CPSC 3720 W02-1: The Tar Pit, SDLC, & Agile

Dr. Alex Adkins

Who Are You?

ON THE BLANK SIDE: Your name

ON THE LINED SIDE:

- Your Major
- Where You're From
- Something interesting about yourself

Today’s Objectives

- Review & discuss reading
- Discuss software engineering and complexity
- Begin discussion of software lifecycle and processes

Discussion

At your tables
(~5 minutes)

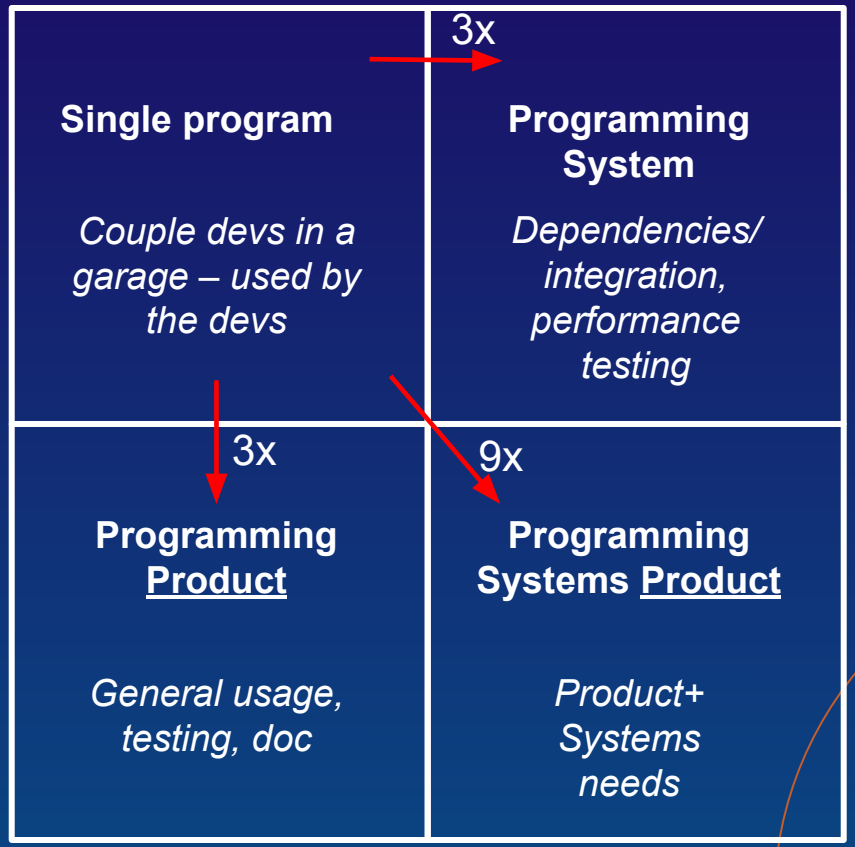
- Do you agree with Fred Brooks' perspective that software programming is a craft? Or do you think it is more of an engineering discipline? Why?

Discussion

At your tables
(~5 minutes)

- How much do you use AI today for programming? Do you think it is making you a better developer?
- What do you think is AI coders' biggest weakness? Will AI replace software developers?

The Tar Pit: Complexity of a Program vs. Product

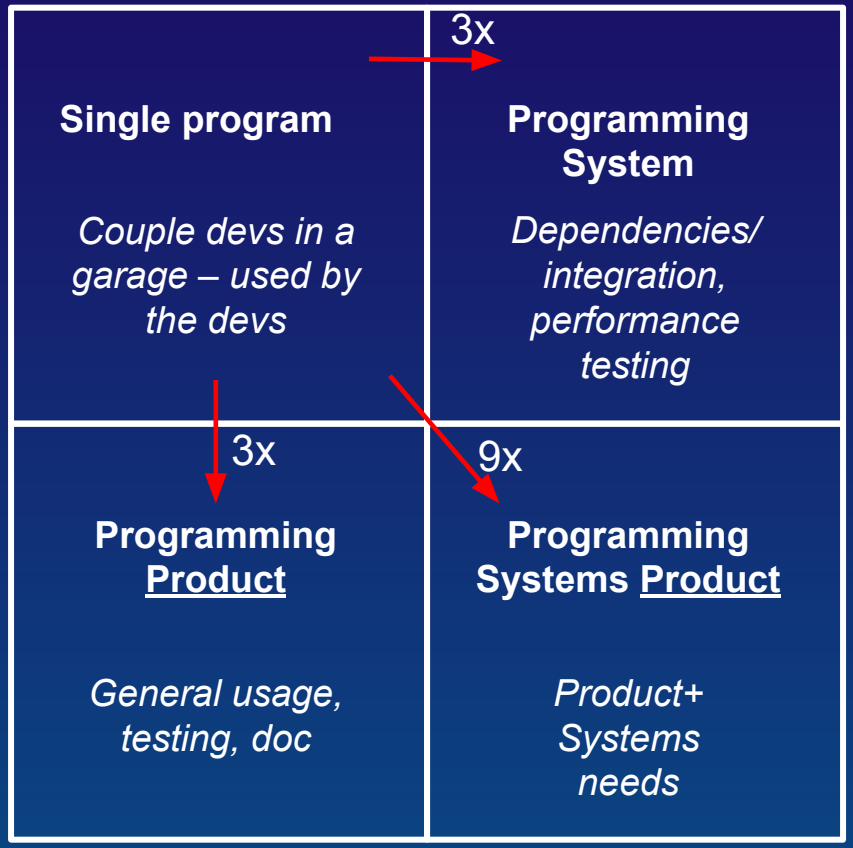


Discussion

At your tables:

- It is generally accepted that it is easier to estimate and build a quality house than it is to estimate and build a quality complex software system
- **Why**?

The Tar Pit: Complexity of a Program vs. Product



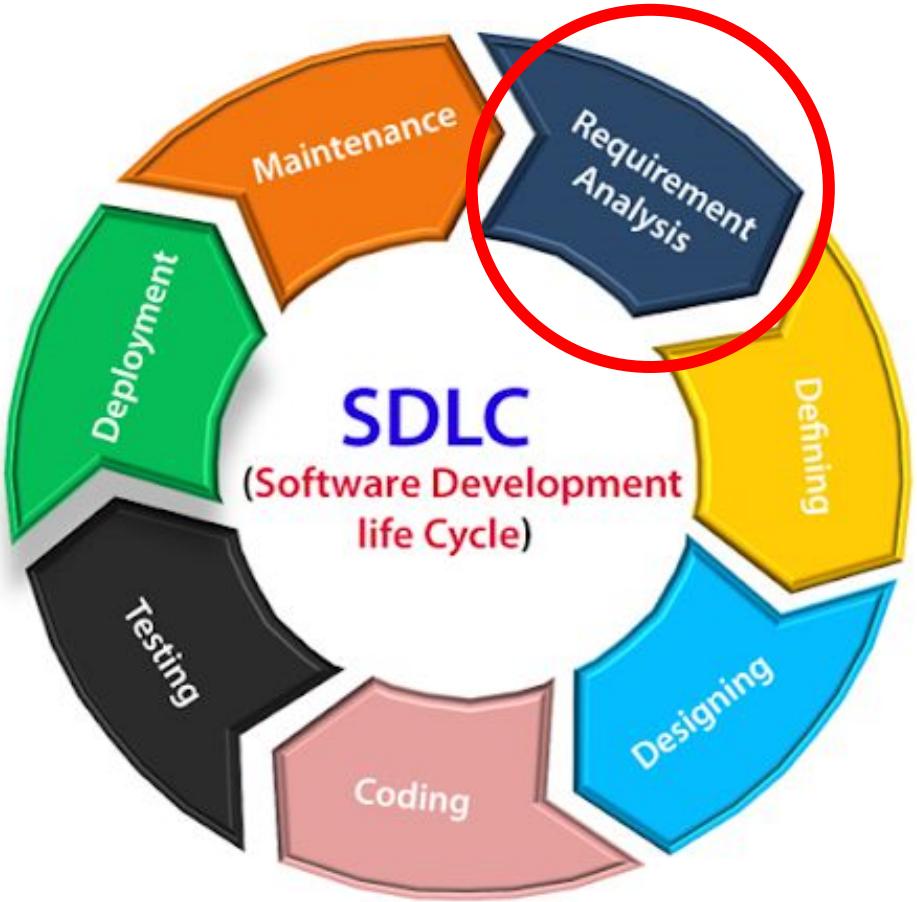
How do we manage this complexity?

