CSC 171 – Module 04

Teams Delivery Features

Project Development Scenarios

- You gave me what I asked for. It’s not what I need.
- The market changed. I don’t want this now. I want something else, yesterday.
- This isn’t what I wanted. You read what I wrote, but that’s not what I meant.
- We need to ship tomorrow. What do you have?

Terms

- Definitions of user story, theme, and epic are not always the same
- User story
  - Something a "user" wants
  - A common format: As a [user], I [want to], [so that].
    - As a customer, I want to be able to zoom content to fit the screen, so that I can see the entire page.
    - As a system, I want to monitor race conditions, so I can reboot the master processor.
  - 3 C's: Card, Conversation, Confirmation
  - INVEST: set of criteria to assess the quality of a user story
    - Independent, Negotiable, Valuable, Estimable, Small, Testable
- Epic
  - A large user story that cannot be delivered within a single iteration or is large enough that it can be split into smaller user stories..
    - As a customer, I want to be able to zoom content, so that I can view it better.
      - Zooming includes "Zoom in", "Zoom out", "Zoom to fit", "Actual Size", "Zoom to Selection"
- Theme
  - A collection of user stories.
  - Commonly, a theme consists of multiple epics and an epic consists of multiple user stories.

Terms – Cont.

- Spike
  - A timeboxed experiment aimed at answering questions or gathering information related to a user story or an epic, rather than at producing shippable product increment.
  - Example: A user story cannot be estimated until some technical question or design problem is resolved.
- MVP (Minimum Viable Product)
  - Has enough functionality for the customer to use, even if it's not the full functionality.
  - To test a product hypothesis with minimal resources, accelerate learning, get the product to early customers as soon as possible, …
  - An example: Airbnb
- MVE (Minimum viable Experiment)
  - An experiment designed to get feedback on how to proceed in your product.
  - An example: https://dzone.com/articles/thinking-about-minimum-viable-experiments
- Feature
  - A client-valued functionality
- Feature set
  - A set of features

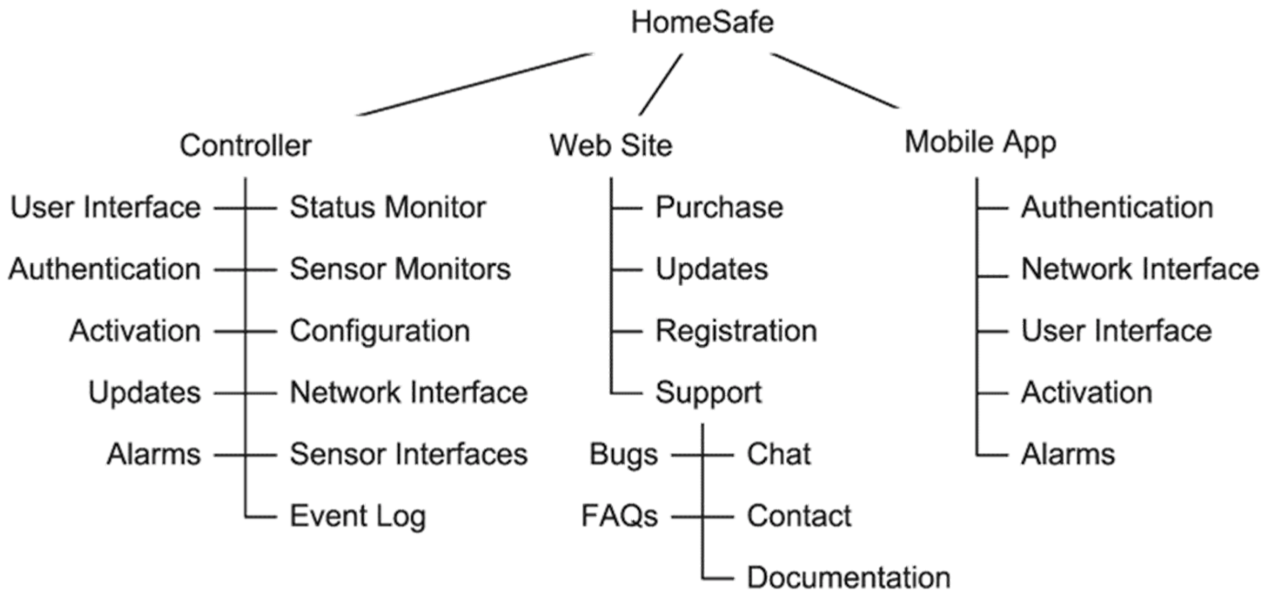
Feature-Driven Development [2]

