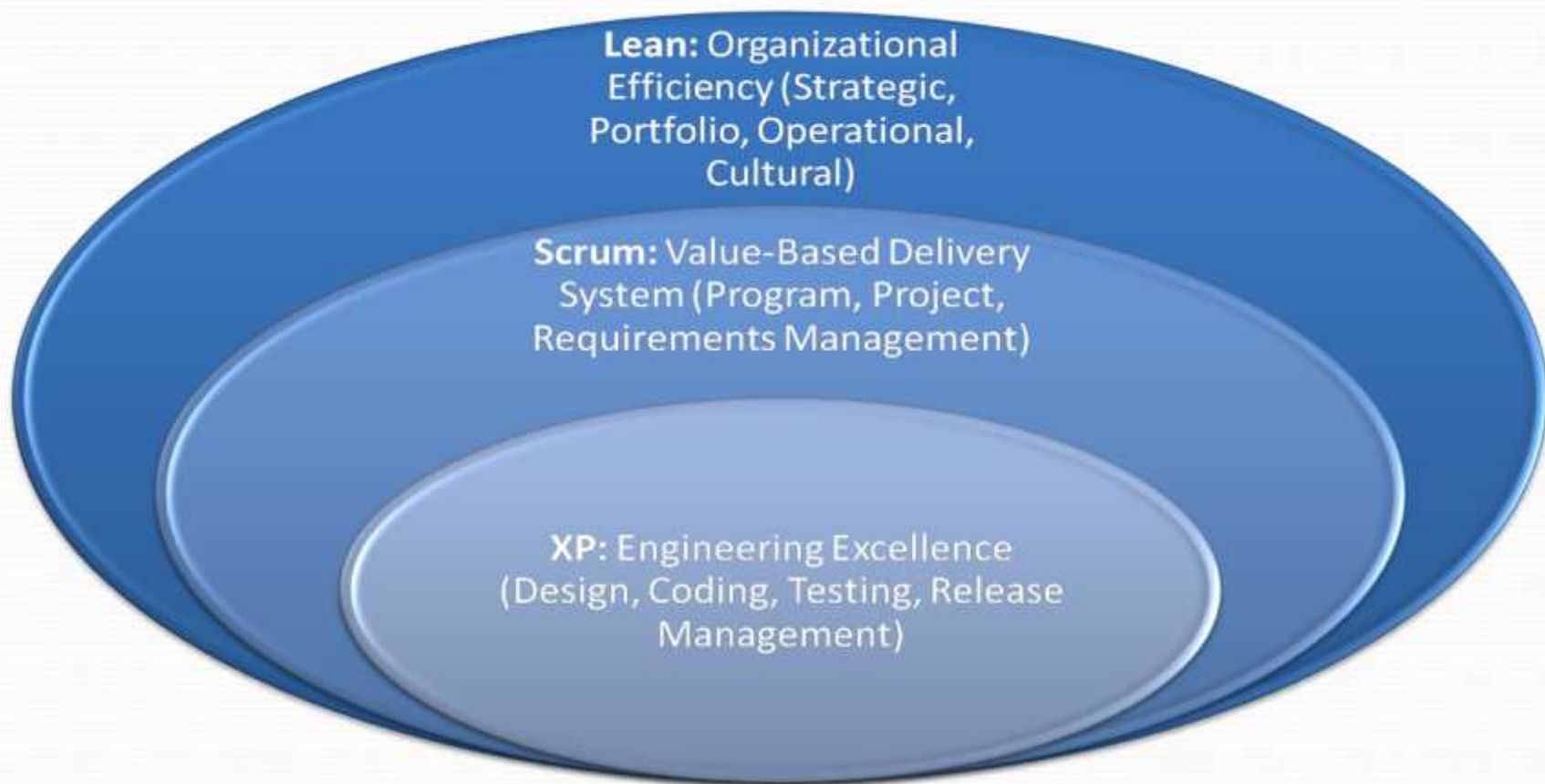
- Focus on delivering business value
- Reduced risk

Agile Development

Traditional Development

What is Agile ?

Agile software development refers to a group of methods based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. These methods focus on all aspects of the development process – engineering practices, incremental delivery, and organizational efficiency.



SCRUM

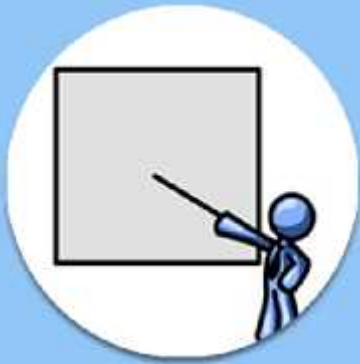
- Not an acronym
- Not a process or a technique
- Iterative, Incremental Framework within which we can employ various processes and techniques
- Scrum (an abbreviated form of scrummage, which is now rarely used, except as a verb), in the sports of RUGBY, is a way of restarting the game, either after an accidental infringement or when the ball has gone out of play. Scrums occur more often, and are of greater importance, in union than in league



Three pillars of SCRUM

Leg	Daily Scrum Meeting	Sprint Review and Planning	Sprint Retrospective
Transperancy	Product Owner, ScrumMaster & Team: 1. What did you do yesterday? 2. What will you do today? 3. What obstacles got in your way?	Team, ScrumMaster, Product Owner	Team, ScrumMaster, Product Owner
Inspection	To inspect progress toward the Sprint goal	To inspect progress toward the Release Goal	To review the past Sprint
Adaptation	To make adaptations that optimize the value of the next work day	to make adaptations that optimize the value of the next Sprint	determine what adaptations will make the next Sprint more productive, fulfilling, and enjoyable.

