SSW-555: Agile Methods for Software Development

Agile Cultures
Use Cases and User Stories
Week 2

Today’s Topics

- Compare organizations and cultures in Plan Driven and Agile organizations.
  - Roles of developers, manages, and executives

Acknowledgement: Introduction to Agile Methods, Ashmore and Runyan, 2015.

“Office Space”

https://www.youtube.com/watch?v=zqjQDP9KX6E

“HBO Silicon Valley”

https://www.youtube.com/watch?v=oyVksFviJVE

Corporate Culture

Plan Driven Cultures	Agile Cultures
Managers assign teams	Self organizing teams
Individuals work for the manager	Individuals work for the team
Individual measured on individual achievements	Individuals measured on team achievements
Manager assigns task	Team members select tasks
Manager responsible for improvement	Team responsible for reflection and continuous improvement
Infrequent deliveries	Frequent/continuous deliveries
Infrequent feedback from customers	Frequent/continuous feedback from customers
“Us and them” e.g. testing, Ops	“Us” e.g. testing, Ops

What can go wrong with the team?

- Dysfunctional teams
  - Fail to self organize: “Tell me what to do…”
  - Hostility to members: “That’s not my job…”
- Inability to adapt to change
  - Losing private office is hard
- Lacking commitment

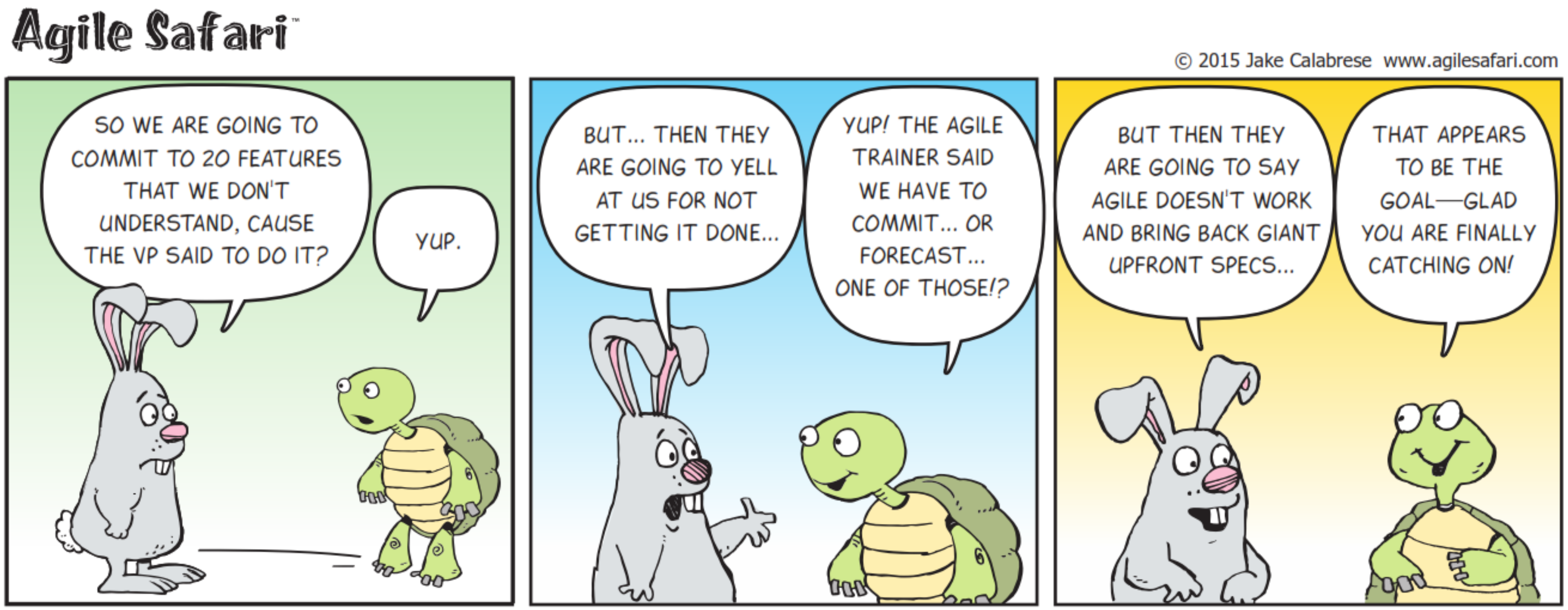
Role of Managers

Plan Driven Cultures	Agile Cultures
Define solutions to be implemented by the team	Asking questions while allowing the team to create the solution
Define roles and responsibilities	Help team to self organize
Leading the effort	Enabling the team while clearing roadblocks to success
Command and Control of the team	Trusting the team
Manager owns the problem	Team owns the problem

