Backlog management for the discovery teams Amdocs IT

July 2017

Dr. Dror Zernik, Sandeep Yadav & Shalhevet Shahuda

Dr. Agile: *Responsible Agile Transformation*

Dror.z@dr-agile.com

Time	Activity	Comments
9:00 – 09:30	Intro+ Agile principles, day goals – product process	What we want to achieve; high level
09:30– 10:30	PO, SM PjM… team definition	
10:30– 12:00	Marshmallow challenge core backlog terms: MMF thinking; MVP, Good US	
12:00-12:30	Our projects… re-defining	
12:45-13:30	Lunch	
13:30-14:30	Techniques for breaking user stories	
14:30-15:15	Backlog preparation	
15:15-15:30	Retrospective	

Team rules

- Be proactive
- Speaking & listening balance
- Voting rules – consensus
- Meeting standards – no screens; silent communication.

  Pre-notify if on call…
- Be on time ?

Workshop goals

AS the Pos, SMs and Technology Leaders of the Amdocs Pune IT who are going to lead the Discovery teams

WE WANT to:

1. **Learn the core techniques for managing an Agile Backlog**
2. **Learn how to plan an Agile-product roadmap?**

SO THAT we can better manage our backlogs & releases in order to improve the likelihood of meeting **our business goals**

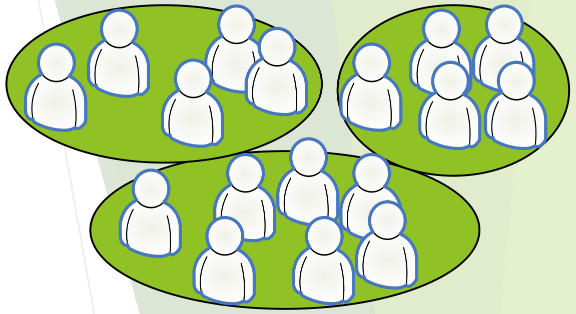
Common language & mind-set

Backlog examples

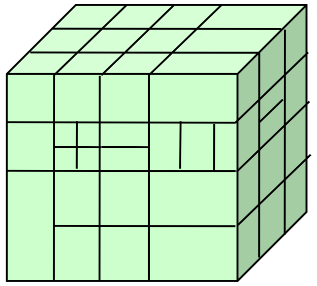
Techniques and practices

Start building an Agile Roadmap

Scrum implementation in a nutshell



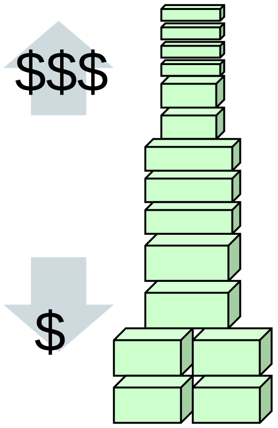
Split your product



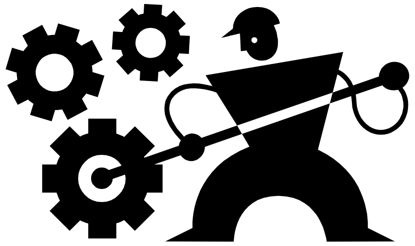
~~Large group spending a long time building a huge thing~~

Small team spending a little time building a small thing

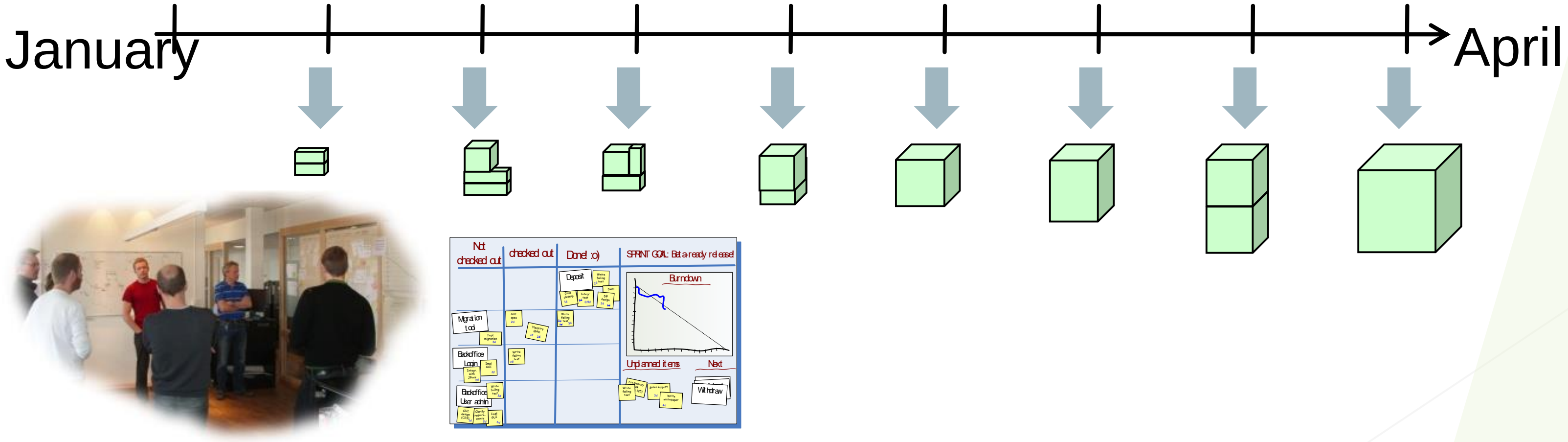
... but integrating regularly to see the whole



