CSc 102 - Project

Zelalem S. Shibeshi

Adapted from Dane Brown

Together we can change the world

Introduction to Software Development

Module Description:

- This module is designed to introduce students to the fundamentals of software development through the engaging and practical task of building a simple game. The module aims to provide students with a hands-on experience in software development activities, emphasizing both technical and collaborative skills.

Introduction to Software Development

Learning Outcomes:

By the end of this module, you will:

- **Understand the basic principles of software development**
- **Develop problem-solving skills by designing and implementing game logic.**
- **Enhance your ability to work effectively in a team, improving your collaboration and communication skills.**
- **Create a functional game prototype, demonstrating your understanding of the software development lifecycle.**

Administration (Similar to 101)

- Important ***concepts*** are highlight in red
  - ***Test questions*** are often based on these concepts
  - Bold/italics emphasizes my point or improves readability
  - Note (***on paper***) anything else you deem important
  - The module has a section on the course page with a name Project

Assessment

- Will there be a project? Yes
- What will be assessed? Project + ???
- Some test and exam questions? Yes

Project: General Info

- Write an app (game) – Minesweeper, tic-tac-toe, Pac-Man or sudoku (last year – card games- Black Jack, (Snap, poker, solitaire, crazy eights, etc)
- Groups of 5 or 4 (**check your group**).
- Tutor will be assigned as a mentor
- Must be in Java or Android for marks to be awarded
- Presentation close to the last week of the semester

Grading of the code (specifics come later):

- completeness
- functionality
- user interface
- understandability
- originality

Project Deliverables

- Project proposal document ~1-2 pages long
- Code pushed to **GitHub** (plagiarism engine)
- ~5 min presentation
- Possibly more

- Try to meet quickly, assign responsibilities, and start working on your project proposal – the deadline is **11 August**.

Software development projects

What is a Project?

- A project is a one-off, temporary activity with a clear start and end; it has full or part-time resources assigned to it; it has temporary management; and it sets out to deliver something: an IT system, a new product, or whatever.
- **Four characteristics of projects:**
  - finite time
  - people assigned
  - clear roles and responsibilities
  - things to deliver
- Have you ever had this feeling about a project?
  - not enough time, too few people, people not sure what they should be doing, and too much to do
- What causes this feeling of overwhelming pressure?
  - **In simple terms, a lack of planning.**

Project management terminologies

The two most important terms:

- *Milestones* are the end-point of a process activity.
- *Deliverables* are project results delivered to customers.

Software Development

- Is the task of converting an abstract idea into complete software.
- It is a complex task.
- Better handled if considered as a process, called software process (defines smaller steps that can be managed better) – read the file "**software development as a process**" on the course page to see the reason why we consider software development as a process.
- The software process is a systematic and formal way of describing how software should be developed.

Software Development as engineering

“Much of our endeavor in software development is the **design and construction** of software to meet some recognised need – of people, organisations or society at large.”

Organising design and construction can be related to what is commonly known as a ‘software development process’

An engineering approach **consists of well-defined process**

Why do we need SE?

- The world is full of examples of failed software projects. Some of the examples of notable failed big software projects:
- London Ambulance Service (1992)
- Healthcare.gov (2013)
- Canada Government website portal (2013)
- The Australian Census (2016)
- The Australian Stock Exchange (2017)
- Boeing 737 MAX (2018-2019)

https://en.wikipedia.org/wiki/List_of_failed_and_overbudget_custom_software_projects

Software process (1/2)

- Process consist of activities/steps to be carried out in a particular order.
- Software process deals with both technical and management issues
- There are different types of process
  - Process for software development:
    - produces software as end-result
    - multiple such process may exist
    - usually a project follows a particular process

Software process (2/2)

- Process for change and configuration management
  - resolving requests for changes
  - defining versions, their compositions
  - release control
- Process for managing the above process themselves
  - improving the processes based on new techniques, tools, etc.
  - standardization and certifications (ISO, PMI)

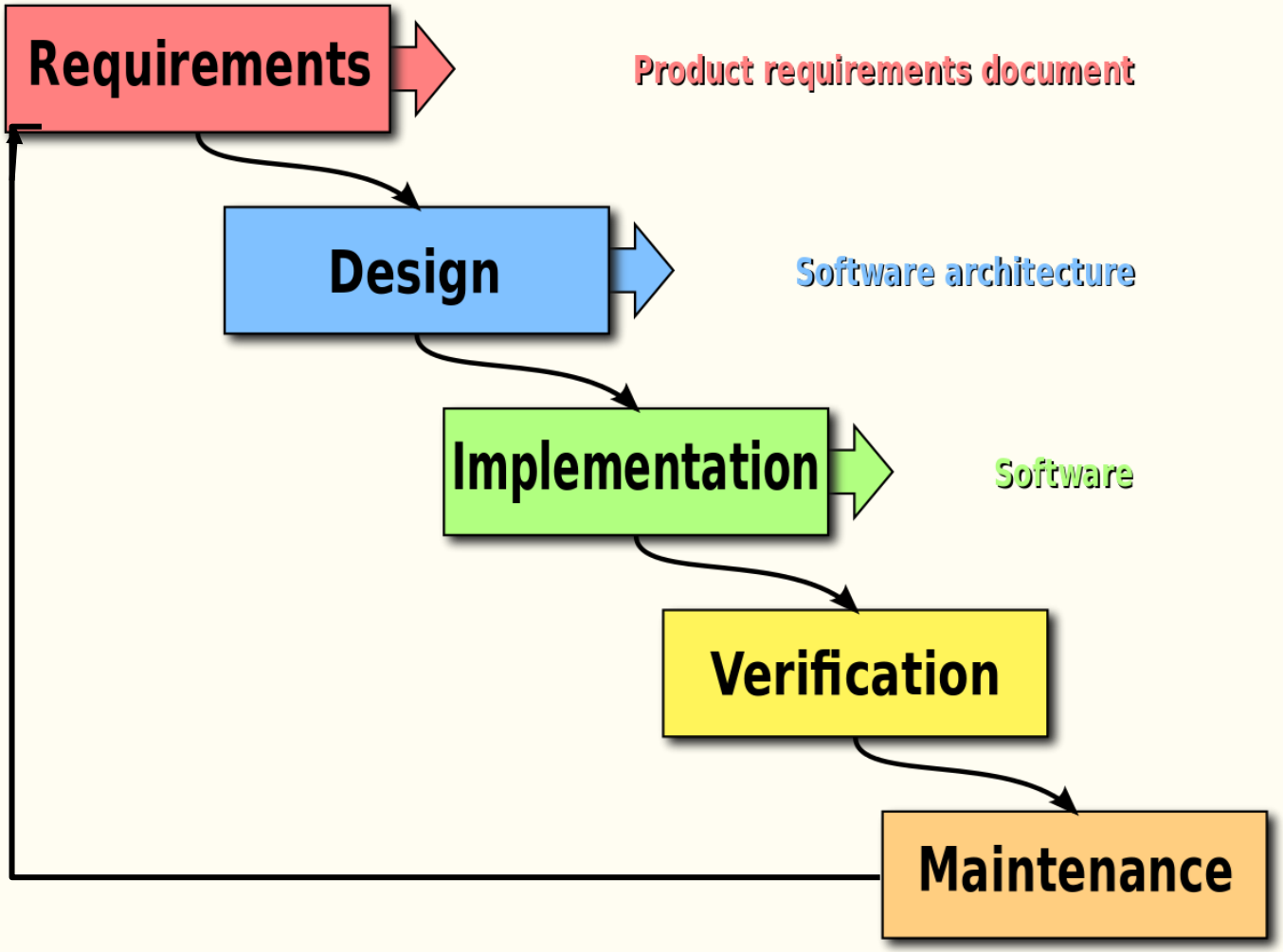
Software Development Life Cycle (SDLC)