Role of Executives

Plan Driven Cultures	Agile Cultures
Clear, unchanging direction defined early in the process	Embrace changes
Manage by business case	Rapid delivery and inspection
Attempt to understand entire problem and outcome in advance	Quick prototype solutions to understand impact

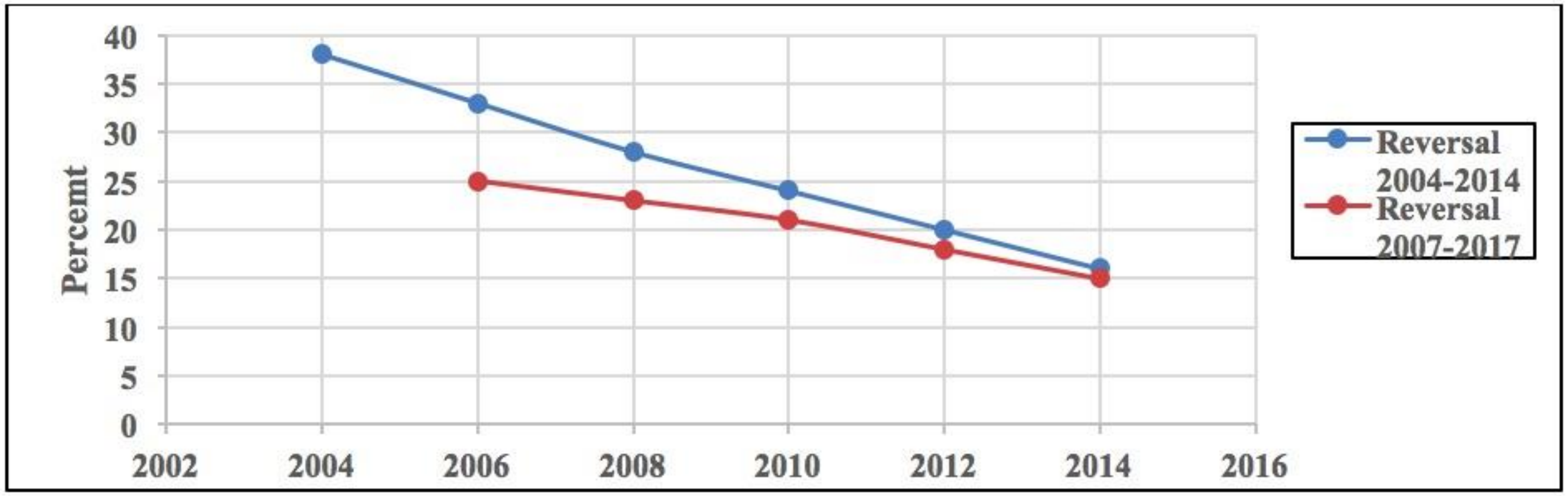
Agile doesn't work for everyone



Agile Safari™
© 2015 Jake Calabrese www.agilesafari.com
SO WE ARE GOING TO COMMIT TO 20 FEATURES THAT WE DON'T UNDERSTAND, CAUSE THE VP SAID TO DO IT?
YUP.
BUT... THEN THEY ARE GOING TO YELL AT US FOR NOT GETTING IT DONE...
YUP! THE AGILE TRAINER SAID WE HAVE TO COMMIT... OR FORECAST... ONE OF THOSE!?
BUT THEN THEY ARE GOING TO SAY AGILE DOESN'T WORK AND BRING BACK GIANT UPFRONT SPECS...
THAT APPEARS TO BE THE GOAL—GLAD YOU ARE FINALLY CATCHING ON!

http://agileforall.com/category/agile-safari/

Agile Reversal Rate



https://www.infoq.com/articles/reifer-agile-study-2017

- Poor preparation, mismatches, and a lack of senior management support.
- Be patient. The old proverb that Rome was not built in a day holds true.

