STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

SSW-555: Agile Methods for Software Development

Agile Cultures

Use Cases and User Stories

Prof. Jim Rowland
Software Engineering
School of Systems and Enterprises

Today's topics

Goal: Compare organizations and cultures in Plan Driven and Agile organizations

Compare roles of developers, managers, and executives

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

"Office Space"

"HBO Silicon Valley"

Corporate Culture

Plan Driven Cultures	Agile Cultures
Managers assign teams	Self organizing teams
Individuals work for the manager	Individuals work for the team
Individuals measured on individual achievements	Individuals measured on team achievements
Manager assigns tasks	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 offices 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

Where would you rather work?

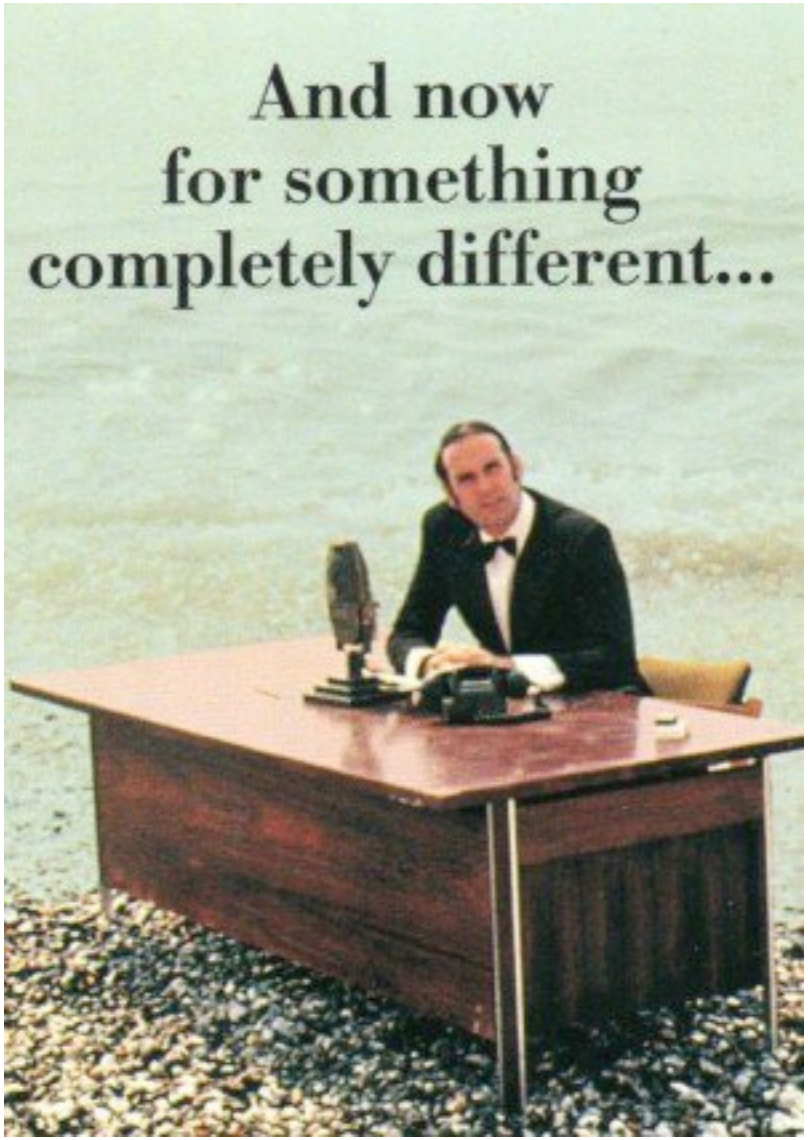
Agile doesn't work for everyone…

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


And now
for something
completely different...

Gathering requirements

Goal: Gather features and requirements for a new system

Compare and contrast three different approaches

- **Traditional Requirements** - Plan Driven method requirements
- **Use Cases** – RUP approach for capturing requirements
- **User Stories** – agile approach for capturing requirements

Beware: Use Cases and User Stories may be a contentious issue!

"*A user story is to a use case as a gazelle is to a gazebo*" – Alistair Cockburn

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

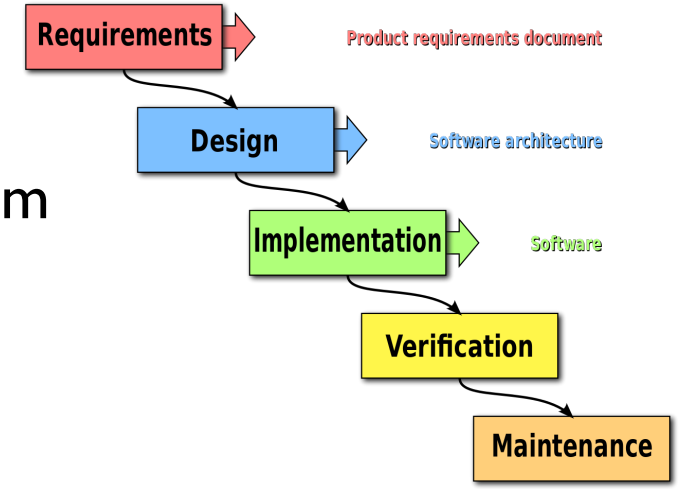
Plan driven requirements

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



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

Business Requirements Document (BRD)