A framework that describes the order in which activities are performed at each stage of a software-based project.

A model that outlines the stages, activities, and tasks involved in the creation, development, and maintenance of software systems.

A related term is **software development process,** which is specific implementation of an SDLC.

Software Development Life Cycle (SDLC)



Requirements (and analysis)

What are requirements?

Requirements are documented needs and expectations of **stakeholders** that the software must fulfill. They serve as the foundation for designing, building, and validating the software product.

Requirements include:

- functionality
- behaviour (e.g states)
- Performance (non-functional)

Requirements (and analysis)

We get the required information from different sources.

- Customer
- Users
- System manuals and reports
- Other similar system

Requirements (and analysis)

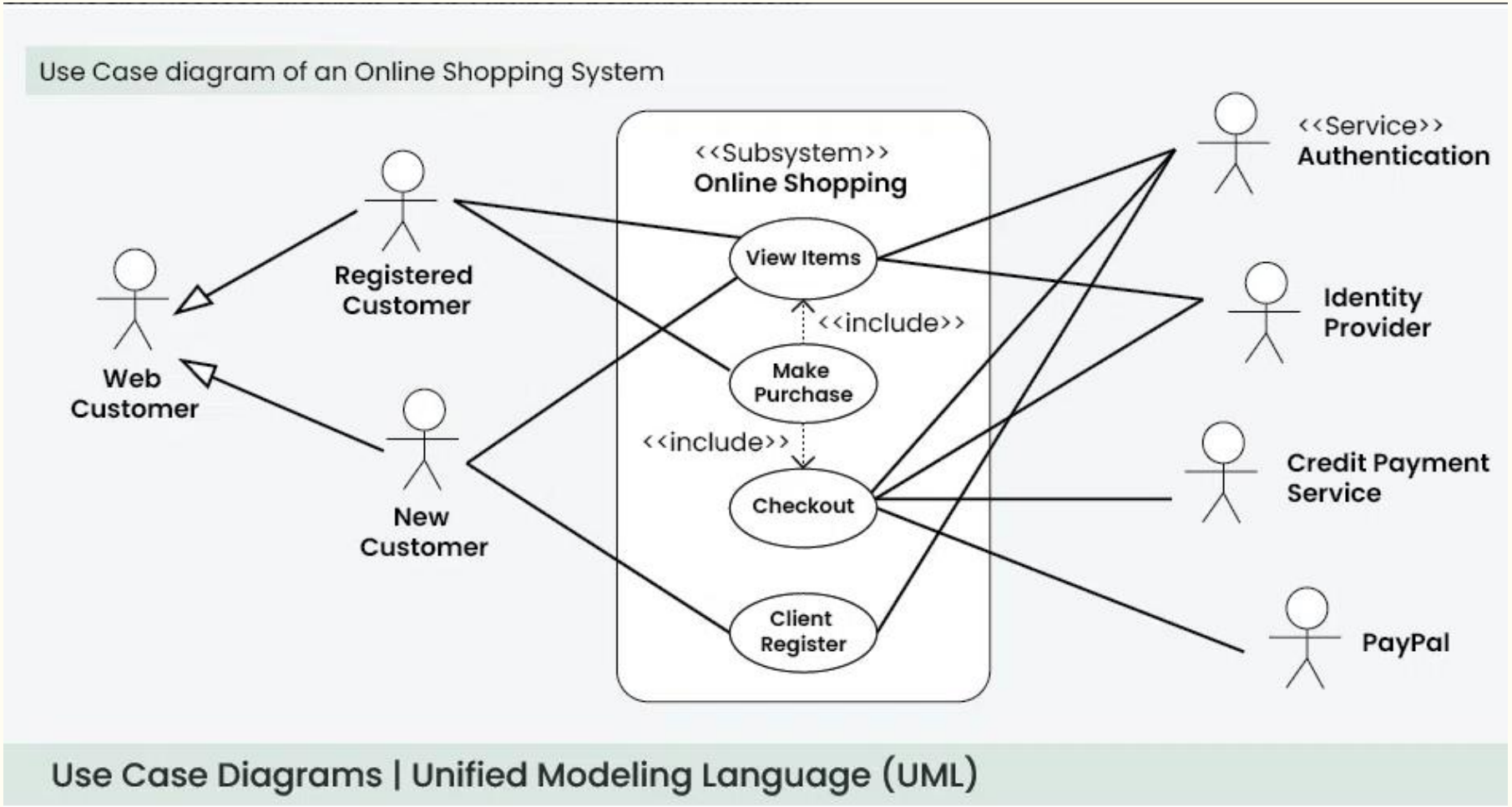
What needs to be analyzed?