谚语

Agile doesn’t work for everyone

- Not all teams successfully transition to Agile Methods
- Failure may be attributed to:
  - Lack of clarity across the team
  - Using only the worst of Waterfall and Agile together:
    - Forcing frequent short deliverables without the Agile advantages
    - Inadequate training or support for Agile Methods
      - Yesterday: product manager; today: scrum master
    - Failing to use automated testing and/or continuous integration
    - Continuing to plan everything in advance and not allowing change
    - Failing to change employee performance metrics
    - Failing to inspect and adapt – Andy Hunt

Conclusions

Today's Topics Cont'd

Gather features and requirements for a new system

DICK DISCOVERS WHY IT'S NOT ALWAYS EASY BEING A BUSINESS ANALYST
SO I UNDERSTAND YOU NEED A HORSE.
THAT'S RIGHT, A HORSE. RIGHT AWAY.
BUT YOU DON'T NEED TO RIDE IT.
OF COURSE NOT. I SAID I NEED A *HORSE*.
AND IT SHOULD HAVE HORNS, AND YOU'RE GOING TO NEED TO MILK IT.
WHAT ARE YOU, A HALFWIT? WHAT ELSE WOULD I DO WITH A HORSE?
I'LL SEE WHAT I CAN DO.

Requirements: Plan Driven

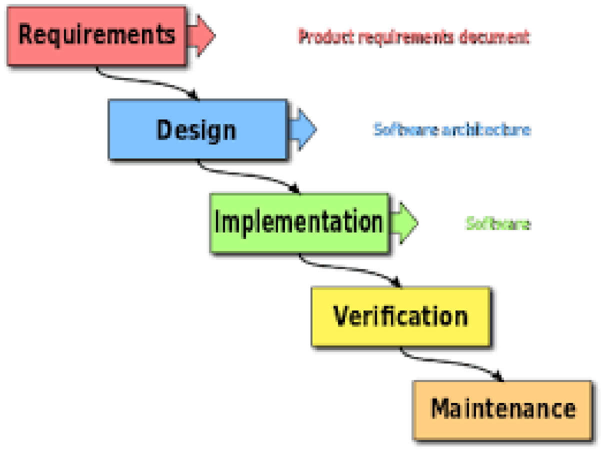
Use Cases: RUP approach

User Stories: Agile approach

http://www.agileadvice.com/2015/04/15/referenceinformation/summary-of-user-stories-the-three-cs-and-invest/

Plan Driven Requirements

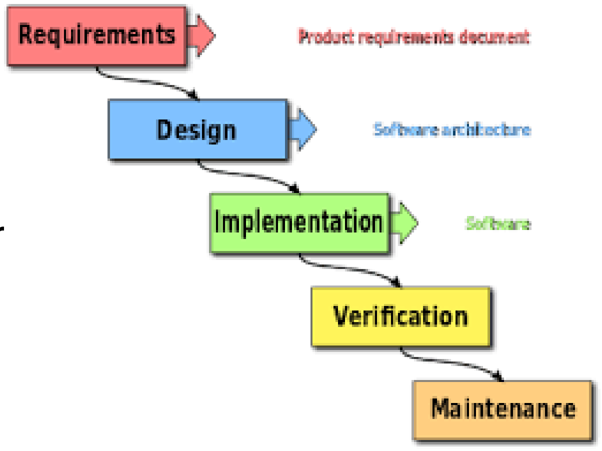
- Gathering requirements is typically the first step in a plan driven SDLC
- Business Analyst interviews the customer to extract features required from the system
  - *The system **shall** allow the user to enter her name, address, and email*
  - *The system **shall** be available 24x7x365 with 0.99% reliability*
- Business Requirements Document (BRD)
  - Formal contract between the customer and the team delivering the product



https://upload.wikimedia.org/wikipedia/commons/thumb/e/e2/Waterfall_model.svg/1280px-Waterfall_model.svg.png

Business Requirements Document (BRD)

- All requirements are gathered and reviewed by the business analysts before handing over to development
  - Little or no discussion between customers and developers
- Describes all of the features needed by the customer
  - Functional requirements, e.g. features
  - Non-functional requires, e.g. availability, …
- Limitations
  - Assumption of completeness
  - Difficult to change
  - Not created by developers and thrown over the wall to the developers



https://upload.wikimedia.org/wikipedia/commons/thumb/e/e2/Waterfall_model.svg/1280px-Waterfall_model.svg.png

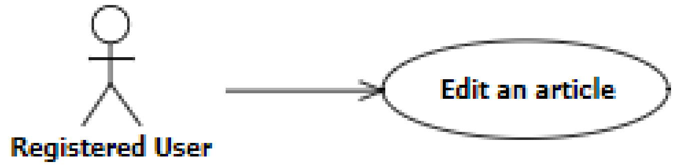
Example Domain

MSS: Music Streaming System

- Spotify, Apple Music, Pandora, …
- Users choose music to stream to their phone, laptop, or device
- Web based solution

Use Cases

- Developed by Ivar Jacobson in 1980s for OO based methods
- Part of UML and RUP (Rational Unified Process)
- Each ***use case*** describes:
    - A scenario
    - The actors in the scenario
    - Actors may be people or systems
    - A sequence of actions between the actor and the system
    - An observable result of value to a particular actor
    - Actor’s intention/what does the actor want?



