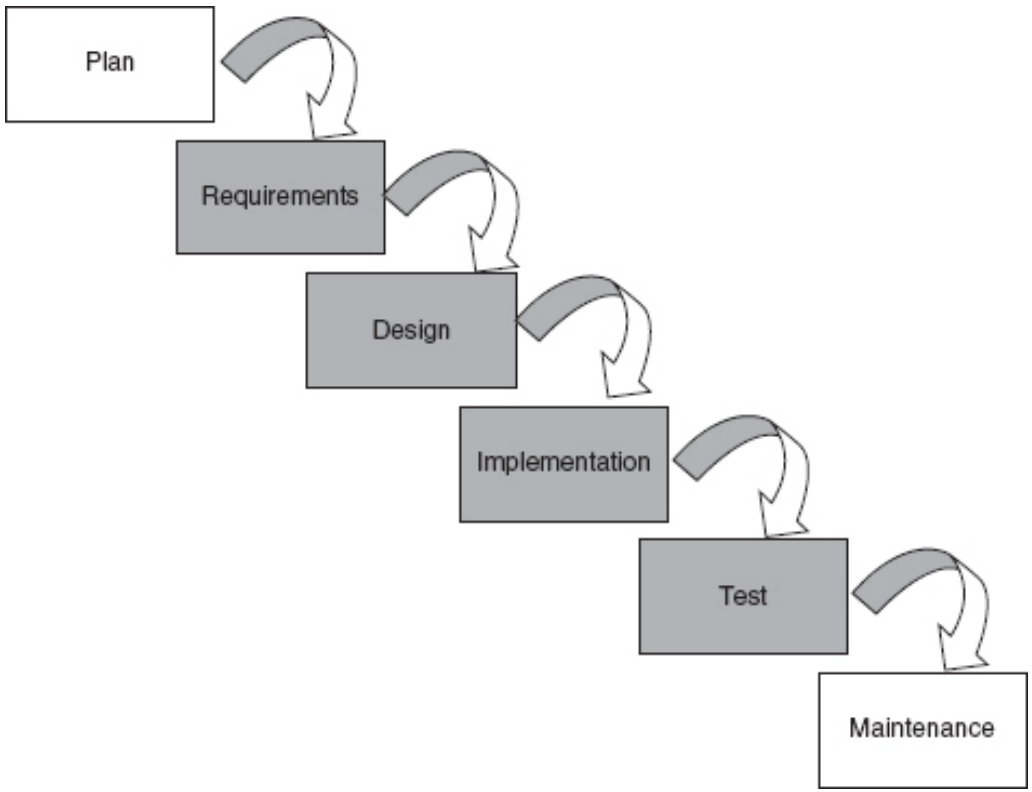
Objectives for today

- Software processes; Waterfall
- Requirements
- Use cases

Software Lifecycle Process Model

- Booch called his method a "Partial Lifecycle" model
- What would be a "full" lifecycle model
- The process model is an abstract representation of the tasks required to achieve the goals of the project
- The SPM defines
  - Which tasks are to be performed
  - Input and output of each task
  - How the tasks should sequence and flow
- Individual tasks in the process model are called *activities*
- Typical activities: Planning, Design, Implementation, Testing, Maintenance

Waterfall Model



- One of the earliest published *software processes*[22]
- Development occurs in distinct phases, with the output of one activity serving as the basis for the next

[22] Winston W Royce. "Managing the development of large software systems". In: *proceedings of IEEE WESCON*. vol. 26. 8. Los Angeles. 1970, pp. 328–338.