Software Development Process

- **Software Process**: a way of breaking down this overall software development work into manageable sub-tasks
  - Systematic and somewhat formal

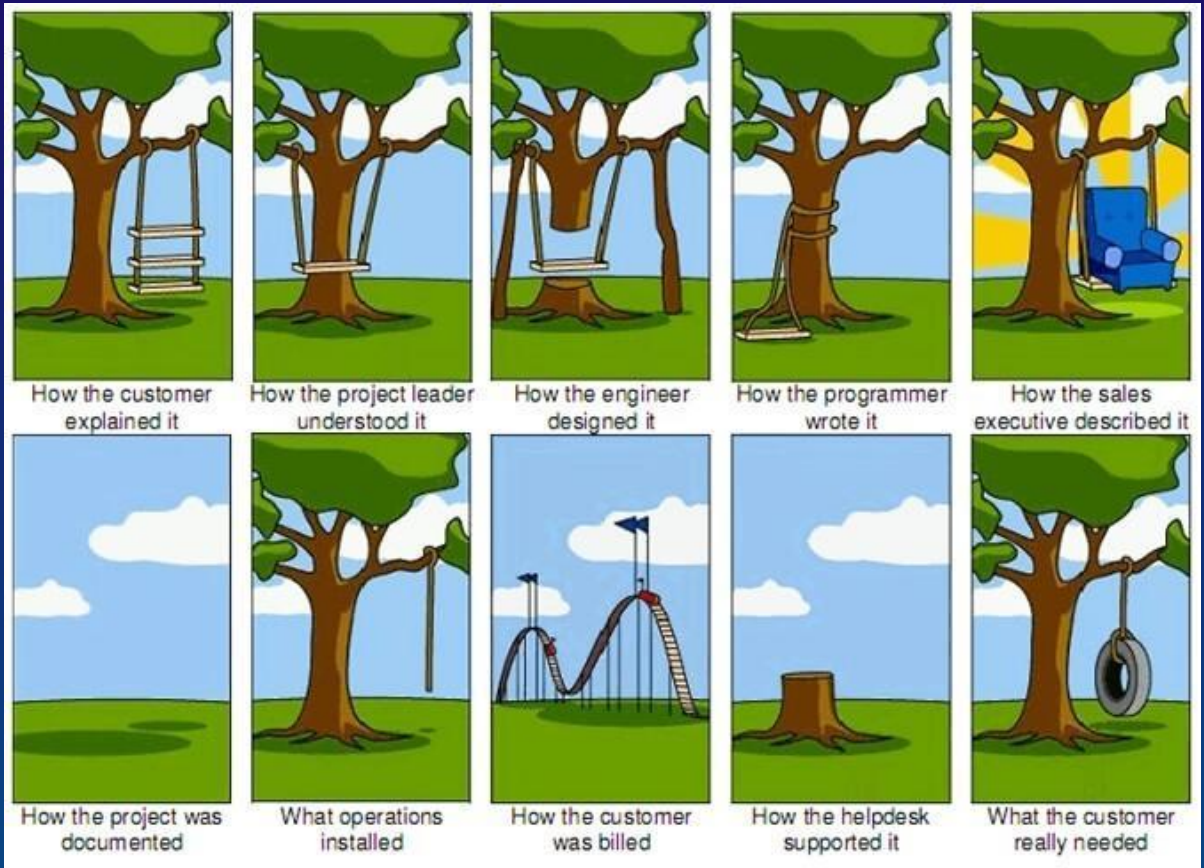
Software Development Process Steps

Requirements Analysis



SDLC: Requirements Analysis

Requirements Analysis



SDLC: Requirements Analysis

- The WHAT and the WHY
- Understanding what the customer wants
  - Or what they THINK they want
  - ASK WHY
- Focus on the business problem you are trying to solve
- Understand what is most important to the customer to enable prioritization
- Can be documented in various ways depending on the process:
  - Formal requirements specifications
  - Wireframes
  - Use case documents
  - Prototypes

How do we know we got it right?

- How do we validate the requirements?
  - Avoid producing a good apple when an orange is required

Validation vs Verification

- Project Management Book of Knowledge (IEEE Standard)
  - defines these terms:

- **Validation**: The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers.

- **Verification**: The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process.

Validation vs Verification

- **Validation**: Are we producing the right product?
- **Verification**: Are we producing the product right?

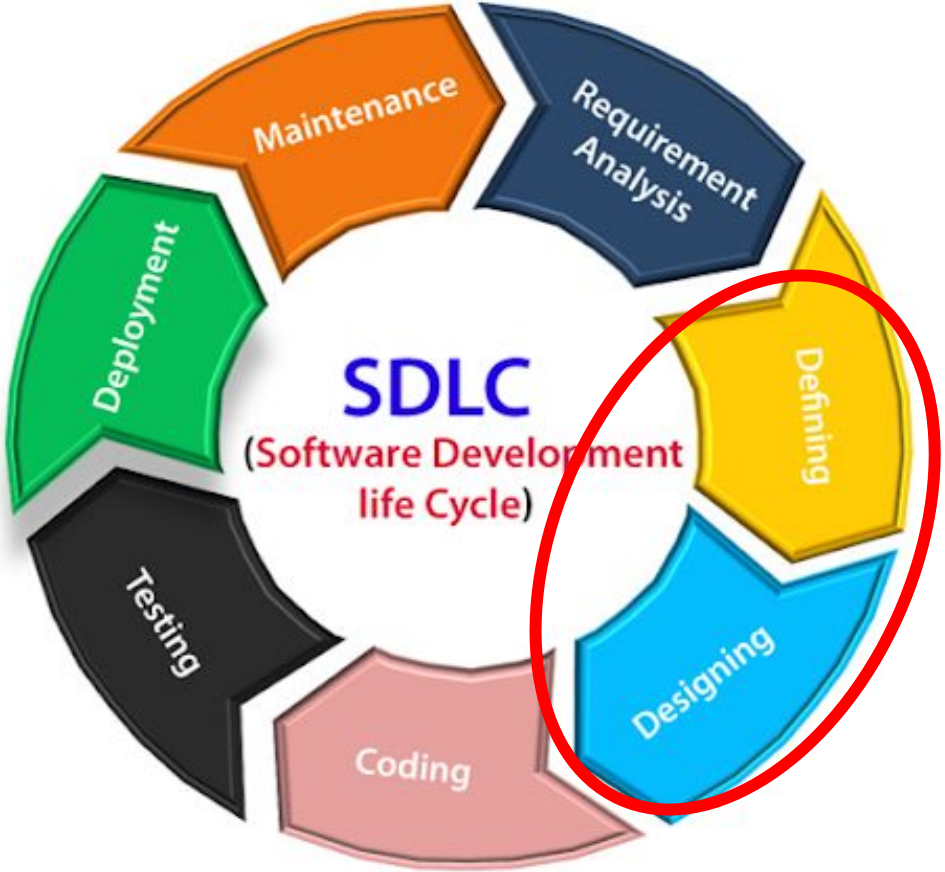
Discussion

At your tables:

- Which is more important?
- Validation
  - Producing the right product
- Verification
  - Producing the product right


