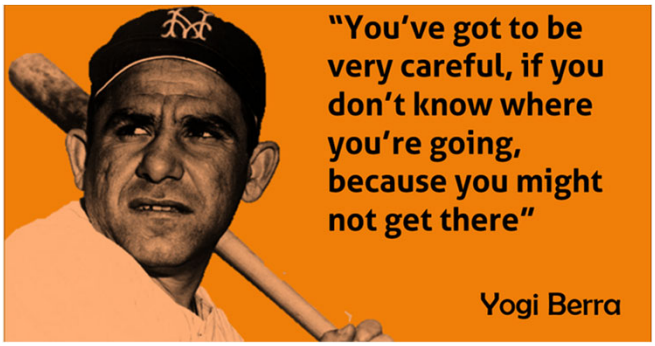


Software Process

CSC 326

1

1

Software Process

"The set of activities and associated results that produce a software product"



What are the activities in software process?

Sommerville, SE, ed. 8

2

2

Example of Process Decisions

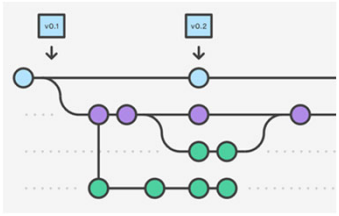
- Whether/Where/How to….



Require approval for all changes to requirements



Review Code



Use Version Control

3

3

Example of Process Decisions

- Whether/Where/How to….
  - Write down all requirements
  - Require approval for all changes to requirements
  - Use version control for all changes
  - Track all reported bugs
  - Review requirements and code
  - Break down development into smaller tasks and schedule and monitor them
  - Planning and conducting quality assurance
  - Have daily status meetings
  - Meet on discord
  - …

4

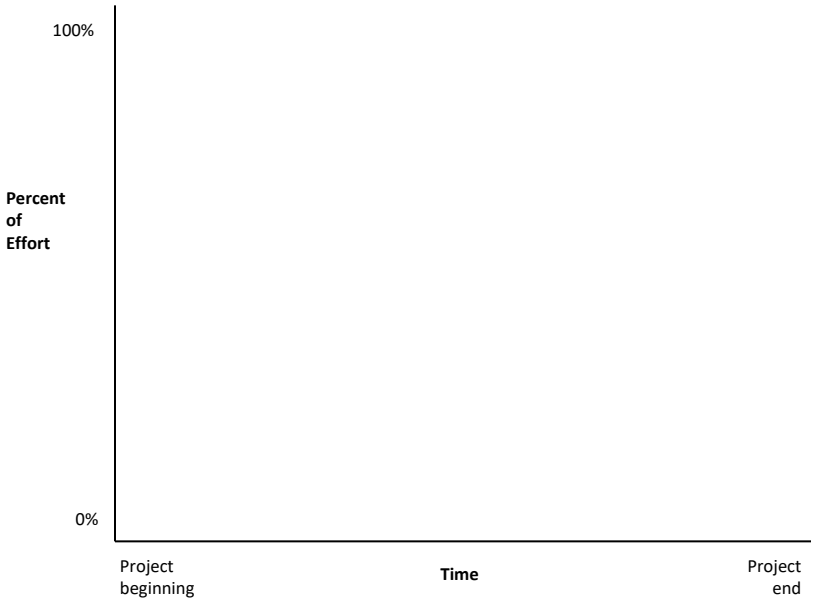
4

Example process issues

- **Change Control:** Mid-project informal agreement to changes suggested by customer or manager. Project scope expands 25-50%
- **Quality Assurance:** Late detection of requirements and design issues. Test-debug-reimplement cycle limits development of new features. Release with known defects.
- **Defect Tracking:** Bug reports collected informally, forgotten
- **System Integration:** Integration of independently developed components at the very end of the project. Interfaces out of sync.
- **Source Code Control:** Accidentally overwritten changes, lost work.
- **Scheduling:** When project is behind, developers are asked weekly for new estimates.