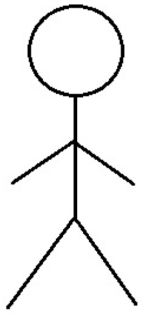
https://en.wikipedia.org/wiki/Use_case

Use Case Method

1. Identify the actors
2. Identify the use cases
3. Identify actor/use case relationships
4. Outline scenarios

1. Identify the actors

- Who uses the system?
- Who gets information from the system?
- Who provides information to the system?
- Who supports the system?
- What other systems interact with this system?
- Remember that actors may be other systems
- MSS Example:
  - Users, marketing, content owners, streaming servers, …



2. Identify the Use Cases

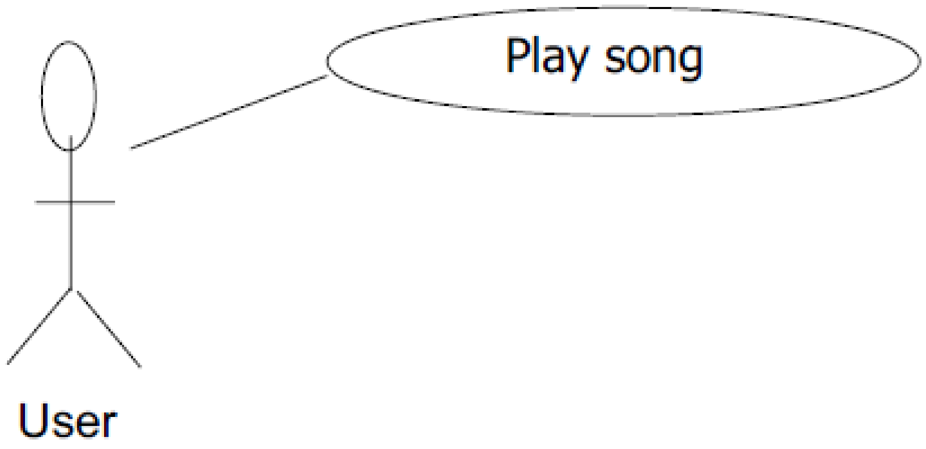
- What are the intentions of each actor with respect to the system?
- Give a descriptive name
  - Start with an action verb
  - Describe the goal or intent
- Give a one-sentence description
  - E.g. Play song– user plays a song on device

MSS: Identify the Use Cases

- Use Case: Play song– user plays a song on device
  - The user should be able to play a song on her device and control playback with start, stop, pause, forward, back
- Use Case: Explore music – user explores available music
  - The user should be able to explore different music alternatives that can be played on the device
- Use Case: Collect user profile – gather information about the user
  - The system should collect demographic information from users

3. Identify Actor/ Use Case Relationships

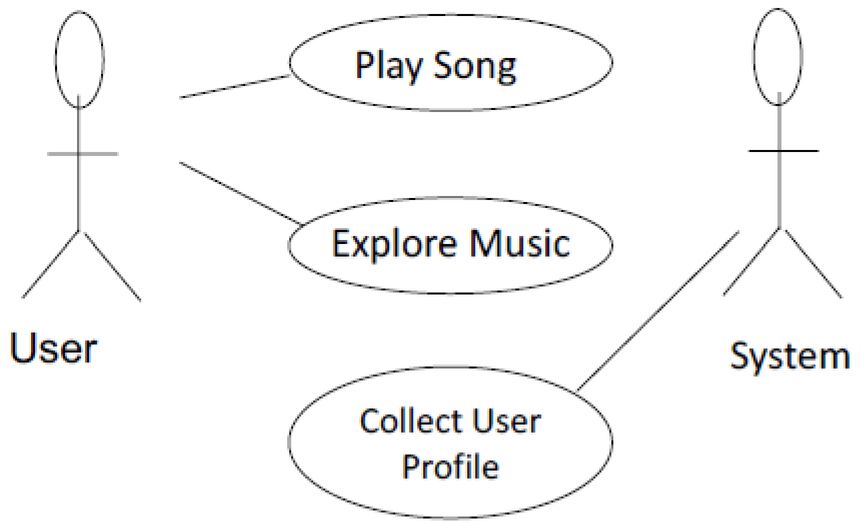
- Draw a diagram showing relationships between actors and use cases



