Week 4 - Monday

COMP 3100

Last time

- What did we talk about last time?
- Software processes
- Waterfall model
- Prototyping
- Risk management and the spiral model

Questions?

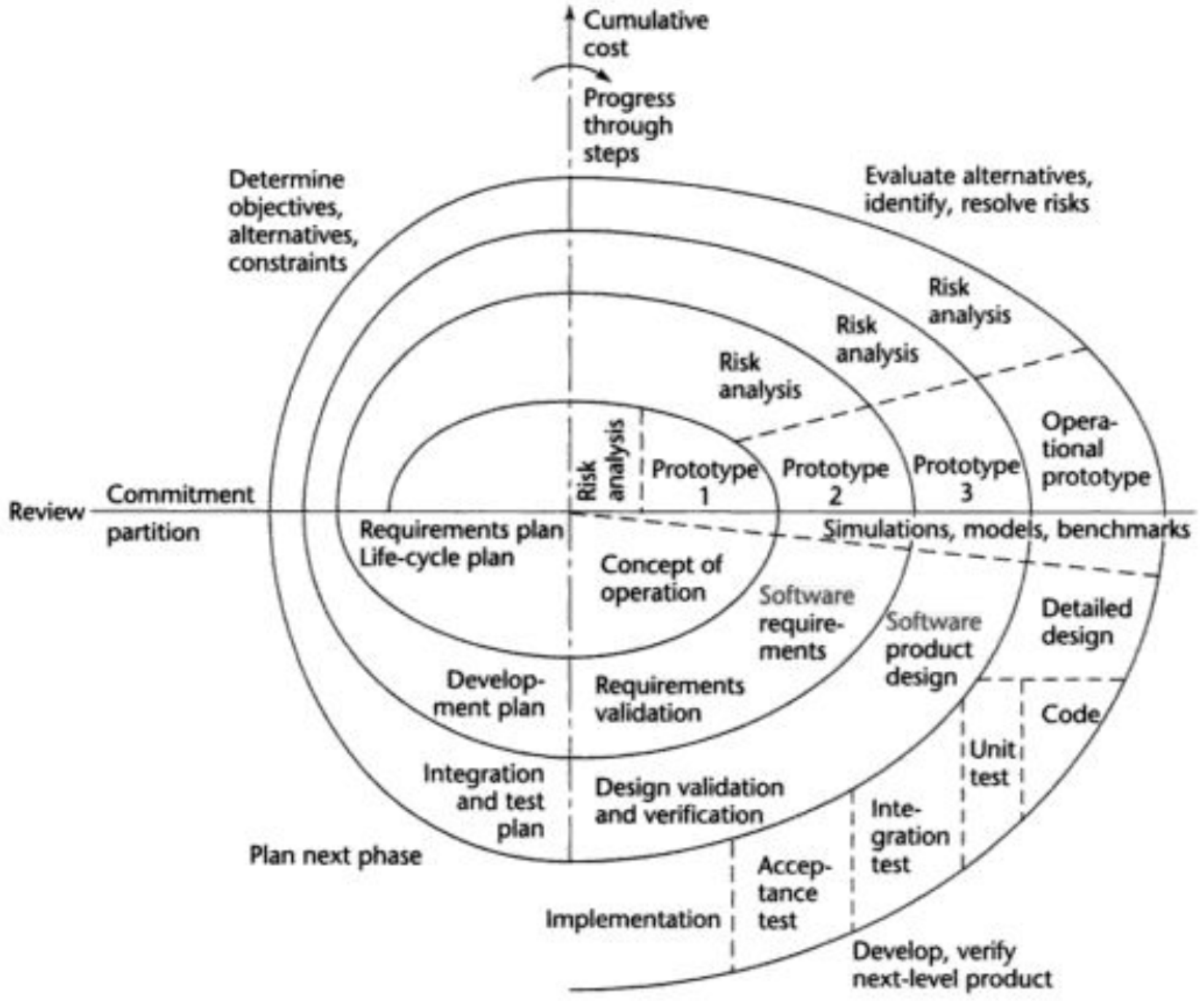
Iterative and Incremental Processes

Iterative and Incremental Processes

- An **iterative process** contains repeated tasks
  - Example: While debugging code, you might run tests, do fixes, run tests, do fixes, and so on
- An **incremental process** produces output in parts
- Processes can be either iterative or incremental, both iterative and incremental, or neither
- The purest version of waterfall is *neither*
  - It's not iterative because each phase is separate and not repeated
  - It's not incremental because a working product is only available at the end

