CERTIFIED SCRUM PRODUCT OWNER COURSE

Course Workbook

Name:

v. 1.5.2

COURSE AGENDA

This is what you can expect to see during these two days (the times are approximate):

	Day 1	Day 2
9 – 12	Business Value Game	Project kick-off / planning exercise - Vision - Writing user stories - Collaborative exercises to analyze and prioritize the user stories in the product backlog
12 – 13	Lunch	
13 – 15	Agile thinking Agile principles	Release planning
15 – 17	Scrum framework Scrum roles in detail	Elective topics, Q&A

Breaks will be held roughly every 60-90 minutes (for 10-15 minutes each). If you have requests for particular break times, come talk to me about.

CERTIFICATION

According to Scrum Alliance, CSPO certification requires approved participation on a CSPO course run by a Certified Scrum Trainer (CST).

For me to send your name forth to Scrum Alliance, Scrum Alliance requires you to participate actively on this course for the full two days. If you have something that absolutely requires you be absent for a portion of the class, talk to me and we'll agree what to do.

After the course, I will send your name and email to Scrum Alliance. You will then receive an email from Scrum Alliance containing an Internet link that you must use to confirm your certification.

Once you've done that, you have formally become a Certified Scrum Product Owner.

Congratulations!!

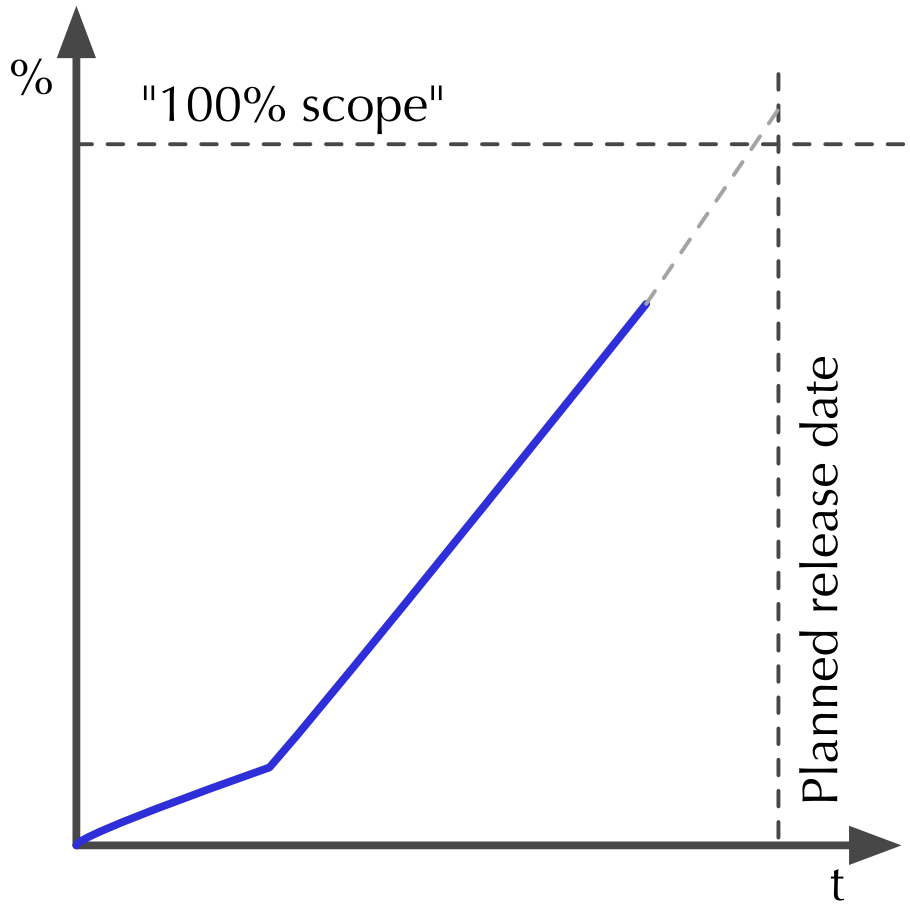
BUSINESS VALUE GAME

Please write here the key ideas you learned from the Business Value Game.

If you're interested in playing the Business Value Game, you can find it here:
http://www.agilebelgium.be/businessvaluegame/

WHY AGILE FOR PROJECT MANAGEMENT?

Please fill up the diagram as we go through it and write your notes to it.

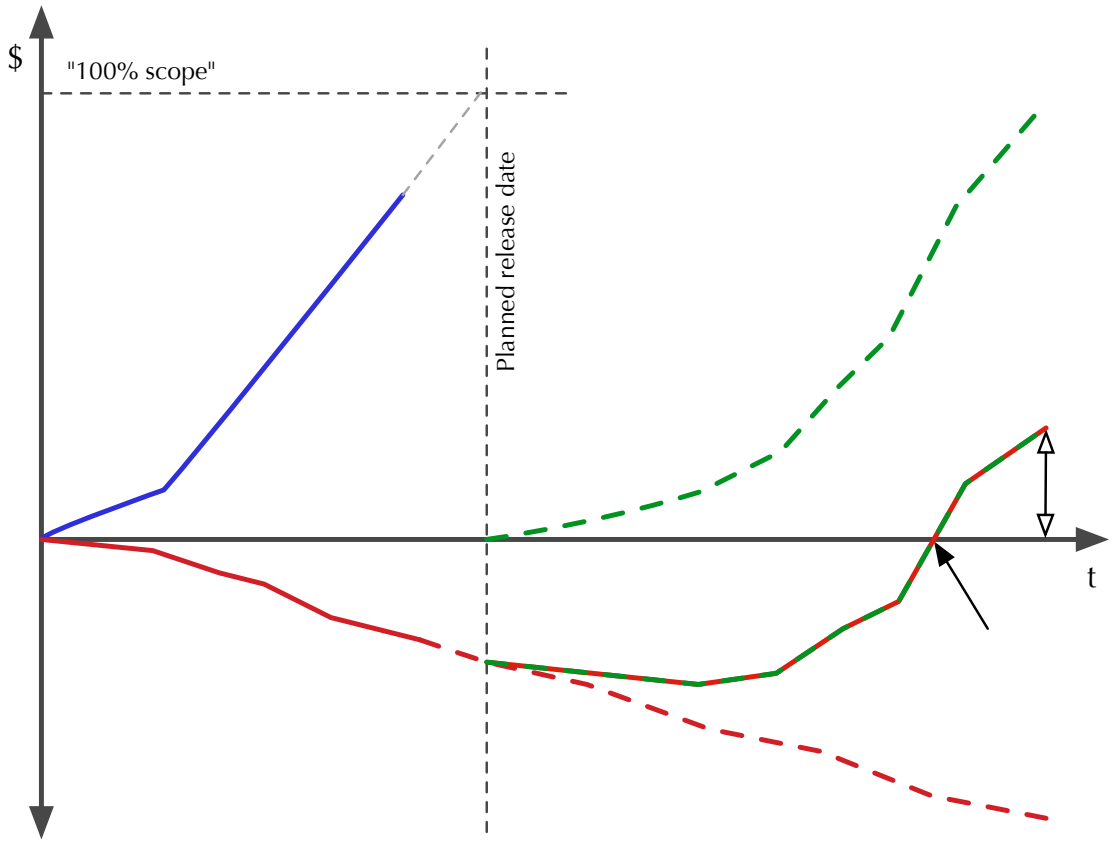


"There are those who look at things the way they are, and ask why... I dream of things that never were, and ask why not?"