Use Case: Play a song

3. Identify Actor/ Use Case Relationships

- Draw a diagram showing relationships between actors and use cases



4. Outline Scenarios

- Describe sequence of events in basic flow (sunny day scenario)
- Describe sequences of events in alternate flows (rainy day scenarios)

Play Music:

- Basic Flow: User selects song and plays song from beginning to end
- Alternate Flow: User selects song but song is not available for playing
- Alternate Flow: User fast forwards through first 30 seconds

Use Case Template

1. Name
2. Brief description
3. Actors
4. Basic flow
5. Alternate flows

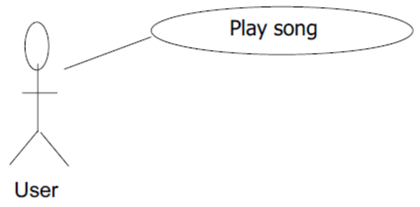
MSS: play song example

1. Name: Play song
2. Brief description: The user selects a song from MSS and plays the song
3. Actors: User
4. Basic flow:

1. User visits MSS home page
2. User selects a song from available content
3. User pushes start button
4. Song is streamed to the user’s device
5. Song plays on user’s device

5. Alternate flows

1. User selects song but doesn’t start playing
2. Song selected by user selected is not available for streaming
3. User fast forwards past first 30 seconds of song



Use Case Summary

- Use cases describe the scenarios in terms of actors and actions
- Specify "all possible" scenarios
- Typically include detailed instructions on how to accomplish those scenarios

1. Name
2. Brief description
3. Actors
4. Basic flow
5. Alternate flows

Use Cases vs. User Stories

- **Use Cases** – RUP approach for capturing requirements
- **User Stories** – agile approach for capturing requirements

  - "*A user story is to a use case as a gazelle is to a gazebo*" – Alistair Cockburn
  - "A use case is to a user story as an Elephant is to an Elevator" -- Bob Forrest, as he pointed out, "gazelle" and "gazebo" have 4 letters in common)

http://alistair.cockburn.us/A+user+story+is+to+a+use+case+as+a+gazelle+is+to+a+gazebo

User Stories: Agile Approach

- **Customers** communicate their needs via short statements
  - **Customers** provide the user stories with help from developers
  - Each User Story is a reminder for the customer and developer to discuss the issue
- Each statement describes the goal of an actor
  - "as a user I want to ..."
- **Customer** should decide **priority** of each user story

MSS Play song:

As a user I want to be able to play a song so I can listen to it from beginning to end so I can hear the song

Priority 1

3 C's of User Stories

C

- **The Card**
- User stories are frequently written on 3*5 cards

C

- **Conversation**
- A user story is a promise of a conversation between the customer and developers

C

- **Confirmation**
- Each user story includes tests to confirm that the feature has been delivered

The Card

- The Card, or written text of the User Story is best understood as an invitation to conversation, must address the "*who*", "*what*" and "*why*" of the story.
- The Card is usually follows the format similar to the one below

As a <user role> of the product,
I can <action>
So that <benefit>

http://www.agileadvice.com/2015/04/15/referenceinformation/summary-of-user-stories-the-three-cs-and-invest/

The Conversation

- The collaborative conversation facilitated by the ***Product Owner*** which involves all stakeholders and the team.
- As much as possible, this is an ***in-person*** conversation.
- The conversation is where the real value of the story lies and the written Card should be adjusted to reflect the current shared understanding of this conversation.
  - Customer resolve ambiguities
  - Developer estimates the effort

http://www.agileadvice.com/2015/04/15/referenceinformation/summary-of-user-stories-the-three-cs-and-invest/