Scrum Roles / Descriptions



Scrum Master

- Is responsible for ensuring that the Scrum Team adheres to Scrum values, practices, and rules
- Coaches/leads SCRUM to be more productive and produce higher quality products
- When the Scrum Master helps make these changes, this is called “removing impediments”
- However, the Scrum Master does not manage the Scrum Team; the Scrum Team is self-organizing



Product Owner

- The Product Owner is the one and only person responsible for managing the Product Backlog and ensuring the value of the work the Team performs
- This person maintains the Product Backlog and ensures that it is visible to everyone
- For in-house development efforts, the Product Owner could be the manager of the business function that is being automated



Team

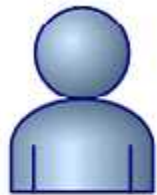
- Teams of developers turn Product Backlog into increments of potentially shippable functionality every Sprint
- Teams are also cross-functional; Team members must have all of the skills necessary to create an increment of work
- Teams are also self-organizing. No one – not even the Scrum Master -- tells the Team how to turn Product Backlog into increments of shippable functionality
- The optimal size for a Team is seven people, plus or minus two

The Scrum framework



And, it's not just for development

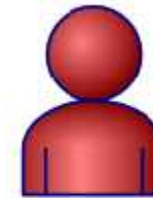
Product Backlog



Product Owner/Business Contact

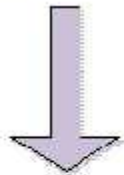


Team



Scrum Master

1. <FEATURE_001>
2. <FEATURE_002>
3. <DEFECT_001>
4. <DEFECT_002>
5. <>
6. <>
7. <>
8. <>
9. <>
10. <>



Pricitized list of Features, Defects to be fixed etc.

Product Backlog

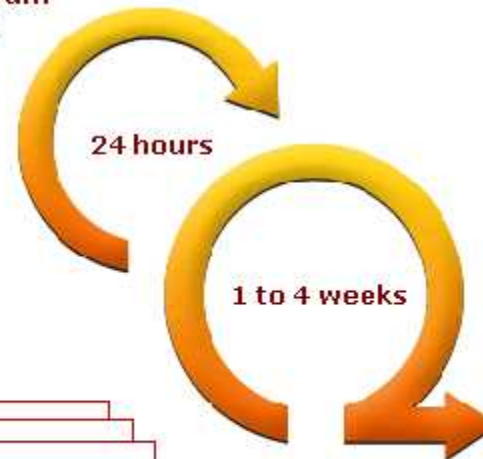
Sprint Planning Meeting

Team identifies stories to deliver the high-priority items for the sprint.

Sprint Backlog

Stories will be further broken down into various tasks

Daily Scrum Meeting



24 hours

1 to 4 weeks



Tasks

Sprint: Review

Sprint Retrospective

var 1.3
10/27/2009

Finished Work

Sprint 0

Major Elements of a Story

- Description
- Epic/Feature/MMF
- Story Points
- Business Value Points
- Validation Criteria
- Tasks
 - # of hours (actual and remaining)
 - Ownership
 - Status

Epic, Feature and Stories

- Epic/Feature: Collection/container of stories
- Story: Fundamental unit of work in SCRUM
 - “Describes some item of value to a user or product owner”
- Stories can be both functional and non-functional

.....> Analysis...

Epic

Feature

MMF

User Story

Right sized
User Stories

Validation Criteria

- Typical story format:
 - As a <user type>
 - I want to <task>
 - So that <business benefit>

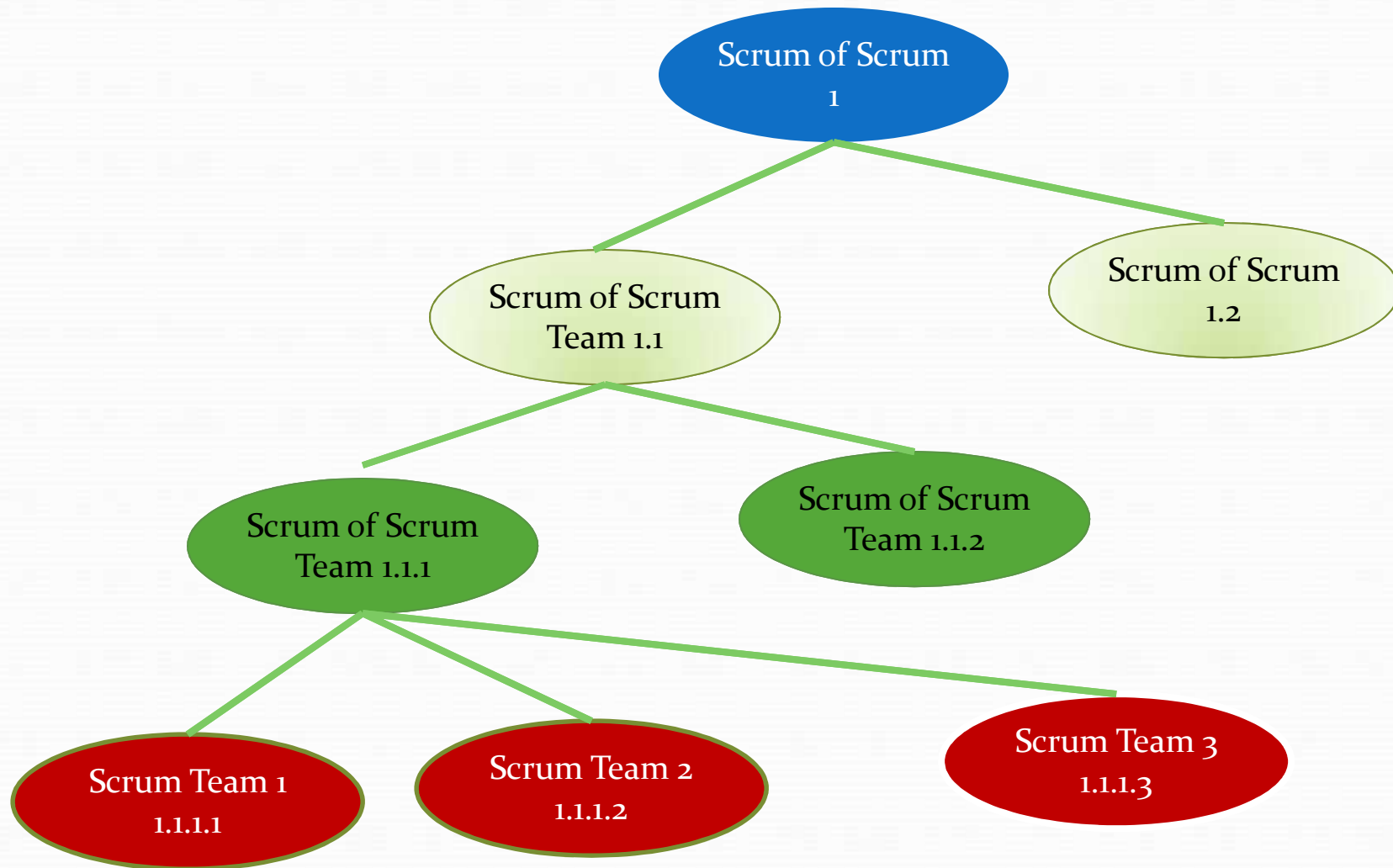
e.g.