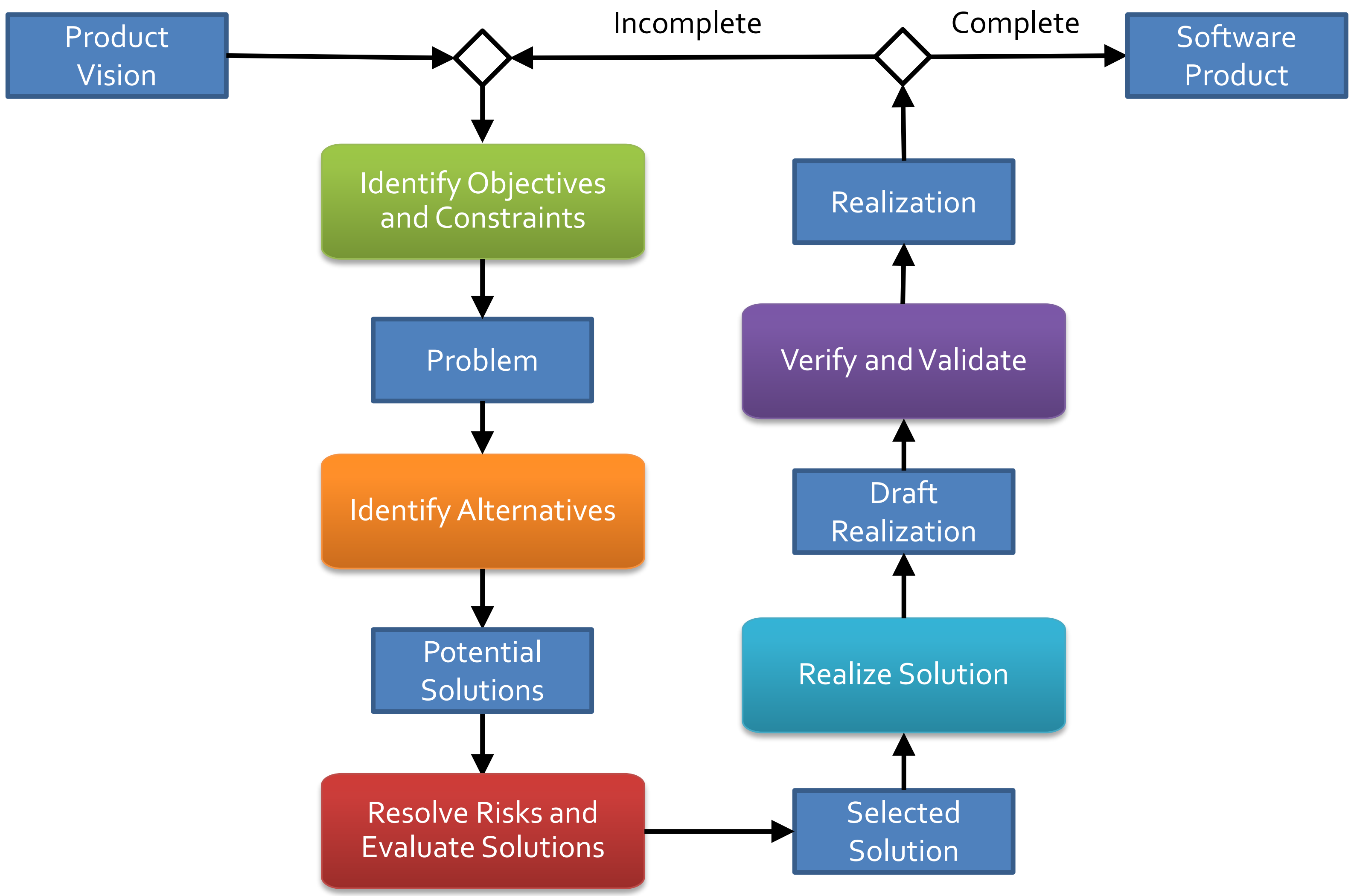
Review: spiral model

- The spiral model is built around risk management
- Multiple cycles are used
- Each cycle starts by looking at goals
- Then evaluate different approaches to the goals in terms of risk
- The model on the right shows how the spiral model can be applied to waterfall



Spiral model viewed as an iterative process

- Other than its focus on risk, people also talk about the spiral model because it's an early example of an iterative process



Iterative processes

- Iteration is the main way you get quality
  - It's just so hard to get it right the first time!