Waterfall Discussion

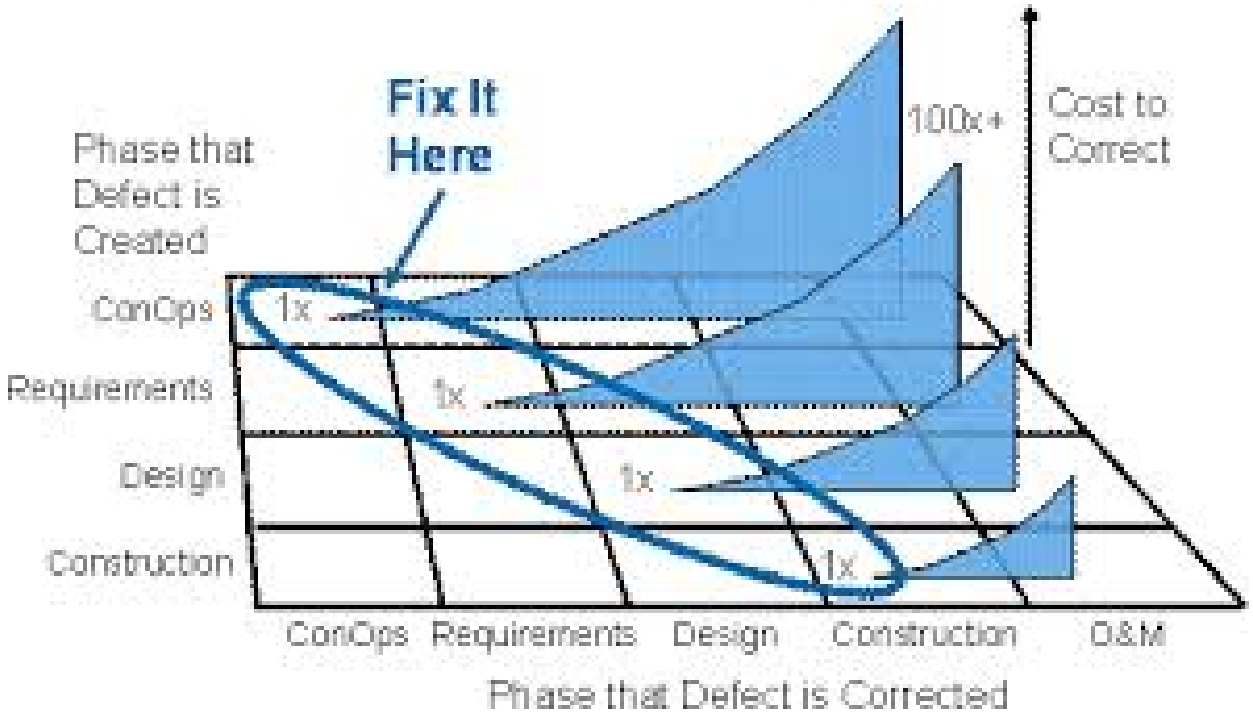
- Waterfall is also called the *document-driven* approach due to the many documents that must be produced while executing it
- Activities occur in a strict linear sequence
- Early electronic computers could occupy an entire room
  - Even simple programs required extreme amounts of labor
  - Not conducive to iterating, making changes
- "Big Design Up Front" process
- Different specialists work in different "silos"
- Formal plan and timeline can be disrupted if one phase runs beyond schedule

Waterfall Thinking

- What do people really mean by *waterfall*?
- It is a kind of rigid thinking that says requirements, scope, design, can all be known in advance.
- Evidence shows that "...poor practice for most software projects, rather than a skillful approach...associated with high rates of failure"[23]

[23] Craig Larman. *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Interative Development*. Prentice Hall PTR, 2012.