YEAH, I GOT ALL THE REQUIREMENTS WRONG
SO IF YOU COULD DO IT ALL OVER, THAT WOULD BE GREAT
makeameme.org

Define and Design



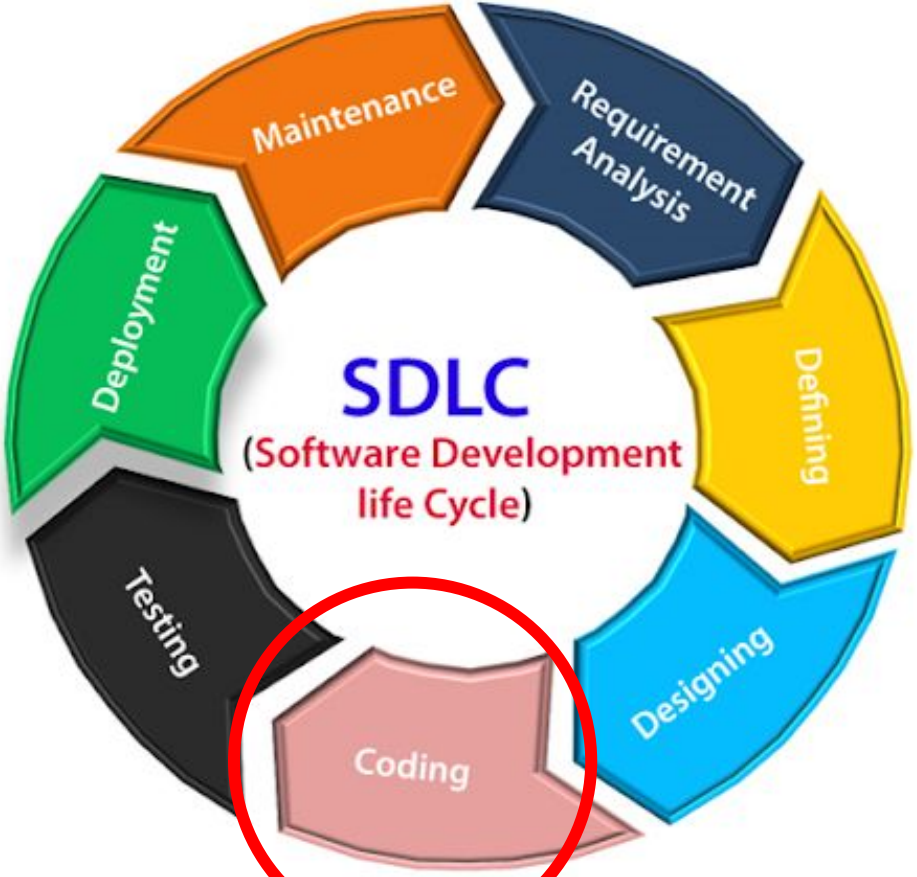
Define and Design

- The HOW
- Need to understand both the business and non-functional requirements
- Depending on the type and size of system you will have various layers of design:
  - System architecture
  - Deployment architecture
  - Object/Class Designs
  - Database designs
  - UI designs

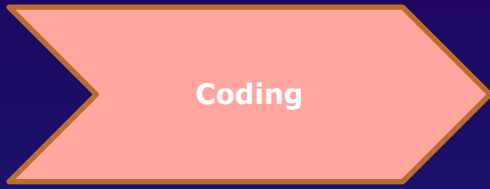
Define and Design

- Can be documented in various ways depending on the process and the type of system:
  - Formal design specifications
  - UML
  - State and Transition diagrams
  - Wiki documents
  - Contract documentation (for APIs)
  - ERDs (Entity Relationship Diagrams)
  - Data Flow Diagrams (DFDs)
  - Whiteboarding sessions and photos!

Coding


Maintenance
Requirement Analysis
Deployment
Defining
SDLC
(Software Development life Cycle)
Testing
Designing
Coding

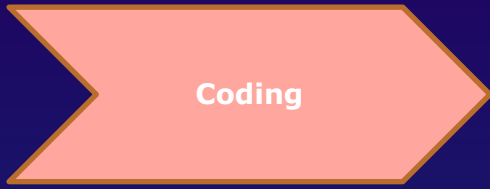
Coding



- AKA Implementation

- Depending on organization you could have different standards and methods:
  - Language requirements
  - Pair programming
  - Code standards
  - Code reviews
  - Tool usage requirements
  - Internal and external frameworks
  - Unit testing requirements

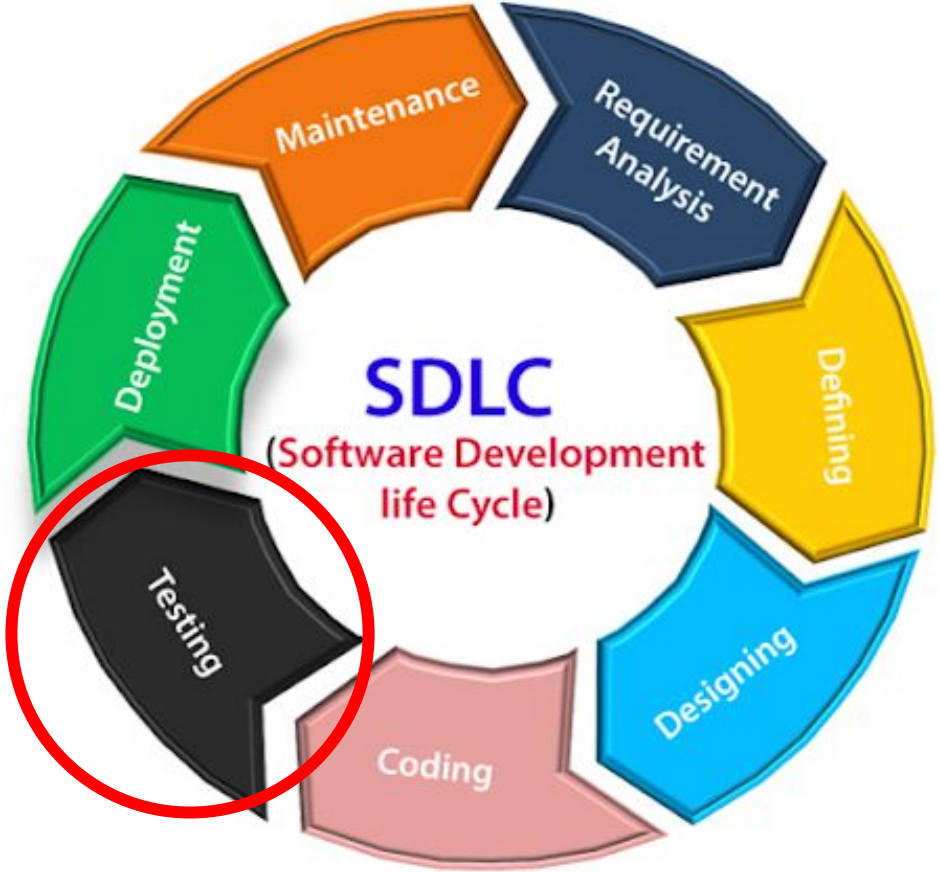
Coding



- Dev/Coding Tools:
  - Configuration management (GIT)
  - IDEs
  - 3rd Party Tools
  - API development tools
  - Documentation tools