Formal contract between the customer and the team delivering the product

Business Requirements Document (BRD)

All requirements are gathered and reviewed by the business analysts before handing over to development

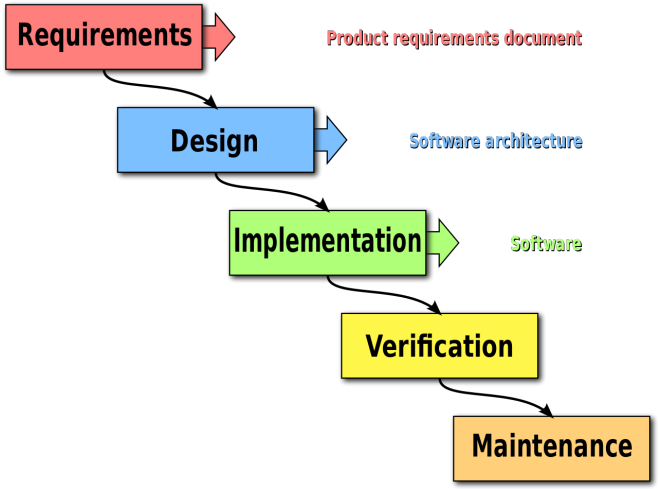
- Little or no discussion between customers and developers

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?

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

2. 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

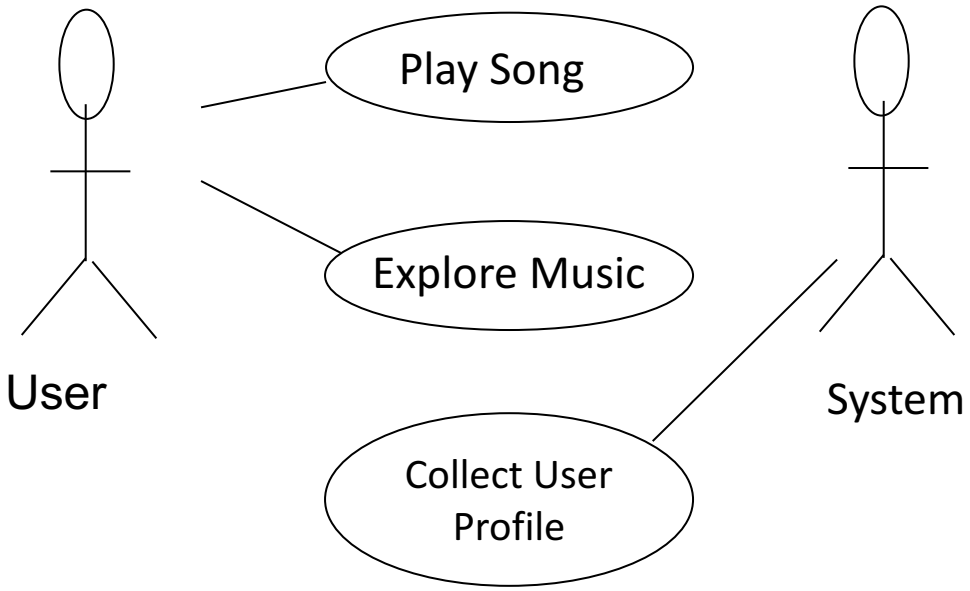
The user should be able to explore different music alternatives that can be played on the device

Use Case: Collect user profile – gather information about the user

The system should collect demographic information from users

3. Identify Actor/Use Case Relationships in a Use Case Diagram

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)

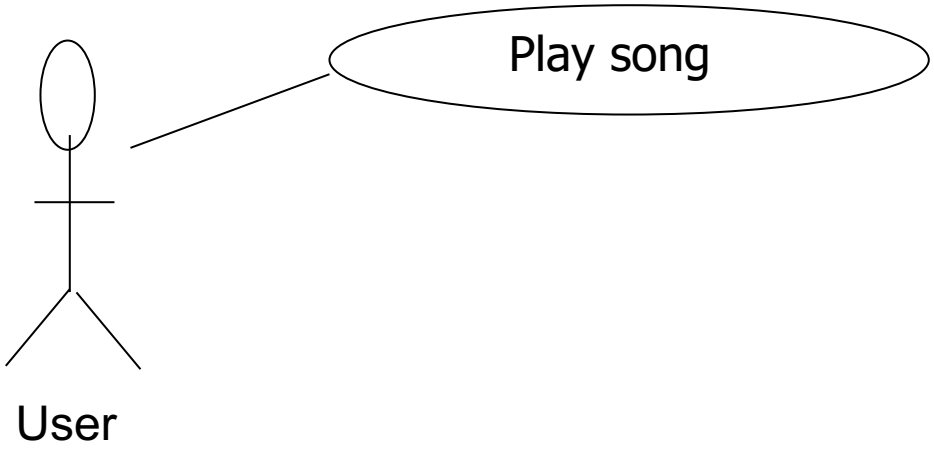
Play Music:

- 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



Play song use case 1 of 3

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

Play song use case 2 of 3

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

Play song use case 3 of 3

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

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 3x5 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

Developers help to elaborate user stories

Sometimes user stories need elaboration to explain exactly what is needed

- Customer and developers resolve any ambiguities

A **developer** estimates the **effort** required for each user story

MSS Play Song:

- Effort: 2 story points

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

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 detail to allow testing

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

User story layout

Front

Title		
AcceptanceTest	Priority	Story Points
Description		

Back

Elaboration of the User Story

With additional details

3x5 index card

User story: play song

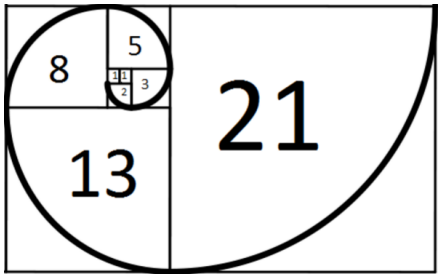
Title: Play song		
Acceptance Test: playSong	Priority: 1	Story Points:2
As a user I want play a song so that I can hear the song from beginning to end		

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

$Task_1$ requires twice as much effort as $Task_2$

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

Planning poker

Goal: estimate relative effort for each user story

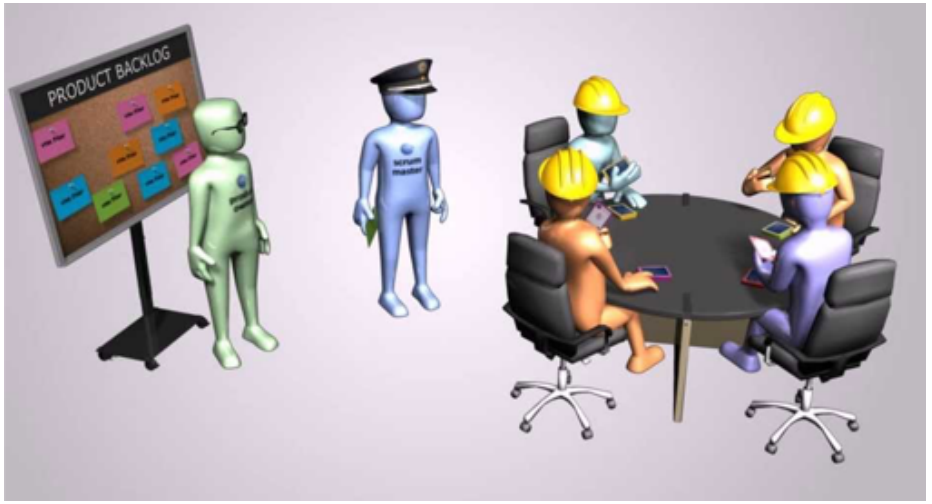
Participants:

- **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



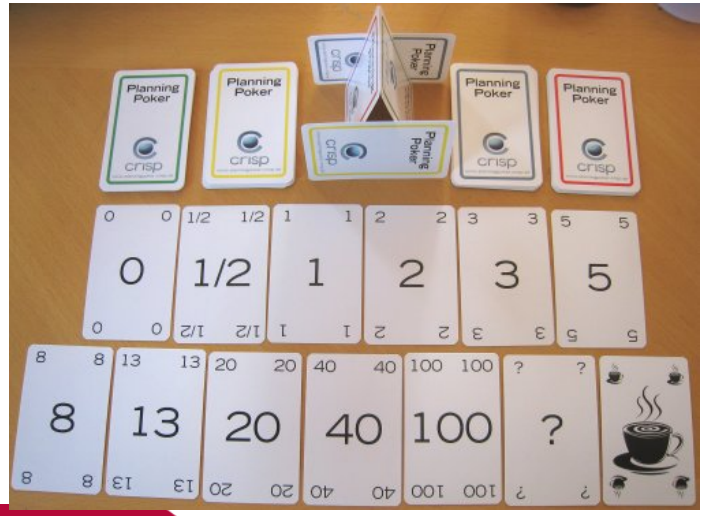
https://www.youtube.com/watch?v=UJ-NnDficnE

Planning poker

Consensus process similar to Wideband Delphi (Boehm)

1. Each expert provides an estimate simultaneously
2. All estimates are shared, perhaps with explanations
3. Continue until consensus reached

Planning Poker uses special playing cards to simplify estimating, and to encourage reasonable estimates



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/agile-development/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/agile-development/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/agile-development/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/agile-development/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/agile-development/0321482751

Questions?