- Discover the boundaries of the new system (or software) and how it must interact with its environment within the new problem domain
- Detect and resolve conflicts between (user) requirements
- Negotiate priorities of stakeholders
- Prioritize and triage requirements
- Classify requirements information into various categories and allocate requirements to sub-systems
- Evaluate requirements for desirable qualities

Requirements (and analysis) (OOA)

- The object-oriented paradigm emphasizes modularity and re-usability.
- The goal of an object-oriented approach is to satisfy the "open–closed principle" (achieved through creating a subclass).
- **This model organizes requirements around objects (incorporating behaviors and states) , and uses**
  - Use cases - describing scenarios (of standard domain functions system must provide).
  - Object models – shows class relations

Requirements (and analysis) (OOA)



Requirements (and analysis) (OOA)

- The outcome of requirement analysis is typically a comprehensive document or a set of documents known as the Requirements Specification. A document that serves as a guide for the entire project, ensuring that all stakeholders have a clear understanding of what is to be built.
- The Requirements Specification usually includes:
  - **Functional Requirements**
  - **Non-Functional Requirements**
  - **Technical Requirements**
  - **Assumptions and Constraints**
  - **Acceptance Criteria**
  - **Traceability Matrix**
- Requirements describe **what** the software system should do.

Design

- How the system is to be built on concrete technologies (solution model)
- Applies implementation constraints to the analysis model
- Design explains how the software system will fulfill those requirements.

The design of a software project includes specifying all:

- **risks**
- **technologies**:
  - data design – database schema, data models, etc.
  - software architecture -- everything about the software
  - interface representations -- storyboards
  - algorithmic details -- object diagram (UML)

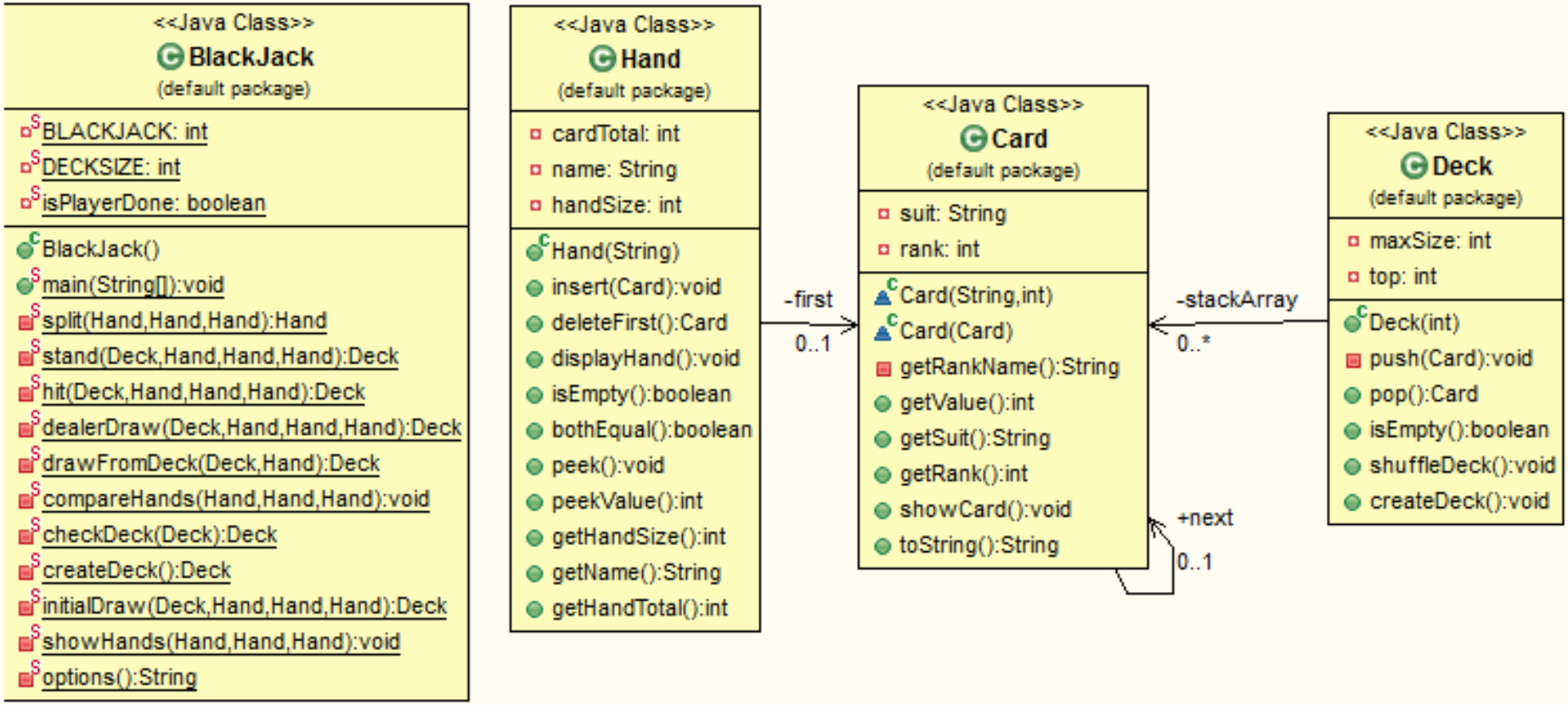
Design (OOD)

- Focuses on software architectures, architectural patterns, and design patterns using OOD principles.
- Identify multiple design options that can fulfill the requirements. Assess each option based on criteria like performance, scalability, maintainability, and cost.

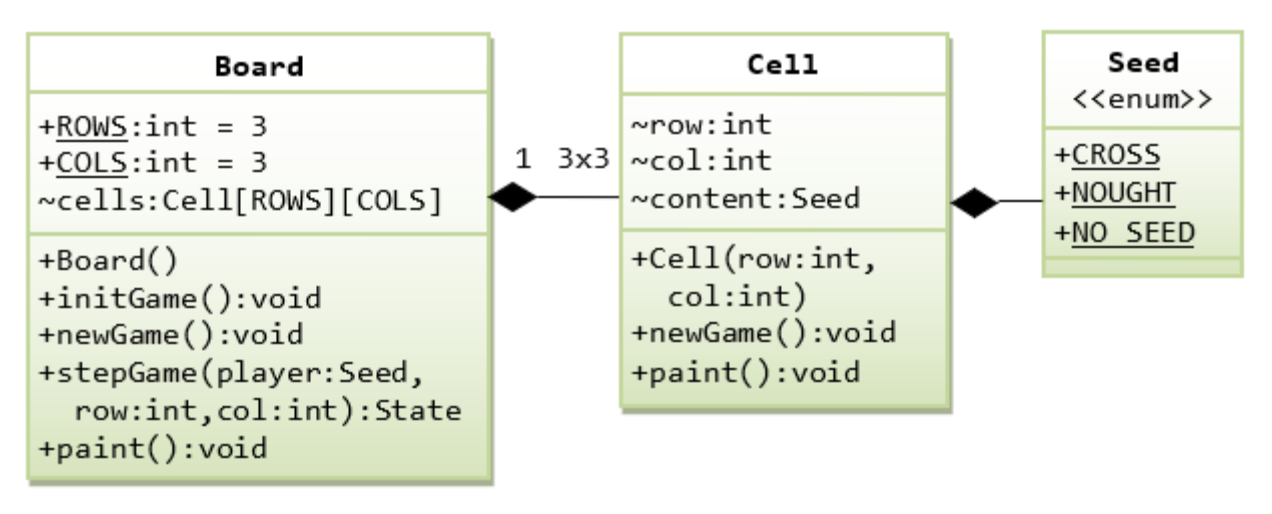
- **Artifacts**: The Design phase produces various artifacts such as diagrams (e.g., UML diagrams), specifications, and design documents.