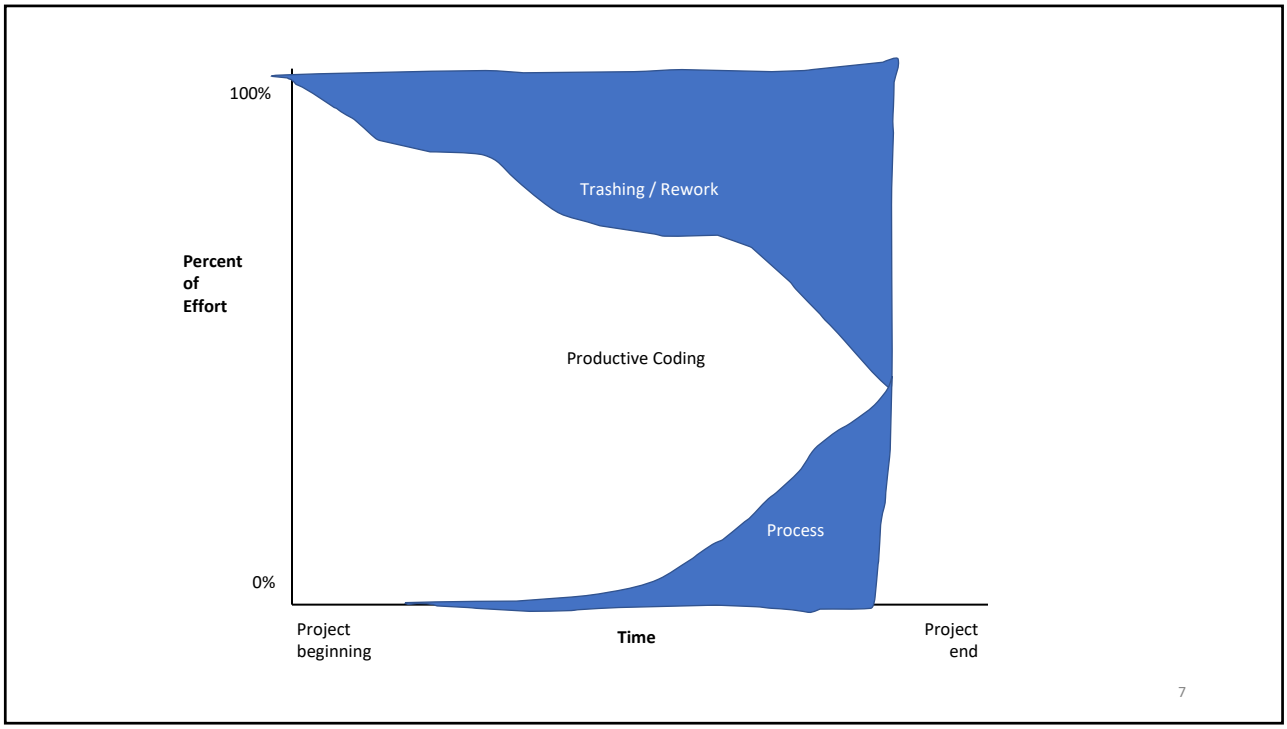
5

5

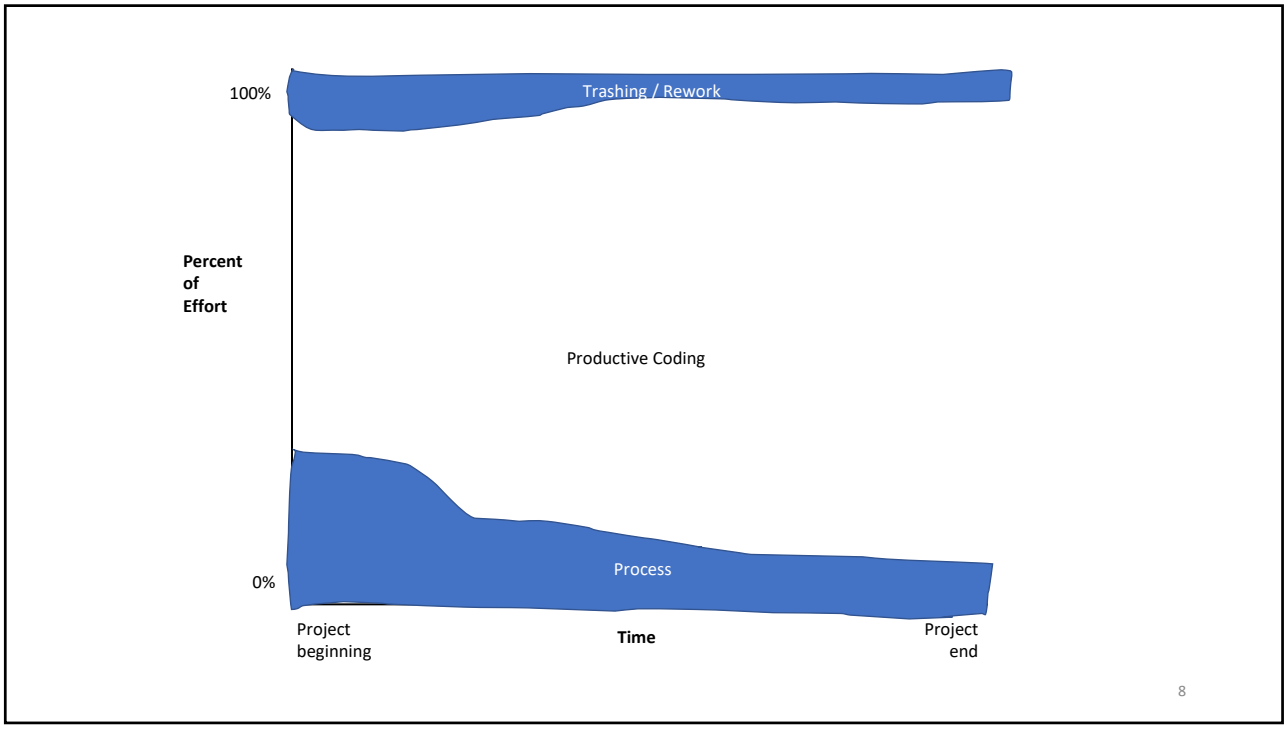


6

6


100%
Trashing / Rework
Percent
of
Effort
Productive Coding
Process
0%
Project
beginning
Time
Project
end
7

7


100%
Trashing / Rework
Percent
of
Effort
Productive Coding
Process
0%
Project
beginning
Time
Project
end
8