– Robert Kennedy

WHY AGILE FOR BUSINESS?

As this topic is discussed, update the diagram below with conceptual levels and elements that are related to each level.

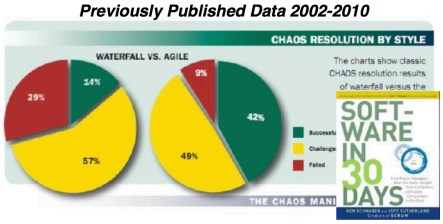


Agile Success Rates 2011-2015

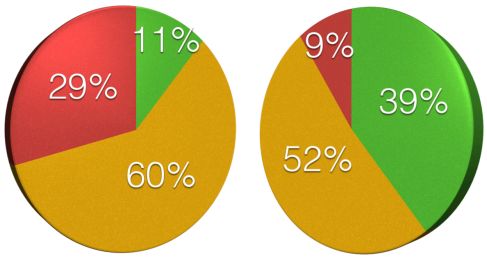
Standish Group International, Inc.

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

Previously Published Data 2002-2010



Waterfall vs Agile 2011-2015 Data



scruminc.

AGILE VALUES AND PRINCIPLES

MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

PRINCIPLES OF LEAN SOFTWARE DEVELOPMENT

1. Eliminate waste
2. Amplify learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build integrity in
7. See the whole

THE 7 WASTES

Production	Product Development
Inventory	Work in progress
Overproduction	Unnecessary features
Extra processing	Unnecessary activities
Transportation	Hand-offs and task switching
Waiting	Delays
Motion	Relearning
Defects	Defects

12 PRINCIPLES OF THE MANIFESTO

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity – the art of maximizing the amount of work not done – is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

COMPLEXITY AND CAUSALITY

Quoting from http://www.noop.nl/2008/08/simple-vs-complicated-vs-complex-vs-chaotic.html,

My car key is simple.

It took me about three seconds to understand how my car key works. OK, maybe that's not quite correct. Mine has a battery in it. If I take it apart it might take me another three hours to understand its details. But yeah, I'm smart, I'll manage.

My car is complicated.

It would take me years to understand how my car works. And I don't intend to. But if I did, then some day in the far future I would know with certainty the purpose of each mechanism and each electrical circuit. I would fully understand how to control it, and I would be able to take my car apart and reassemble it, driving it exactly as I did before.

Car traffic is complex.

I can travel up and down the same street for twenty years, and things would be different every time. There is no way to fully understand and know what happens around me on the road when I drive, how other drivers operate their vehicles, and how the people in the streets interact. I can make guesses, and I can gain experience in predicting outcomes. But I will never know for sure.

Car traffic in Lagos (Nigeria) is chaotic.

When things get too complex, they easily become chaotic. Traffic in Lagos is so bad, it is not even predictable. Poor infrastructure and planning, heaps of waste, pollution, lack of security, floods, and many more problems make it one of the worst places in the world to be, as a simple car driver.

BUILD YOUR OWN SCRUM

With your table group, illustrate the Scrum framework in a way that makes most sense to you. You will do this part using cut-out pictures given by the instructor.

Below, you can draw your diagram at the end of the exercise.

INSTRUCTOR'S SCRUM

If you want to do this exercise in your own training sessions, you can find the latest version from http://weisbart.com/byos/

SCRUM ROLES VS. TRADITIONAL PROJECT MANAGER

In small groups, discuss the different kinds of responsibilities and activities traditional project managers typically have. Then discuss how those responsibilities and activities would map to Scrum roles. Please write your mapping to the diagram below.

Product Owner responsibilities

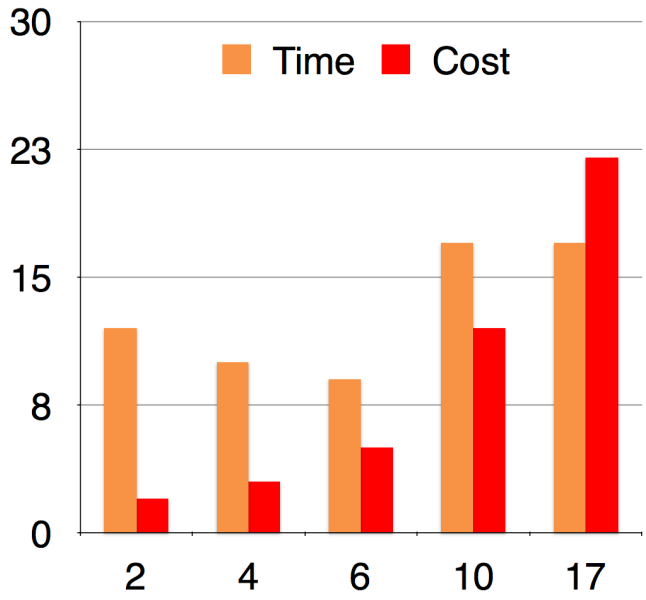
ScrumMaster responsibilities

Development Team responsibilities

Responsibilities not related to Scrum

IDEAL TEAM?

Due to communication saturation, as the size of the Team increases, their ability to get things done decreases.



From Scrum Inc. CSM slide deck, original source:
http://www.qsm.com/process_01.html
(491 projects)

SCRUM TEAM HYPERPRODUCTIVE PATTERN LANGUAGE

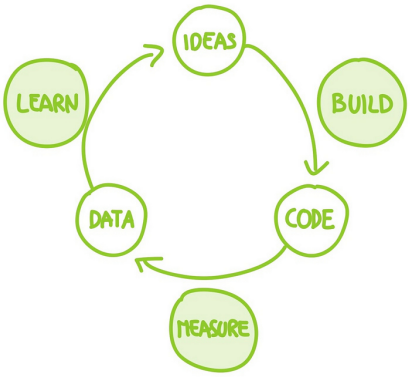
Teams that finish early accelerate faster:

- Stable Teams – Retain knowledge and team formation, aim for 80% annual stability
- Yesterday's Weather – Do not pull too much work into a Sprint
- Swarming – Getting work done quickly by collaborating on items at the same time
- Interrupt Pattern – Manage interruptions during the sprint
- Daily Clean Code – Deliver a defect-free product at Sprint end, and every day
- Scrum Emergency Procedure - Stop the line when plans are no longer realistic
- Scrumming the Scrum – Put the one improvement item to the top of Sprint Backlog
- Happiness metric – Do not overburden the Team, happy people are more productive

Adapted from: Scrum, Inc. CSM course materials. Original source: Teams That Finish Early Accelerate Faster: A Pattern Language for High Performing Scrum Teams
47th Hawaii International Conference on System Sciences (HICSS)
By Jeff Sutherland, Neil Harrison, Joel Riddle, January 2014

BUILDING A BUSINESS CASE

Nothing is worth building, unless we know its value. Or at least have high enough a confidence that it's worth moving on. Lean Startup gives a good framework for understanding and managing the early life of a product or development idea.