What we'll be doing in 3720

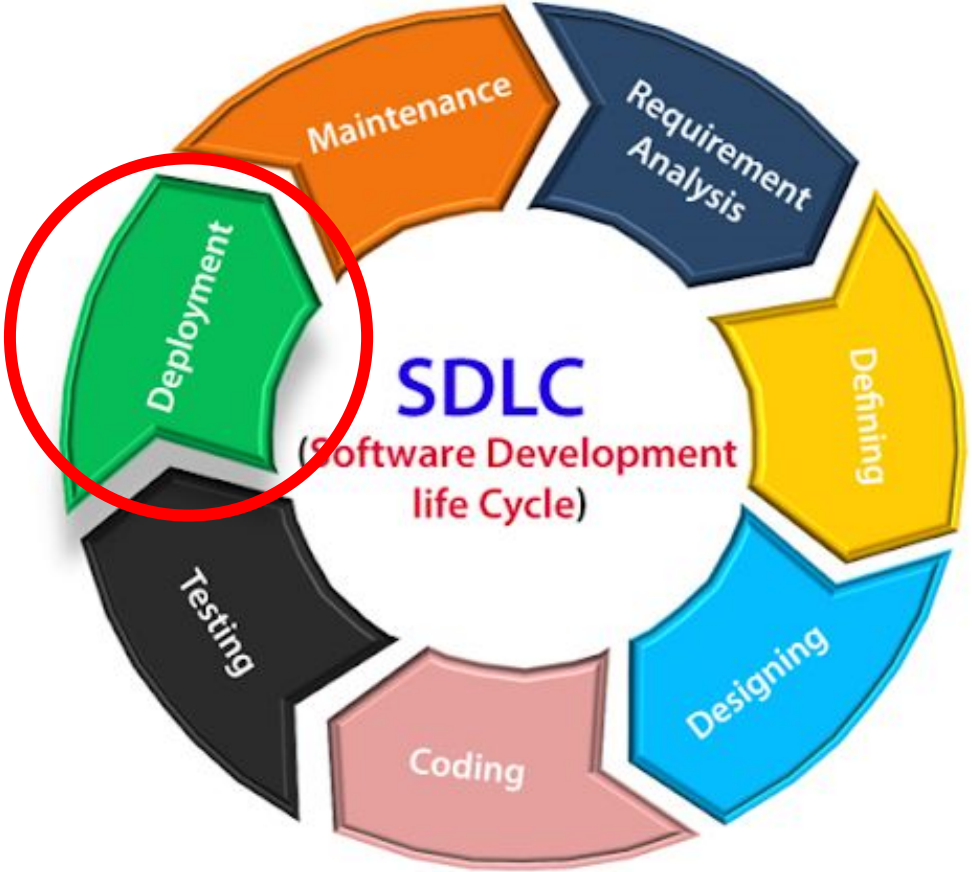
Testing


Maintenance
Requirement Analysis
Deployment
Defining
SDLC
(Software Development life Cycle)
Testing
Designing
Coding

Testing

- AKA Quality Assurance – the validation AND verification
- Depending on organization and process you will have different approaches:
- When you test:
  - Unit Testing
  - Integration Testing
  - System Testing
  - Performance Testing
  - Regression Testing
- How you Test
  - Test Driven Development
  - Continuous Testing
  - Automated vs. Manual

Deployment

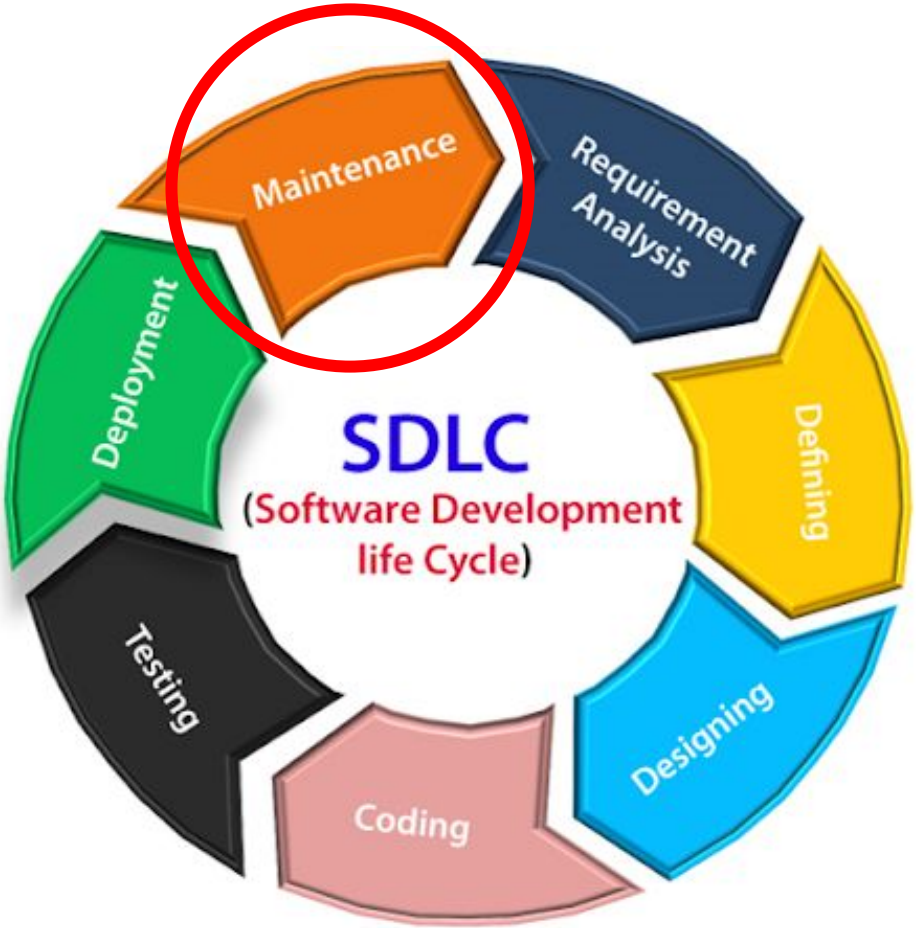

Maintenance
Requirement Analysis
Deployment
Defining
SDLC
(Software Development life Cycle)
Testing
Designing
Coding

Deployment

- Deployment is the process of putting your software into production for use by the customer

- Deployment will vary based on type of software being developed:
  - **Commercial “on-premise”** – Customer IT will deploy the software
  - **Commercial SaaS** – Company dev/operations will deploy the software
  - **Internal** – Company dev/operations will deploy the software

Deployment


Maintenance
Requirement Analysis
Defining
Designing
Coding
Testing
Deployment
SDLC
(Software Development life Cycle)

Maintenance

- Fixing bugs in production that are found by the customer
- The deployment model will dictate how the maintenance is delivered
  - Emergency Patch fixes
  - Maintenance Releases
  - Major Releases
- -OR-
  - Continual Deployment (more about this later)

Putting it all Together

- The listed steps are used in most software engineering projects
  - However, they can be done in many different ways

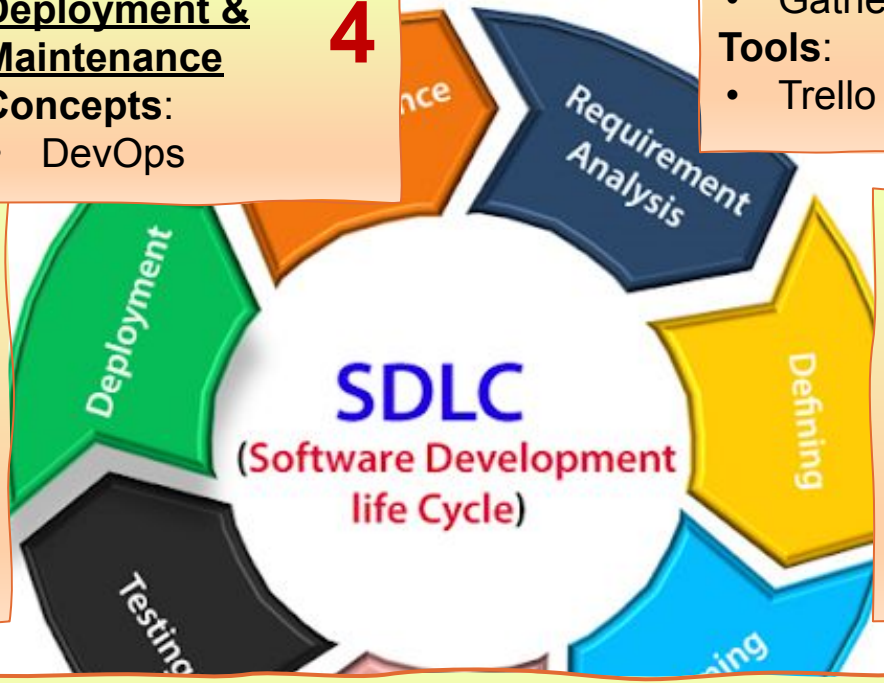
- The SDLC in use will vary company to company based on their business needs and culture
  - It is not “one size fits all”

3720 Software Engineering: Where are We Going?

Requirements Analysis 1
Concepts:
- Epics/Stories
- Gathering Reqts

Tools:
- Trello

Deployment & Maintenance 4
Concepts:
- DevOps



Coding and Testing 3
Concepts:
- Code Mgmt
- Test techniques

Tools:
- GIT
- AWS API Dev Tools
- Postman

Define/Design 2
Concepts:
- Microservices
- Cohesion & Coupling
- API First and API Specification