The Confirmation

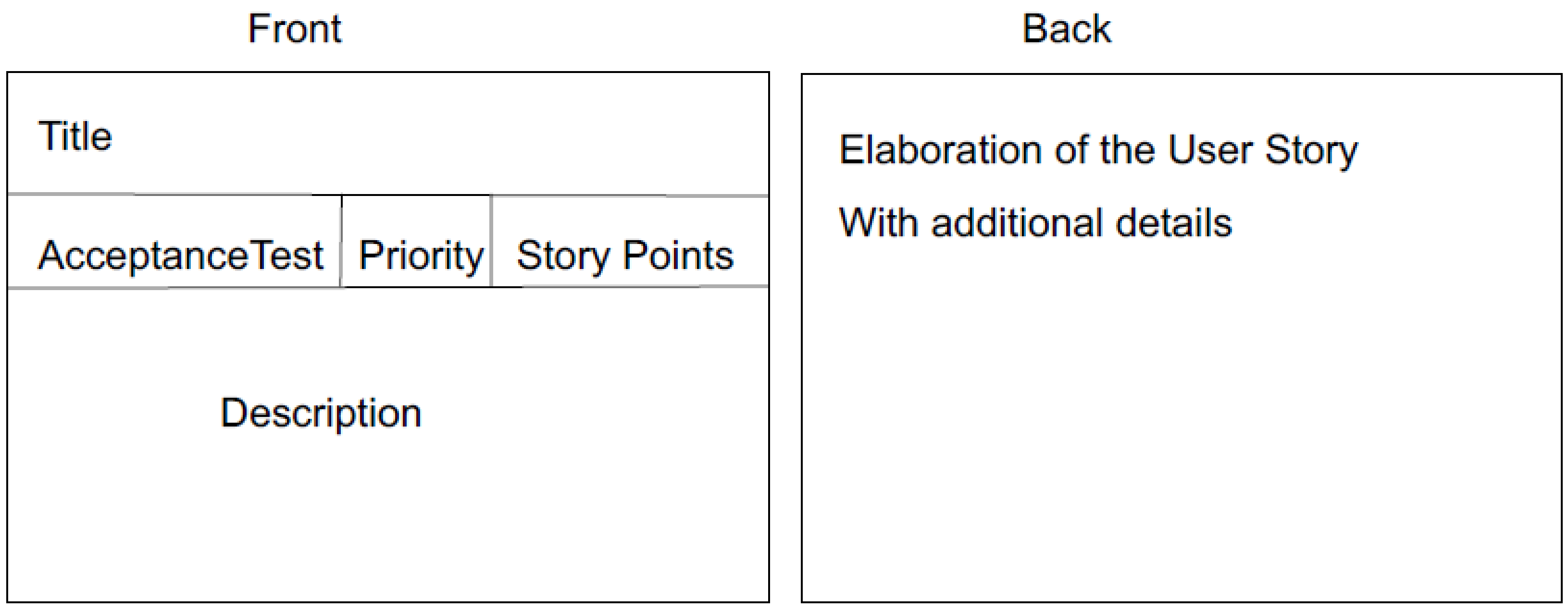
- The Product Owner must confirm that the story is complete before it can be considered “done”
- The team and the Product Owner check the “doneness” of each story in light of the Team’s current definition of “done”
- Specific acceptance criteria that is different from the current definition of “done” can be established for individual stories, but the current criteria must be well understood and agreed to by the Team.
- All associated acceptance tests should be in a passing state.

http://www.agileadvice.com/2015/04/15/referenceinformation/summary-of-user-stories-the-three-cs-and-invest/