8

Hypothesis

- Process increases flexibility and efficiency
- Upfront investment for later greater returns

9

9

Process of Building a House



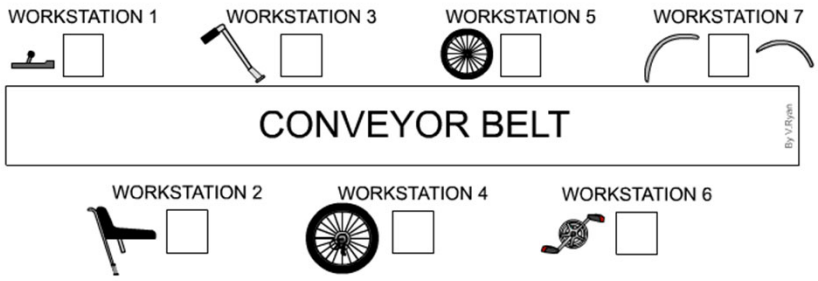
Same Life Cycle



10

10

EXAMPLE BASIC PRODUCTION LINE



11

11

Software Process Models

© NC State Software Engineering Faculty

12

12

Definition of Process

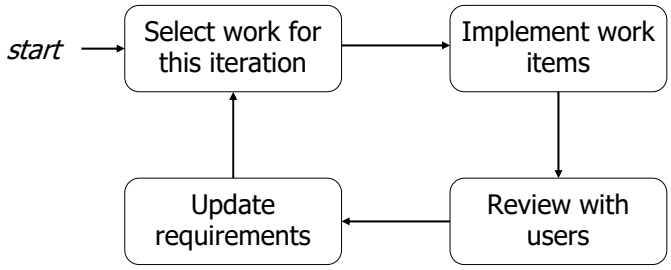
- A process meets the goals of a project by:
  - Orchestrating activities to achieve the goal
  - Defining artifacts to be produced by the activities
  - Determining the roles and skills needed
  - Setting criteria for progressing from an activity
  - Specifying events for planning and tracking

13

13

An Iterative Process

- Typically, diagrams show only the activities (as boxes)
  - This particular process shows what is to be done, not how to do it

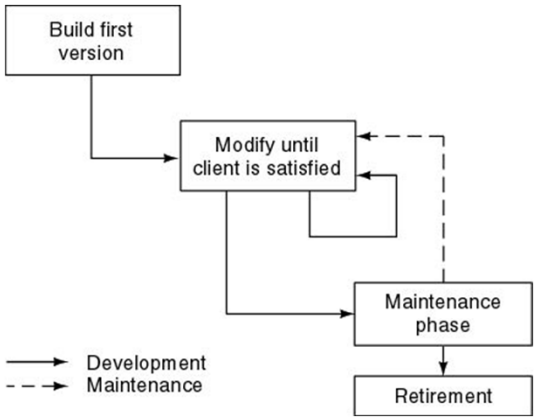


14

14

Build and Fix Model

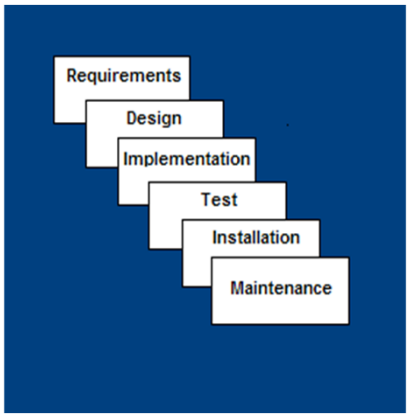
- Problems
  - No specifications
  - No design
- Totally unsatisfactory
- Need life-cycle model
  - "Game plan"
  - Phases
  - Milestones



15

15

Plan-Driven Process Models (Waterfall Model)



- **Requirements** – defines needed information, function, behavior, performance and interfaces.
- **Design** – data structures, software architecture, interface representations, algorithmic details.
- **Implementation** – source code, database, user documentation.
- **Testing** – test completed project.

16

16

When to use the Waterfall Model

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.

17

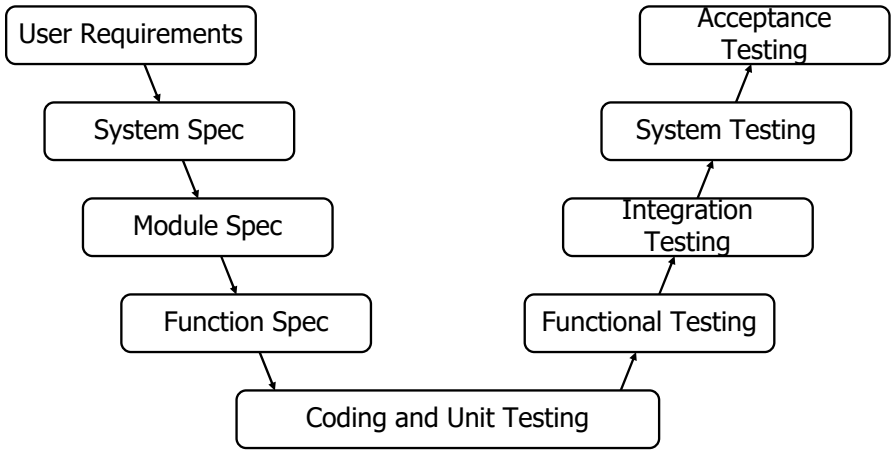
17

V-Model

18

18

V Model: How do it works?