Optimize process



Split time



Henrik Kniberg

The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value

Individuals and
ons

Over

Process and

Early Feedback

ftware

Con
doc

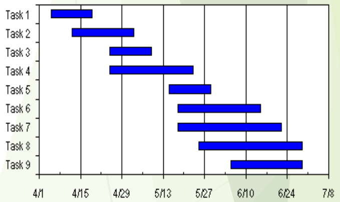
Team empowerment

Customer collaboration

Contract negotiation

Responding to
char

Following
plan

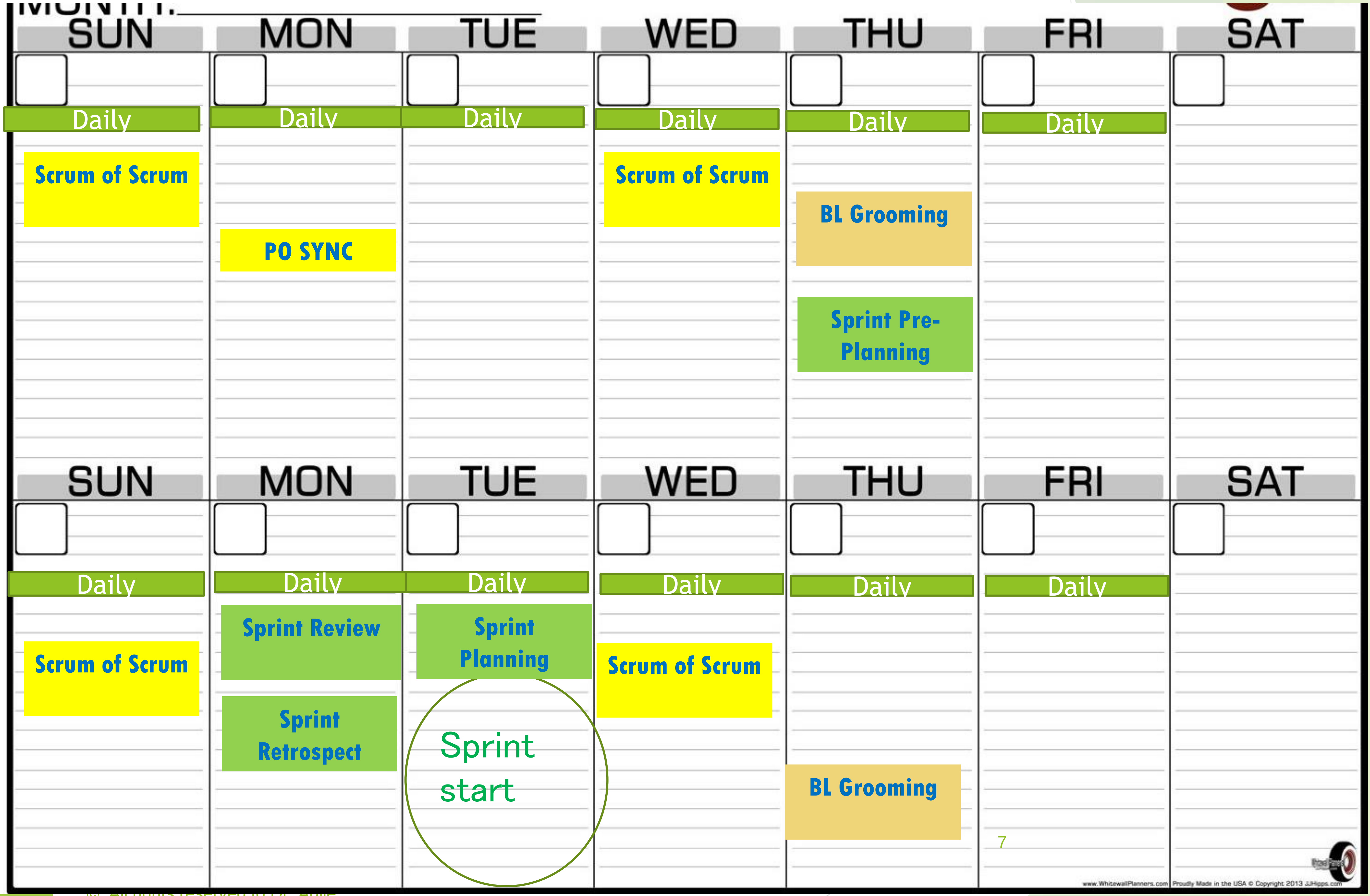


Continuous improvement

While there on the right, we value the items on the left more

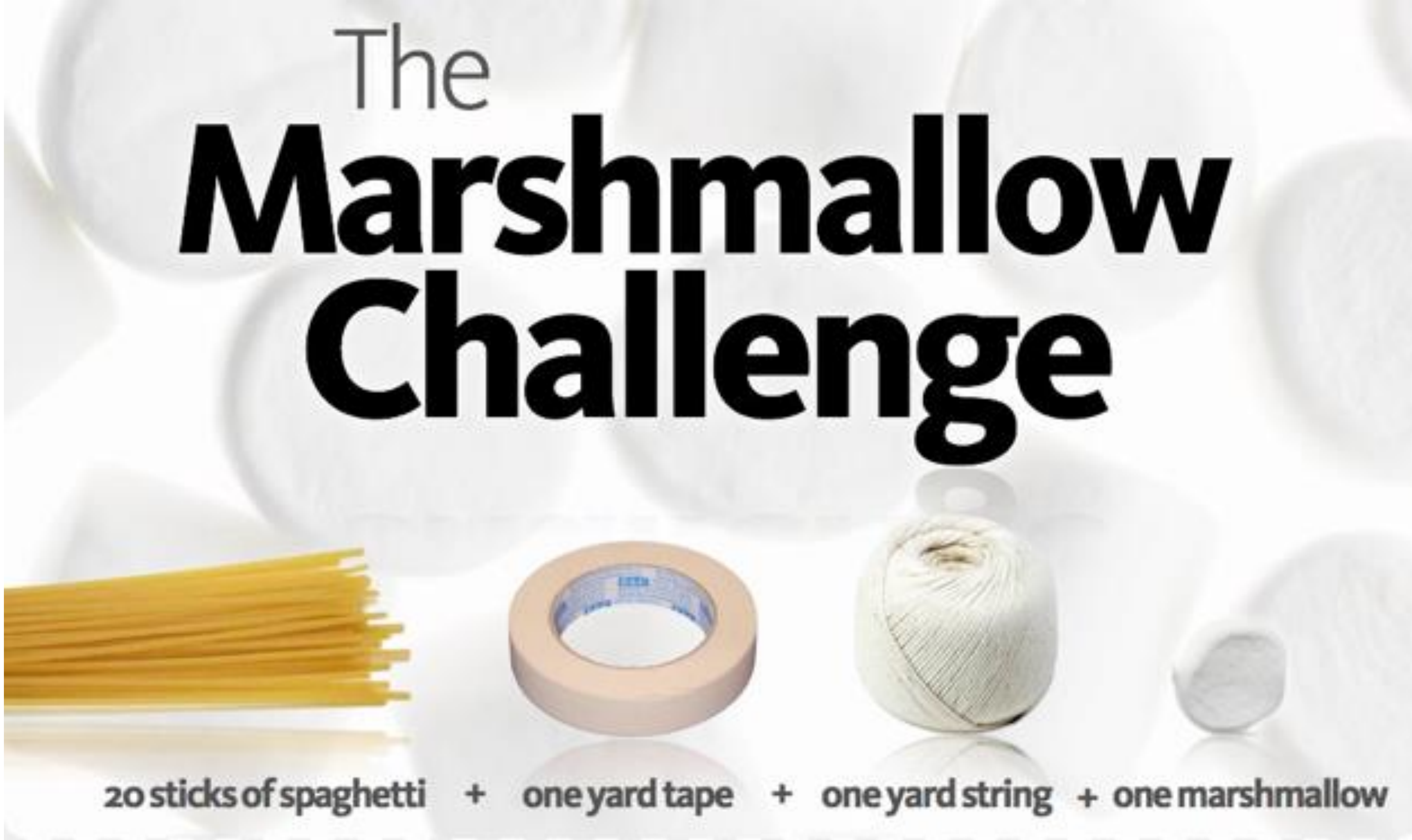
(http://www.agilemanifesto.org)

Scrum Meetings – for a 2 wks sprint



SUN	MON	TUE	WED	THU	FRI	SAT
Daily Scrum of Scrum	Daily PO SYNC	Daily	Daily Scrum of Scrum	Daily BL Grooming Sprint Pre-Planning	Daily	
Daily Scrum of Scrum	Daily Sprint Review Sprint Retrospect	Daily Sprint Planning Sprint start	Daily Scrum of Scrum	Daily BL Grooming	Daily	

MONTH:
www.WhitewallPlanners.com | Proudly Made in the USA © Copyright 2013 JJHipps.com


The
Marshmallow
Challenge
20 sticks of spaghetti + one yard tape + one yard string + one marshmallow

Ted

The Rules

Build the Tallest Freestanding Structure

The winning team is the one that has the tallest structure measured from the table top surface to the top of the marshmallow. That means the structure cannot be suspended from a higher structure, like a chair, ceiling or chandelier.

The Entire Marshmallow must be on top

The entire marshmallow needs to be on the top of the structure. Cutting or eating part of the marshmallow disqualifies the team.

Use as Much or as Little of the Kit

The team can use as many or as few of the 20 spaghetti sticks, as much or as little of the string or tape. The team cannot use the paper bag as part of their structure.

Break up the Spaghetti, String or Tape

Teams are free to break the spaghetti, cut up the tape and string to create new structures.

The Challenge Lasts 18 minutes

Teams cannot hold on to the structure when the time runs out. Those touching or supporting the structure at the end of the exercise will be disqualified.

Scrum Master Role - SM

- Ensures that the team is moving to Scrum
- Organizes and facilitates Sprint meetings
- Responsible for protecting the team
- Focuses on solving impediments
- Motivates the team
- Pushes for visibility

IS NOT:

- Manager/ baby-sitter, owner of Jira..

Key ownership – HOW? (process)

Product Owner’s Role - PO

Responsibilities:

- Relationship with the Stakeholders
- Get detail level of requirement
- Hold clear vision of product and business & Clearly communicate the business value & requirements to the Team
- Backlog - Creating and maintaining
- Priorities - Sort the Product backlog items
- Review - Approve the result

Key ownership – WHAT to do?

Product manager

Responsibilities:

- Overall deployment at customer site
- 1st line support?
- Working closely with PO to define requirements
- Working closely with SM to remove impediments
- Working with management (and SM) to ensure visibility

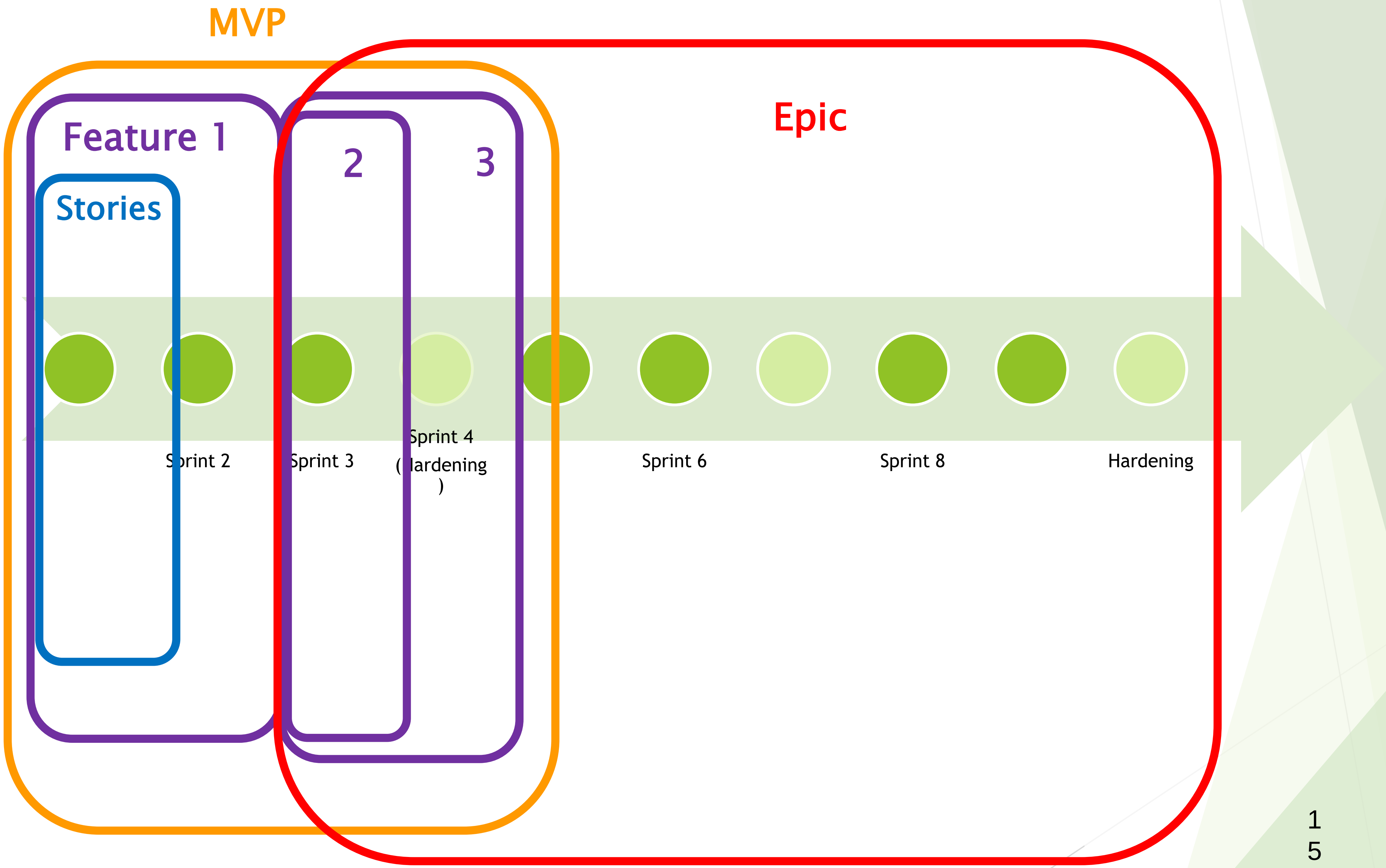
What problem do user story address?