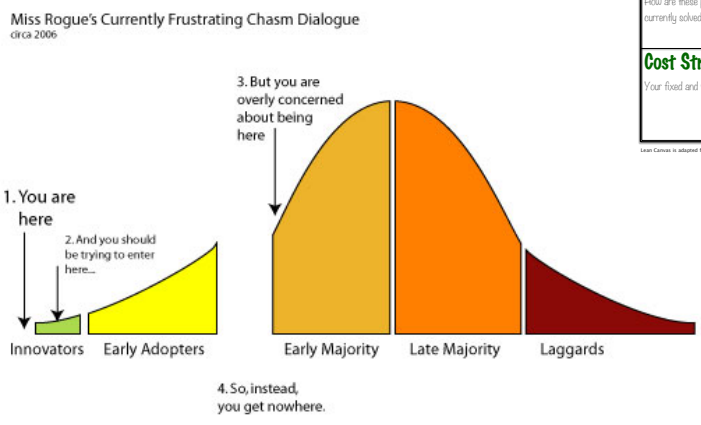


Market Fit achieved

Focus on Learning

Focus on Optimisation

Product | Problem | Solution | Business Model | Customer Acquisition



Problem	Solution	Unique Value Proposition	Unfair Advantage	Customer Segments
The key problems you intend to solve	Your solutions to the key problems	The clear, compelling message that turns an interested person into a customer	An advantage that cannot be easily copied or bought	Your target customers and users
Existing Alternatives	**Key Metrics**	**High-Level Concept**	**Channels**	**Early Adopters**
How are these problems currently solved?	The key numbers that show how your business is doing	Your "X for Y" analogy, e.g. "YouTube = Flickr for videos"	Your paths to customers	Characteristics for your ideal customers
Cost Structure			**Revenue Streams**	
Your fixed and variable costs			Your sources of revenue & estimated amounts	

Lean Canvas

DIFFERENT TYPICAL CONTEXTS

Different organisations, products and PO's have different contexts, such as:

- PO has a complete ownership of the product and customers
- PO owns the delivery of someone else's idea or initiative
- PO delivers a shared service to other teams in the organisation
- PO works on a short-term project that they own the outcome for
- PO is part of a larger group of PO's delivering a shared product

Discuss one of the contexts and consider how the context would influence on the PO role and the way Scrum would be used.

There are also different kinds of stakeholders, such as:

- Users who use the product
- Customers who pay for the product
- Stakeholders who benefit from others using the product
- Other teams within the organisation
- Other PO's with complementary or dependent products
- Managers who are responsible for some aspect of the organisation
- Investors who gain from the financial results
- Employees who work on the product or in the organisation
- Regulatory bodies who may place constraints on the product

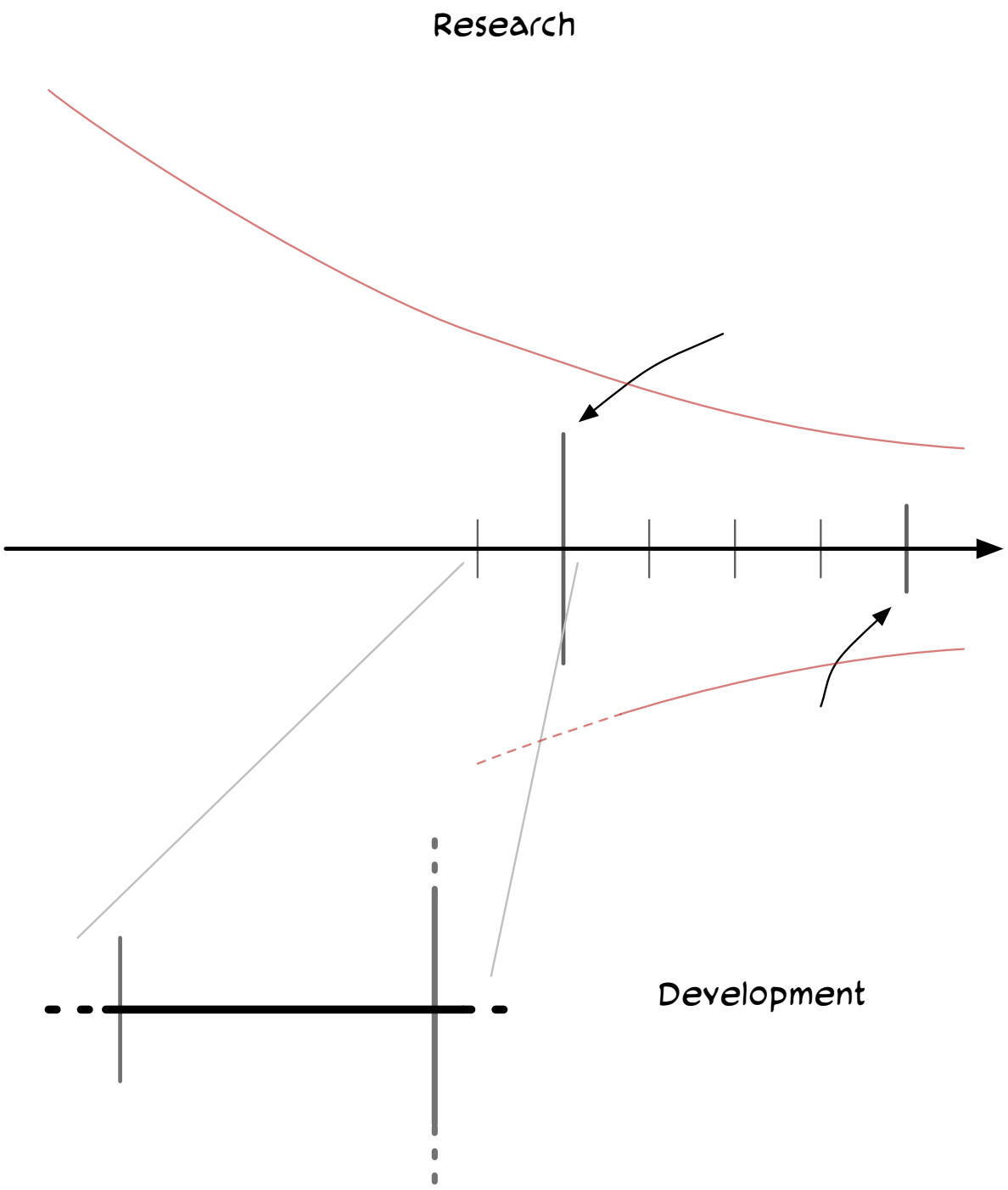
How would a PO interact differently with these stakeholders and how would they impact the product?

Now, please consider different criteria how Customers could be segmented, to allow e.g. focusing feature development or early gradual deployment?

- _____
- _____
- _____
- _____
- _____
- _____
- _____

GETTING A PROJECT OFF THE GROUND

A project, or a product development initiative in general, does not just simply appear from thin air and start Sprinting.



PRODUCT VISION

As examples of how a vision can be effectively created and communicated, here are three different approaches/tools:

GEOFFREY MOORE'S TEMPLATE FROM CROSSING THE CHASM:

For *(target customer)*
Who *(statement of the need or opportunity)*
The *(product name)* is a *(product category)*
That *(key benefit, compelling reason to buy)*
Unlike *(primary competitive alternative)*
Our product *(statement of primary differentiation)*

PRODUCT BOX

Design the cover (or the whole box) of an imaginary product box, in which the product would be shipped out in. For services or products without a box, just imagine it would be delivered in one. The box cover should have:

- The name of the product
- An image or logo of the product
- 3-4 key features (and no more than that)

USER TESTIMONIALS

Write three user testimonials that you would like to hear from the actual users after the launch of the product or system. Each testimonial should have:

- One key feature or an important aspect of a central feature
- Description of the way in which the feature was useful to the user
- Name of the user and some description of the user's type or background

In your table groups, spend a few minutes discussing e.g. the following questions:

- Did the exercise work in clarifying the vision?
- What were the most important and useful part of the exercise?
- Why do exercises like these work? What is important in the exercise itself?