19

19

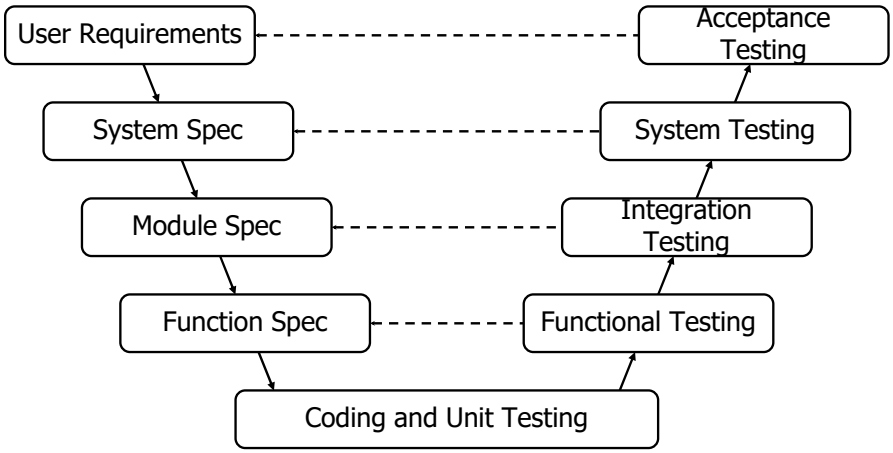
Overview of V Processes

- V processes were introduced in the 1950s
  - The V-shaped diagram and the name came later
- Underlying principles
  - Tests can be specified before coding begins
  - Testing can begin as soon as there is code to test
- V processes are plan-driven

20

20

Paired Specification and Testing Levels

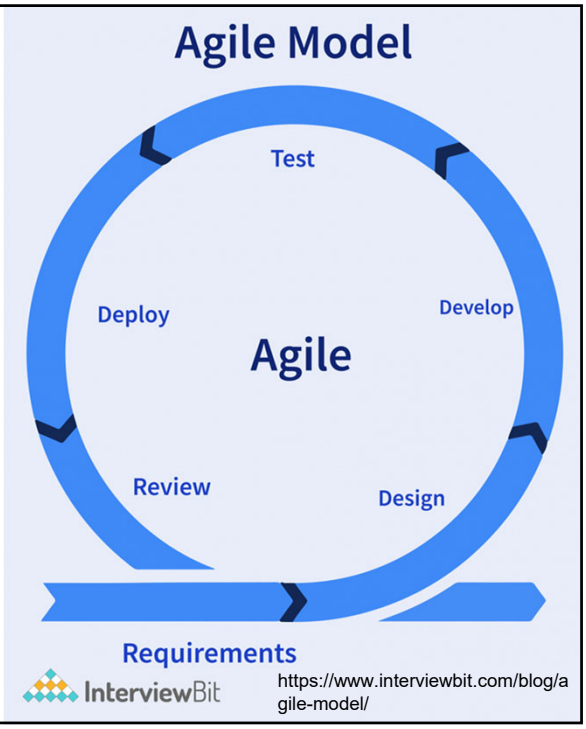


21

21

Agile Process Models

- Anticipation of change
- Planning done in small iterations
  - Gather requirements and plan for small chunks of functionality



https://www.interviewbit.com/blog/agile-model/

© NC State Software Engineering Faculty

22

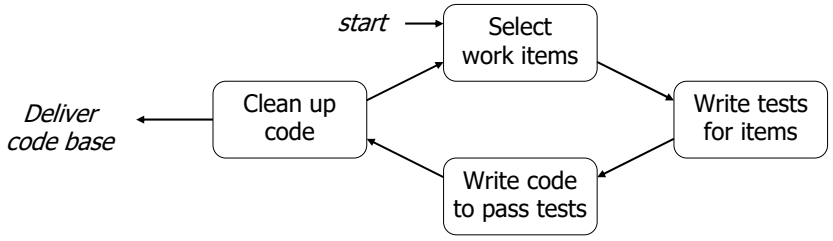
Contrasting Agile and Plan-Driven Development

- Agile
  - Individuals and interactions
  - Working Software
  - Customer Collaboration
  - Responding to Change
- Plan Driven (Plan then Build)
  - Processes and tools
  - Comprehensive documentation
  - Contract negotiations
  - Following a plan

23

23

What Does this process do?