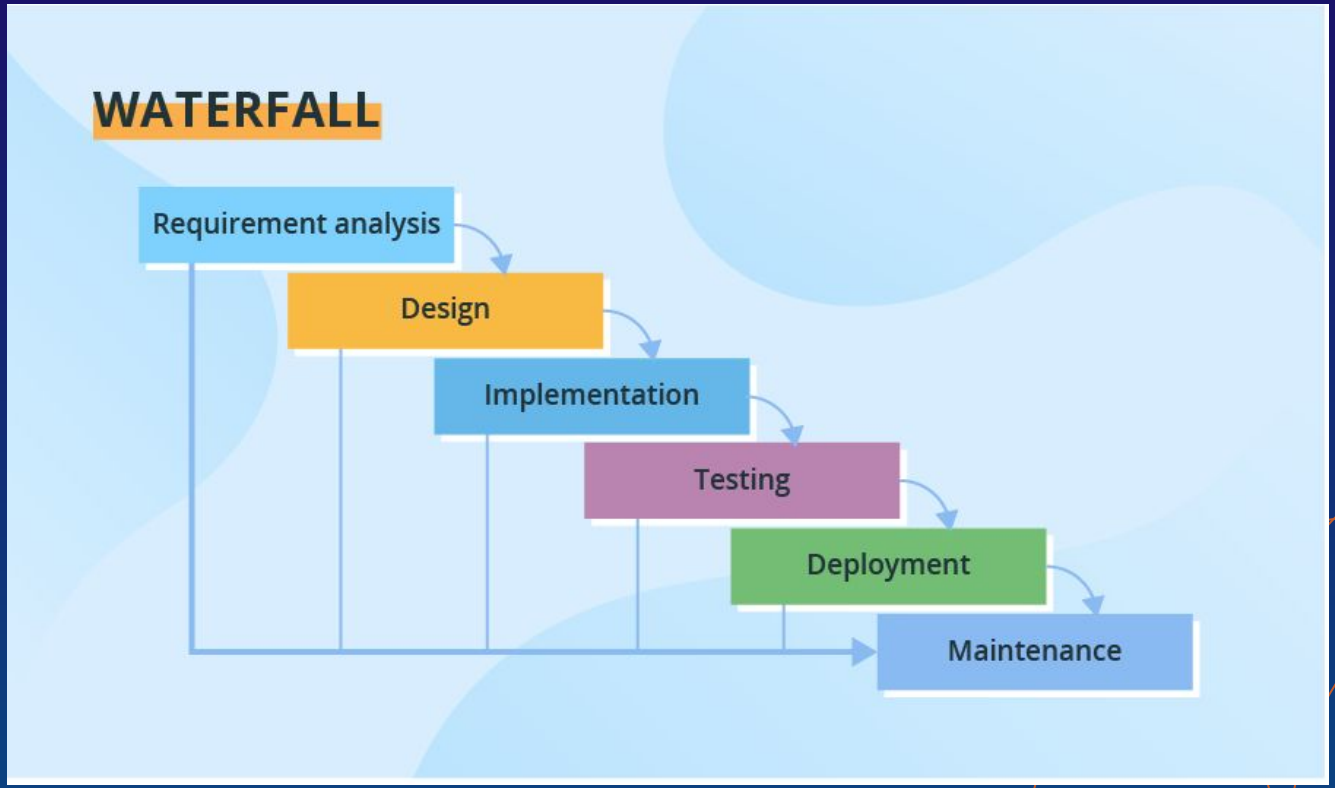
Tools:
- Postman

FOUNDATIONAL CONCEPTS
- Use of a **software process** to address software complexity; **Agile/Scrum** for class
- **Effective Teaming** is essential throughout SDLC; Conway's Law + Team=Software
- Software **planning and estimation** is hard! Relative sizing and storypoints

Process Models: SDLC Approaches

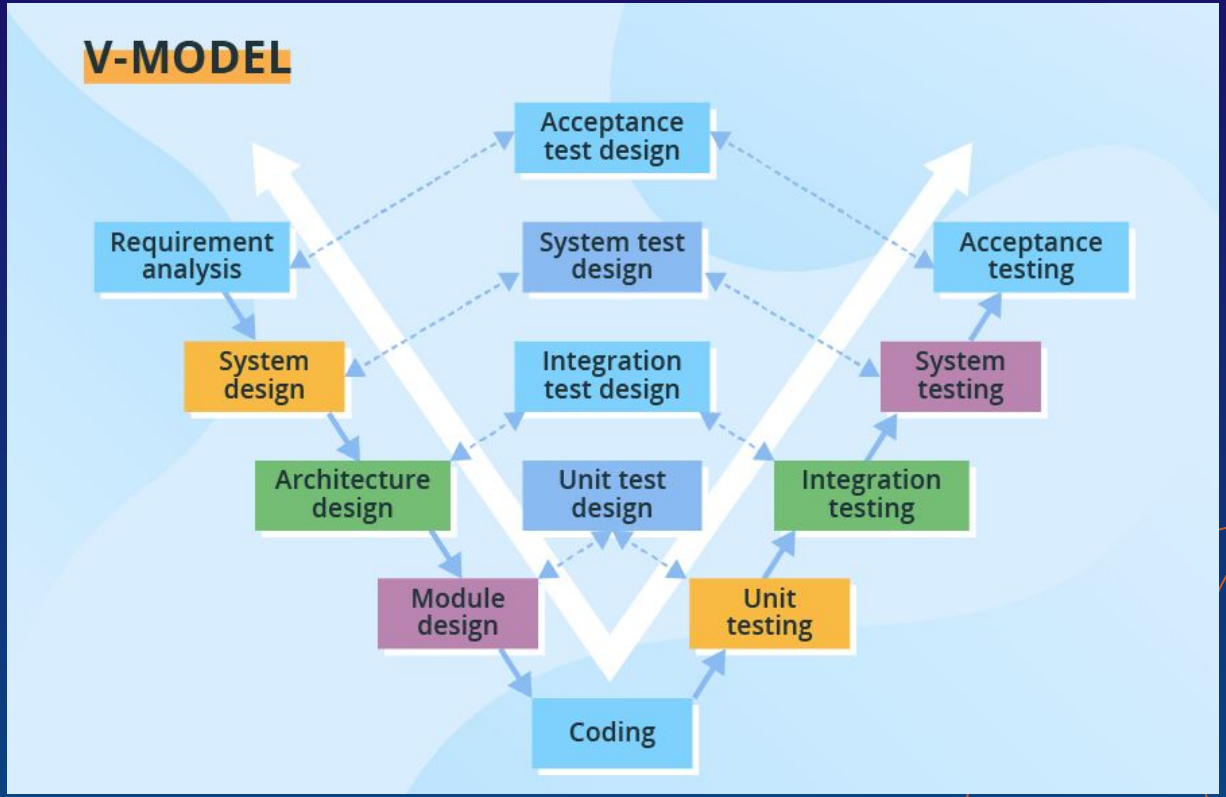
- Waterfall
- V-Model
- Incremental
- Iterative
- Spiral Model
- RUP
- Agile:
  - Scrum
  - XP
  - Kanban
  - Lean