Design -- Example UML

Classes specify the various object that represent things in the system.
Types of objects: entity (e.g players), control (controlling other objects e.g game master), interface (for GUI)

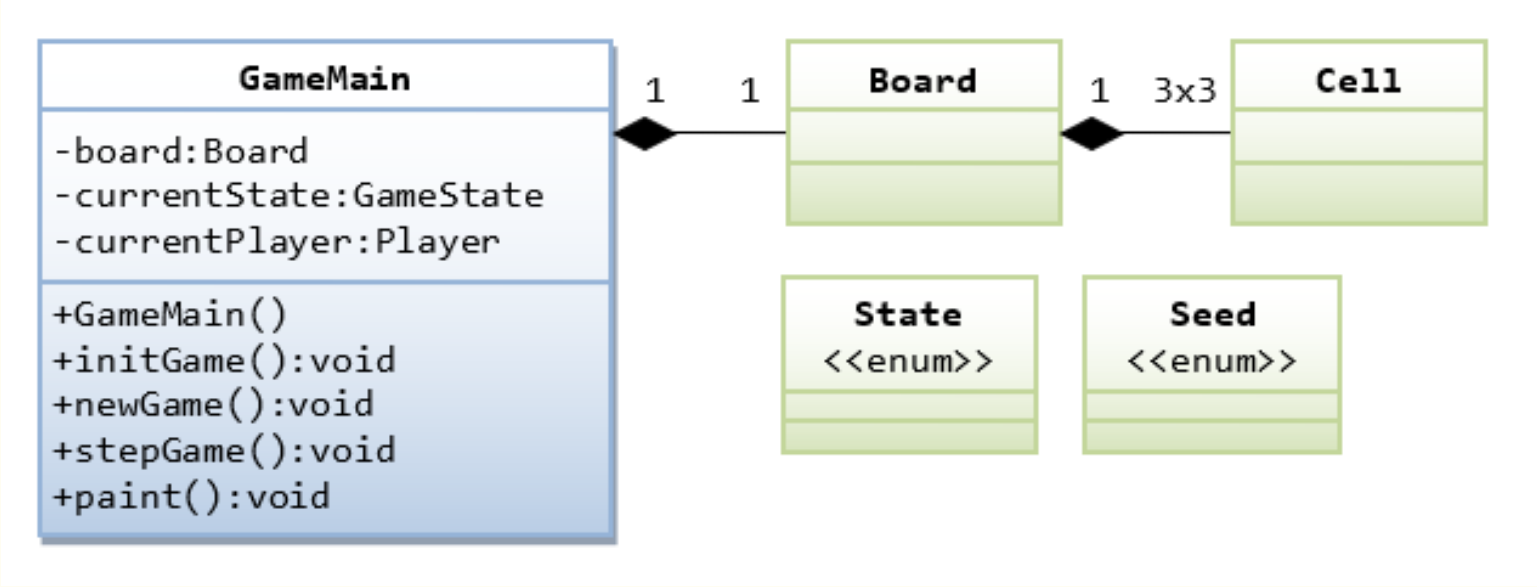


Design -- Example UML (Tic-Tac-Toe)
Class Diagram



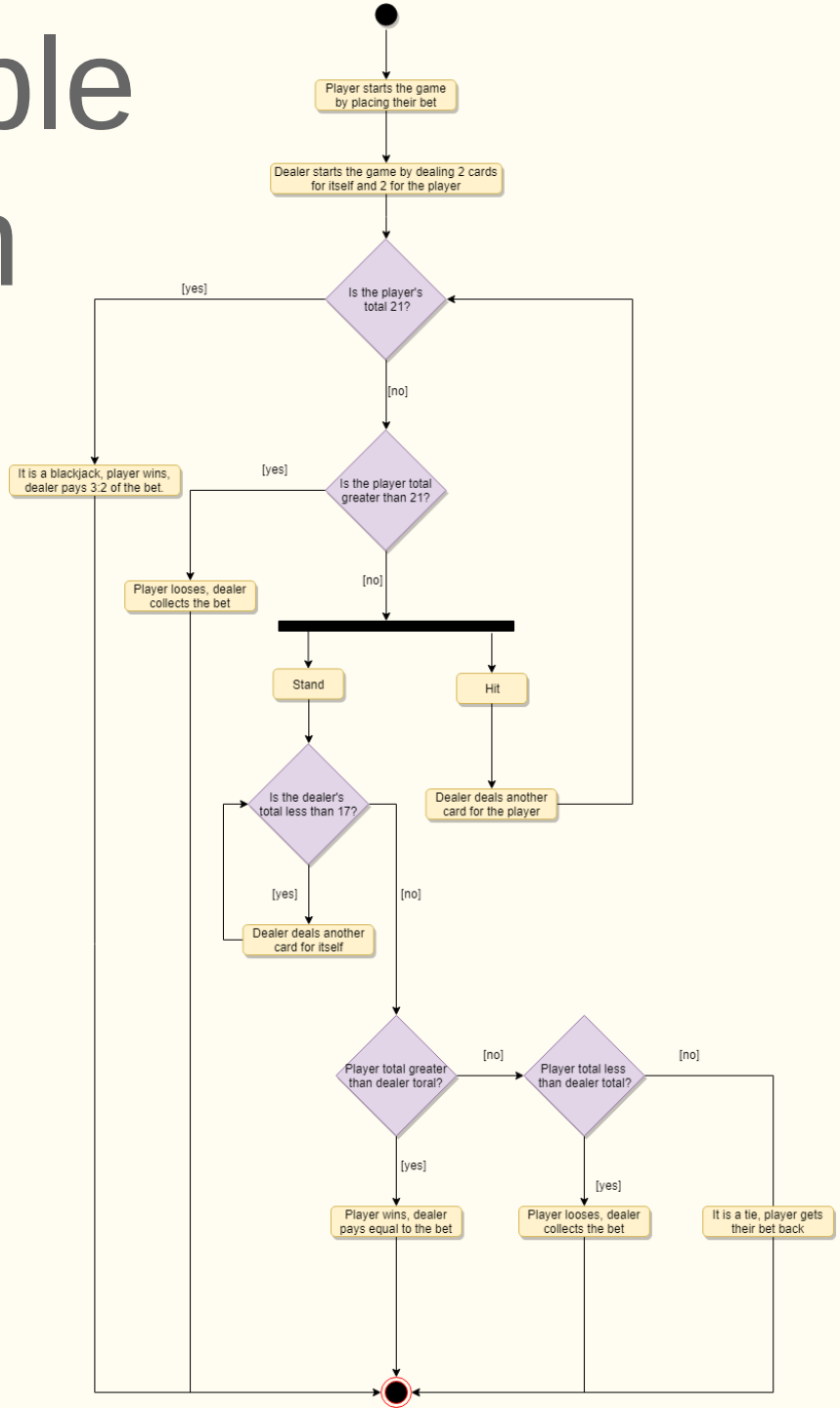
Enums are used to define a set of named constants that are used to represent a specific set of values