- An iterative and incremental software development process
- Five basic activities
  - Development overall model (an example domain model)
    - High-level walkthrough of the scope
    - Creating detailed domain models of each domain area
    - Merging domain area models into overall model
  - Build feature list
    - Features are small pieces of client-valued functions in the form
      - <action> <result> <object>
      - e.g., Calculate the total of a sale
  - Plan by feature
    - Development and assign feature ownership
  - Design by feature
    - Milestones: domain walkthrough, design, design inspection
  - Build by feature
    - Milestones: code, code inspection, promote to build
- Milestones of a feature

Domain Walkthrough	Design	Design Inspection	Code	Code Inspection	Promote To Build
1%	40%	3%	45%	10%	1%

Traditional Planning

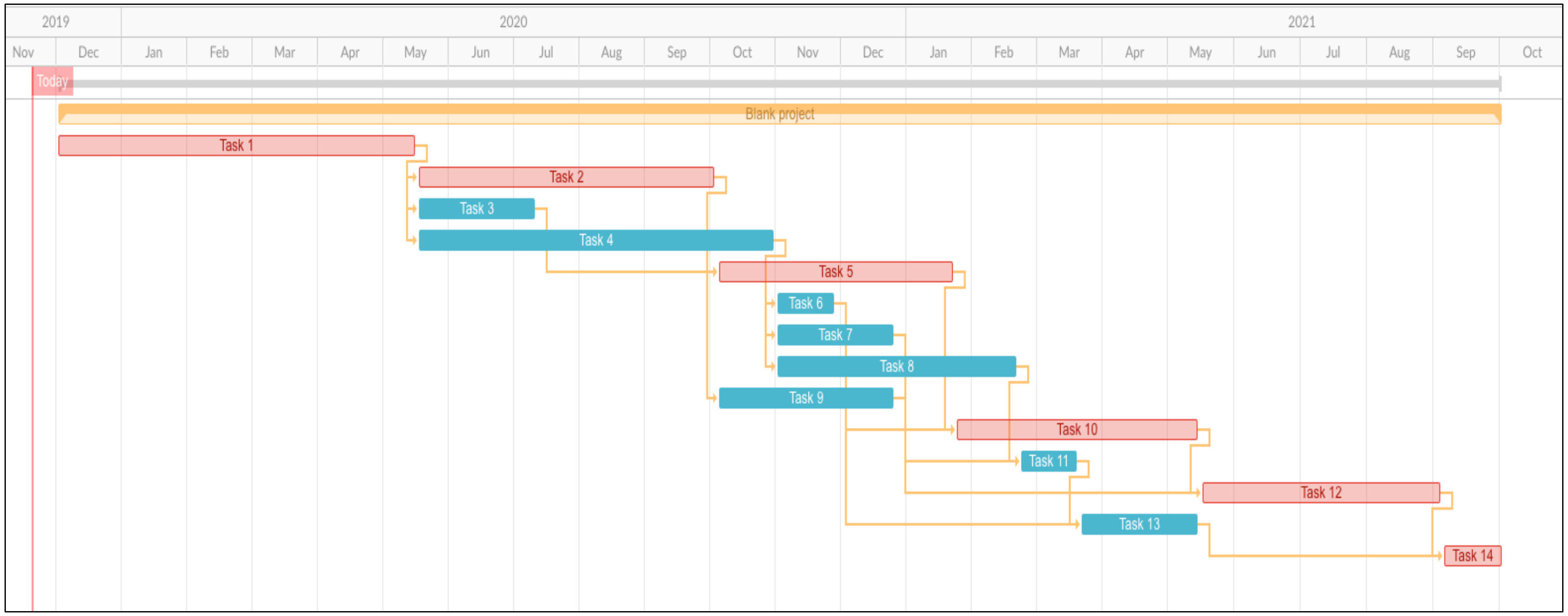
- Traditional processes
  - Most planning activities occur "up front"
  - Organizing deliverables into Work Breakdown Structures (WBS)
    - A hierarchical decomposition of the work needed to produce deliverables
    - Leaf nodes are called work packages requiring between 8 and 80 person-hours
  - Scheduling project using Gantt Charts or Critical Path Methods
- An example WBS