Waterfall process Model



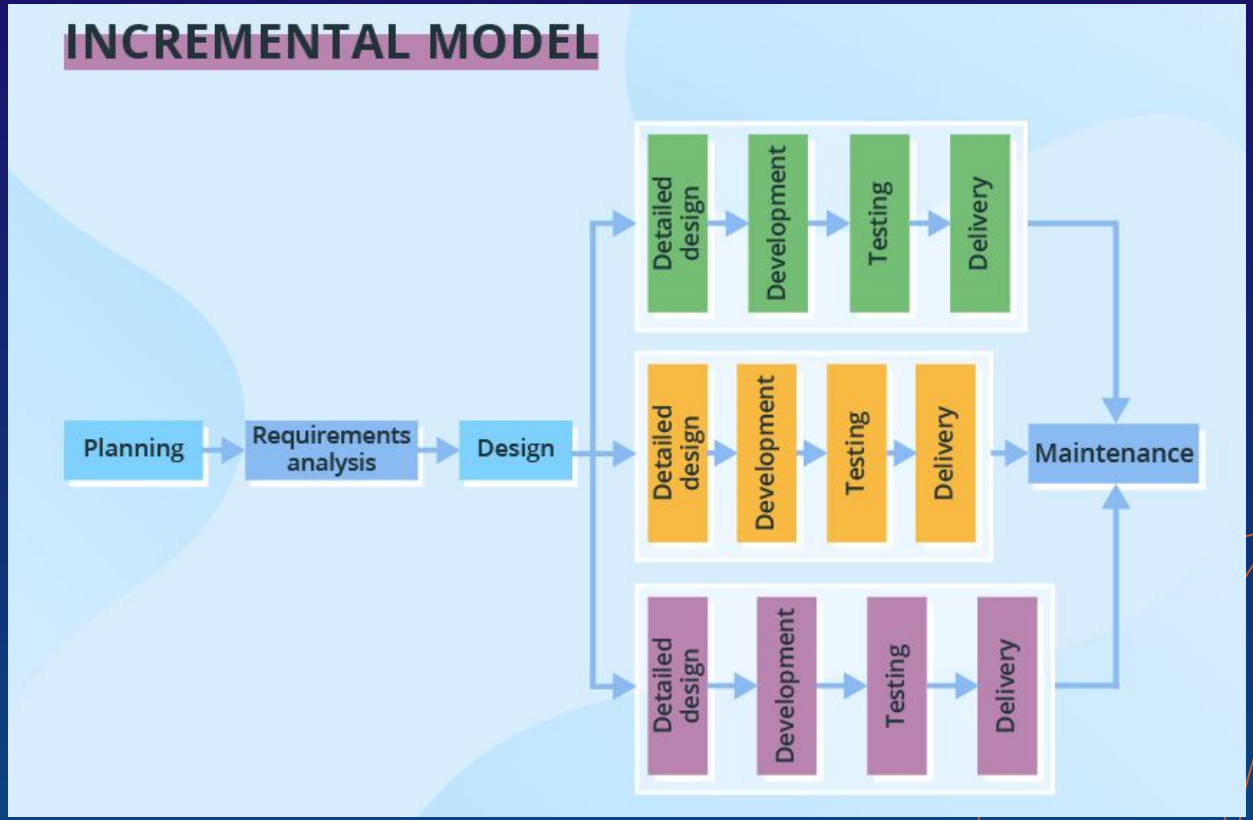
Source: https://www.scnsoft.com/blog/software-development-models

V-Model



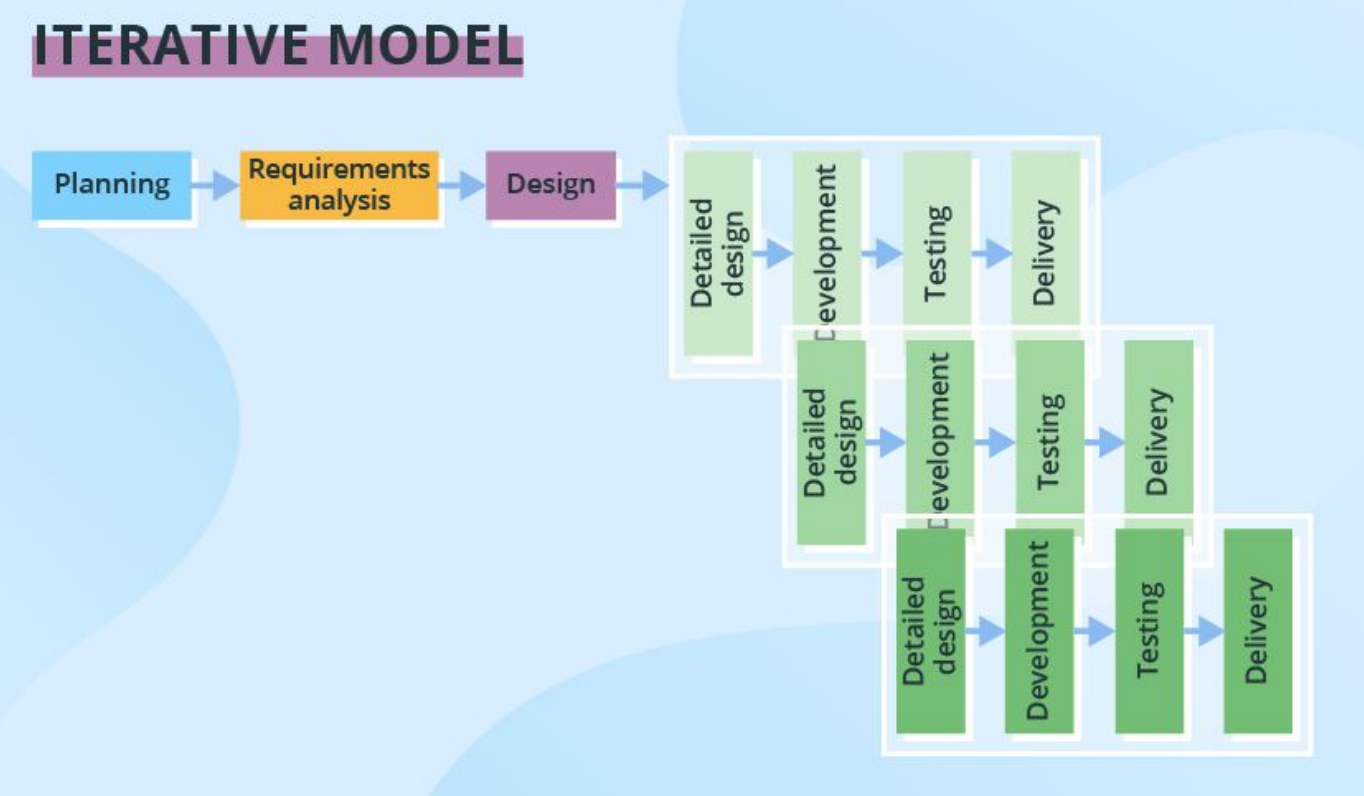
Source: https://www.scnsoft.com/blog/software-development-models

Incremental Model



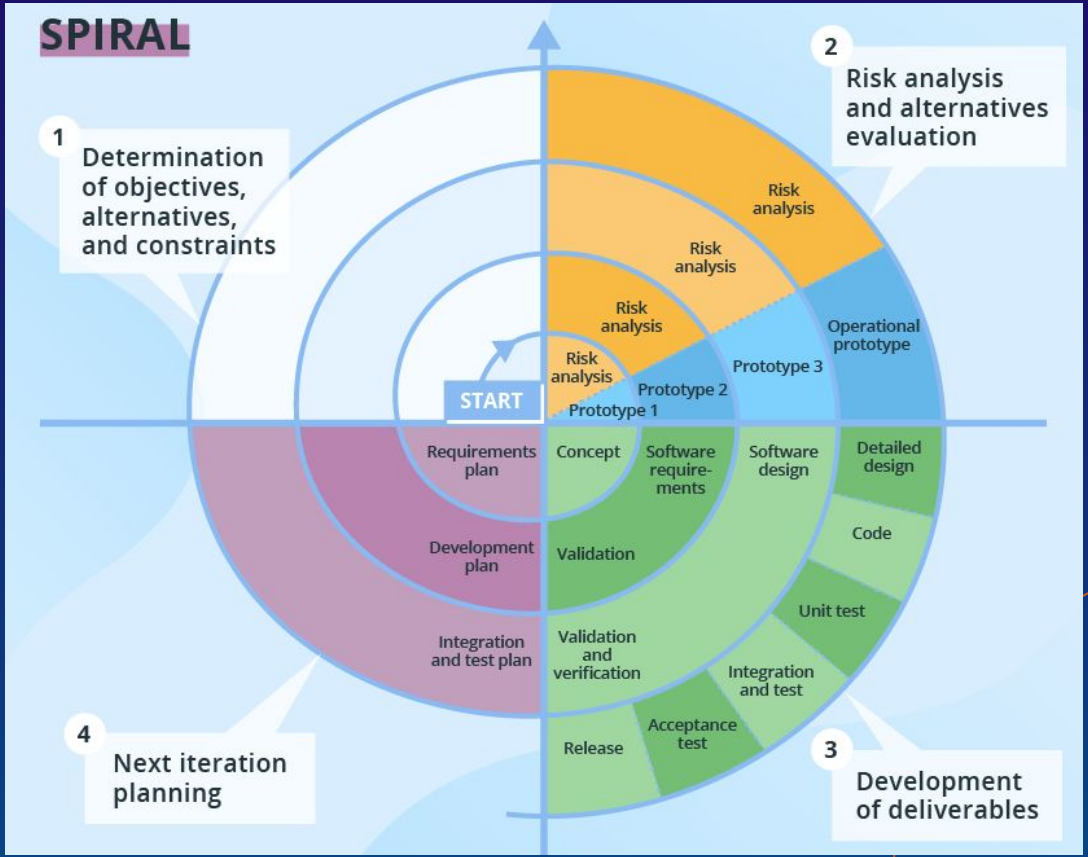
Source: https://www.scnsoft.com/blog/software-development-models