Design -- Example UML



Possible values for state: PLAYING, DRAW, CROSS_WON, NOUGHT_WON

Design – Example Activity Diagram



Implementation

The main tasks during implementation include:

- Source code writing
- Compilation
- Testing and debugging
- Database Development
- Documentation
- **Version Control**
- Code Review

- **Artifacts**: Produces source code, compiled executables, and unit test cases.

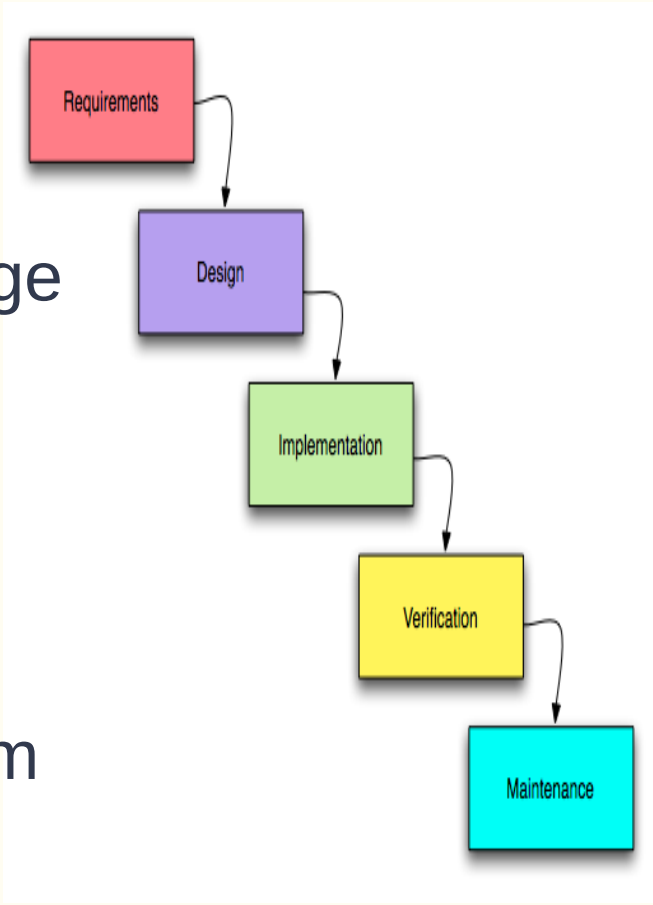
SDLC Frameworks

- Are methodologies used to plan, structure, and control the process of developing an information system.
- These frameworks define the stages and activities required for software creation, from initial conception to deployment and maintenance. Examples include:
  - Waterfall
  - Incremental
  - Spiral
  - Agile

- Choosing a specific process model in software development is essential – it serves as a structured framework that guides the entire software development lifecycle.
- Different process models offer distinct approaches to managing projects, each with advantages and drawbacks.

Waterfall model

- Waterfall is the oldest and easiest of the structured SDLC methodologies.
- There are strict stages and each stage needs to be completed first before going to the next stage. **There is no going back.**
- Each **stage** relies on information from the previous **stage** and has its own project plan.



What does this mean?

- Prone to delays as each **stage** needs to be reviewed and fully signed off before the next **stage** can begin
- Once a stage is completed, problems can’t be fixed until you get to the **maintenance stage**.
- If a **requirement** is wrong or missing, it won’t become apparent until the late **stages** (phases) of the life **cycle**.

Waterfall Strengths

- **Easy** to understand/use
- **Provides structure** to inexperienced staff (newbies)
- **Milestones** are normally well understood
- Requirements are **stable/static**
- Good for management control (plan, staff, track)
- Works well when **quality is more important** than cost or schedule

Waterfall Weaknesses

- **All requirements must be known upfront**
- Deliverables created for each **stage** are considered frozen – **inhibits flexibility**
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of **stages**
- **Integration** is one big bang at the end
- Little opportunity for **customer** to preview the system **(until it may be too late)**

When to use Waterfall?

- 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
- Works well for smaller straightforward projects
- **A first year card game project** 😀

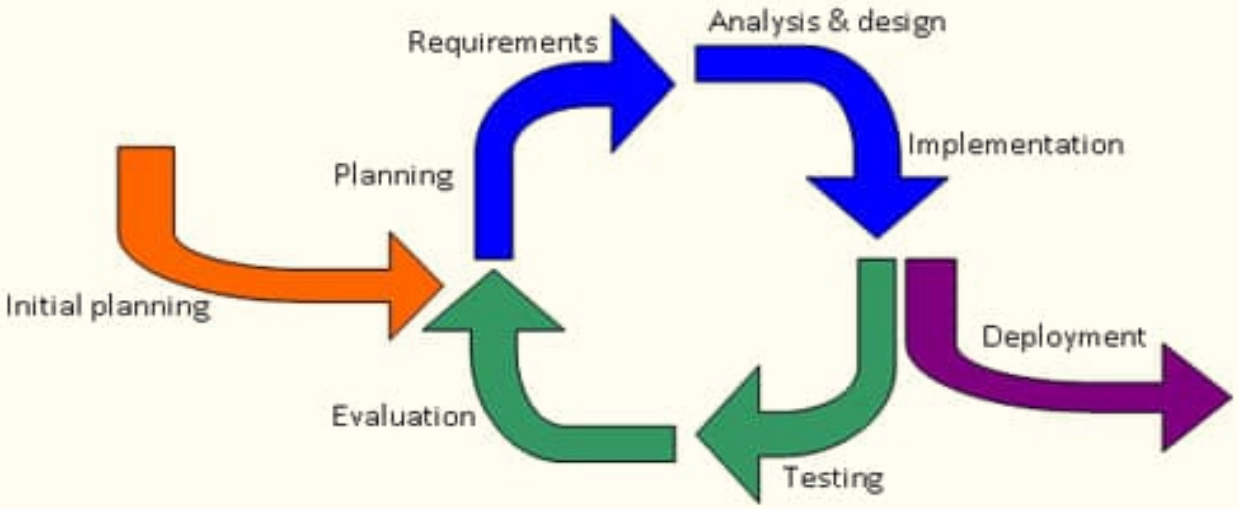
Good Model for a Card Game?