Gantt Charts

- Tasks
  - Represented by rectangles, whose widths are proportional to tasks' durations
  - The left side is positioned at the earliest possible start time
  - Added in vertical direction
- Dependencies
  - Represented by arrows
- Time
  - Increases in the horizontal direction

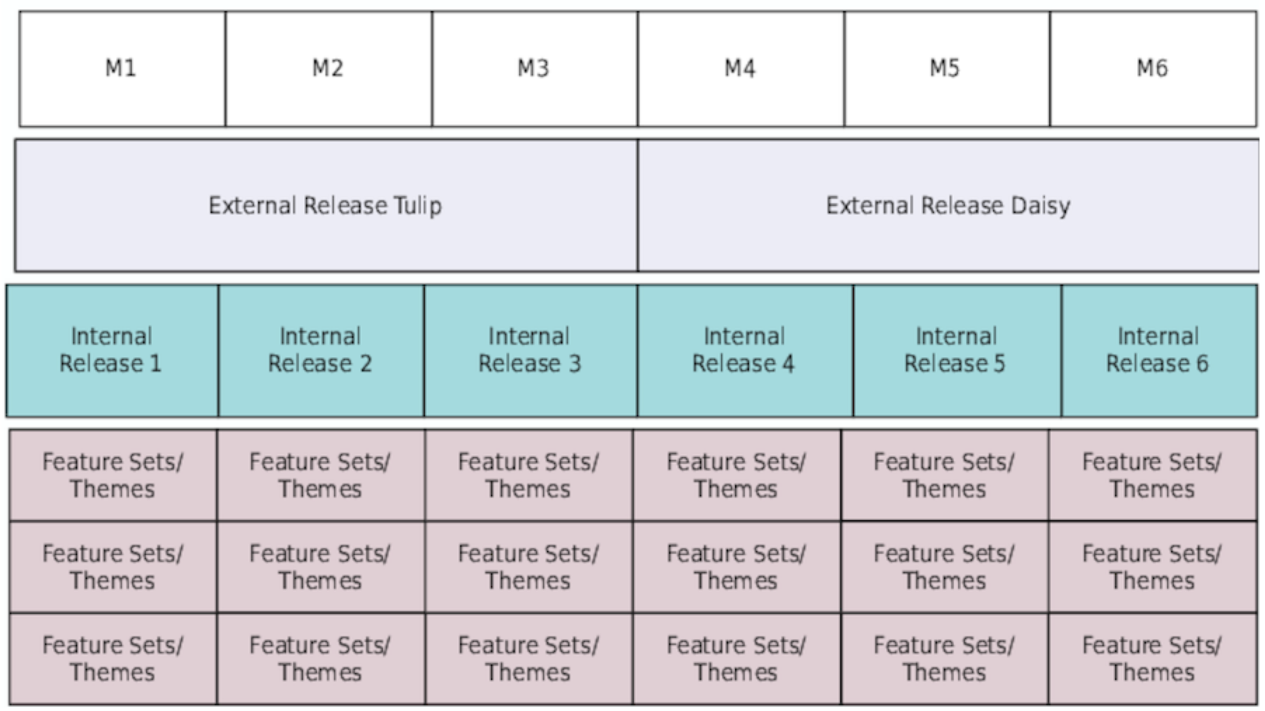
- Should Gantt charts be used with agile development?
  - Some think they should not: e.g., Jeff Sutherland's opinion
  - Some think they can be used in the context of enterprise-grade scale agile development