Iterative Model



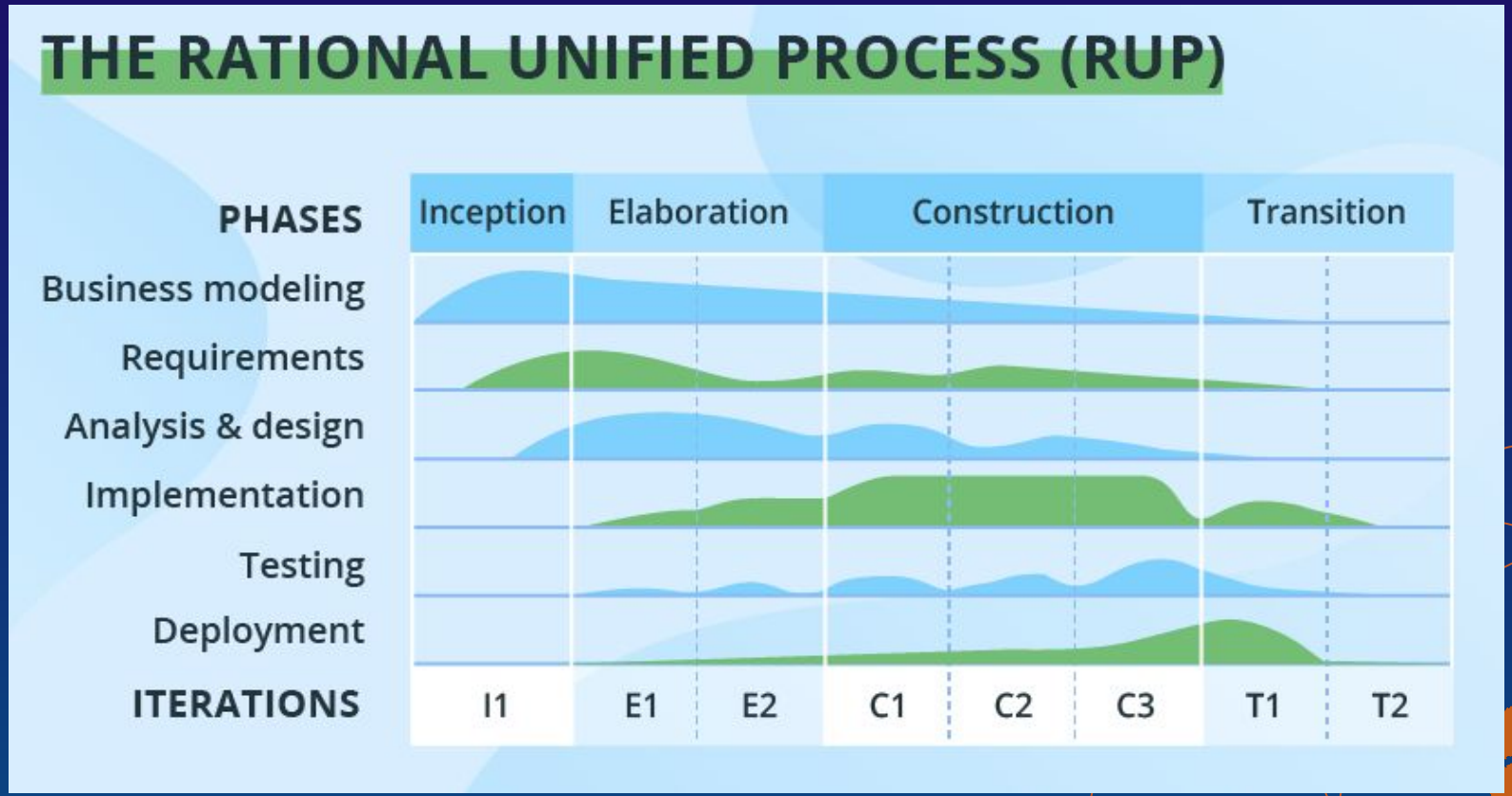
Source: https://www.scnsoft.com/blog/software-development-models