As a client relationship manager I want to provide an image editing software to my clients so that they can effectively edit the geographical images.

Story Points/Sizing

- Estimate NOT upfront – make them along the way
- Example – how long it takes to drive somewhere?
 - Better precision, closer destination!
- If I leave this room and arrives at Newport News airport
 - What is the single point estimate for the time?
 - What are the 1%, 60% and 98% times?
- Unit less measurement
 - 1,2,3,5,8,13,20,40,100
- Story points to clarify the story acceptance criteria – more discussions on the requirements
 - clarity for development

Enterprise SCRUM Adoption



Enterprise SCRUM Adoption – an e.g.



Organizational Changes on SCRUM adoption

- Increased collaboration as Business Owners becomes more involved in Software Development
- Increased responsiveness and adaptability necessitated by iterative development in short sprints.
- Increased accountability because shippable code has to be produced at the end of each sprint. Empowerment is with those who have accountability.
- Organizational structure may need change as roles descriptions, and roles may not be applicable in new methodology.
- Team is expected to be multi-skilled. Cross Training becomes necessary. Adaptability becomes key attribute in team selection. Specialists may be uncomfortable.
- Workplace may need change to accommodate increased collaboration.
- Reward System may have to change. Glory is for Team , not individuals.

Shea's Work Systems Model for Change

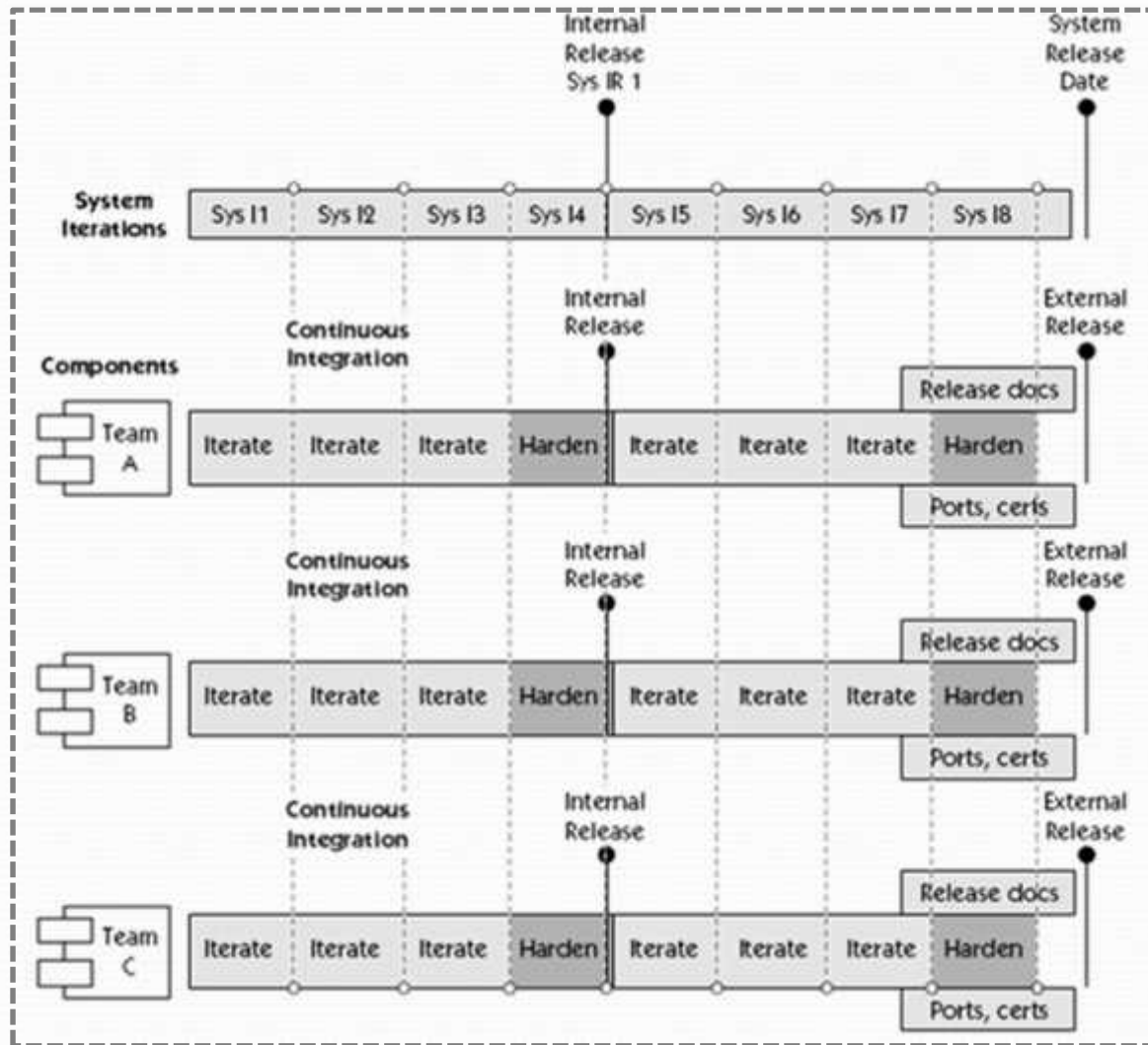
Work Systems Model



Work Systems. Source: Copyright Shea & Associates, Inc.

The Agile Release Train (ART)

A synchronized agile release train



- All component teams are synchronized to the same iteration schedule
- Continuous integration has been applied at the system level, so the system now has iterations and internal releases available for customer assessment just like the components do. This gives the team the objective evidence it needs to adjust system and component scope in real time and to shake out the issues that lie around the interfaces of the system.
- The risk is addressed early, rather than late, in the release cycle
- The slowest team no longer drives the train. Its relatively slower progress is detected early, scope is adjusted or resources are added.

Source: *Scaling Software Agility: Best Practices for Large Enterprises*, Addison Wesley, 2007

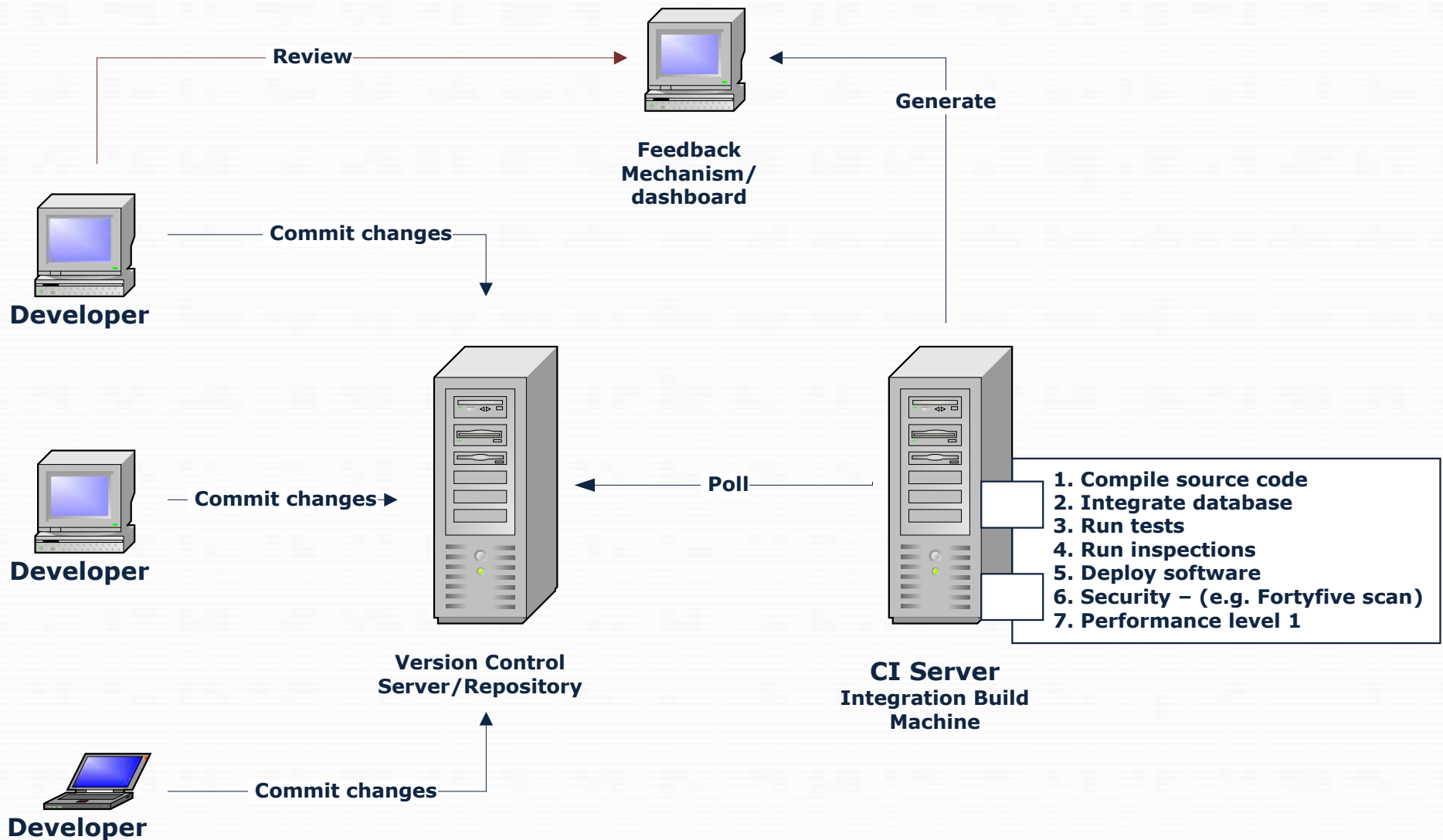
Continuous Integration (CI) and QA

- Biweekly, weekly or daily builds
- Faster and in small increments
- Each integration is verified by an automated build to detect integration issues
- Improves the overall quality of software product