mis **Communication**

Software development – communication problem

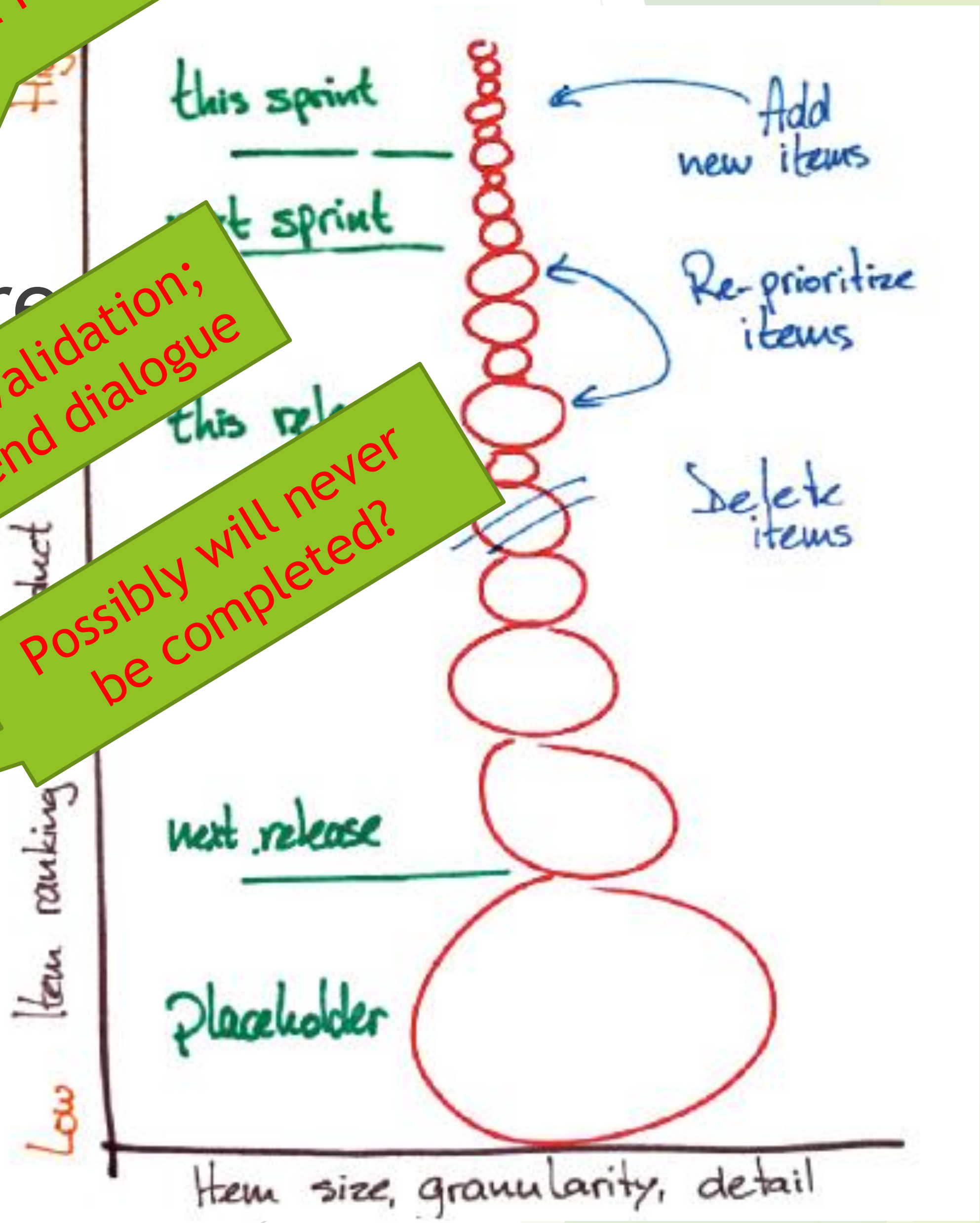
How the customer explained it

From Stories, sprints, to Features, MVPs …. And Epics



The Product and Release Planning Process (Grooming/ Discovery)

- Planning and committing
- Product –
  - MMF – Minimal Marketable Feature
  - Breaking into Good User Stories
  - Grouping Into MVP – Minimal Viable Product
  - Epic – a cross-team, multiple MMFs, capability