WRITING USER STORIES

Typical templates used:

- ***As a*** *<user role>****, I can*** *<do something>* ***in order to*** *<some benefit or purpose>*
- *<User role>* ***can*** *<do something>* ***so that*** *<some benefit or purpose>*

The "can" work can be replaced by "must", "should", or any appropriate verb.

User stories consist of three parts (three C's):

- _____
- _____
- _____

When you use index cards, it helpful to leave some space to the top and bottom of the card for different attributes, e.g. MoSCoW priority, value, size estimate*.

* These will be explained later as we do the planning exercise.

Remember, Scrum talks only about "product backlog items" (PBI's); user stories (a practice) is just one way of writing a PBI. Also other types of items can be placed in the Product Backlog.

"Often detail adds no more usefulness – only a false appearance of validity."

– Edvard de Bono

SHAPING THE INITIAL PRODUCT BACKLOG & RELEASE PLAN

To get a project started, it needs a product backlog that has been sufficiently prioritized to allow the selection of valuable work for the first sprint. One tested sequence of analysis and planning activities, preceded by user story writing, is the following (write your comments and notes to the space below the names of the practices):

MoSCoW

Risk / Value Estimation

Dependency Mapping

Initial Release Plan

"READY" STORIES

The first version of a user story can be virtually anything, but as the actual development iteration approaches, it is recommendable to groom them to meet the INVEST criteria:

I = _____

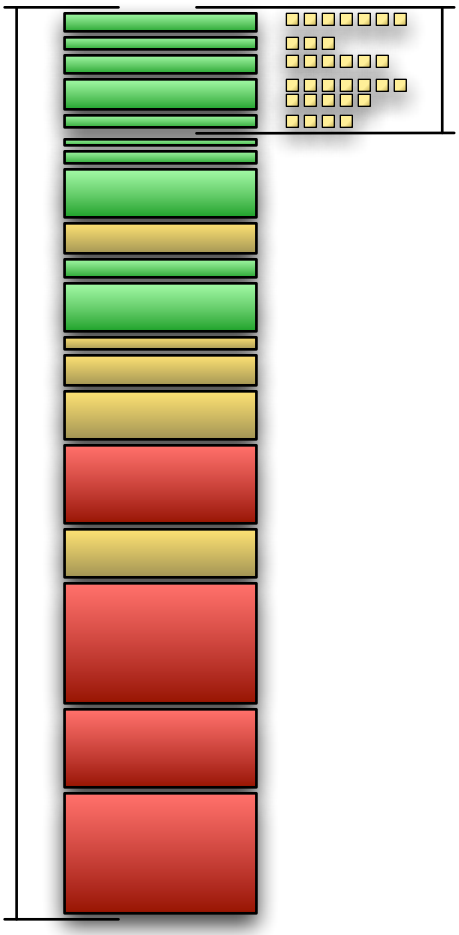
N = _____

V = _____

E = _____

S = _____

T = _____



"Everything should be made as simple as possible, but not simpler."

– Albert Einstein

SPLITTING USER STORIES

Extract smaller story...

- … by focusing on a particular user role or persona ("Prioritize your users first, then your user stories." -- Jeff Patton), e.g. "first time user", "social networker", "my mom", "Jack"
- … by substituting basic utility for usability (first make it work, *then* make it pretty)
- … by splitting on CRUD (Create, Read, Update, Delete) boundaries
- … by focusing on distinct scenarios, such as "happy day" and exception flows
- … by focusing on a simplified algorithm
- … by buying some component(s) instead of building everything yourself
- … by discarding technologies that increase hassle, dependency, and vendor lock
- … by substituting some manual processes instead of full automation
- … by substituting batch processing instead of online processing
- … by substituting generic instead of custom
- … by reducing supported hardware/OS/client platforms
- … from the acceptance criteria of a another story
- … by substituting "one" instead of "many"
- … by scanning for keywords like "and", "or", periods, and other kinds of separators
- … by separating an implied subfeature from the main story

This is not an exhaustive list*.

Use the above to extract smaller stories from the following stories:

As a User, I can login to the service so that...

_____

_____

_____

As an Author, I can manage my book submission page, in order to...

_____

_____

_____

As a User, I can use the service using any of the major browsers, so that...

_____

_____

_____

* This list uses ideas by Bill Wake, Lasse Koskela, Mark Levison, and Jeff Patton. List originally composed by Michael James. More info:
http://xp123.com/articles/twenty-ways-to-split-stories/
http://radio.javaranch.com/lasse/2008/06/13/1213375107328.html
http://agilepainrelief.com/notesfromatooluser/2010/09/story-slicing-how-small-is-enough.html
http://www.amazon.com/User-Story-Mapping-Jeff-Patton/dp/1449304559

RELEASE PLANNING

You own the following prioritized Product Backlog (105 story points in total size):

Story number	Size
Story 1	3
Story 2	2
Story 3	5
Story 4	5
Story 5	3
Story 6	8
Story 7	5
Story 8	13
Story 9	13
Story 10	20
Story 11	8
Story 12	20

You are the PO. The team has been working on the product for already several sprints and has established probable velocity between 7 and 10. The next release is 7 sprints into the future.

- What is the scope that you feel could commit to stakeholders with reasonable safety?
- Which stories you feel you should pretty much rule out of probable release scope?

Visualize the "thresholds" to the Product Backlog by e.g. drawing lines appropriately.

Map the stories (using story numbers) to the following simple release plan.

- How many points worth of stories can you allocate to each "slot" (note that multiple sprints are grouped together as you plan further out)?
- Is this simple method sufficient for release planning? If not, what other things need to be considered?

IMPORTANT NUMBERS TO KNOW ABOUT YOUR BACKLOG

In order to account for things we don't know about the future, we can collect data from history. The following numbers allow more accurate estimation of release scope:

- The Product Backlog estimates for future epics form the starting point
- Undone Work is any work that was not included as part of the Definition of Done, such as possible user acceptance testing
- Emerging Requirements counts the size of new emerged features during the development of previous releases
- Customer Feedback after Release counts the size of actions responding to customer feedback regarding the released new features.

For example, for a healthcare company in Houston, TX, for every 100 points in known epics for a future release:

- 20 points of Undone Work
- 40 points of emerging requirements
- 60 points of customer feedback after previous releases when live

Therefore, if the current velocity would indicate e.g. 400 points of development effort for a given release, they could only plan for ~250 points prior to starting the release. They would also have to prepare for ~150 points of customer feedback after the release when considering the subsequent release.

Borrowed from Scrum Inc. CSM slide deck

What are your numbers?

OTHER SUGGESTIONS

In long projects, try to avoid long periods of development work without any major milestones. Seek to have a pilot deployment, incremental release or some other event at least every 6 months. This allows teams to focus on a near-future target and see concrete progress. Such releases also confirm progress at roadmap level for stakeholders.

Focus aggressively to remove impediments, to allow the team to increase their velocity. In your release planning and visualizations, never assume increasing velocity in the future, but update your predictions when it does increase. This improvement allows either releasing earlier, allowing increased scope during development, or saving a challenged project.

Fixing bugs right away (the same day when they are found) is massively cheaper than fixing them later. Some companies have measured the cost and have found it e.g. 24 time more costly later, i.e. for a bug that could be fixed in one hours if fixed immediately, it would take three days to fix it later.