The **waterfall model** will work well for this purpose since the **requirements are very well known with a clear and stable product definition.** Technology involved includes **Java and a pc** or laptop and allows for **portability**. An example of a card game will usually be an **existing** popular one and thus it is **straightforward to implement.**

Other models can also work but you need to motivate why

Incremental model

- Just like iterative Waterfall?
- No, it is more like breaking down tasks into **milestones**
- With each **cycle**, the next **group** of **milestones** are tackled



Model 1: Typical iterative development process

Incremental Strengths

- **Group** and develop **high-risk** or **major functions** first
- **Each cycle delivers a working (part of) system**
- Customer can respond to each build
- Lowers initial delivery cost
- Initial product delivery is faster
- Customers get important functionality early
- Risk of changing requirements is reduced

Incremental Weaknesses

- Requires good planning and design (**abstraction**)
- Some customers would still want to see the full product/program upon the first **cycle**
- Requires early definition of a complete and fully functional system to allow for the definition of **cycles**
- Well-defined module interfaces are required (some will be developed long before others)

When to use Incremental?

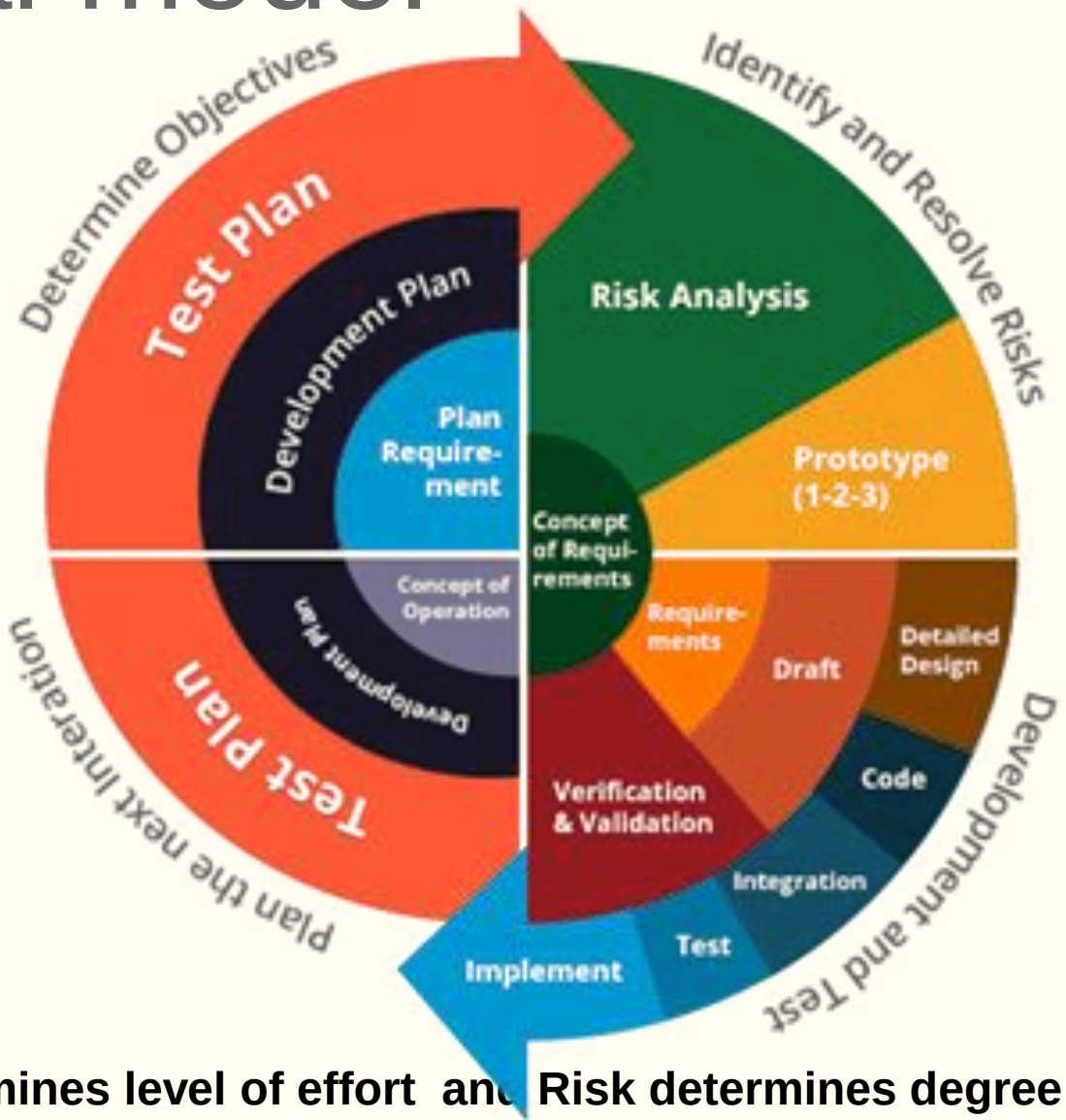
- Customers need early realization of benefits
- Most of the requirements are known upfront but are expected to evolve over time
- A need to get **basic functionality** to the market early
- Remember: Difficult milestones first (for realistic goals)
- On projects which have lengthy development schedules
- On a project with new technology (impatience)
- **A first year card game project** 😀

Iterative vs. Incremental?

- **Iterative** model is an approach where the focus is to rapidly develop a working **prototype** first but often not good quality or incomplete. e.g. write parts of chapter 1,2,3,4,5 and significantly change them during each (next) **cycle**, often making it better

- Whereas features are added during each **cycle** of the **incremental** model. e.g. write chapter 1 then 2 then 3.