  - Software development still involves significant trial and error
- Even the waterfall model usually has iterative steps in practice
- Prototype evolution is iterative
- The spiral process is iterative
- The problem with iteration is **rework**
  - Redoing or throwing out previous work

Incremental processes

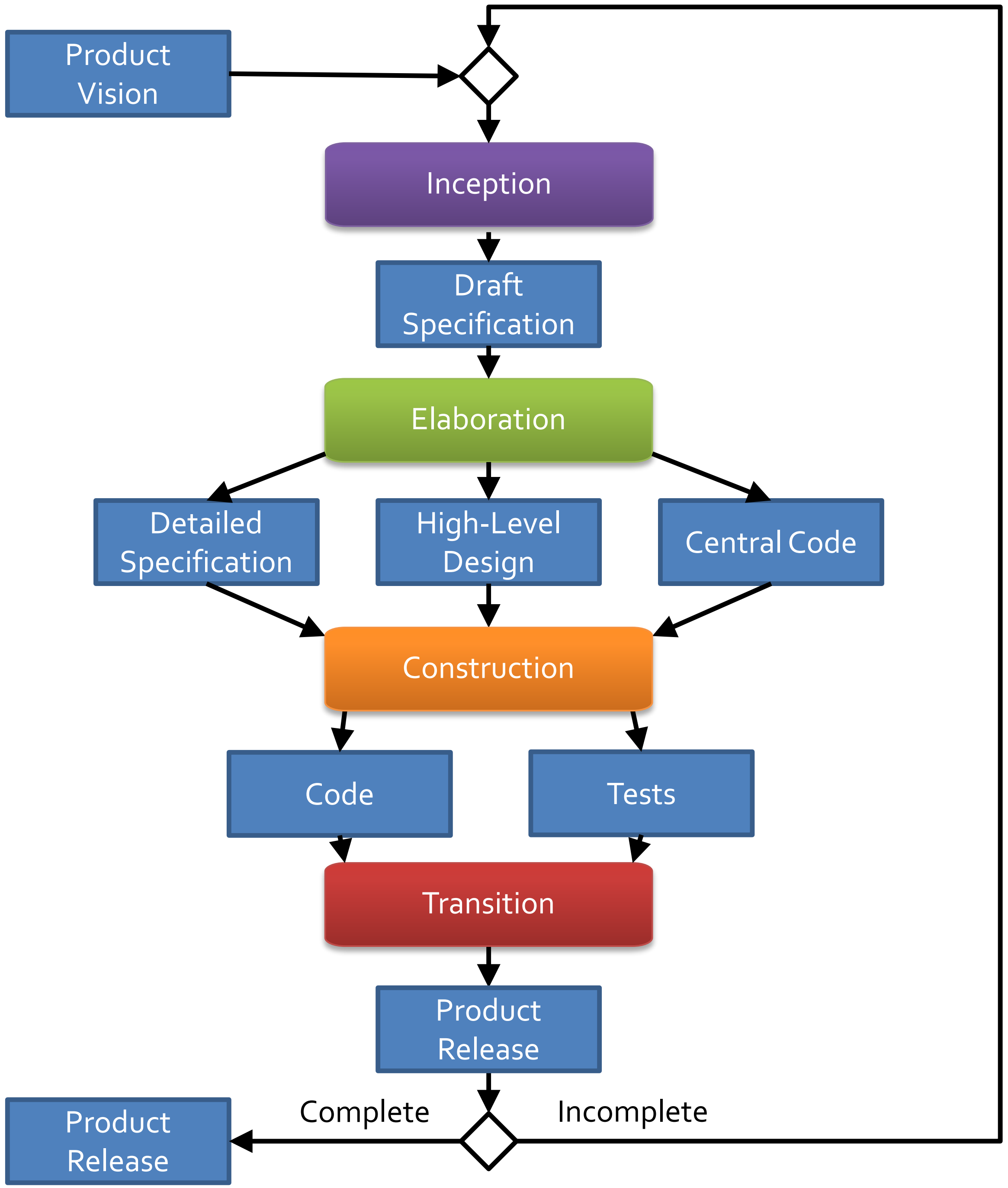
- Iteration is found lurking everywhere to greater or lesser degrees, but being incremental is more binary
- To be incremental, final products must be produced along the way
- Waterfall is *not* incremental because the products produced along the way are just used for the next step