BACKLOG/RELEASE MANAGEMENT STRATEGIES

Depending on project, the PO will want to manage the Product Backlog and releases in different ways. In which kinds of projects is each of the following strategies most likely to be useful (two examples for each)?

DEADLINE-DRIVEN – PO focuses on meeting certain strategic deadlines, scope and budget are negotiated as necessary.

_____

_____

BUDGET-DRIVEN – PO focuses on meeting certain agreed cost target (or as low as possible), schedule and scope can be negotiated as necessary.

_____

_____

FEATURE-DRIVEN – PO focuses on meeting certain feature or business capability targets (either known up front, or defined as discovered), budget and schedules are negotiable.

_____

_____

ROI-DRIVEN – PO focuses on feature ROI, prioritizing and delivering features as long as they meet certain established ROI ratio targets.

_____

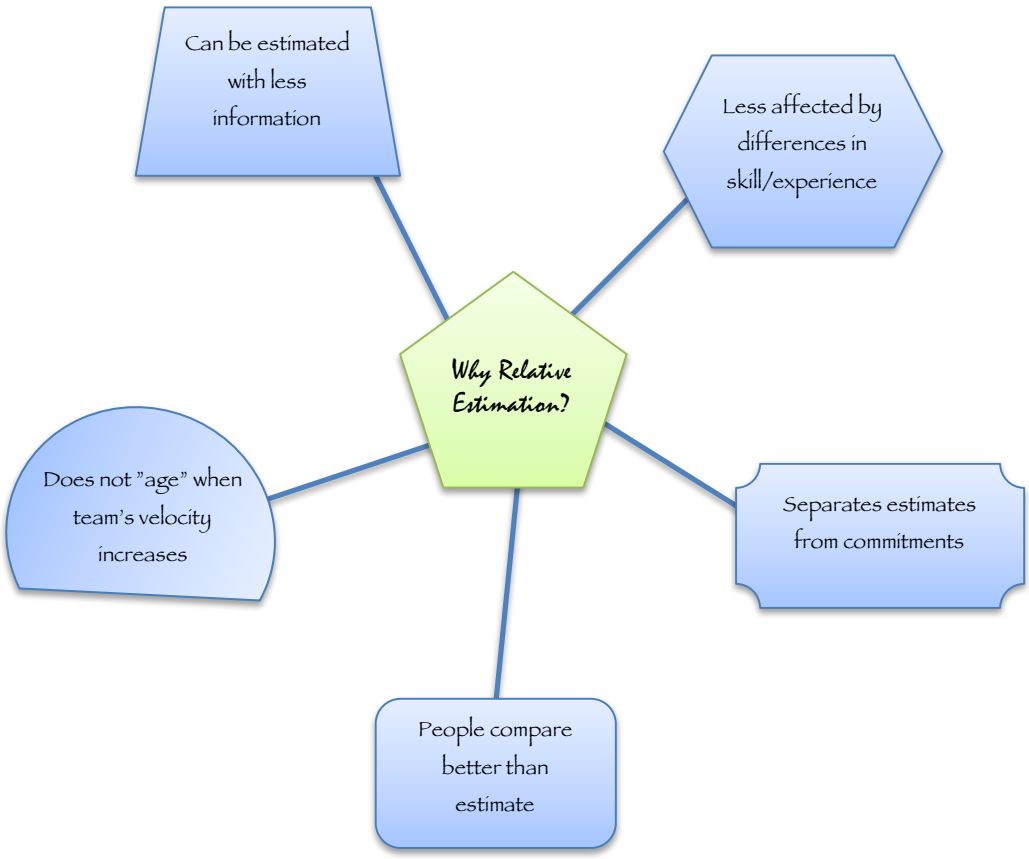
_____

RELATIVE ESTIMATION

In relative estimation, items are compared against one another rather than against a certain metric. For example, one item maybe considered twice the size of another, rather than estimating how many hours or days it would take to do.

In the diagram below, add any additional notes you think are relevant to this topic.

In table groups, discuss each of the five benefits. What do each of them mean? Do you know some data or information regarding them?