Value of CI

- Reduce risks
- Reduce repetitive manual processes
- Generate deployable software at anytime and at any place
- Enable better project visibility
- Establish greater confidence in the product

Continuous Integration (CI) Contd.



Metrics

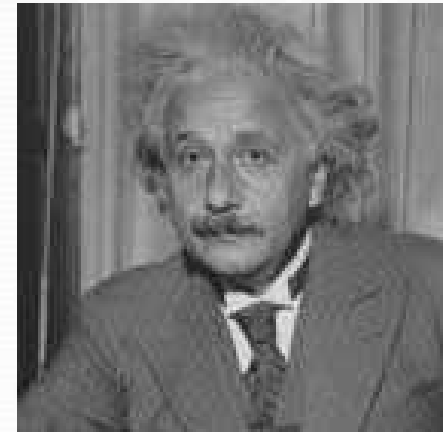
"Not everything that counts can be counted,
and not everything that can be counted
counts."

- Albert Einstein

Some things we can measure.

Some things we can't.

And just because we can measure
something doesn't make it more
real or significant.



Ref: http://www.ideachampions.com/weblogs/archives/2008/08/not_everything.shtml

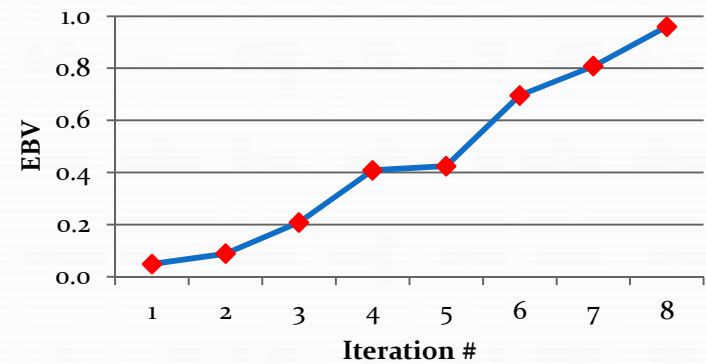
Heuristics for Choosing Metrics

- Affirms and reinforces Agile Principles
- Follows trend not numbers (aggregate above team)
- Less is More - A focused set of Metrics and Diagnostics
- Measure outcome not output
- Make it easy to collect
- Reveals rather than conceals its context and significant variables
- Provides Fuel For Meaningful Conversation
- Provides feedback on a frequent and regular basis
- Measure value or processes
- Encourage "Minimum Deliverables"

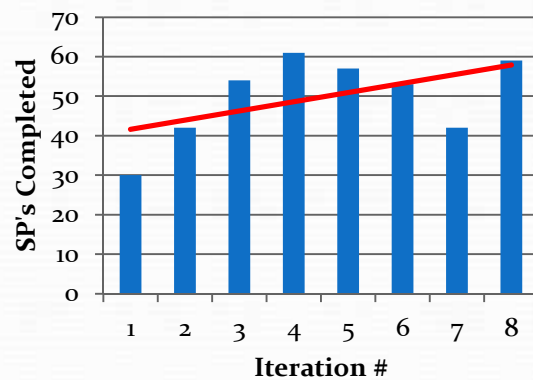
SCRUM Metrics

Metric	Definition	Level
Earned Business Value	Percentage of Estimated BV that has been achieved	Iteration, Release
Velocity	Rate at which SBIs 's are completed per Iteration	Iteration, Release
Burn Up	# of SBIs s completed against plan	Daily within Iteration
Requirements Stability Index	1- (# of Backlog Items Changed/ # of Total Backlog Items)	Iteration Release
Defect Density	# of defects reported per SP	Iteration, Release