Spiral Model

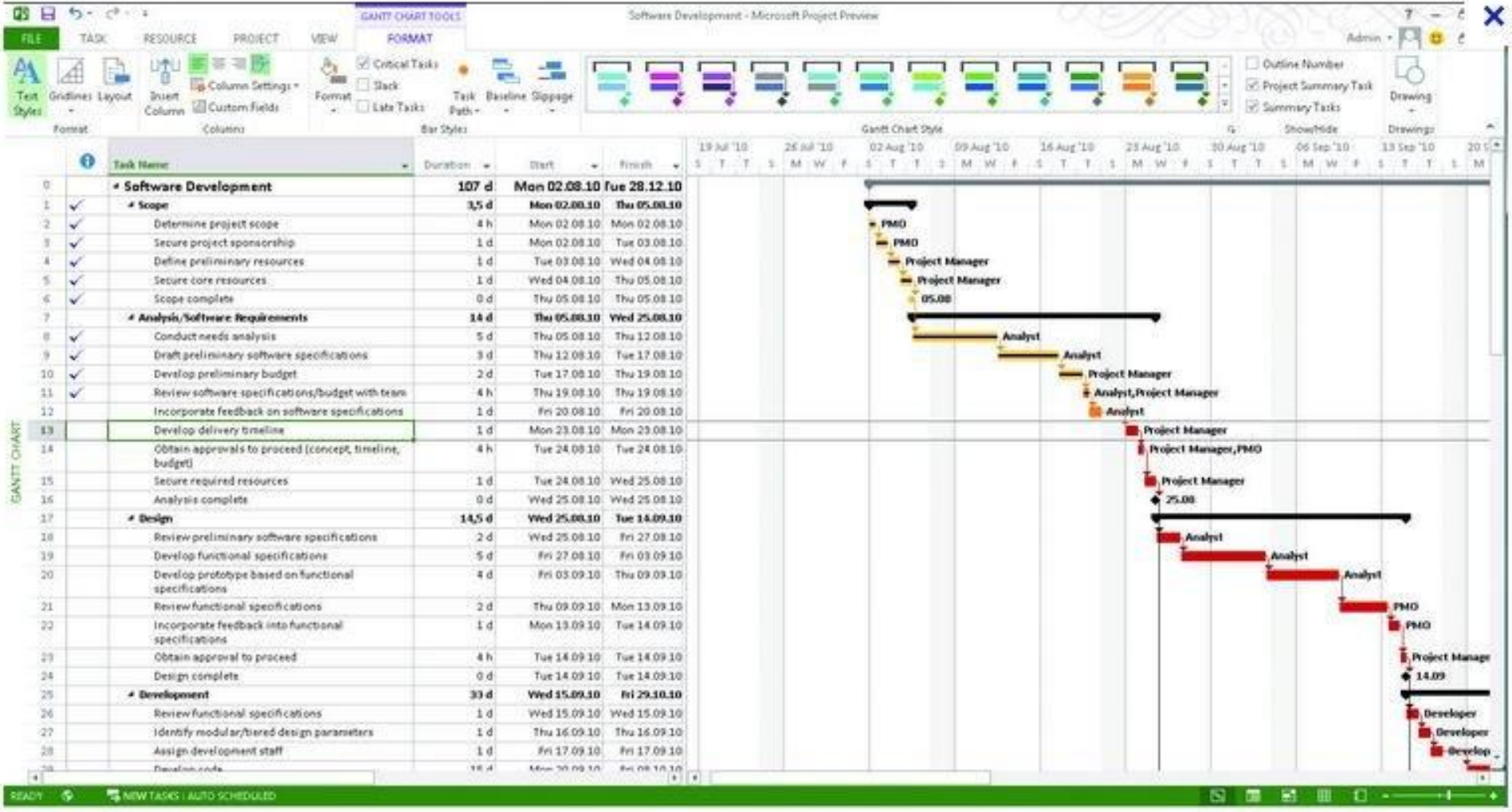


Source: https://www.scnsoft.com/blog/software-development-models

RUP



Source: https://www.scnsoft.com/blog/software-development-models


Software Development - Microsoft Project Preview
GANTT CHART TOOLS
FILE
TASK
RESOURCE
PROJECT
VIEW
FORMAT
Task Name
Duration
Start
Finish
0 Software Development 107 d Mon 02.08.10 Tue 28.12.10
1 Scope 3,5 d Mon 02.08.10 Thu 05.08.10
2 Determine project scope 4 h Mon 02.08.10 Mon 02.08.10
3 Secure project sponsorship 1 d Mon 02.08.10 Tue 03.08.10
4 Define preliminary resources 1 d Tue 03.08.10 Wed 04.08.10
5 Secure core resources 1 d Wed 04.08.10 Thu 05.08.10
6 Scope complete 0 d Thu 05.08.10 Thu 05.08.10
7 Analysis/Software Requirements 14 d Thu 05.08.10 Wed 25.08.10
8 Conduct needs analysis 5 d Thu 05.08.10 Thu 12.08.10
9 Draft preliminary software specifications 3 d Thu 12.08.10 Tue 17.08.10
10 Develop preliminary budget 2 d Tue 17.08.10 Thu 19.08.10
11 Review software specifications/budget with team 4 h Thu 19.08.10 Thu 19.08.10
12 Incorporate feedback on software specifications 1 d Fri 20.08.10 Fri 20.08.10
13 Develop delivery timeline 1 d Mon 23.08.10 Mon 23.08.10
14 Obtain approvals to proceed (concept, timeline, budget) 4 h Tue 24.08.10 Tue 24.08.10
15 Secure required resources 1 d Tue 24.08.10 Wed 25.08.10
16 Analysis complete 0 d Wed 25.08.10 Wed 25.08.10
17 Design 14,5 d Wed 25.08.10 Tue 14.09.10
18 Review preliminary software specifications 2 d Wed 25.08.10 Fri 27.08.10
19 Develop functional specifications 5 d Fri 27.08.10 Fri 03.09.10
20 Develop prototype based on functional specifications 4 d Fri 03.09.10 Thu 09.09.10
21 Review functional specifications 2 d Thu 09.09.10 Mon 13.09.10
22 Incorporate feedback into functional specifications 1 d Mon 13.09.10 Tue 14.09.10
23 Obtain approval to proceed 4 h Tue 14.09.10 Tue 14.09.10
24 Design complete 0 d Tue 14.09.10 Tue 14.09.10
25 Development 33 d Wed 15.09.10 Fri 29.10.10
26 Review functional specifications 1 d Wed 15.09.10 Wed 15.09.10
27 Identify modular/tiered design parameters 1 d Thu 16.09.10 Thu 16.09.10
28 Assign development staff 1 d Fri 17.09.10 Fri 17.09.10
19 Jul '10
26 Jul '10
02 Aug '10
09 Aug '10
16 Aug '10
23 Aug '10
30 Aug '10
06 Sep '10
13 Sep '10
PMO
Project Manager
05.08
Analyst
Analyst,Project Manager
Project Manager,PMO
25.08
14.09
Developer
GANTT CHART
READY
NEW TASKS : AUTO SCHEDULED

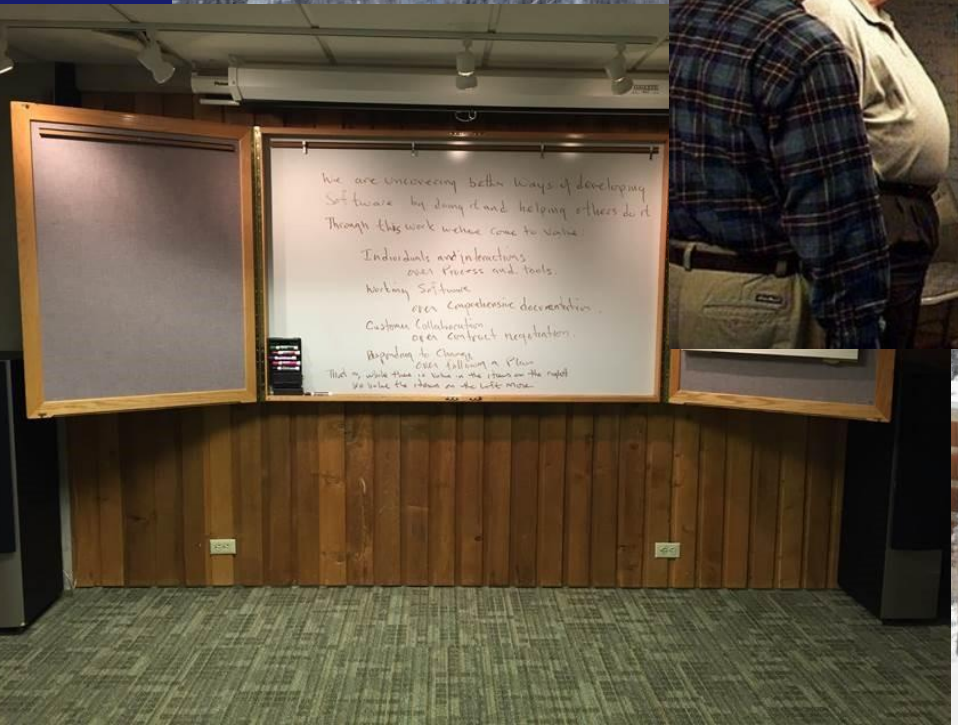
Discussion

At your tables:

- What are some of the issues with the waterfall or other very formal software development processes?

Process Concerns

- Classical methods of software development have many disadvantages:
  - Huge effort during the planning phase
  - Poor requirements conversion in a rapidly changing environment
  - Treatment of staff as a factor of production


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 process and tools
Working Software
over comprehensive documentation
Customer Collaboration
over contract negotiation
Responding to Change
over following a Plan

- Kent Beck, who co-created eXtreme Programming (XP)
- Mike Beedle, co-author of Agile Software Development with Scrum
- Arie van Bennekum, owner of Integrated Agile
- Alistair Cockburn, IT strategist and creator of the Crystal Agile Methodology
- Ward Cunningham, inventor of wiki and first to coin term technical debt
- Martin Fowler, software practitioner, and partner at Thoughtworks
- James Grenning, author of Test-Driven Development
- Jim Highsmith, creator of Adaptive Software Development (ASD)
- Andrew Hunt, co-author of *The Pragmatic Programmer*
- Ron Jeffries, co-creator of eXtreme Programming (XP)
- Jon Kern, who still helps organizations with agile today
- Brian Marick, a computer scientist and author of several books on programming
- Robert C. Martin, also known as "Uncle Bob," who consults via Clean Coding
- Steve Mellor, a computer scientist also credited with inventing Object-Oriented System Analysis (OOSA)
- Ken Schwaber, who co-created Scrum with Jeff Sutherland
- Jeff Sutherland, the inventor, and co-creator of Scrum
- Dave Thomas, programmer, and co-author of *The Pragmatic Programmer*

The Agile Manifesto

Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

Source: www.agilemanifesto.org

Agile

- The 2001 Agile Manifesto is closest to a definition of Agile workflow
  - Includes a set of 12 principles
- Agile methods are considered
  - Lightweight
  - People-based rather than Plan-based

- No single Agile method
  - Scrum
  - XP
  - Kanban
  - Lean

12 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.

Next Time

- More Agile!
- Agile Reading Assignment Homework on Canvas
  - Due before next class

- Quiz 1 (Lessons 1-4) on Tuesday Sept 10
  - ~15 minutes, closed note, 26 points
  - Will be on your computer – take the tech test quiz