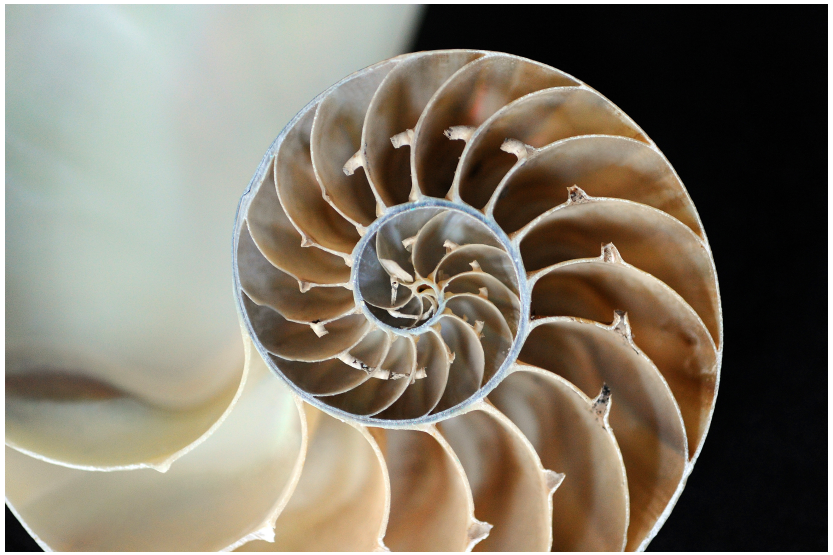
Cost of Fixing a Bug



- The total cost to correct a bug (time spent, monetary, lost business, etc.) increases the later the bug is detected (image credit[24])
- A process that delivers working software early can help to mitigate the effects of *defect cost increase*

[24] `https://ops.fhwa.dot.gov/publications/seitsguide/section3.htm`

Iterative and Incremental Development[25]



- Two important notions in software development are *increments* and *iterations*
- *Iteration*: Is one pass through the waterfall process
- *Incremental*: Delivering additional features in small units
- *Evolutionary Model*: Iterative model where the software gradually evolves over time, allows for changes to requirements and design based on customer feedback

[25] C. Larman and V. R. Basili. "Iterative and incremental developments. a brief history". In: *Computer* 36.6 (2003), pp. 47–56.

Process Iteration

- Changing requirements are a basic fact of developing systems
- *Incremental Delivery*: software delivered in increments, each new increment adds a small piece of the required functionality
- Most important requirements are delivered in early increments
- Requirements are frozen once work has started on a given increment, this is called a *timebox*
  - Typical timebox might be 2-3 weeks
- A fundamental principle of *agile* development is that working software should be delivered as early as possible

Managing Chaos

- According to Brechner[5], project management is about *limiting chaos*
  - Helpful way to differentiate between different software processes
- Waterfall: chaos is managed by a detailed plan and milestones
- Scrum: all work done in *time-boxed* sprints
  - A *timebox* is a period of 2-3 weeks where all requirements are frozen
- Kanban: limit the current work in progress

[5] Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.

Requirements

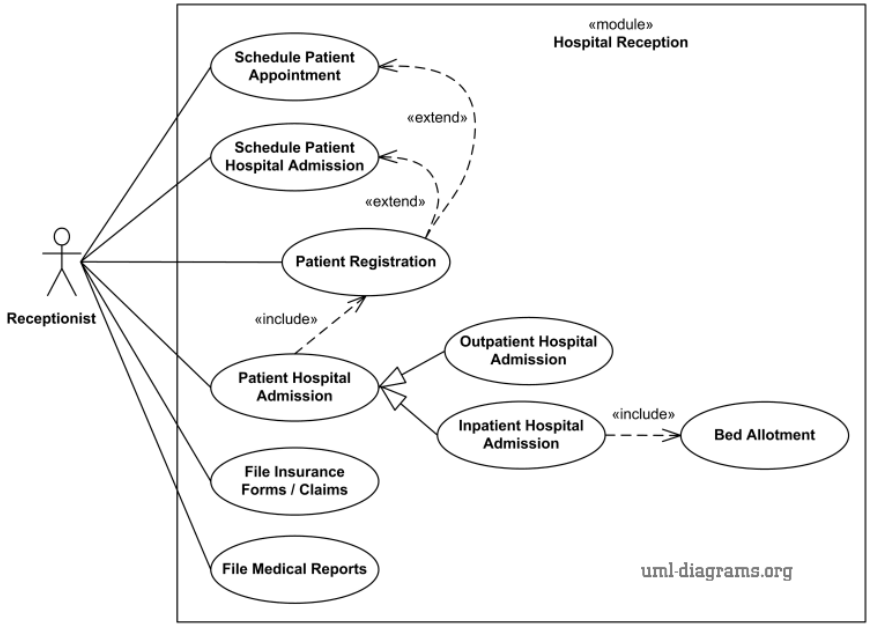
- What should be the output of the planning phase?
- "Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a property or attribute. They may be a constraint on the development process of the system." – Sommerville[6]
- *Requirements Elicitation*: Process of collecting requirements from stakeholders
- *Functional* requirements say what the system will do and *Non-functional* requirements describe what the system won't do

[6] Ian Sommerville. "Software Engineering. International computer science series". In: *ed: Addison Wesley* (2004).