Spiral model



Risk determines level of effort and Risk determines degree of details

Spiral Strengths

- Provides **early indication of insurmountable risks**
- Users see the system early because of **rapid prototyping tools**
- **Critical high-risk functions are developed first**
- The design does **not** have to be perfect
- **Users** can be closely tied to all **lifecycle** steps
- **Early and frequent feedback** from users
- Cumulative costs assessed frequently

(Reminds me of a complex version of incremental model)

Spiral Weaknesses

- Time spent evaluating risks too long/expensive for **small or low-risk projects**
- Therefore, does **not** work well for smaller projects
- The model is complex
- Cost: Risk analysis requires highly specific expertise.

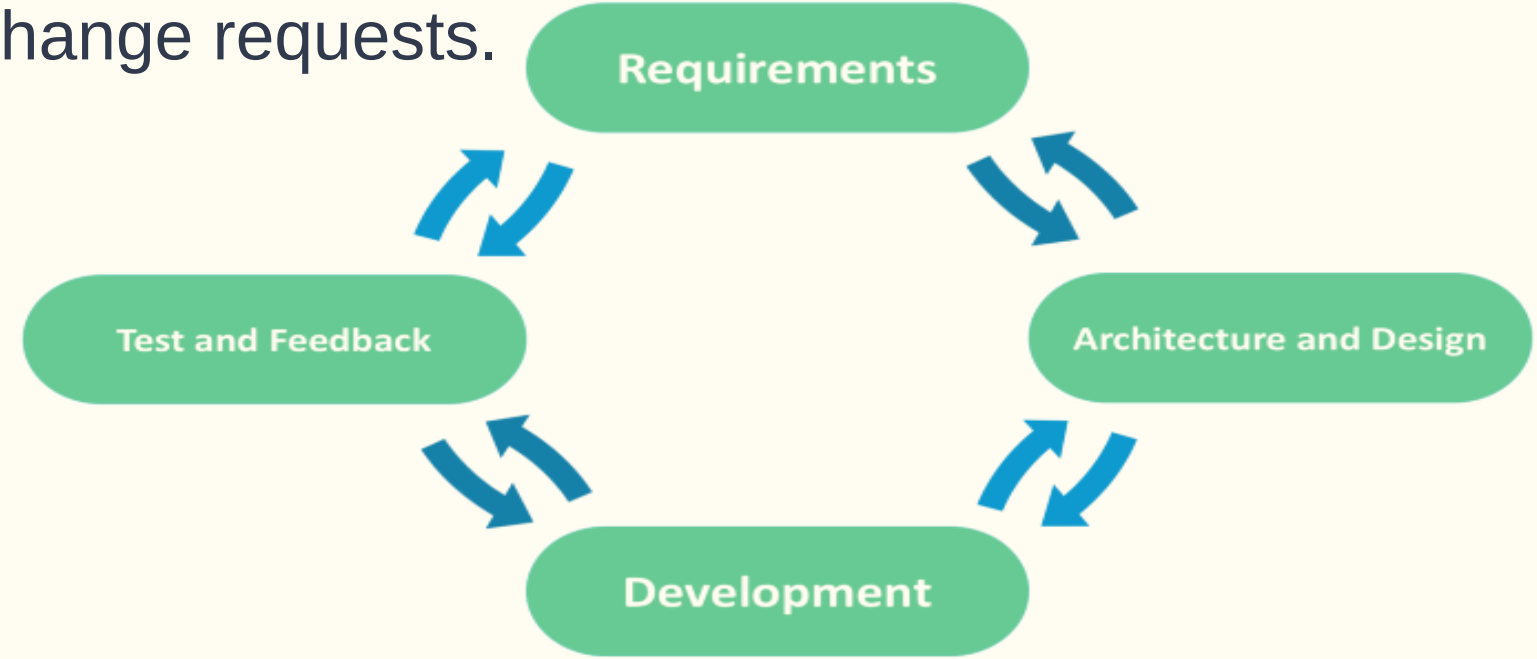
When to use Spiral?

- When rapid creation of a prototype is appropriate
- When costs and risk evaluation is important
- For **medium to high-risk** projects
- Even more control between **cycles** are required than incremental -- shorter increments

The US military has adopted the spiral model for its Future Combat Systems program.

Agile model

- Agile development was proposed to tackle issues with iterative waterfall, which includes handling customer change requests during project development and the high cost and time required to incorporate these changes. It is designed to help projects adapt quickly to change requests.



Agile models

The Agile Model refers to a group of development processes.

- **Scrum**
- Rapid Application Development (RAD)
- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Feature Driven Development (FDD)
- Crystal Clear
- Dynamic Software Development Method (DSDM)
- Rational Unified Process (RUP)

Scrum

- Scrum, like all of the agile models, is both **iterative** and **incremental**.
- Scrum encourages teams to learn through **experiences**, **self-organizing** while working on a problem, and reflect on their wins and losses to continuously improve.

Scrum

- Prioritized wish list called a **product backlog** -- a placeholder for all the future development
- During **planning stage**, decides how to implement the top chunk called a **sprint backlog**
- The team is given a certain amount of time, called a **sprint**, to complete its work
- The team meets **each day** to assess the progress
- **2-4 weeks** work is ready to hand to a customer
- The sprint ends with a sprint review
- The next sprint then begins (based on customer input)

When to use Agile?

- **Dynamic** users -- continuous changes
- Can afford to have team and client **meetings often**
- When many changes are sure to be implemented because of the **many iterations**
- **Unlike** the **Waterfall model**, it requires only initial planning to start a project

- **A crazy first year card game project** 😀

Investigate the rest yourself

- Rapid Application Development (RAD)
- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Feature Driven Development (FDD)
- Crystal Clear
- Dynamic Software Development Method (DSDM)
- Rational Unify Process (RUP)