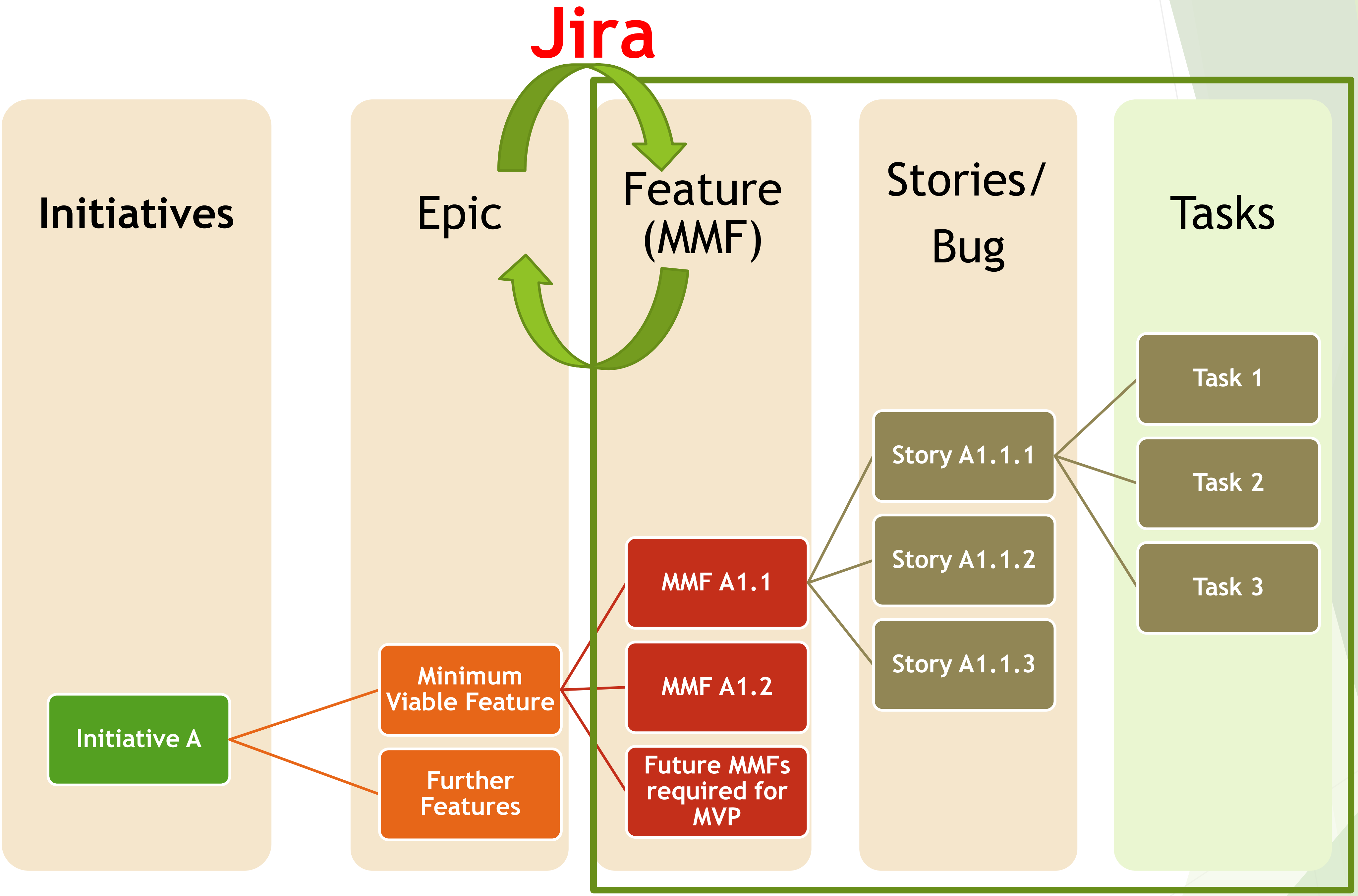


Internal feedback

Business validation; end-to-end dialogue

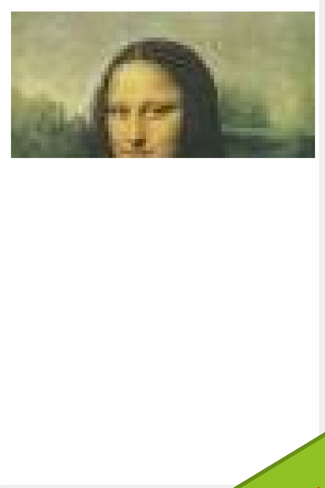
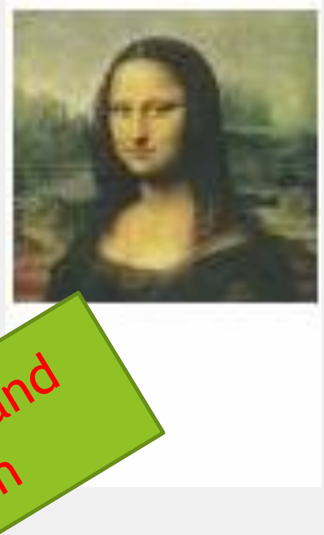
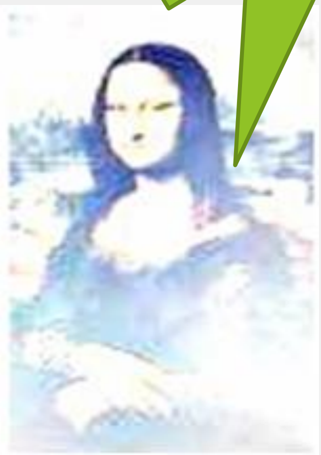
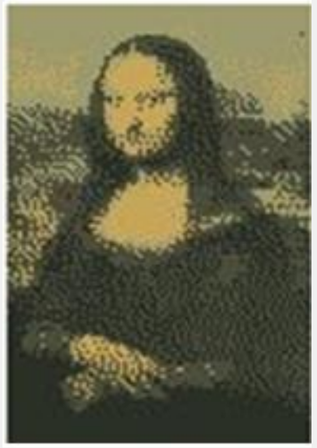
Possibly will never be completed?

Hierarchy Backlog



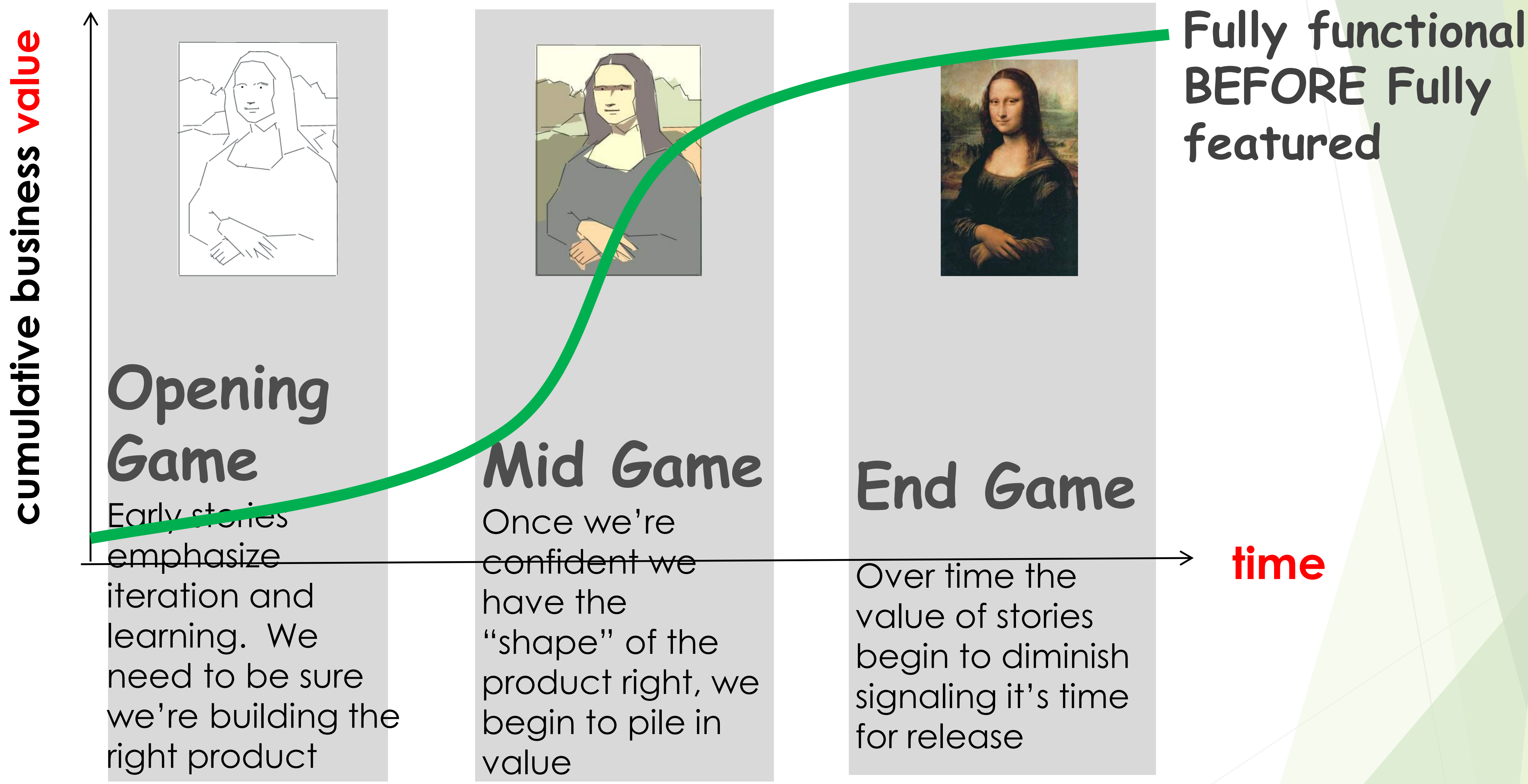
- **MMFs accumulate into a MVP – minimally viable product/release, and cannot really be delivered on their own…**
- **The MVP can…**

Incremental vs. Iterative

	Delivery 1	Delivery 2	Delivery 3
Incremental plan			
Iterative plan			

Infrastructure and application

Looking at the release of business value over time lets us see what's going on here



Fully functional BEFORE Fully featured

Opening Game

Early stories emphasize iteration and learning. We need to be sure we're building the right product

Mid Game

Once we're confident we have the "shape" of the product right, we begin to pile in value

End Game

Over time the value of stories begin to diminish signaling it's time for release

All the layers, but smaller – Minimally Marketable Feature (MMF)

http://www.levensmiddelenkrant.nl/uploads/foto/0_61_hamburger_(Large).jpg

http://us.123rf.com/400wm/400/400/cokemomo/cokemomo1206/cokemomo120600007/13939893-mini-hamburgers-party-food.jpg

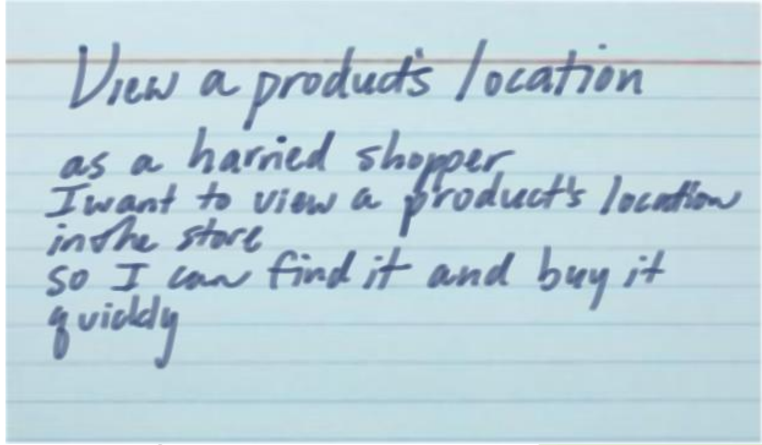
Elephant Carpaccio & Walking skeleton

User Stories are multi-purpose

Stories are a:

- User’s need – focuses on the user need rather on system attributes – the problem we are trying to solve
- Token for a conversation
- Support iterative development
- Right size as planning item
- Business value? In prioritization

* ***Kent Beck coined the term user stories in Extreme Programming Programming Explained, 1999 1999***



View a product's location

as a harried shopper
I want to view a product's location in the store
so I can find it and buy it quickly

Good User Stories

"are a promise for a future conversation"