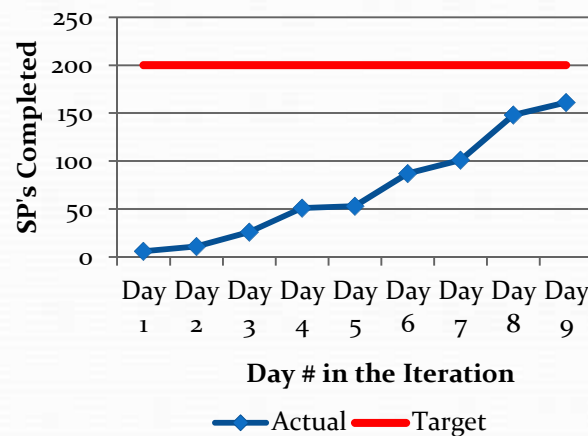
Earned Business Value



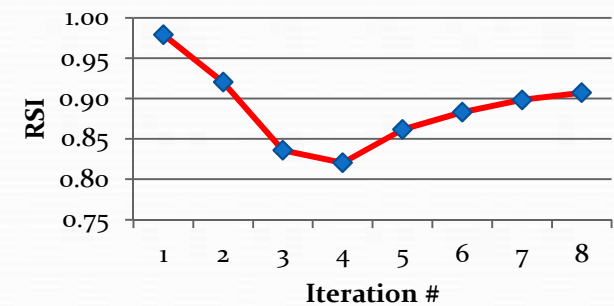
Velocity



Burn Up



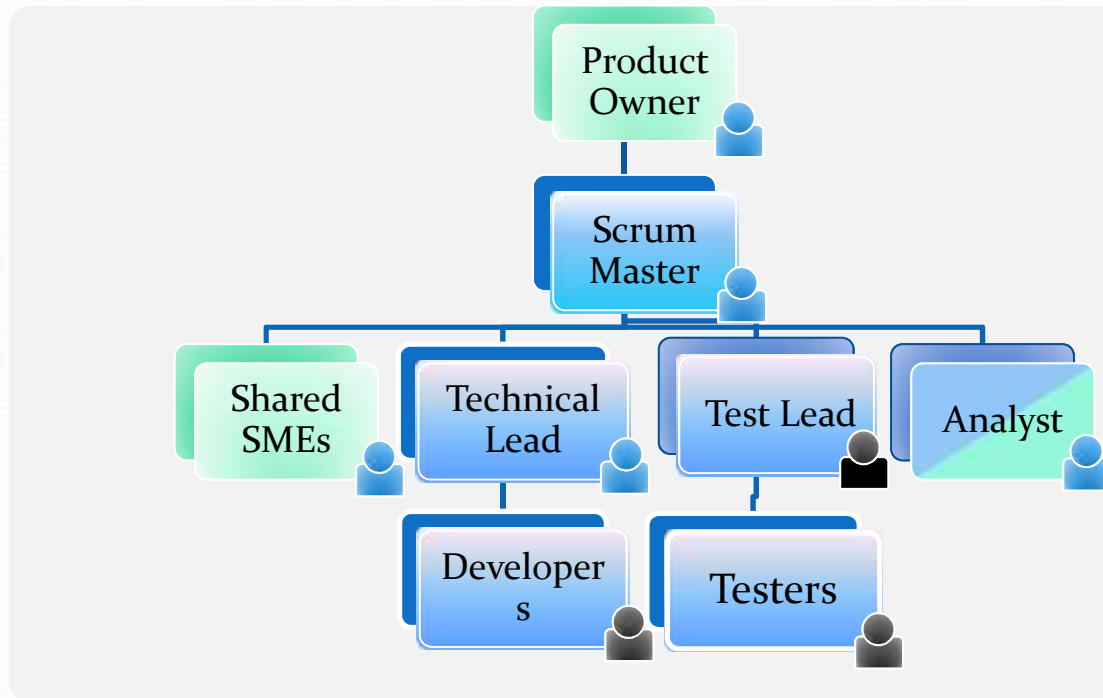
Requirements Stability



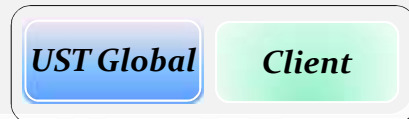
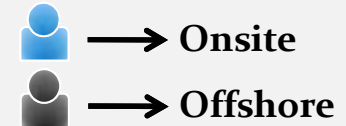
SCRUM in a Distributed Team Environment

- More documentation needed than for co-located teams
 - Documentation should not replace communication
- Teams at all locations should commit to some overlapping working hours every day
 - Will ensure sufficient real time interaction
 - Will improve team bonding and trust due to the spirit of give and take
- Co-locate by rotating key personnel between different locations
 - After an important release, or to review product road map, or perform training, etc.
 - Will result in creating a better cultural understanding of each team's environment
- Should use collaboration tools like Video conferences, Web cam, web meetings, TFS, etc., extensively
 - To improve team bonding
 - Reduce communication gap
- Strong leadership required in all locations to ensure that respective teams are communicating effectively
- Teams may conduct separate Scrum activities (planning, Daily standup) between onshore and offshore
 - Allows for team members to work directly in each location
 - Representatives of each location will attend others' meetings