Agile Planning

- Plan at several levels
  - Portfolio plan
    - The mixture of products that define the organization's strategy
  - Product plan (roadmap)
    - Include feature sets / Themes, internal and external releases for near future (e.g., next 6 months)
  - Release (project) plan
  - Iteration plan
    - Such as the plan created during sprint planning meeting
  - Daily plan
    - Every day during daily stand-up
- Plans are updated on regular basis
  - e.g., Product roadmap updated bi-weekly

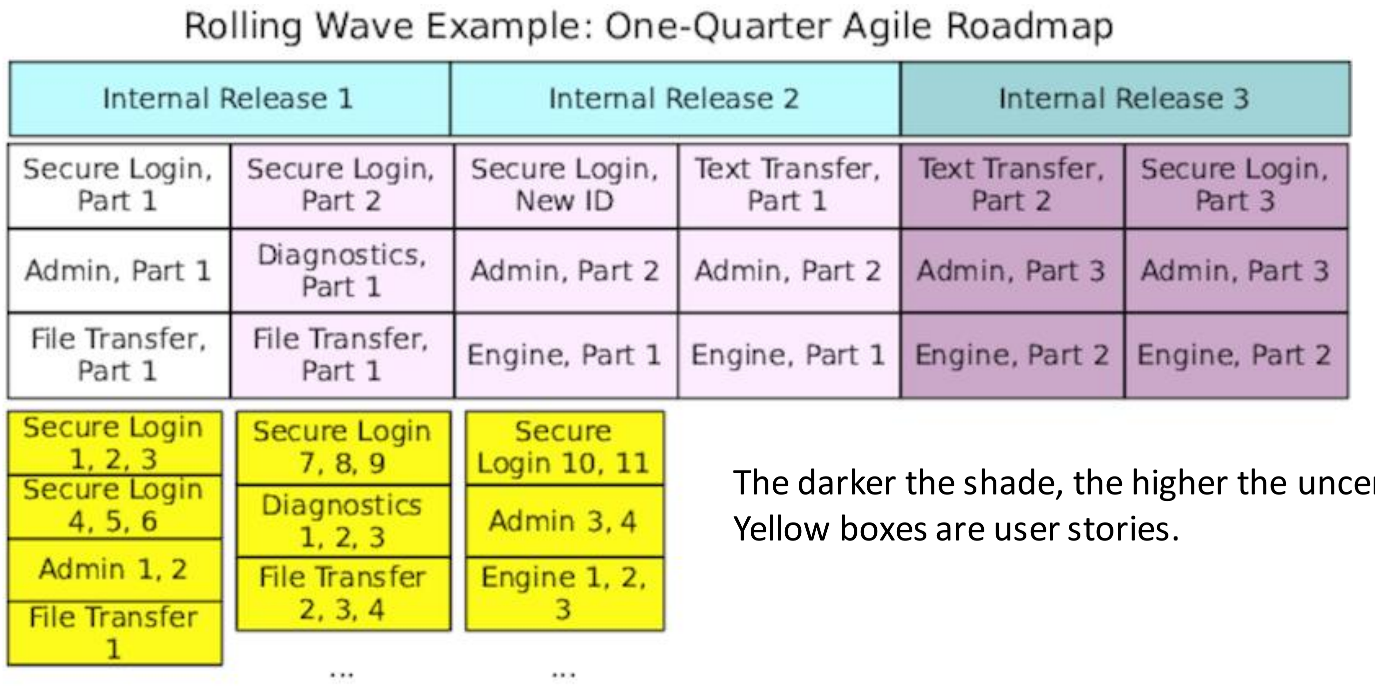
Product Roadmap Example [1]