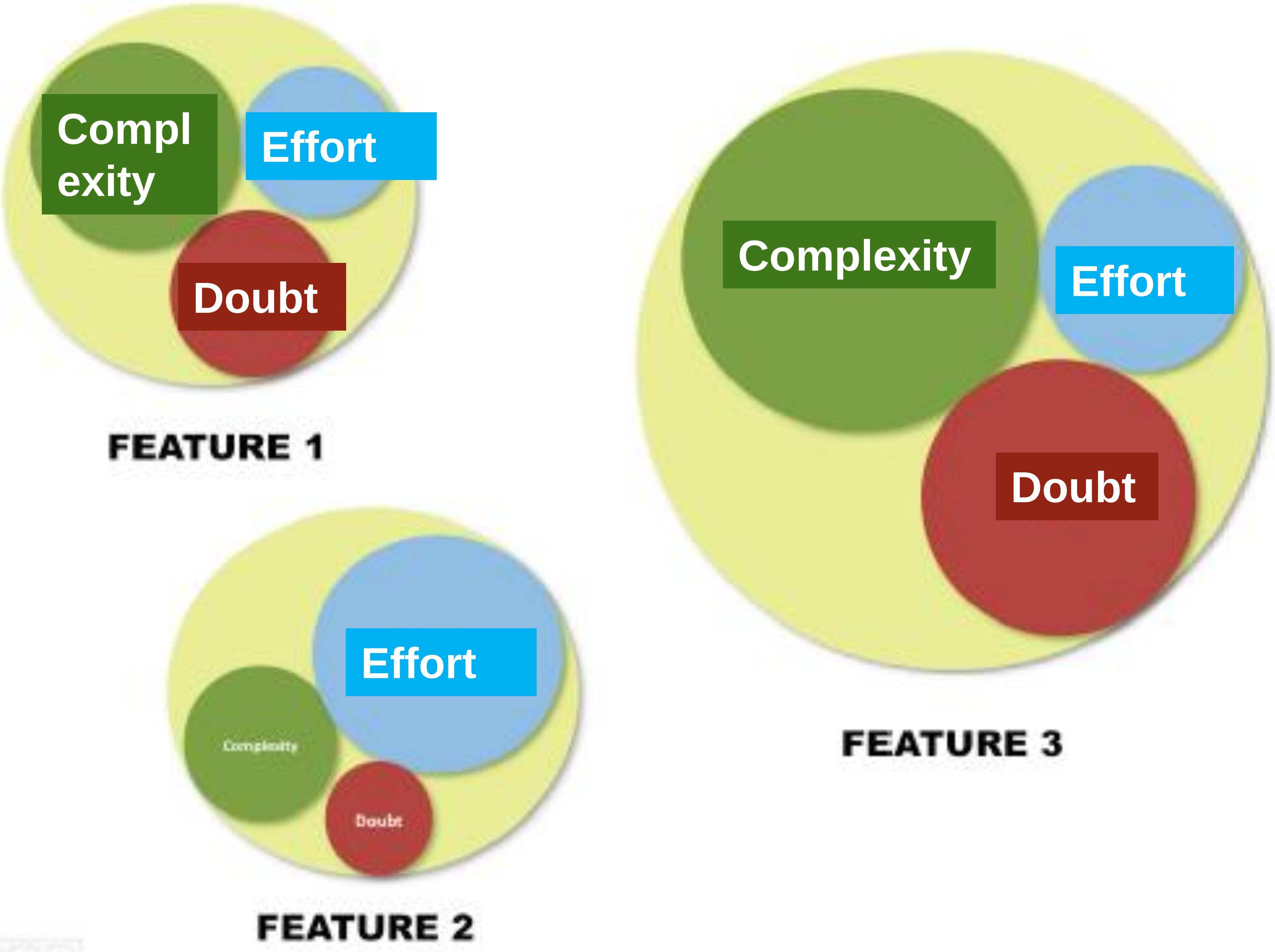
"As a <role> I can <function> so that <WHY?>."

- May assume template form:
  - Card - short
  - Confirmation – Definition of Done
  - Conversation – Agile modeling

Good User Story: INVEST

I	Independent	self contained, no inherent dependency on another user story.
N	Negotiable	Can be changed and rewritten.
V	Valuable	Must deliver value to the end user.
E	Estimable	The team can estimate its size.
S	Sized appropriately Small < 5 days + QA	Must be small to enable several stories in sprint.
T	Testable	Test is developed concurrently.

Relative Estimates in Story Points



Physical assumptions

- In order to get into higher (SWJF) priority – features must be as small as possible.
  Find the feature that is so small that it can compete;
  - Minimal Viable Feature;
  - Minimal Marketable Feature
- Find the feature that brings maximal value
  - Reduces risk, hence integrative
  - Typically – an end-to-end “walking skeleton”

A User-Story Done Criteria (DoD)

- Define Acceptance Test
- Exceptions from standard DoD
- What is excluded → becomes a user story…

- Example:
  - An application that shows 20 fields of a query to the BE.

Standard Done Criteria

- Passed acceptance test
- Standard quality procedures:
  - Code review;
  - Unit test;
  - Regression tests; (smoke = this version is QA ready)
- Automation test was implemented
- Checked in
- Deployed on Staging / Lab / Test environment (not on Dev environment)
- Merge – frequent and OFTEN; at least once a sprint. Preferably once a week.
- More?

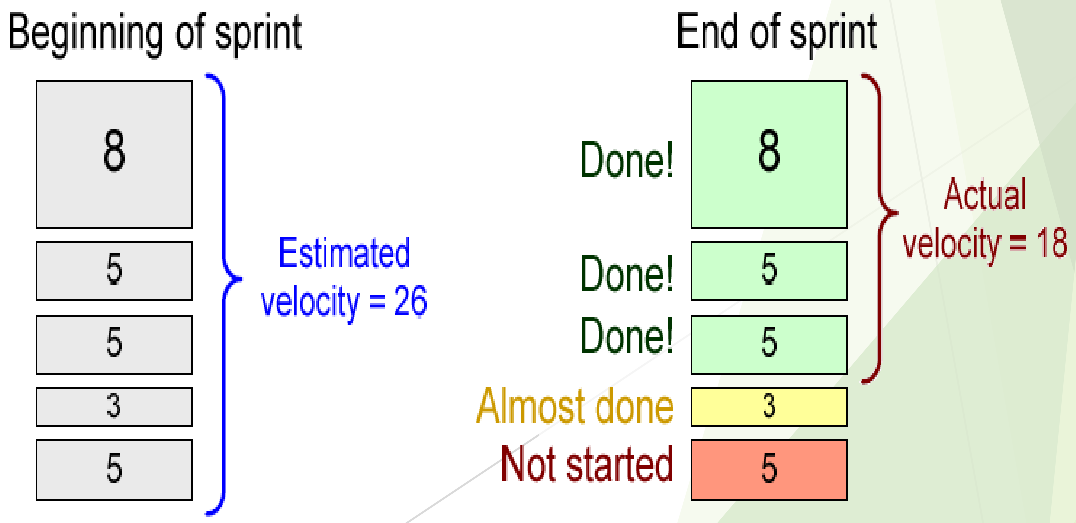
Specific-exceptional Done Criteria

- Documents
- POCs
- Research
- NF stories (*abilities: usability, scalability, performance, high availability, safety etc.')
- Weaker cases? When we cannot?

Backlog Estimation (Story Points)

- Done by the Preparation (Discovery) team
- Rough estimate (range)
- Estimates may be wrong
- Team Velocity is per Backlog estimates
- Team estimates (Scrum planning) are not related to Backlog estimates (Backlog preparation)
- Velocity: Done is Done
- Feature estimates – top down and bottom up.

Story Point	T-shirt Size
1	XS
2	S
3	M
5	L
8	XL
13	XXL
20	XXXL



Let’s practice

- Pick 2 projects…

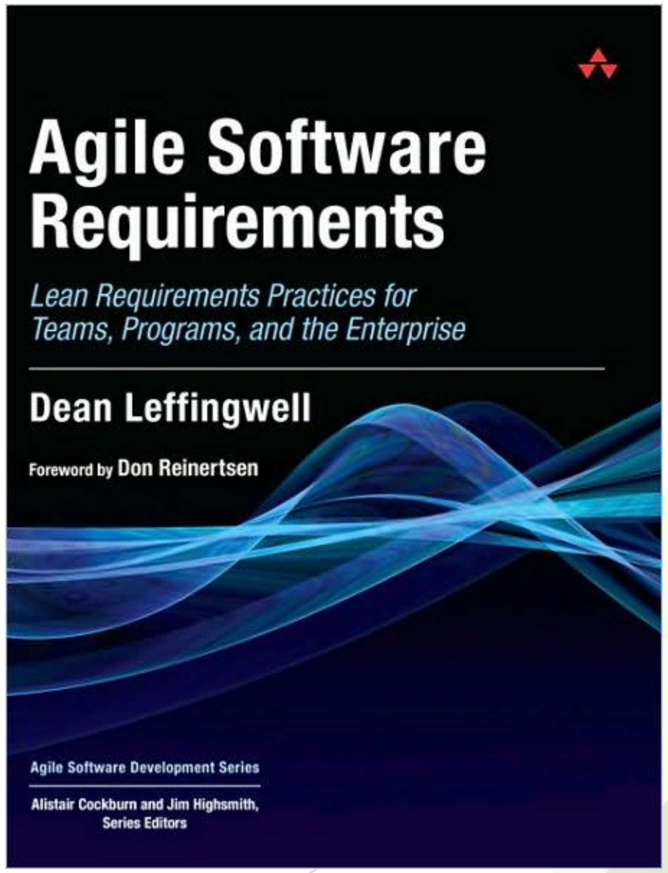
Before we continue

- How do I preserve the larger picture? How do I know there is a coverage of the entire requirements?
- How to cope with non-story data? (e.g. Tables), Safety, technical needs
- How to write a User story?
  - Title
  - Explain
  - DOD

Let’s practice: Sprint 0 starts in 60 minutes

- Create your item map…
- Do you have a clear version roadmap?
- Can you break epics into stories vertically?
- Clear DoD?