24

24

Terminology, I of II

- Process Model:
  - A class of processes; e.g., Agile, Plan-Driven
- Framework:
  - Partial specification of a process class; e.g., Scrum
- Practice:
  - Specific actionable descriptions of an activity

25

25

Terminology, II of II

- Values:
  - Principles that guide individuals and groups
- Culture:
  - Group norms and values, along with commonly used practices
- Method:
  - General term for process and culture

26

26

Project Risks

27

27

Process Helps Manage Risks

- We use term risk informally for a potential challenge
  - Risk can be thought of as the cost of recovering if something goes wrong
- Process models tend to focus on specific risks
  - Agile methods are iterative, so they manage risk related to requirements
  - Plan-driven methods seek to manage risks related to design

28

28

The Scrum Framework

Scrum is an example of an Agile process model. There are others, like Kanban, XP…. and they often work together

29

Context

- Scrum focuses on teamwork
  - It has rules for iteration planning and progress reviews
- The Framework does not tell developers how to do their work
  - XP focuses on how to do the work
- Recommendation: Use Scrum together with XP
  - Within the planning and review structure of Scrum, use XP for the work

30

30

The Scrum Framework

Reference: "The Definitive Guide to Scrum"
by Jeff Sutherland and Ken Schwaber [2020]

http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

31

31

81% of Agile Projects Use Scrum or a Scrum Hybrid

2021 State of Agile Survey

- 66% use Scrum by itself
  - How developers do their work is not specified
- 15% use a Scrum hybrid with another method
  - 9% ScrumBan (Scrum+Kanban)
  - 6% Scrum+XP

32

32

Structure Provided by Scrum

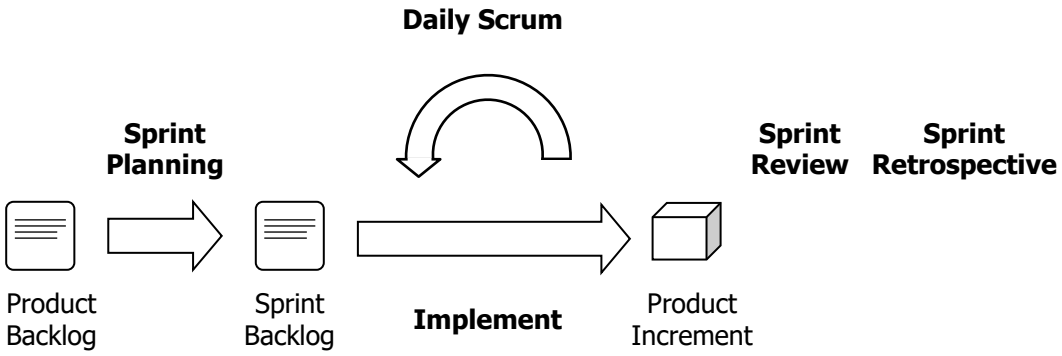
- Development is keyed to iterations, called sprints
- The rules of Scrum specify planning and review events during a sprint
  - Daily Scrum and Sprint Review events help to keep implantation on track
- Sprints and events are ”time boxed”; that is, they are time limited
- Scrum also specifies roles and artifacts

33

33

Overview of a Sprint: Events and Artifacts

Arrows show events, with one exception: Implement is not a Scrum event



34

34

The Team Project

Project Outline/Milestones

Your goal is to use good software engineering practices to create a high quality product. As part of the project, you will plan your tasks; write requirements; create a design for the new parts of the system; create system and acceptance tests for the new functionality; implement and write automated tests to demonstrate high quality; perform inspections on requirements, design, implementation, and tests; explore non-functional requirements and constraints; and present a live demo of your system all while following good process, working collaboratively with your team, and maintaining existing system functionality.

Details about each project milestone are listed below is forthcoming.

- Iteration 0 - Team Forming & Requirements
- Iteration 1
  - Iteration 1A - Design & System Test
  - Iteration 1B - Implementation & Test
- Iteration 2
  - Iteration 2A - Design & System Test
  - Iteration 2B - Implementation & Test
- Final Project
  - Team Demo
- Reflection
- Team Member Peer Evaluation

The high-level iterations are like sprints. At the end of each iteration, working software with a new feature is delivered.

35

35

The Scrum Team: Roles

Product, The Developers, Scrum Master

Remember…. scrum is about teamwork!

36

36

Roles: The Product Owner

- The **product owner** represents the voice of the customer
  - Manages the product backlog of work items (requirements)
  - Sets the goal for what to accomplish during a sprint
- The **product owner is an individual**, not a team
  - In other contexts, the corresponding role is product manager

37

37

Roles: The Developers

- Developers have the flexibility to choose how they implement work items
  - Review events hold them accountable
- Developers deliver potentially shippable product “increments” per sprint

38

38

Roles: The Scrum Master

- Scrum Master is a coach (servant-leader)
  - Organizes and moderates all events
  - Respectfully keeps events on track
- Additional responsibility during Daily Scrums
  - Takes responsibility for removing external impediments
- The Scrum Master is an individual, not a team

39

39

Events

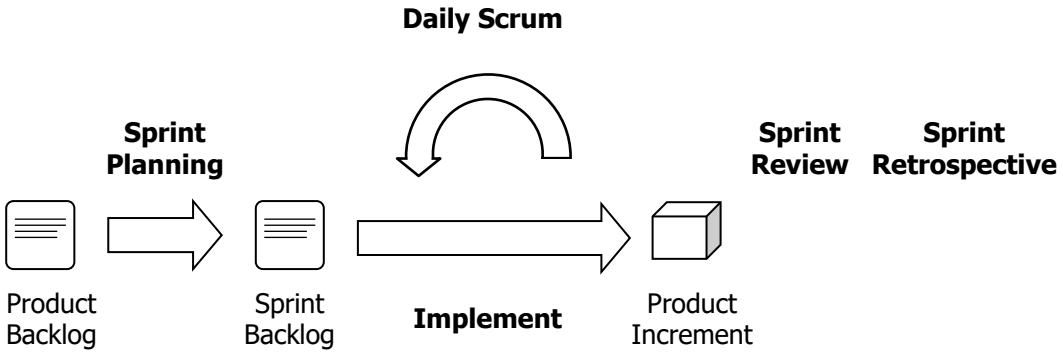
Sprint Planning, Daily Scrums, Sprint Review, Sprint Retrospective

40

40

How are requirements handled?