M1	M2	M3	M4	M5	M6
External Release Tulip			External Release Daisy		
Internal Release 1	Internal Release 2	Internal Release 3	Internal Release 4	Internal Release 5	Internal Release 6
Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes
Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes
Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes	Feature Sets/ Themes

"When people try to plan in great detail, they tend to over plan. That creates waste."
"Accuracy of a plan decreases rapidly the further we attemp to plan beyond where we can see"

Rolling-ware Product Roadmap [1]



Rolling Wave Example: One-Quarter Agile Roadmap

Internal Release 1		Internal Release 2		Internal Release 3	
Secure Login, Part 1	Secure Login, Part 2	Secure Login, New ID	Text Transfer, Part 1	Text Transfer, Part 2	Secure Login, Part 3
Admin, Part 1	Diagnostics, Part 1	Admin, Part 2	Admin, Part 2	Admin, Part 3	Admin, Part 3
File Transfer, Part 1	File Transfer, Part 1	Engine, Part 1	Engine, Part 1	Engine, Part 2	Engine, Part 2

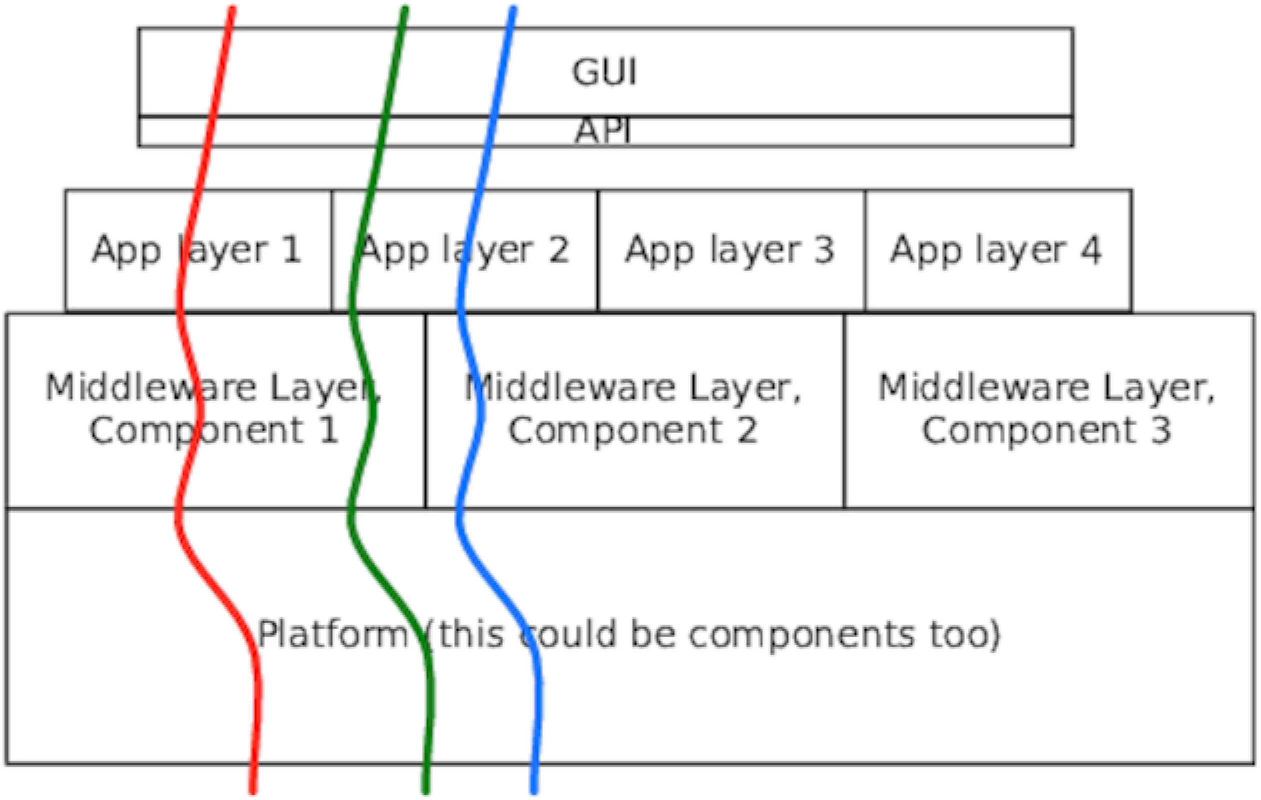
Secure Login 1, 2, 3	Secure Login 7, 8, 9	Secure Login 10, 11
Secure Login 4, 5, 6	Diagnostics 1, 2, 3	Admin 3, 4
Admin 1, 2	File Transfer 2, 3, 4	Engine 1, 2, 3
File Transfer 1	...	...