11 Patterns for Splitting User Stories



Agile Software Requirements

Lean Requirements Practices for Teams, Programs, and the Enterprise

Dean Leffingwell

Foreword by Don Reinertsen

Agile Software Development Series

Alistair Cockburn and Jim Highsmith, Series Editors

Pattern 1 – Workflow Steps (INCREMENTAL WORKFLOW)

- Identify specific steps that a user takes to accomplish the specific workflow, and then implement the workflow in incremental stages

- As a POC user, I can easily observe and predict the power supply and see if the machine is charging
- Generate an US image using one probe

…I can see how much time is left if idle.
…I get alerts when reaching a threshold.
…I see charging state

Source: Dean Leffingwell, Agile Software Requirements, Addison-Wesley, 2010

Pattern 2 – Business Rule Variations

(TREAMED BRANCHES)

- Some business rules are pretty complex. If this is the case, break the story into several stories to handle the business rule complexity

- As a POC operator the power consumption during an US operation may vary…

- …power save mode
- …alternative operations
- …ensure save results

Source: Dean Leffingwell, Agile Software Requirements, Addison-Wesley, 2010

Pattern 3 – Major Effort

- Sometimes a story can be split into several parts where most of the effort will go into implementing the first one

- As a user, I can securely export data.
- Create the first US image

- Print
- Can export it to a readable format
- Can allow for direct interaction with standard.

Source: Dean Leffingwell, Agile Software Requirements, Addison-Wesley, 2010

Pattern 4 – Simple Complex (MMF)

- What’s the simplest version that can possibly satisfy the customers need

- As a user, a 2-hour battery is enough.

- …specifying a max delay between LDAP and UC admin
  …including confirmation of update.
  …report conflicts/errors.
  …etc.

Source: Dean Leffingwell, Agile Software Requirements, Addison-Wesley, 2010

EX2: Sprint starts in 15 minutes – use the patterns

- Break the epic into most valuable stories…
- Which pattern did you use?
- Can you break the stories into smaller ones?

Pattern 5 – Variations in Data (DATA TARGET & SOURCE)

- Data variations and data sources are another source of scope and complexity. Consider adding stories just in time after building the simplest version

- Probes

- …customers who want their messages
- …graphical
- …more interfaces, etc…

Source: Dean Leffingwell, Agile Software Requirements, Addison-Wesley, 2010

Pattern 6 – Data Entry Methods (UI)

- Sometimes complexity is in the user interface rather than the functionality itself. Build the simplest possible UI first

- System configuration
  …using simple system access permissions
  …with a fully configured UI.

Source: Dean Leffingwell, Agile Software Requirements, Addison-Wesley, 2010

Pattern 7 – Defer System Qualities (System ABILITIES)

- Sometimes the initial implementation is not all that hard. Do the simple thing first and then add attributes like scalability and speed later.

- System return time from idle
- Multiple images / minute
- Serviceability & diagnostics

- …display update time
- allow for x time delay for scale
- etc…

Source: Dean Leffingwell, Agile Software Requirements, Addison-Wesley, 2010

Pattern 8 – Bulk operations (CRUD)

- Words like manage or control give away that the story might cover multiple operations

- Configure
- Connectivity
- New technology

- …I can sign-up for an account
- …I can edit any account settings
- …I can cancel any account

Source: *Dean Leffingwell, Agile Software Requirements, Addison-Wesley, 2010*

Pattern 9 – Use Case Scenarios (USE_CASE LIKE)

- If use cases have been developed to represent complex user-to-system or system-to-system interactions, the story can often be split according to the scenarios in the use case

- From board to UI
- Clinical use cases

- …Use Case/Story #1 (happy path) Notify and register UC service within LDAP.
- …Use Case/Story #2 (alternate scenario) Handle data validation errors

Source: Dean Leffingwell, Agile Software Requirements, Addison-Wesley, 2010

Pattern 10 : *None, one, many (ITERATOR)*

- First story consider what would happen if no events or objects exist (usually just a simple error case)
- Second story for the case of one object or event existing.
- Final story you can then add how it changes when multiple objects or events in play.

  - Empty list handling…
  - Only one… probes
  - Last one…
  - Too many, need a scroll bar

Pattern 11 – Break Out; a Spike (SPIKE)

- If the story is too large or overly complex, or if the implementation is poorly understood, build a functional or technical spike to figure it out.

- Research PCI Gen1

- …Research the available technologies for
- …Create mockups

Source: Dean Leffingwell, Agile Software Requirements, Addison-Wesley, 2010

The 11 patterns

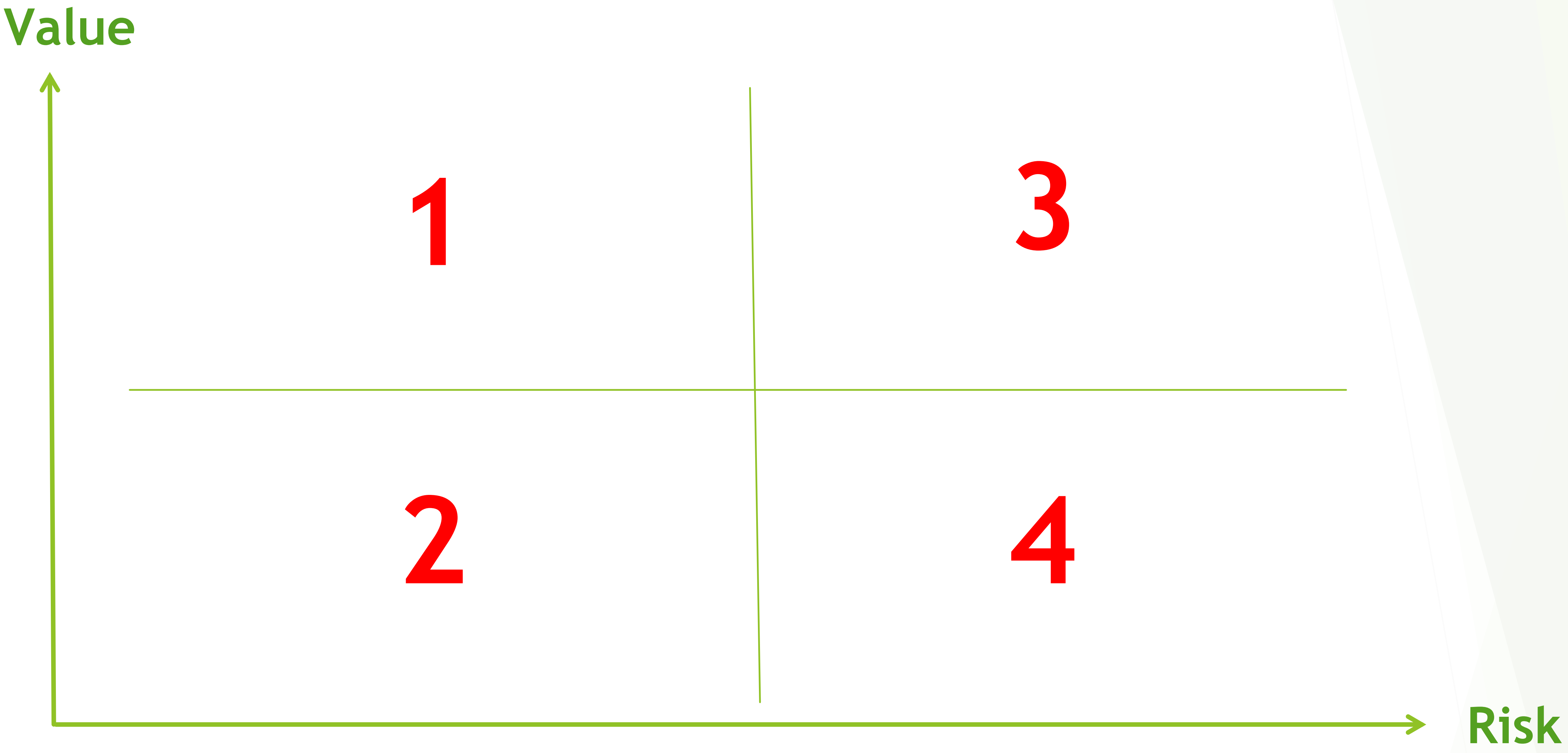
- Workflow stages
- Trimmed branches
- Major effort
- MMF
- Data Source & target
- UI
- System Abilities
- CRUD
- Use Case Pieces
- Iterator
- Spike

Which Pattern to Use

- **Choose the split that lets you de-prioritize or throw away a story.** The 80/20 principle says that most of the value of a user story comes from a small share of the functionality. Helps to get rid of low value stuff
- **Choose the split that gets you more equally sized small stories.** The split that turns an 8 point story into four 2 point stories is more useful than the one that produces a 5 and a 3.
- Choose the split that completes an **MMF earlier**

Some more tricks and techniques

- Ask the team what they can accomplish with half budget /estimate
- 5 Why's to identify the real purpose of a MMF and gather more information (Maybe you can have better solution, less output for more outcome)
- If you feel that you can't break it more (in a sense of bringing business value) think about something that's small enough to get feedback on quality (the smallest testable story)


Value
1
3
2
4
Risk

How to prioritize stories within an MMF

1. **Reduce risk**

   Business first, technology, budget & time.

2. **Earlier feedback**

   Integration as a way of feedback;

   Business feedback – demo;

   Quality, compliance enablers

3. **Larger degree of freedom**

   Integrate first; walking skeleton; high level architectural decisions first.