User Story Components

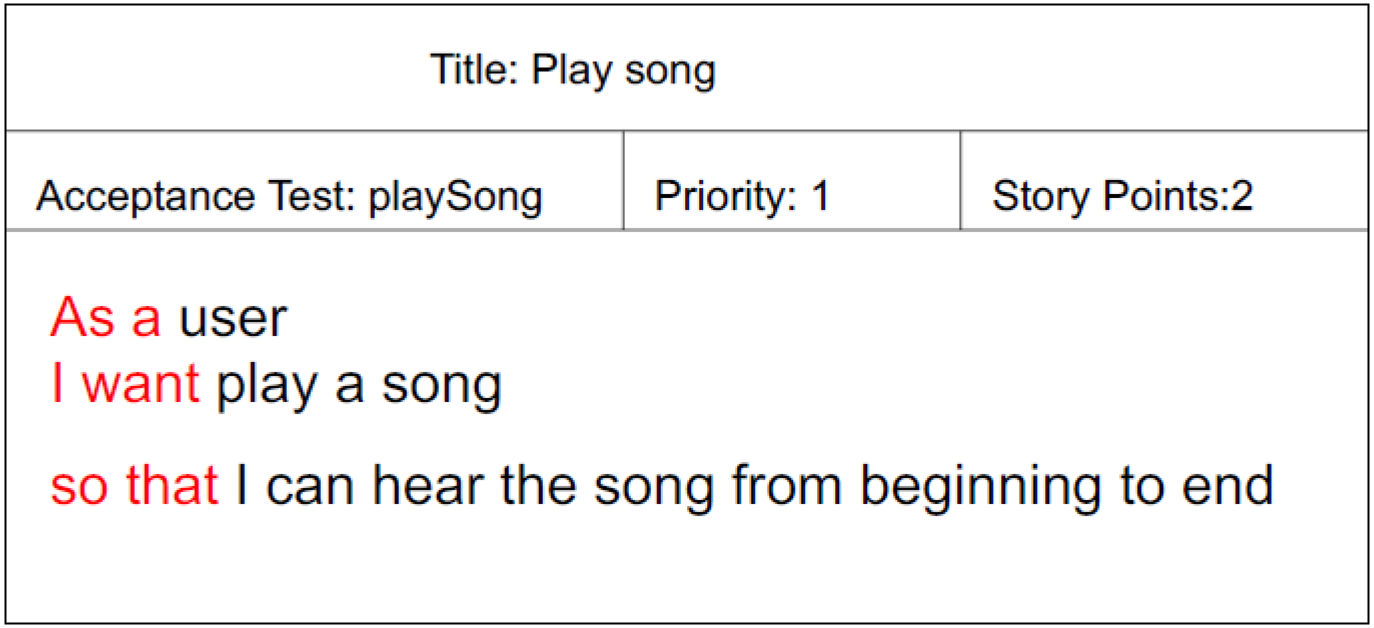
- **Title** – a short handle for the story. Present tense verb in active voice is desirable
- **Acceptance test** –the name of a method to test the story
  - How to determine if the functionality is provided?
  - Acceptance test helps to flesh out the details of the user story
- **Priority** – decided by the customer
- **Story points** – estimated time to implement expressed in relative units
- **Description** – one to three sentences describing the story

User Story Layout



3x5 index card

User Story: Play Song



Title: Play song		
Acceptance Test: playSong	Priority: 1	Story Points:2
As a user I want play a song so that I can hear the song from beginning to end		

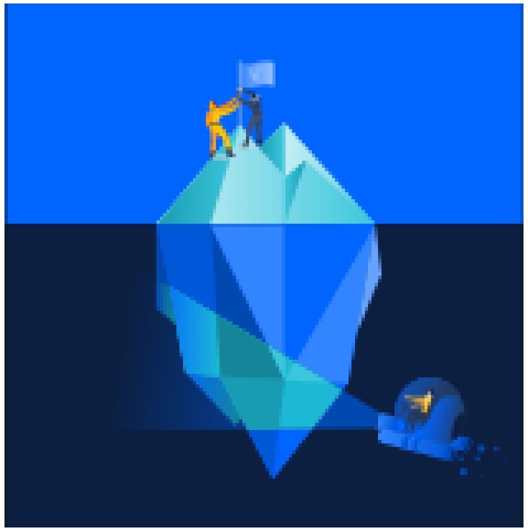
1= high priority, 2 story points = 2 days

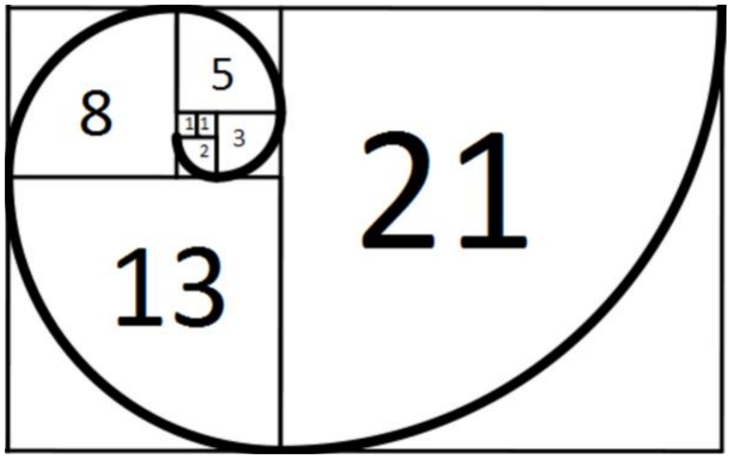
Estimating Time for User Stories

- Developers estimate how long each story will take
- Estimates are expressed in ***Story Points***
- Relative measure of effort
  - Task1 requires twice as much effort as Task2, e.g. developers can deliver 3 story points per day
- But, how do they know how long a new story will take?
  - previous experience
  - similar stories on project
  - use a consensus process to compare estimates