- Specifically, where is the product backlog updated?

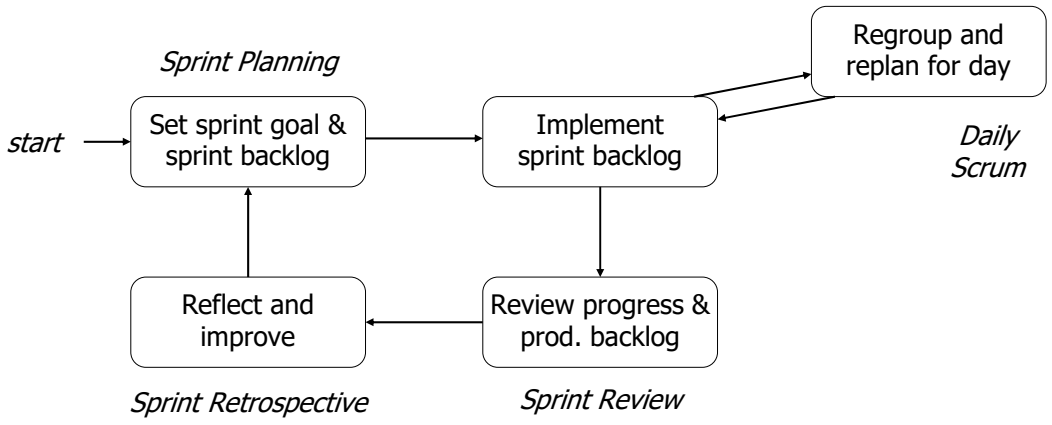


41

41

Activities Associated with Scrum Events

Activities are represented by boxes; events appear as labels next to activities



42

42

Events: Sprint Planning

At most 8 hours for a one-month sprint

- Attendees: The Scrum Team and users
- Led by product owner, set a Sprint Goal, which then cannot be changed
- Select a Sprint Backlog of work items to meet the goal
  - Users define the value/priority of backlog items
  - Developers estimate effort for work items and what will fit in a sprint
  - Selection of work items is based on a cost-benefit analysis of the items

43

43

Events: Daily Scrum

15-minute review to keep the project on track

- Attendees: Developers, with Scrum Master as moderator
- Each developer answers 3 questions:
  - What did I do yesterday?
  - What will I do today?
  - Any obstacles? (a.k.a. Blockers)
- The Scrum Master takes responsibility for removing any external obstacles

44

44

17.01 Activity: Daily Scrum

Pretend you're all developers.

Have a daily scrum about CSC 326 (we will take 5 minutes). If you didn't work on CSC 326 yesterday, answer for the most recent day you did work on CSC 326.

- What did I do yesterday?
- What will I do today?
- Any obstacles? (a.k.a. Blockers)

45

45

Example: Removing External Obstacle

- Obstacle: Developers find a bug in a video subsystem from a supplier
  - The bug could delay the project
- The Scrum Master takes responsibility for removing the obstacle
  - The Scrum master works with the purchasing manager
  - Together, they contact the supplier
  - The contract is large, so the supplier expedites a fix

46

46

Events: Sprint Review

At most 4 hours for a one-month sprint

- Attendees: Scrum Team and users
- Review product increment for this sprint
- Update Product Backlog based on user feedback
- **Note: this is where the Product Backlog (Requirements) are updated**

47

47

Events: Sprint Retrospective

At most 3 hours for a one-month sprint

- Attendees: Scrum Team
  - **Plan process improvements** for the next Sprint
  - This event is *not about the product*

48

48

Artifacts

Product Backlog, Sprint Backlog, Product Increment

Scrum Artifacts

For transparency of information, artifacts are open to all team members

- Product Backlog:
  - Ordered list of all work items that might be needed for a product
- Sprint Backlog:
  - Items selected for meeting the Sprint Goal that is set during planning
- Product Increment:
  - Sum of all Product Backlog items that are completed during a Sprint

Frequently Asked Questions

about SCRUM

51

51

What if the Sprint Backlog is Too Optimistic?

- Situation:
  - Developers overestimate what they can accomplish during a sprint
  - That is, they realize that they cannot implement the full Sprint Backlog
- Response:
  - Developers can renegotiate the Sprint Backlog with the Product Owner

52

52

How Does Scrum Scale?

- Situation:
  - The development team is too large for a 15-minute Daily Scrum
  - With more than 7 developers, 15 minutes means less than 2 per person
- Response:
  - On approach is to split the team into subteams
  - Each subteam has its own Scrum Masters
  - The Scrum Masters from the subteams then have a Scrum of Scrums

What if Some Features Don't Fit?