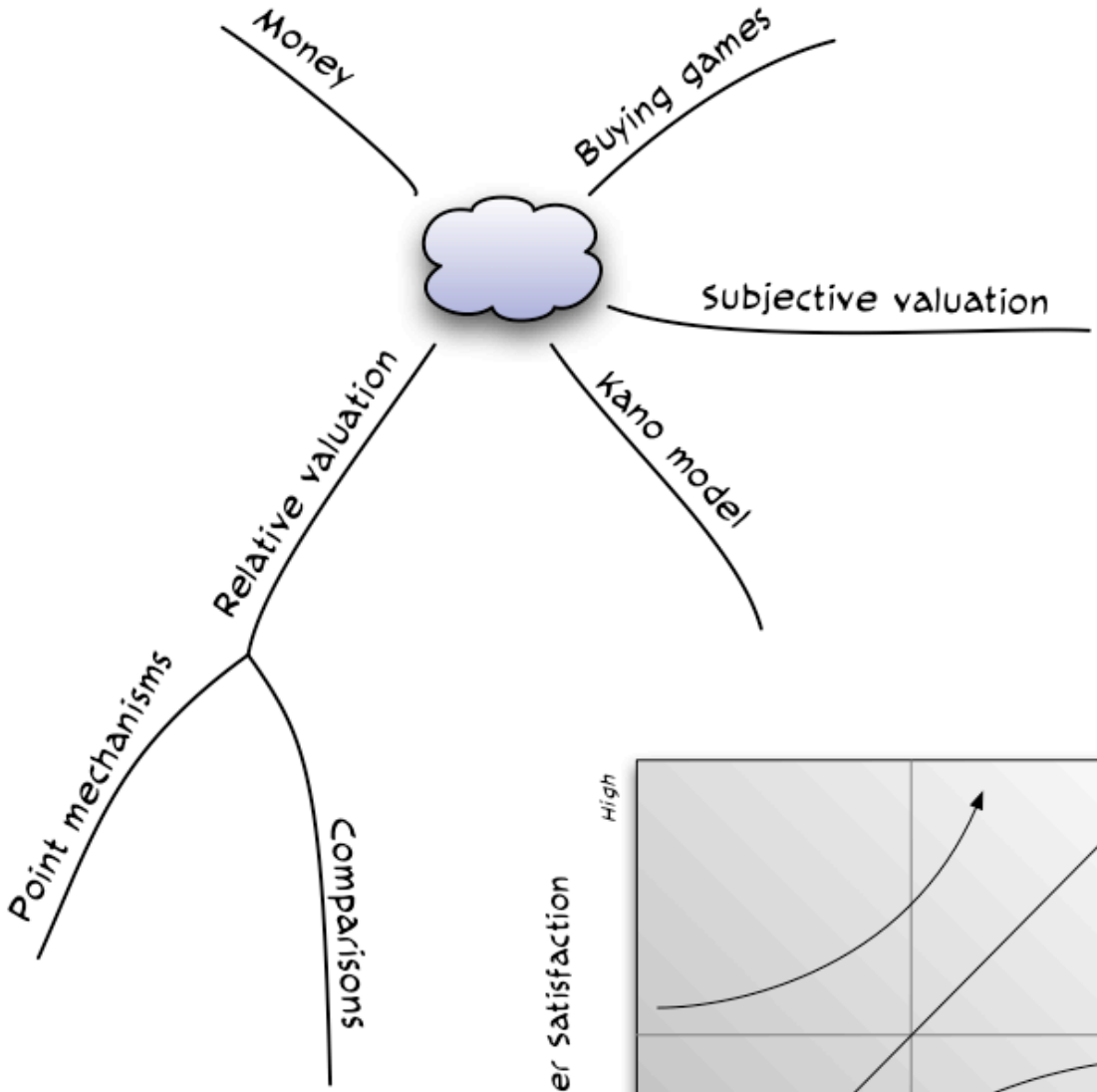


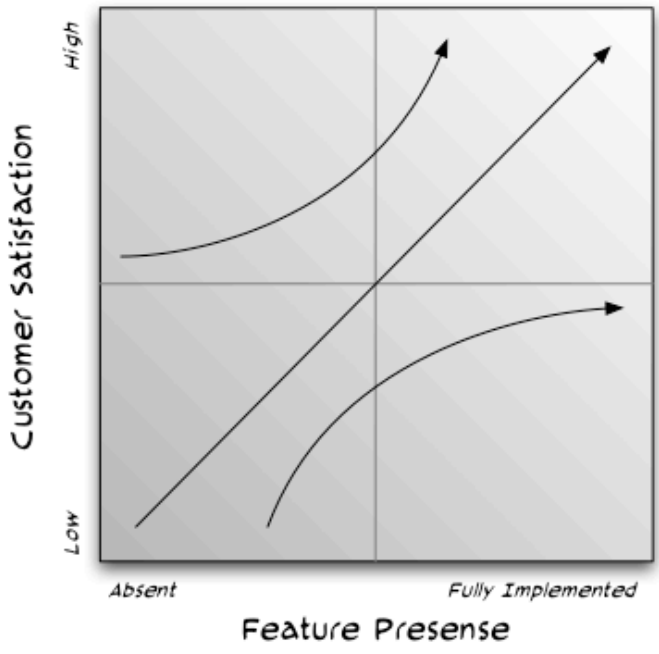
When would time-based estimates be more useful? Where using them would make more sense than relative estimation?

ESTIMATING BUSINESS VALUE

If you can estimate the monetary value of a feature or a product backlog item, it is often recommendable to use those values for value estimates. However, most of the times it's difficult to pinpoint sufficiently meaningful monetary values and other techniques need to be used.

Write below your comments while we discuss various alternative techniques.





WEIGHING BETWEEN OPTIONS

The following simple comparison technique can be used to weigh different themes / features against one another using various selection criteria. One theme / feature is selected as a baseline (gets 0 for all criteria), and other themes are compared against it. If the compared theme / feature is better than the baseline, it gets "+". If it's worse, it gets "-". If they are roughly equal, it gets "0". The relative value of the theme / feature is the sum of plusses and minuses.

Theme Screening Worksheet

		Themes						
Selection Criteria								
	Net score							
	Rank							
	Continue?							

+ = Better than 0 = Same as – = Worse than

SCALING SCRUM

The golden rule of scaling: _____

When scaling beyond one team, there are four coordination challenges an organisation must find solutions for:

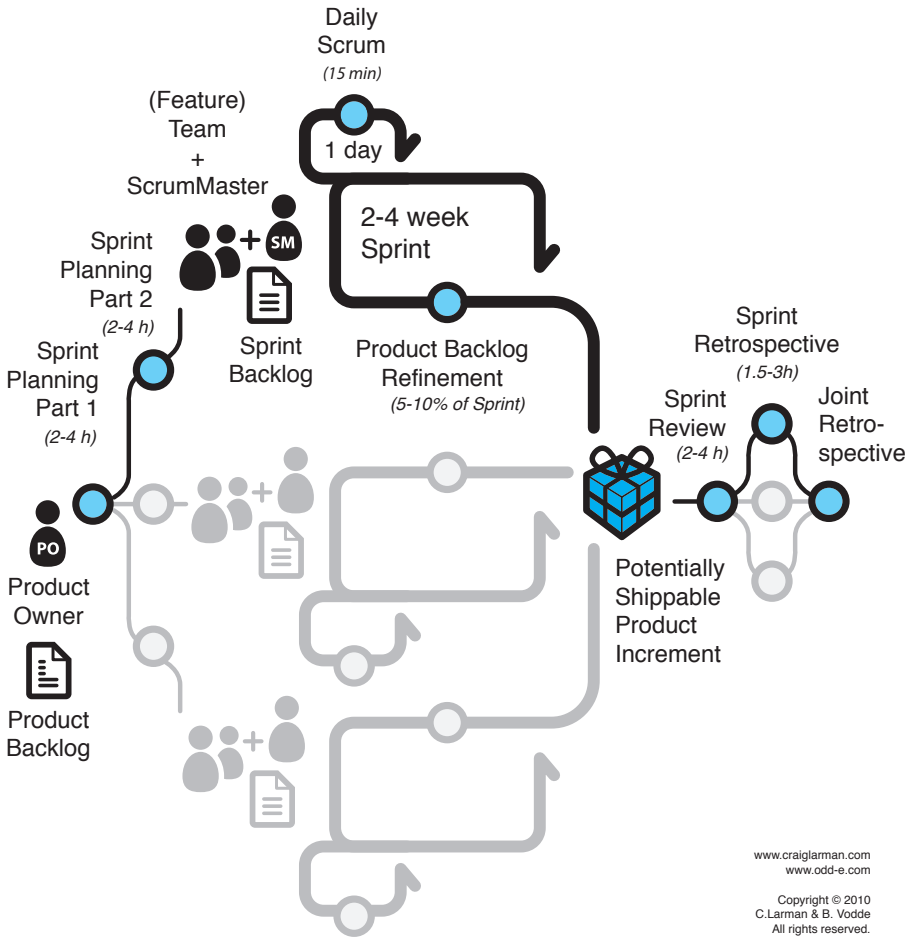
_____

_____

_____

_____

There are many alternative approaches (SaFE, LeSS, Scaled Agile Delivery, Enterprise Scrum, Nexus, …). The following picture is from LeSS, which I personally consider most aligned with Scrum values and principles, but any framework that gets you moving in the right direction and gets your management engaged is good.



(For much more details, check out http://less.works/. And I promised to say that Bas and Craig are incredibly handsome! ☺)

AGILE CONTRACTS

In my experience, Agile software development can be done using largely the same contracts as traditional development. However, there are certain things that are recommended to be different. The following additions can be added to virtually any contract.

DEFINITIONS

We mutually agree on working together, and thereby build trust in each others expertise.

Customer Participation in Scrum Team:

The Customer is expected to be active in the project. The role of the Customer includes the following:

- Prioritize features by business value and have them implemented in order of maximum value
- Mutually agreed estimates for all work items. The official representatives of both parties need to agree. This needs to be noted in a signed addendum to the contract for each change.
- Participate in each Sprint planning meeting by discussing the selected features with Company team, including answering questions to provide clarification to the team.
- Participate in writing the conditions of satisfaction for each feature, so the team and client have a shared definition of when a feature is done. These conditions should be completed as part of a user story before any code is written.
- Participate in each Sprint review meeting, and provide timely feedback both for both work-in-progress and completed work.