Rational Unified Process

- The **Rational Unified Process (RUP)** is a process that is both iterative and incremental
- Pure RUP is now rarely used, but it was a step in the evolution of modern agile methods
- RUP creates products in increments called **releases** made during a **cycle**
  - Each release is a working product
- Each cycle has four phases: inception, elaboration, construction, and transition
- Each phase has iterations divided into five workflows: requirements, analysis, design, implementation, and test

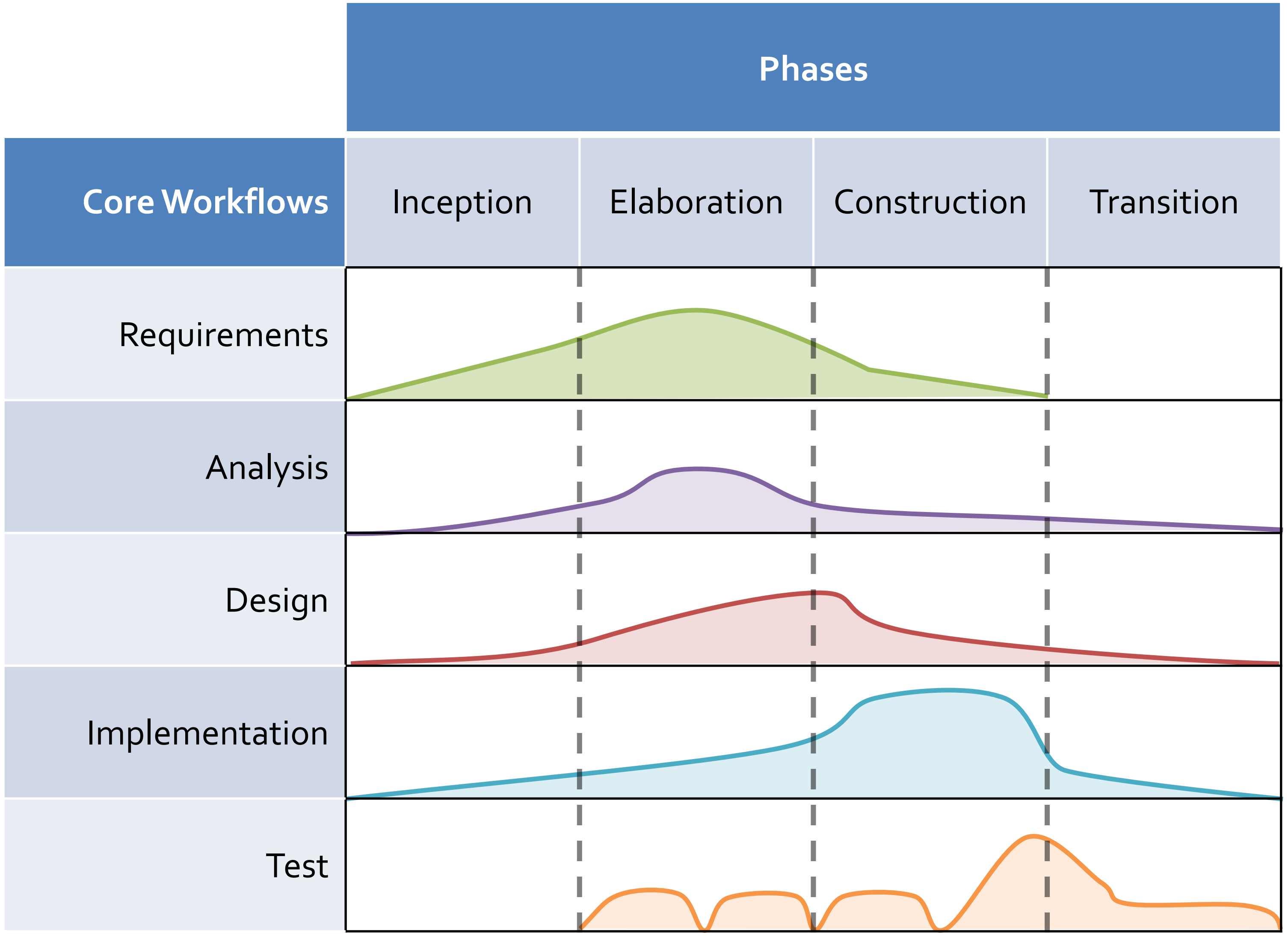
RUP phases

- **Inception:** Product vision is developed
- **Elaboration:** Specifications become more detailed as core code is written
- **Construction:** Majority of code is written and tested, refining designs as needed
- **Transition:** Customers test the product



Workflows in the RUP

The core workflows of requirements, analysis, design, implementation, and test are done throughout the phases, but some phases have more of one workflow than others



Advantages of RUP

- A lot of work has been done with RUP, and it has templates, checklists, and other tools
- It can be adapted to projects with different sizes
- It has benefits of all incremental processes
  - Usable product early on
  - Total failure less likely
- Its iterative nature makes it easier to schedule work, improve existing product, and incorporate risk management

Disadvantages of RUP

- It's complex, and you have to understand it well
- It has a lot of documentation and management
  - Heavyweight process
- Development cycles are long, making it hard to change requirements

Agile

- Versions of waterfall were the only commonly used software development model until the 1990s
- A lot of people were unhappy with it
- In response, some developers created the Agile Manifesto, a statement about developing software that was diametrically opposed to waterfall
- The ideas caught on, and many developers embraced the idea, creating a series of different methods
- Sometimes businesses claimed to be changing over to agile methods but really just renamed parts of their waterfall approach

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 interactions** over processes and tools*
- ***Working software** over comprehensive documentation*
- ***Customer collaboration** over contract negotiation*
- ***Responding to change** over following a plan*

That is, while there is value in the items on the right, we value the items on the left more.