Story Points

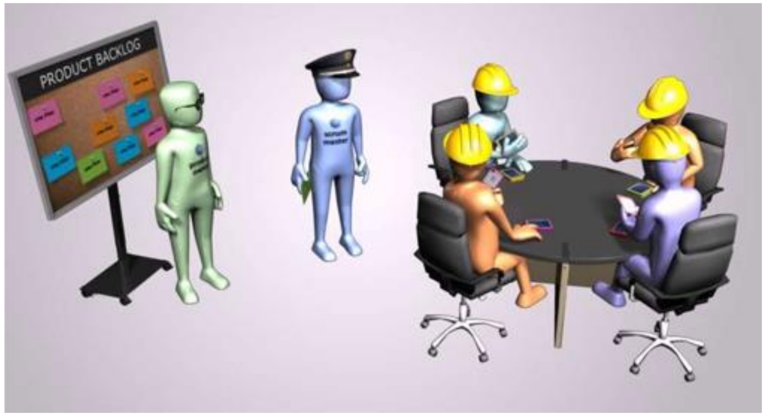
- Estimation is hard: it must take into account a slew of factors.
    - Involving everyone (developers, designers, testers, developers... everyone) on the team is key.
    - Each team member brings a different perspective on the product and the work required to deliver a user story.
- Story points rate the relative effort of work in a Fibonacci-like format: 0, 0.5, 1, 2, 3, 5, 8, 13, 20, 40, 100.
    - E.g. What's the difference between 6 and 7?
    - In Fibonacci serial, a number is the sum of the prior two numbers.





Planning Poker

- **Goal**: estimate relative effort for each user story
- **Participants**:
  - **Developers** estimate effort
  - **Scrum Master** optimizes the process
  - **Product owner** answers questions
- **Process:**
  - For each user story:
    - Describe the user story
    - Each developer assigns effort
    - Continue until consensus



https://www.youtube.com/watch?v=UJ-NnDficnE

Criteria for User Stories

- Each story should **add value** to the customer
  - Customers write user stories (with help from developers if needed)
- Stories need to be **small** enough that several can be completed per iteration
  - Replace big stories with several smaller stories
- Stories should be **independent** (as much as possible)
- Stories must be **testable** – like any requirement, if it cannot be tested, it's not a requirement!
- Include **non-functional** requirements as User Stories

INVEST in User Stories (Bill Wake)

I	• **Independent** in context and scheduling
N	• **Negotiable** between customer and developers
V	• **Valuable** to the customer
E	• **Estimable** value to customer & effort for developers
S	• **Small** in scope
T	• **Testable**: include sufficient details to allow testing

http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

Limitations of User Stories

- User Stories come from XP
  - Small teams with engaged customers
- What can go wrong?
  - Lack of look-ahead
  - Lack of context
  - Lack of completeness

Source: http://proquest.safaribooksonline.com/book/software-engineering-and-development/agiledevelopment/0321482751

User Story Limits and Solutions

- Lack of look-ahead
  - Developers need quick responses from customers
    - What if the appropriate customer isn't available when needed?
    - Anticipate that the answers may not be readily available
    - Developers need to plan far enough ahead to allow time for answers

Source: http://proquest.safaribooksonline.com/book/software-engineering-and-development/agiledevelopment/0321482751

User Story Limits and Solutions

- Lack of context
  - User Stories are expressed in simple, concise statements
  - What if the statement is terse/too concise?
    - Group together related User Stories
    - Provide relevant context

Source: http://proquest.safaribooksonline.com/book/software-engineering-and-development/agiledevelopment/0321482751

User Story Limits and Solutions

- Lack of completeness
  - Gaps may arise as the user stories and product evolve
    - Continue to add User Stories as the project evolves

Source: http://proquest.safaribooksonline.com/book/software-engineering-and-development/agiledevelopment/0321482751

When to use which?

- User Stories are more flexible and easier to write
  - Ideal for small projects:
    - where a dedicated customer responds to questions
    - where a small team can deliver the entire solution
- Use Cases are more rigorous
  - May be better for larger projects
  - Where risks are higher

Source: http://proquest.safaribooksonline.com/book/software-engineering-and-development/agiledevelopment/0321482751

Recap

- Compare organizations and cultures in Plan Driven and Agile organizations.
  - Agile may not work for everyone!

- Requirements in plan driven process
- Use cases in RUP process
- User story in Agile approaches

STEVENS

INSTITUTE of TECHNOLOGY

THE INNOVATION UNIVERSITY®

1870

stevens.edu

Thank You