The darker the shade, the higher the uncertainty.
Yellow boxes are user stories.

Related agile practice: Frequent Release
Release for learning, feedback, and value

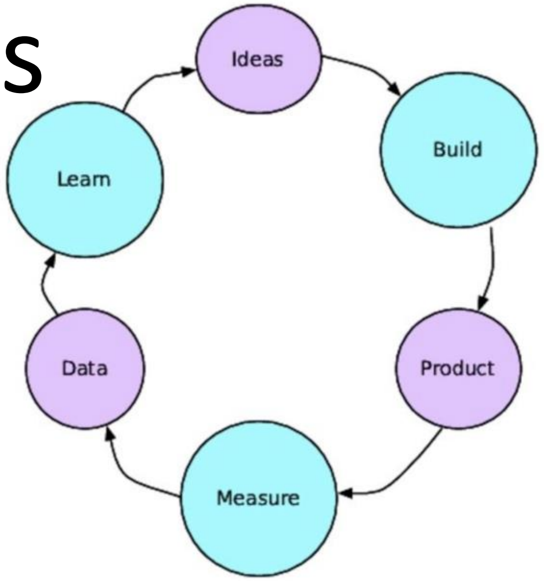
- The more often your team can release internally or externally, the more the team learns about what it takes for the team to release.
- The more often your team practices releasing, the easier releasing becomes.
- The more often the team can release externally, the faster the team will receive customer feedback.

Deliver Value Through the Architecture


GUI
API
App layer 1
App layer 2
App layer 3
App layer 4
Middleware Layer, Component 1
Middleware Layer, Component 2
Middleware Layer, Component 3
Platform (this could be components too)

Deliver value one slice (vertical increment) at a time.
The narrower the slice, the more often the team can release.
Value is realized only after full integration.
"Think about how little you can do to create a walking skeleton of the product, and add more features..."

Recommendations



- Write small user stories and make the value transparent
  - Helps the team see its progress and learning
  - Allows stakeholders see the team's progress and provide feedback frequently
- Consider MVP and MVE to use Build-Measure-Learn loop
- Use a feature parking lot to see possibilities

Idea	Date Added	Value to Us	Why
Engine automation at scale	Jan 12	Might be able to capture the vertical we keep talking about	No one else does this
Cloud-based search	Feb 2	??	Danny, CTO, wants us to do this
Calendar integration	Jun 15	Need to integrate calendar and email at some point	Customers have been requesting this

A Feature Parking Lot Example

References

- [1] Create Your Successful Agile Project, Johanna Rothman, Pragmatic Programmers LLC, 2017. ISBN:9781680502602
- [2] Feature-driven development, https://en.wikipedia.org/wiki/Feature-driven_development