Agile principles

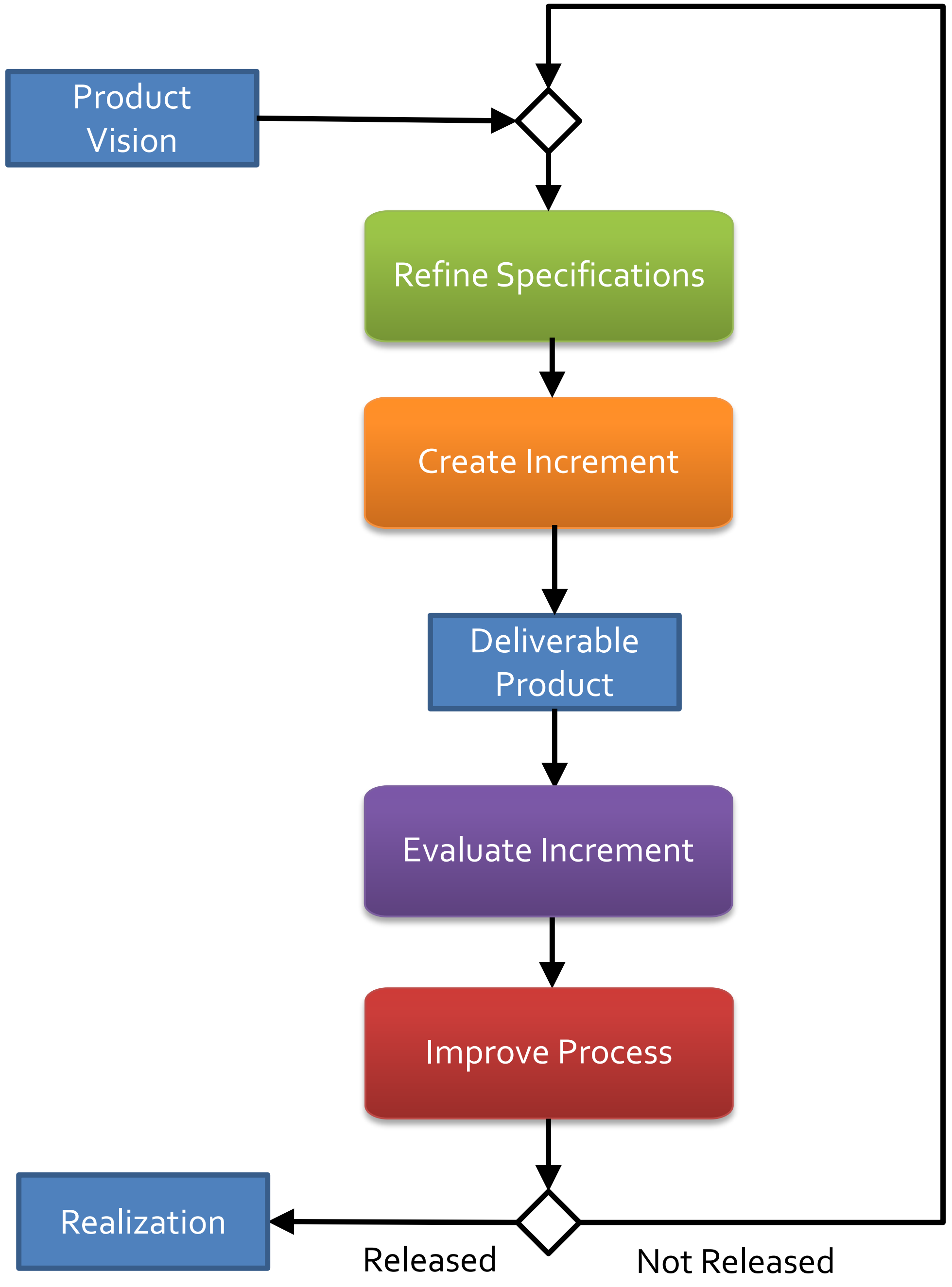
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile characteristics

- The ideas caught on, spawning specific methods such as Extreme Programming, the Crystal Method, Dynamic System Development Method, and Scrum
- These methods *all* have the following characteristics:
  - Incremental process with increments ranging from a week to a few months
  - Customers are closely and continuously involved in the product
  - Lightweight process minimizing documentation and management tasks
  - Test driven, using automated test suites to avoid the problems of frequent code change

Agile lifecycle

- Agile processes are similar, following a lifecycle much like the one on the right



Agile advantages

- Product specifications can change without destroying all the work that's been done
- Customers get a software product quickly
  - With new versions coming frequently
- Bad projects can be canceled early
- Time is saved because of lightweight requirements for documentation and management
- Duplication of effort is usually reduced

Agile disadvantages

- Customers have to be involved constantly, but most customers don't want to spend their time giving feedback
- Continuous refinement of a product can lead to a bad design through an evolution of ideas that seemed like good ideas at the time
- For large projects, it's hard to coordinate many teams on a product that's evolving unpredictably without documentation
- It's hard to predict the outcomes of agile methods