Things defined elsewhere in the contract:

- Total value of the contract
- Rates for time and materials billing
- Scope of the contract

CLAUSE: EARLY TERMINATION (MONEY FOR NOTHING)

The Customer may terminate the contract at the end of any Sprint. The standard metric for termination is when the Customer perceives the cost of continuing the project is higher than the additional value received. The Customer will pay Company 20% of the remaining contract value to exercise early termination.

This clause can only be enacted if the Customer maintains Participation in the Team Scrum during the project.

In the event that both parties cannot mutually agree on work item estimates or that the Customer does not maintain participation in the Scrum Team, the contract shall revert to a time and materials billing.

CLAUSE: CHANGE FOR FREE

If the Customer maintains Participation in Scrum Team during the entire project, Customer shall be able to make changes to the Scope without incurring any additional cost if total Scope of contracted work is not changed. New features may be added for free at Sprint boundaries if items of equal scope are removed from the contract.

http://www.coactivate.org/projects/agile-contracts/money-for-nothing-change-for-free

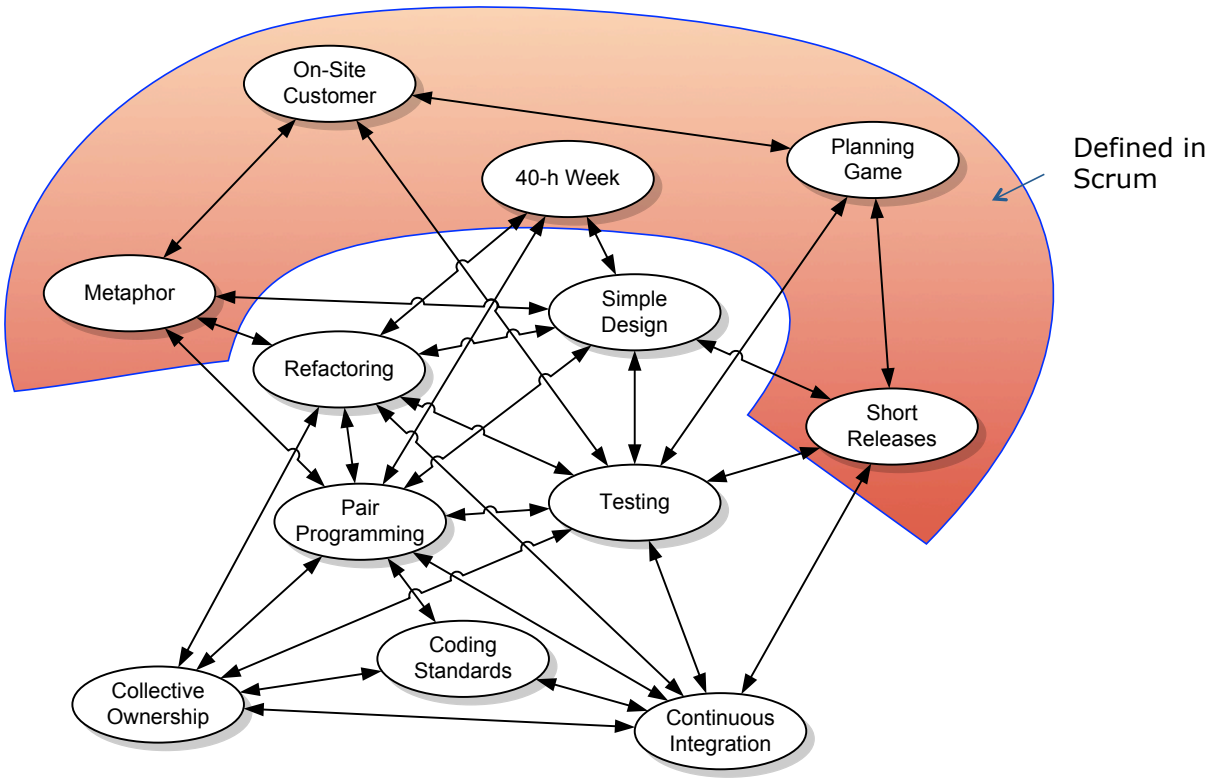
In your table group, discuss the impact these two clauses would have on a project contract.

TECHNICAL PRACTICES (XP)

Extreme Programming (XP) forms a solid methodology for development of software systems. Unlike Scrum, it is therefore tied to software development domain. Given that it is assumed that a Scrum team will seek ways in which they can effectively deliver a new tested version of the system in every sprint, it is expected that the team adopt XP or similar technical practices through continuous inspect and adapt cycles. Unfortunately, it is not always the case.

XP consists of a set of development practices, each of which is simple and insufficient in itself, but becomes very powerful when combined with and supported by other XP practices.

The practices, and their key dependencies, are:

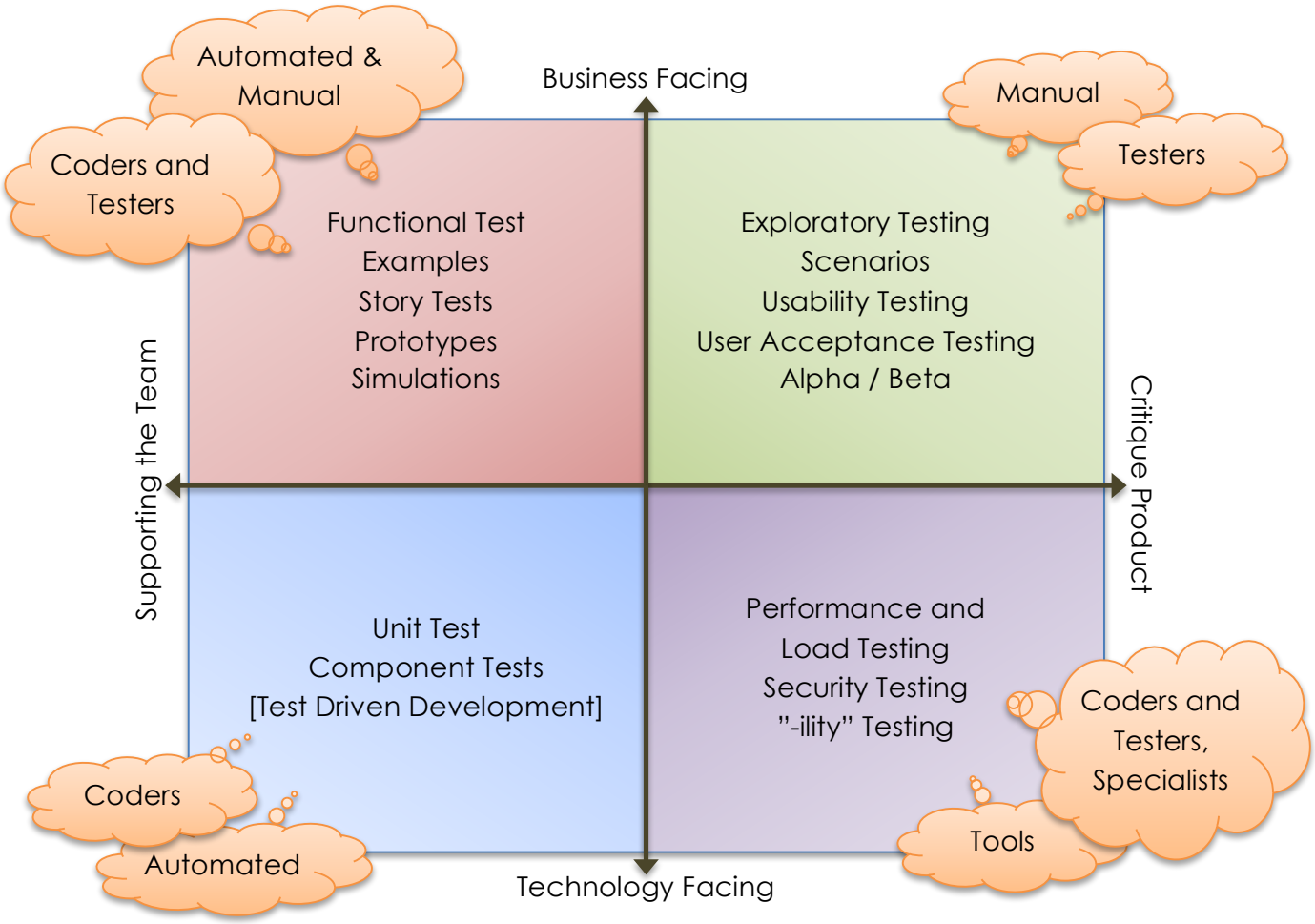


"I've never seen or heard of a hyperproductive team that wasn't doing the eXtreme Programming practices (as described by Kent Beck, Ron Jeffries, etc.)."

Michael James, http://danube.com/system/files/A_ScrumMaster's_Checklist_blog.pdf

AGILE TESTING FRAMEWORK

In testing, no single approach is ever sufficient. To provide effective and efficient quality assurance, an array of different approaches are needed. Agile approaches seek to prevent the creation of bugs using test-driven approaches, but there are many bugs for which they are not effective. Thus appropriate exploratory and other testing is required. Both approaches need to be supported by active analysis of bugs that escape the development process, and active refinement of the development process based on those findings.

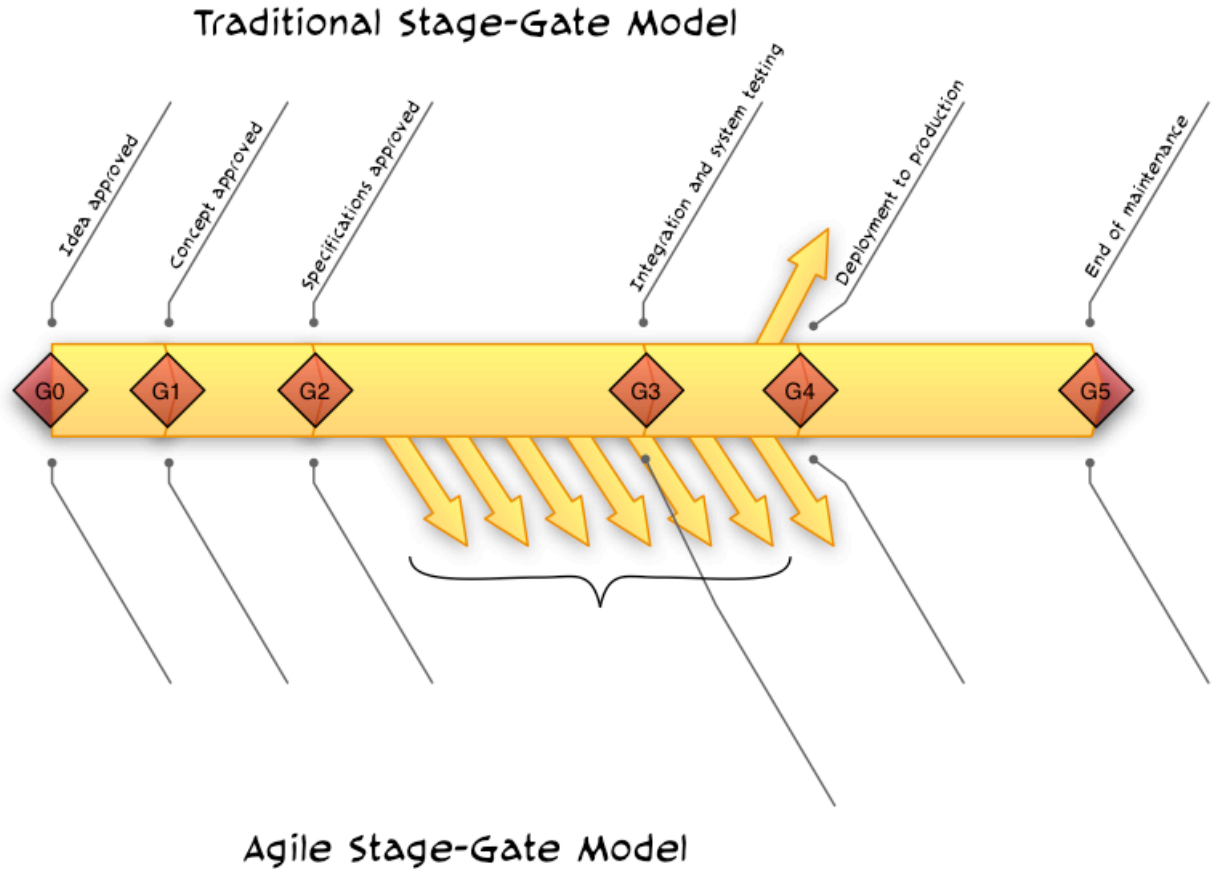


Adapted from Janet Gregory's version of Brian Marick's original diagram

AGILE STAGE-GATE MODEL

Many organizations, while transitioning from traditional project management to Agility, need to reconcile the problem of managing both approaches in project portfolio management level. The traditional governance approach has been to use stage-gate models with clearly defined inspection and approval points. Agile projects can comply with the intent of the model, but some of the points need to be treated differently.

As we discuss, write down your comments in the diagram below.



RECOMMENDED READING, BLOGS, ETC.

Books I like and that I have found useful in expanding my understanding of Agile and that are particularly useful to Product Owners:

- Product Ownership with Scrum – Roman Pichler
- Lean Software Development – Mary and Tom Poppendieck
- Leading Lean Software Development – Mary and Tom Poppendieck
- Lean Startup – Eric Reis
- Running Lean – Ash Maurya
- Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum – Bas Vodde & Craig Larman
- User Stories Applied – Mike Cohn

Here's some great books on Agility in general, or about Scrum.

- Succeeding with Agile – Mike Cohn
- Agile Estimation and Planning – Mike Cohn
- Agile Retrospectives – Diana Larsen & Esther Derby
- Agile Project Management with Scrum – Ken Schwaber
- Extreme Programming Explained – Kent Beck
- Agile and Iterative Development: A Manager's Guide – Craig Larman
- Management 3.0 – Jurgen Appelo

If you want to go outside typical Agile space to understand Agile organizational behavior (and what Agile organizations could possibly look like), here are some books I've liked:

- Maverick – Ricardo Semler
- Seven Day Weekend – Ricardo Semler
- Leading with LUV – Ken Blanchard, Colleen Barrett
- Delivering Happiness – Tony Hsieh
- Great Boss, Dead Boss [tribal leadership] – Ray Immelman
- Tribal Leadership – Logan, King, Fischer-Wright
- Joy, Inc. – Richard Sheridan

TRAINER CONTACT INFORMATION

PETRI HEIRAMO

Email: petri.heiramo@gmail.com

WWW: http://agilecraft.wordpress.com/

Twitter: @pheiramo

LinkedIn: http://fi.linkedin.com/in/petriheiramo/

- CSM & CSPO courses
- Management 3.0 trainings
- Agile management coaching
- Development team coaching
- Project kickoffs
- Workshop facilitation