Use Cases

- Use Case: description of a sequence of interactions between the system and one or more external actor, the actor derives some benefit or achieves some goal by interacting with the system
- *Actor*: an actor can be a user or another external system (e.g. credit card or payment processor, external database).
- *Scenario*: A specific path through a use case: main success scenario, alternative flows, exceptions

UML Use Cases



- UML includes a simple diagram type to describe use cases
- Actors are represented by stick figures, the system boundary is represented by a rectangle and use cases are represented by individual ellipse

Textual format of Use Cases

- Complete specification of a use case is textual
- Takes the form of a structured document containing varying levels of detail depending upon the situation
- The title of a use case is typically a simple phrase beginning with a verb: e.g. *buy something*
- Use cases capture the *behavior* of a system when an actor interacts with it
- When writing use cases we can imagine the system in two ways
  - Actors and Goals
  - Stakeholders and Interests

Actors and Goals

- The primary actor has some goal in interacting with the system
- A scenario is a sequence of interactions with the system
- All interactions relate to the primary actor's goal
- Scenarios may end in success or failure: the intended interaction is called the *main success scenario*
- A use case comprises all of the possible scenarios in achieving the goal
- The interactions can be folded and unfolded depending upon the level of detail required

The *buy something* use case

- In this case the actor and goal are clear
- Actor need not be a specific individual: class of people or systems that can assume a *role*
- What are the *pre-conditions*?
- What is the *main success scenario*?
- How can this use case be specified at varying levels of detail?

Extensions

- Extensions capture the scenarios other than the main success scenario
- Extensions are written below the main success scenario
- An extension occurs at each point in the main success scenario where behavior can branch
- Extension may merge back with main success scenario or end in failure

The Use Case Format

NAZARBAYEV UNIVERSITY
SCHOOL OF ENGINEERING AND DIGITAL SCIENCES

- A possible Use Case template is as follows[7]

```
<name of the use case>
Context: <statement of the goal>
Scope: <the system under consideration>
Primary actor: <description or role>
Stakeholders and Interests: <stakeholders and relevant interests>
Precondition: <expected starting conditions>
Min. Guarantees: <protection of the interests>
Success Guarantees: <state of world if goal succeeds>
Trigger: <initiating event>
Main Success Scenario: <steps of main scenario>
<step #> <description>
Extensions: <extensions ref. main scenario>
<step altered> <condition> <action description>
Related Information:
<other important information for the project>
```

[7] Alistair Cockburn. *Writing effective use cases*. 2001.

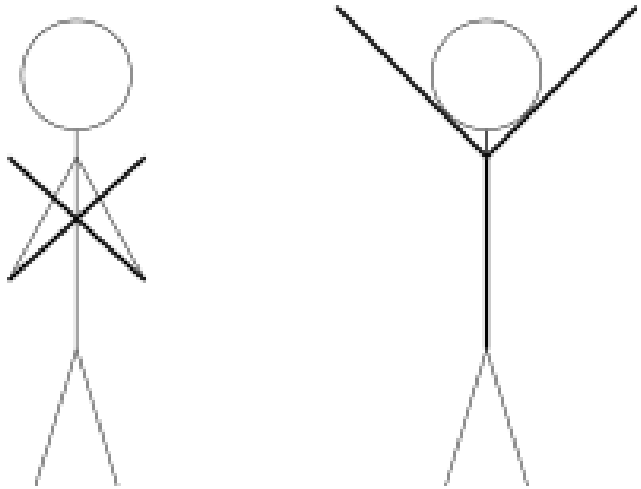
Stakeholders and Interests

- Actors and Goals helps to describe the components (interactions) of the use case
- Internal behavior of the system is captured by a *Stakeholders and Interests* model
- Contract between stakeholders and interests
- Indicates what should be included in the use case
- In addition to fulfilling the goal of the primary actor the system has to protect the interests of the stakeholders

Need for Agile

- Computers and software have advanced significantly
- Computers are more available now and are supported by better development tools
- New set of challenges for software developers
- Agile processes consist of the same *activities* as waterfall, except they overlap/occur in parallel
- Handoffs from one activity to the next occur over longer period of time
  - Greater collaboration between various specialists in development team