Distributed SCRUM Team Structure

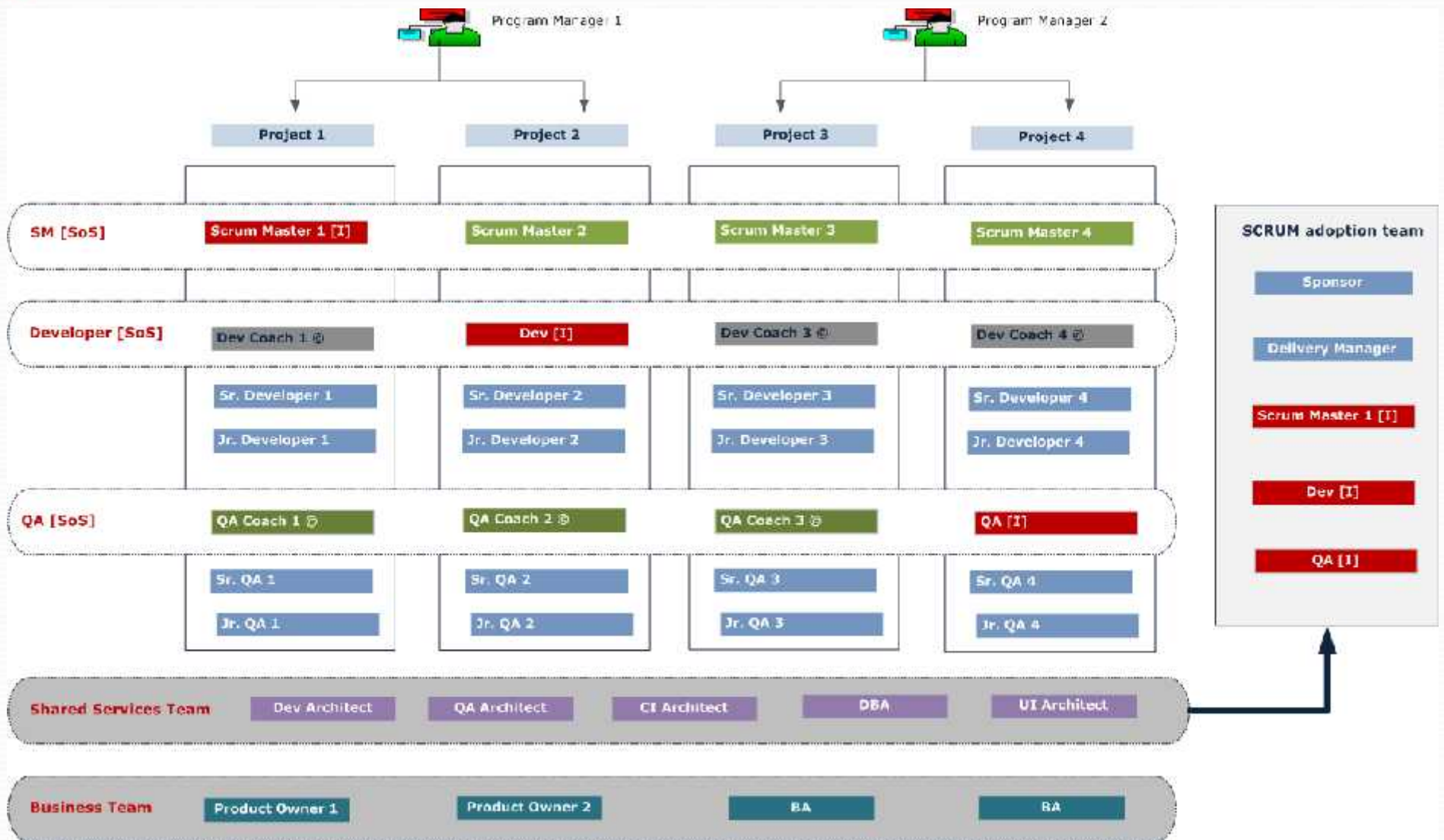


<i>Typical Team</i>	
<i>Role</i>	<i>Count</i>
<i>Scrum Master</i>	<i>1</i>
<i>Analysts</i>	<i>1-2</i>
<i>Tech Lead</i>	<i>1</i>
<i>Test Lead</i>	<i>1</i>
<i>Offshore Dev</i>	<i>4</i>
<i>Offshore Test</i>	<i>2</i>