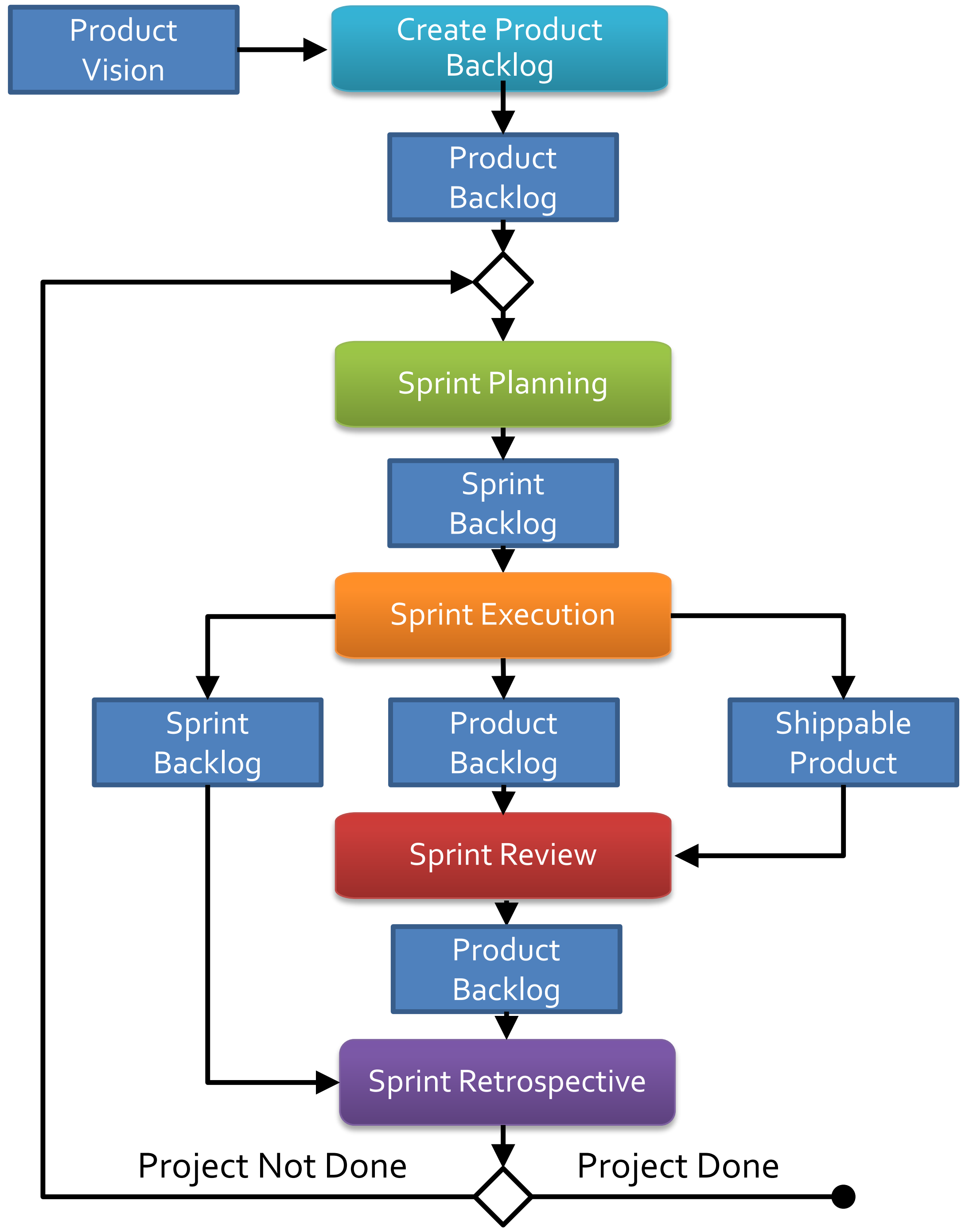
Scrum

Scrum

- Agile methods are popular
- It's hard to get reliable data on how popular because private companies do not have to disclose what they do
- Some agile sites say that over 2/3 of all development is agile or "leaning agile"
  - But agile people are always going to overrepresent agile
- Scrum is one of the most popular agile methods
  - Kanban is another
- Scrum is a process framework that can be adapted to different settings