- Situation:
  - The entire Product Backlog cannot be done during the project schedule
- Response:
  - With any iterative project, the highest priority features during a sprint
  - Inessential features that don't fit within the schedule can be dropped
  - If essential features don't fit, renegotiate with users as early as possible
  - Avoid surprises by involving users in planning, from the start

Extreme Programming (XP)

55

Agile Development with XP

56

Extreme Programming (XP)

- Kent Beck defined XP in the mid-1990s, while on a project for Chrysler
- The name comes from taking common-sense practices to the extreme:
  - Example: "If design is good, we'll make it part of everybody's daily business (refactoring)."
- XP was refined in the years following the Chrysler project
  - There are now XP values, principles, activities, and practices

57

57

Overview of XP

- Beck describes XP as follows:
  - "an always deployable system to which features, chosen by the customer, are added and automatically tested on a fixed heartbeat."
- The "heartbeat" or iteration length is often a week or two

58

58

The Four Main XP Activities

- Listening
- Testing
- Coding
- Design

59

59

Advice Tied to the Four Main Activities

Stay true to the spirit of the values and advice

- Listen: To what users want from a system
- Testing: Make it central to development
- Coding: Write simple readable code
- Design: Refactor to clean up the design and implementation

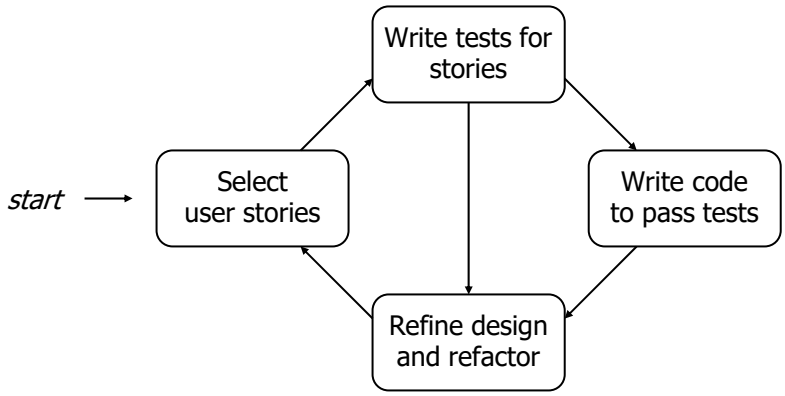
These activities come together in the XP development cycle (next slide)

60

60

The Test-Driven XP Development Cycle

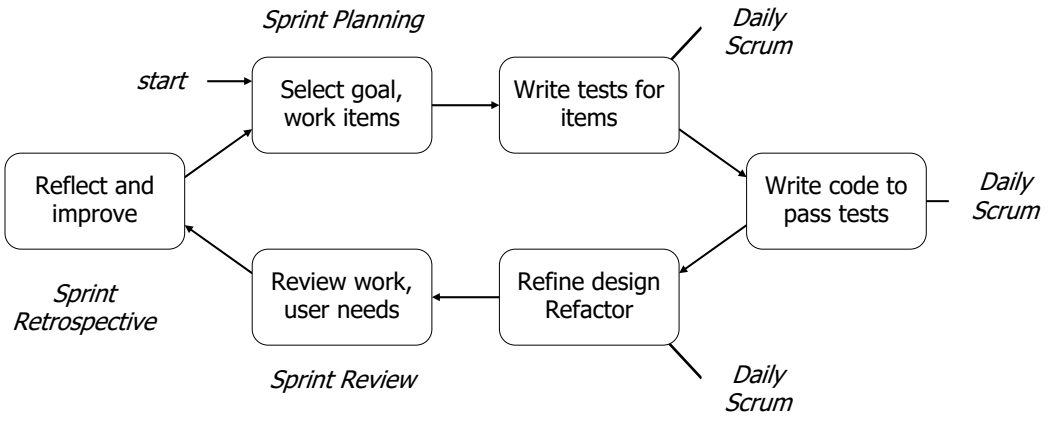
User stories express requirements, as we'll see



61

61

A Scrum/XP Hybrid Process



62

62

Listening: Writing User Stories (Requirements)

User Stories are for Snippets of Functionality

Write from the user's perspective

- User Story Template
- As a <role> I want to <action> so that <benefit>
- Example:
  - As a customer,
  - I want to withdraw cash from an ATM
  - so that I can get cash when the bank is closed

We did this in a previous lecture!

With XP, *Testing* Plays a Central Role

65

65

Testing Supports All Activities

- Three forms of testing
  - Acceptance tests for user stories
  - Unit tests for the code
  - Regression tests for guarding against breaking existing functionality
- Testing after every change increases confidence in the code
  - Aids debugging: either the change was faulty or it broke something
  - Such testing can actually shorten overall development time

66

66

Validation and Verification

- Validation is for functionality and requirements
  - The focus is on ”Am I building the right product?”
- Verification is for implementation correctness
  - The focus is on “Am I building the product right?”
- During test-driven development
  - Before coding, write acceptance tests for validation
  - After coding, write additional unit tests for verification

67

67

Acceptance-Test Template for User Stories

- In effect, acceptance tests define what a product does
- Simple form of the template:
  - Given <some context> when <triggering event> then <expect outcome>
- There may be multiple acceptance tests for a user story
- Write acceptance tests so they are executable

Previously we focused on a checklist-style **acceptance criteria**.
This is about **acceptance tests** associated with the user stories!