Theory X and Theory Y[8]



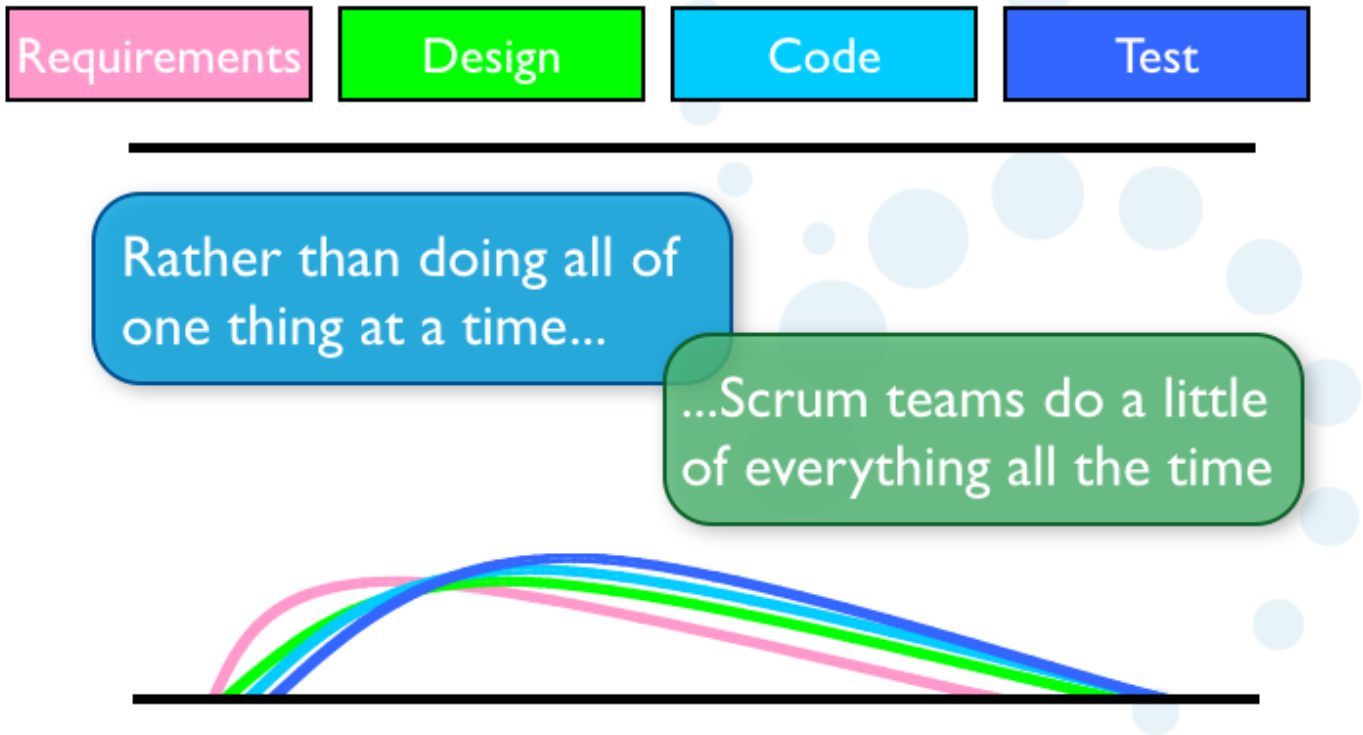
- Two theories of motivation at work
- Theory X: workers need external motivation (rewards and penalities), close supervision
- Theory Y: workers have internal motivation, should have responsibility for what they work on
- Mirrors how people think about Agile versus Waterfall

[8] Douglas MacGregor. *The human side of enterprise*. Vol. 21. 166-171. New York McGraw-Hill, 1960.

Overlapping Activities

NAZARBAYEV UNIVERSITY
SCHOOL OF ENGINEERING AND DIGITAL SCIENCES

Sequential vs. overlapping development



Source: "The New New Product Development Game" by Takeuchi and Nonaka. *Harvard Business Review*, January 1986.

Mountain Goat Software, LLC