Scrum process

- Like other workflows, Scrum can be modeled with an activity diagram showing familiar steps
- Everything is built around a cycle called a **sprint**
- Because sprints repeat, the process is iterative
- Because each sprint produces a shippable product, the process is incremental



Sprints

- Recall that agile methods are built around a product backlog, containing high-level descriptions of the desired features of the product
  - Items can be added to or removed from the product backlog at any time
- Some of the product backlog is chosen for a sprint
  - Making the **sprint backlog**
- The sprint backlog is implemented, making a new shippable product
- A **sprint review** allows customers to give feedback on the product
- The **sprint retrospective** is used to figure out how to do the next sprint better

Scrum roles

- **Product owner (PO)**
  - Responsible for what's in the product
  - Customer representative to the other developers
  - Updates the product backlog
- **Scrum master (SM)**
  - Guides the team through the Scrum process
  - Facilitator and coach
  - Protects the team from outside interference
- **Team members**
  - People who decide how to build the project and build it
  - Typically, everyone works on everything

Scrum artifacts

- **Product backlog**
  - A prioritized list of product features that haven't been implemented yet
  - **Product backlog items (PBIs)** are the elements of this list
  - Priorities are based on business value
- **Sprint backlog**
  - Subset of PBIs
  - Tasks needed to complete them
  - Estimates of effort needed for each one
- **Potentially shippable increment (PSI)**
  - Product that could be shipped to the customer (though maybe without all the desired features)
  - A PBI on the sprint backlog that wasn't finished goes back into the product backlog

Scrum activities

- **Product backlog creation**
  - The PO creates the product backlog for the first time, using customer input
- **Product backlog refinement**
  - The PO constantly adds and deletes PBIs from the product backlog based on feedback from stakeholders
- **Sprint planning**
  - The PO, SM, and other team members select PBIs, maybe with a particular sprint goal
  - PBIs are chosen by priority, taking into account how much can be done by estimating the work for the tasks for a PBI
- **Sprint execution**
  - Everyone performs the tasks to implement the sprint backlog PBIs
- **Sprint review**
  - A product demo where stakeholders discuss what was added and how they feel about it
  - Goal: improving the product
- **Sprint retrospective**
  - The team discusses what went well, what didn't, and how the next sprint can be better
  - Goal: improving the process

Managing the product backlog

- The product backlog is a prioritized list of PBIs
- Each PBI consists of
  - Specification
  - Priority
  - Estimate of effort
  - Acceptance criteria

PBI specifications

- PBI specifications can be less formal and more general than requirements in waterfall
- They could be traditional requirements statements, UI diagrams, use cases, user stories, bugs, design tasks, research tasks, etc.
- They start at broad levels of abstraction and are refined over time
- PBIs are refined into detailed, sprintable PBIs as needed, based on priorities
- Product backlogs should contain enough refined PBIs for two or three sprints

User stories

- We mentioned **user stories** in the discussion about requirements
- User stories are the most popular way of specifying features in Scrum
- User story format:
  - As a <*user role*> I want to <*goal*> so that <*benefit*>.

- Examples:
  - As a course scheduler I want to determine whether students can take other sections of a course so that I can see if I can cancel a section with students already enrolled in it.
  - As a shopper I want to see whether an item is still on sale so that I can buy it more cheaply.
  - As an internet user I want to secure my devices so that I can protect my private information.
  - As an electric utility customer I want to see my usage over several years so that I can analyze it to budget my electricity costs more exactly.

Upcoming

Next time…

- Finish Scrum
- Review

Reminders

- Read Chapter 3: Scrum for Wednesday
- Use my feedback to improve your projects
  - **Project 1 final due Friday before midnight!**
- **Exam 1 next Monday**