Not Everything is a Story

- Non Functional
  - Non stories
  - Constraints
  - Acceptance Testing
  - Compliance / safety
- Wireframes

Release planning… & Story mapping

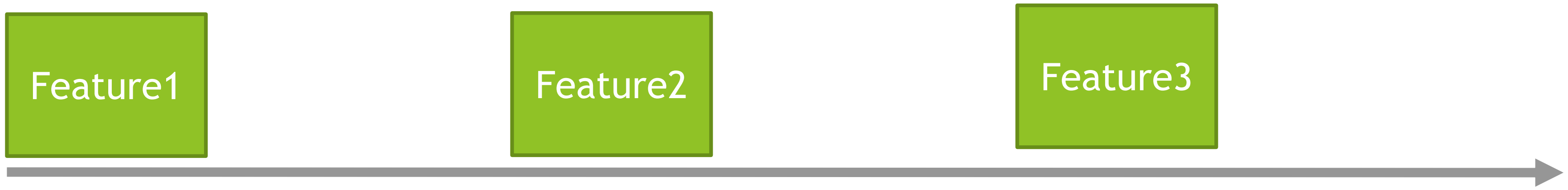
Story Maps – How to create the Mona Lisa and the Elephant Carpaccio

Thanks to
Jeff Patton

Gary Levitt, owner & designer of Mad Mimi

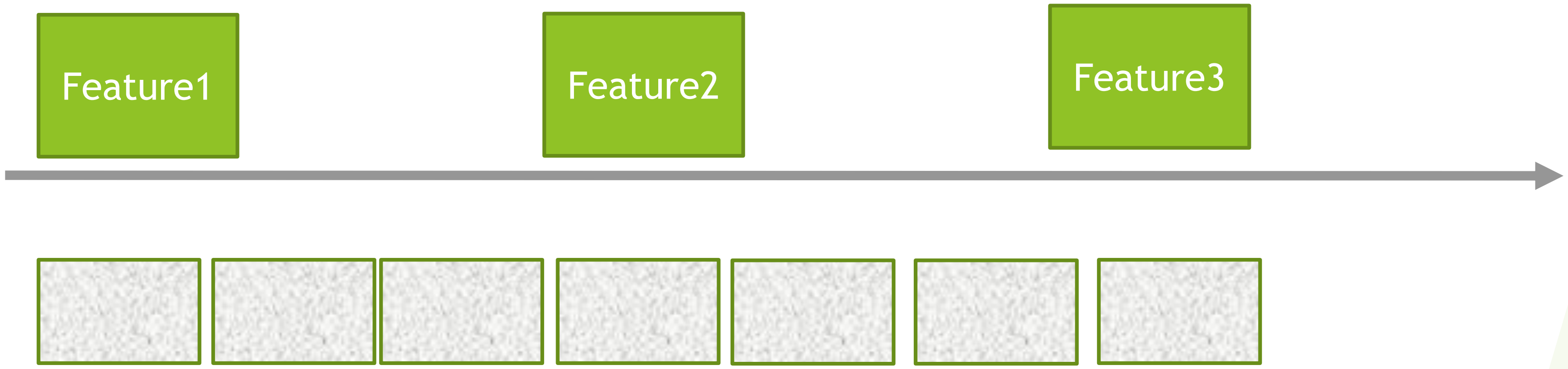
Seeing the bigger picture

- Tell a big story of the product by starting with the major user activities
  - Arrange activities left to right in the order you'd explain them to someone when asked the question: "What do people do with this system?"



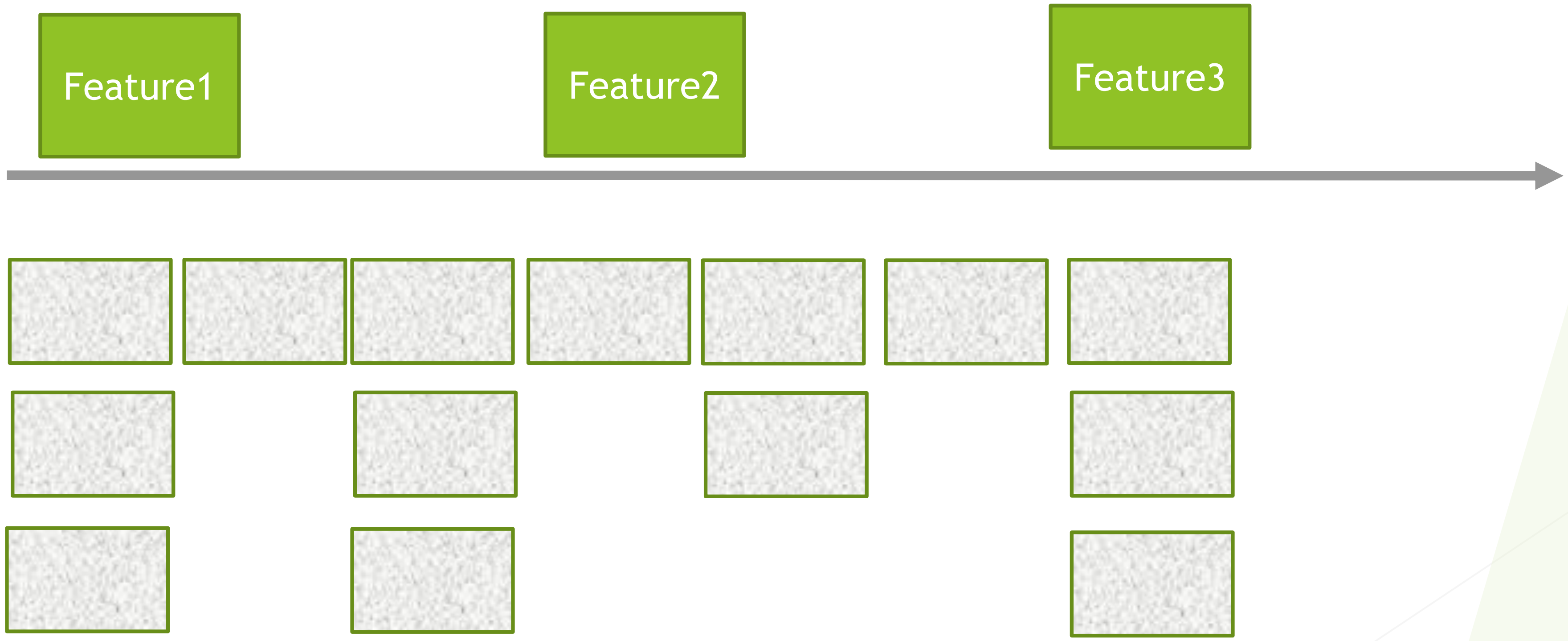
Turning activities into user stories

- Add task-centric stories in under each activity in workflow order left to right.
  - If you were to explain to someone what a person typically does in this activity, arrange tasks in the order you'd tell the story. Don't get too uptight about the order.



More complex stories

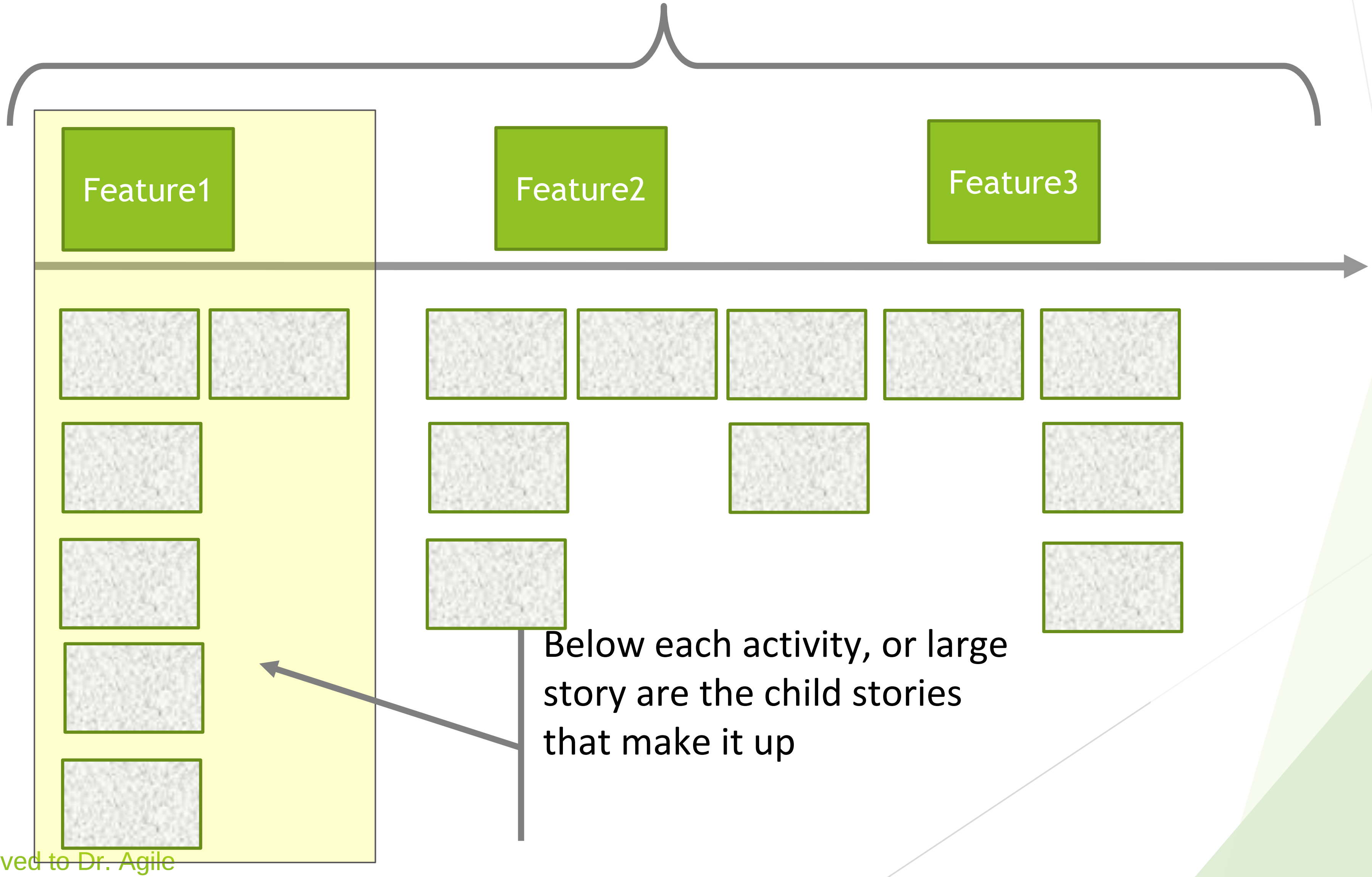
- Overlap user tasks vertically if a user may do one of several tasks at approximately the same time
  - If in telling the story I say the systems' user typically "does this or this or this, and then does that," "or's" signal a stacking vertically, "and then's" signal stepping horizontally.