68

68

General Form of the Acceptance Template

Given <some initial context>
and <some more context>
when <triggering event>
then <expect outcome>
and <another outcome>

- There may be multiple and-clauses for context and outcomes

69

69

Example: Case 1 for Cash Withdrawal

Account is in Credit

Given the account is in credit
and the card is valid
and the dispenser has cash
when the customer requests cash
then update account balance
and dispense cash
and return card

70

70

Example: Case 2

Account is Overdrawn Past the Credit Limit

Given the account is overdrawn
and the card is valid
when the customer requests cash
then display "sorry" message
and return card

71

17.02 Activity: Acceptance Test

User Story for UC4:

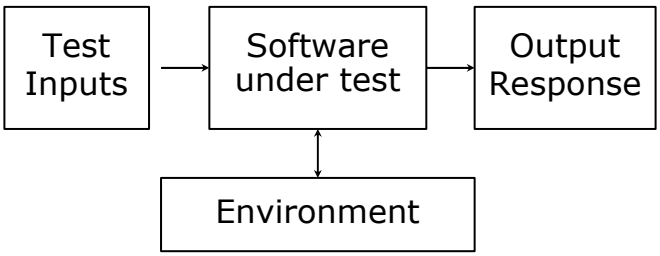
As a coffee machine user,
I want the ability to edit a recipe's cost and ingredients
so that I can make changes over time if needed.

Write an Acceptance Test for this.

72

Relate the Acceptance Template to the diagram

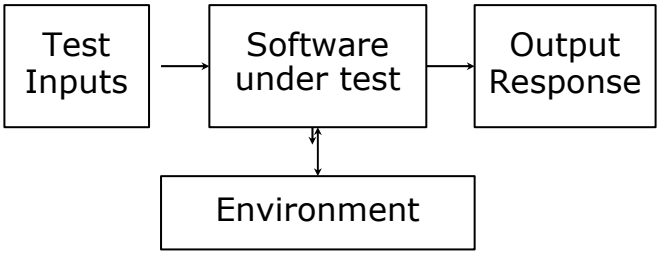
- How does Given … when … then … relate to the elements of testing in:



73

73

Relating the Acceptance Template to Testing



- The “Given” context sets up the environment
- The “when” event corresponds to an input
- The “then” response corresponds to the expected output

74

74

Coding: Write Simple Readable Code

XP Emphasizes Simplicity

- Write code that fits the purpose for which it is intended, nothing more
- Use the simplest solution that could possibly work
- Keep code clean and readable at all times
- Avoid frills and functionality that may end up never being needed

Practice: Pair Programming

- Pair programming is two people on the same task at the same screen
  - The proposed benefit is that code is reviewed all the time
  - Also, if someone leaves, there is a person who knows the code

77

77

Practice: Refactoring: Code Clean Up and Restructuring

Refactoring consists of a sequence of correctness-preserving changes

- Restore modularity, so each module has a specific responsibility
- Put common (duplicated) functionality into a single method or module
- Improve control flow for readability and predictability of behavior
- Ensure that variable and method names reflect their purpose
- Add comments for rationale and what not to change

78

78

Design: When and How Much?

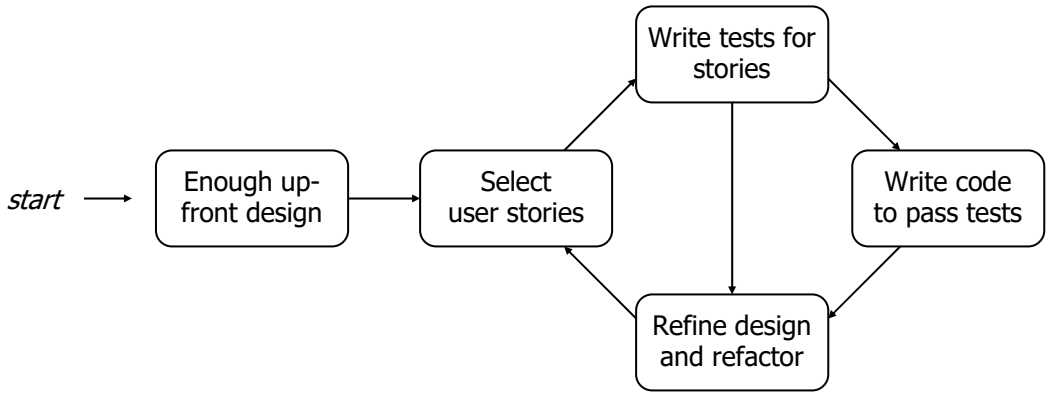
When to Design?

The answer varies from project to project

- Too much design too soon
  - Can lead to wasted effort on something that will not be needed
- Too little design too late
  - Can result in an unwieldy design that needs rework
- How do we avoid either wasted effort or poor design?

Modified XP Cycle with Some Up-Front Design

Note that the XP cycle allows an iteration to bypass coding



81

81

When to Design Depends on Product and Requirements

Specifically, on product complexity and requirements stability

PRODUCT \ REQUIREMENTS	*Stable*	*Changing*
Complex	Design properties into the system up front	Start with a modifiable modular design
Simple	Minimal initial design to set direction	Defer major decisions until needed

82

82