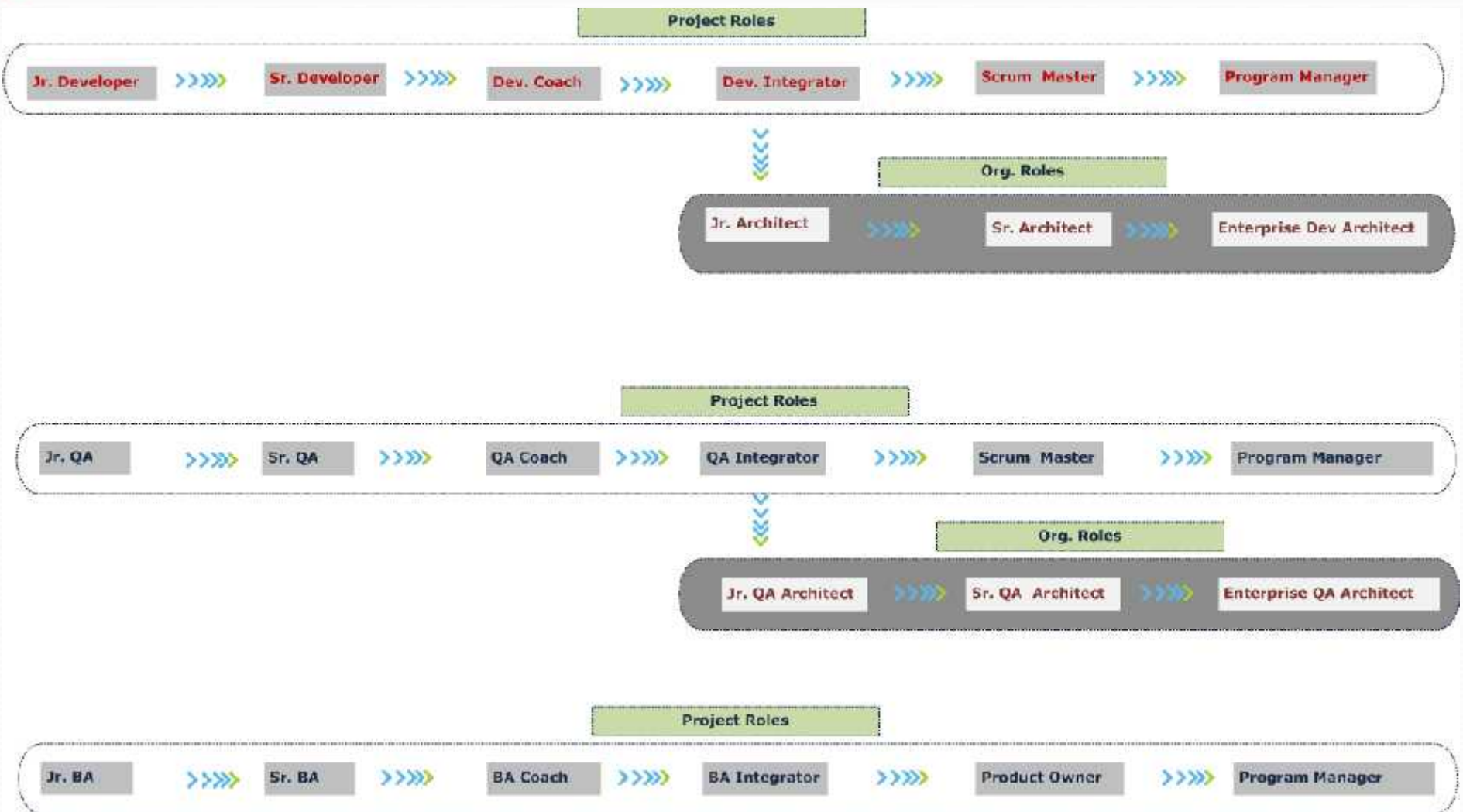


- Suggested for Scrum teams with three or more developers and/or three or more testers
- Cost effective and Leverages economies of UST's Global Delivery model
- On-site technical leads facilitate communication with off-shore developers and testers
- Increased flexibility to cost effectively expand and contract off-shore resources

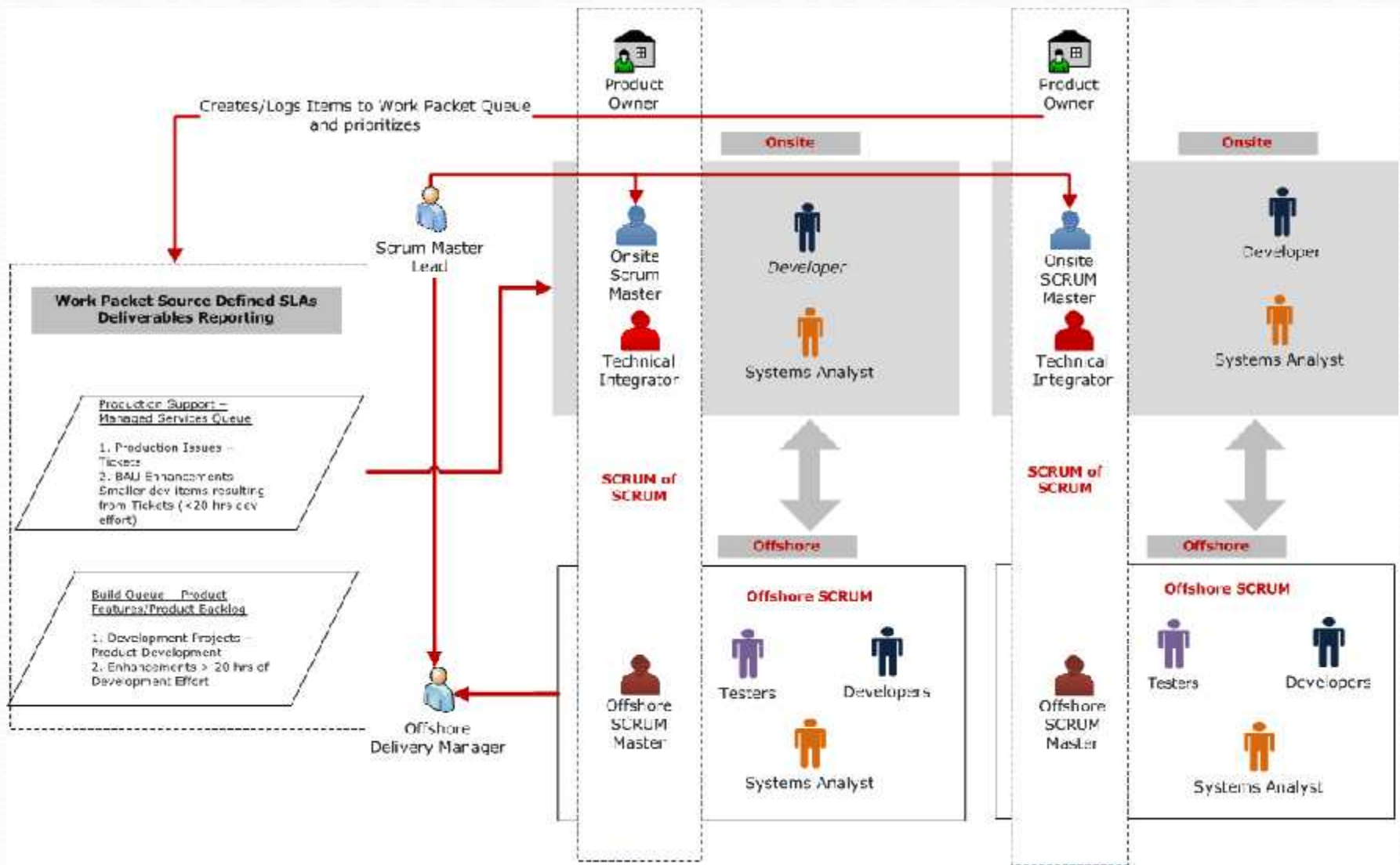
QA Integration in Agile Framework: Proposed Model



Agile/SCRUM Career Path



Distributed SCRUM and SoS for QA Integration





Thanks for your time!



Q & A



Appendix - I