© All rights reserved to Dr. Agile

A typical map

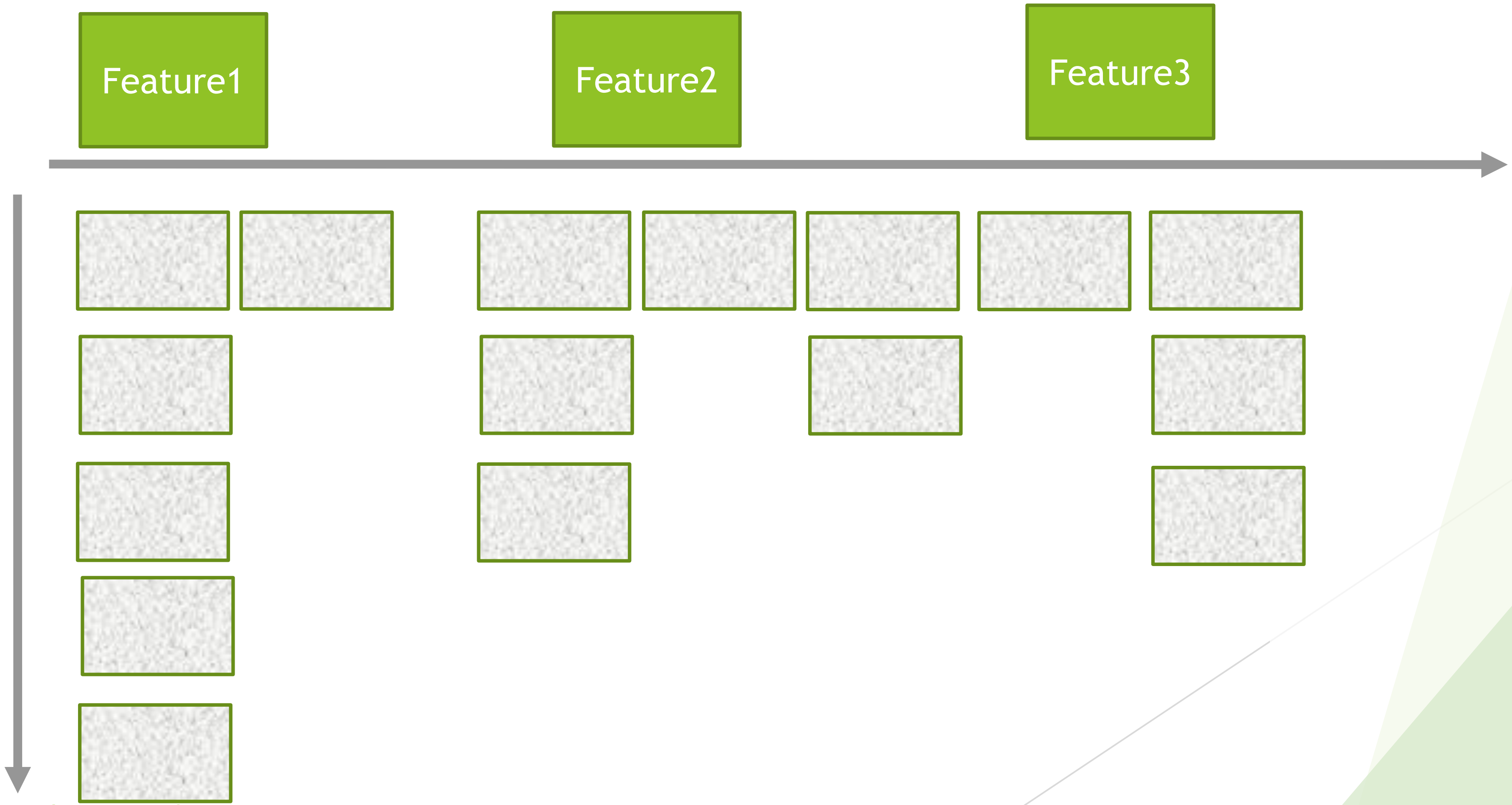
- Reading the activities across the top of the system helps us understand end-to-end use of the system.



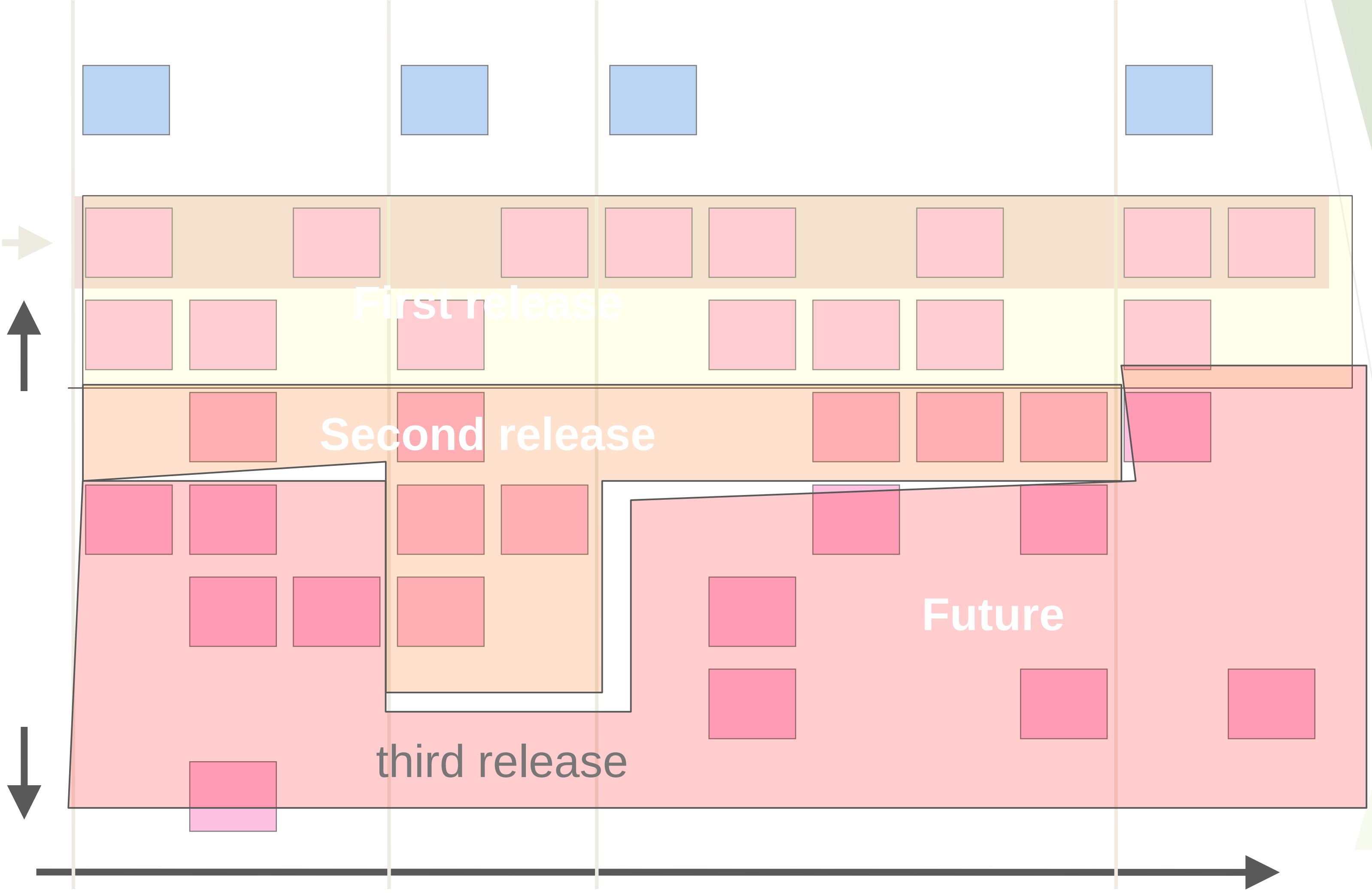
Below each activity, or large story are the child stories that make it up

User story mapping - for release planning

- Add a vertical axis to indicate necessity
- Move tasks up and down this axis to indicate how necessary they are to the activity.
  - For a user to successfully engage in this activity, is it necessary they perform this task? If it's not absolutely necessary, how critical is it?



Now we can start planning the releases



- Choose coherent groups of features that consider the span of business functionality and user activities
- Support all necessary activities with the first release
- Improve activity support and add additional activities with subsequent releases

Summary

- What did we miss?
- Unanswered questions?
- Action items… and the road ahead

Retrospective