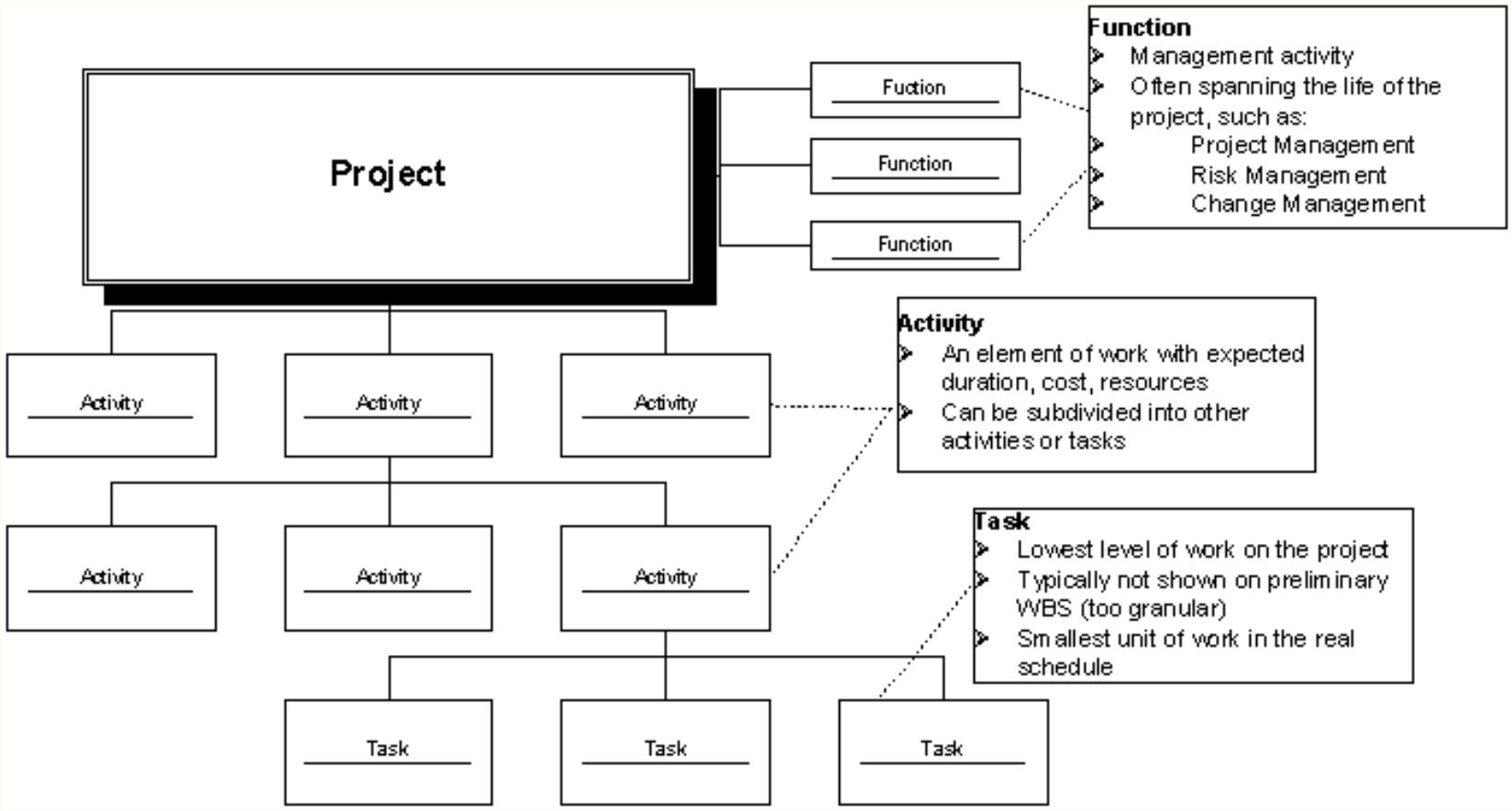
Software development projects

What is a Project?

- A project is a one-off, temporary activity with a clear start and end; it has full or part-time resources assigned to it; it has temporary management; and it sets out to deliver something: an IT system, a new product, or whatever.
- **Four characteristics of projects:**
  - finite time
  - people assigned
  - clear roles and responsibilities
  - things to deliver
- Have you ever had this feeling about a project?
  - not enough time, too few people, people not sure what they should be doing, and too much to do
- What causes this feeling of sometimes overwhelming pressure?
  - **In simple terms, a lack of planning.**

Project Elements

- A Project is divided into functions, activities, tasks



Project management terminologies

The two most important terms:

- *Milestones* are the end-point of a process activity.
- *Deliverables* are project results delivered to customers.

The 5 phases of project management

- Initiation
- Planning
- Execution
- Monitoring and management
- Closing and review

What is project planning?

- Project planning is a discipline addressing how to complete a project in a certain timeframe, usually with defined stages and designated resources. One view of project planning divides the activity into these steps:
  - setting measurable objectives
  - identifying deliverables
  - scheduling
  - planning tasks

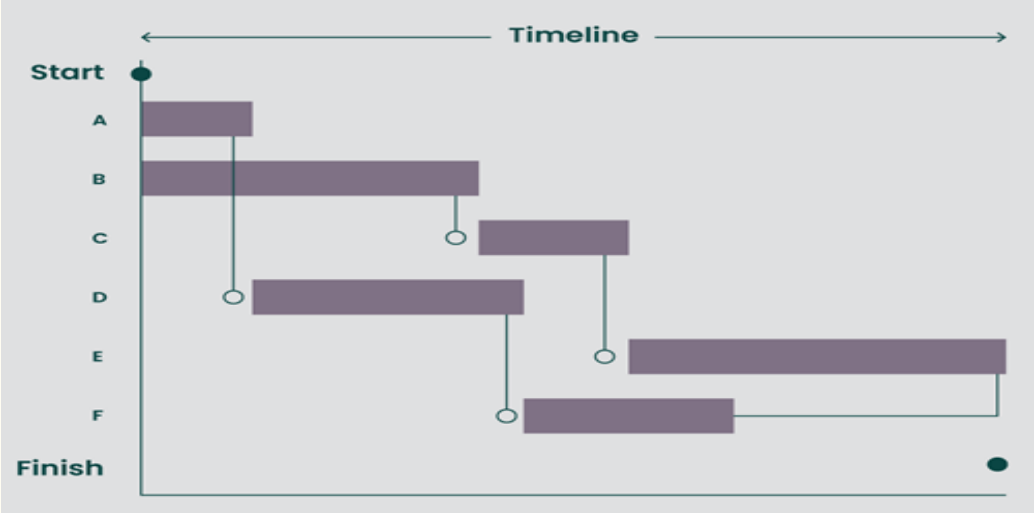
Planning, Estimating, Scheduling

- What’s the difference?
- Plan: Identify activities. No specific start and end dates.
- Estimating: Determining the size & duration of activities.
- Schedule: Adds specific start and end dates, relationships, and resources.

What is project planning?

- Supporting plans may encompass human resources, communication methods and risk management.
- Tools used for the scheduling parts of a plan include Gantt charts and PERT charts.

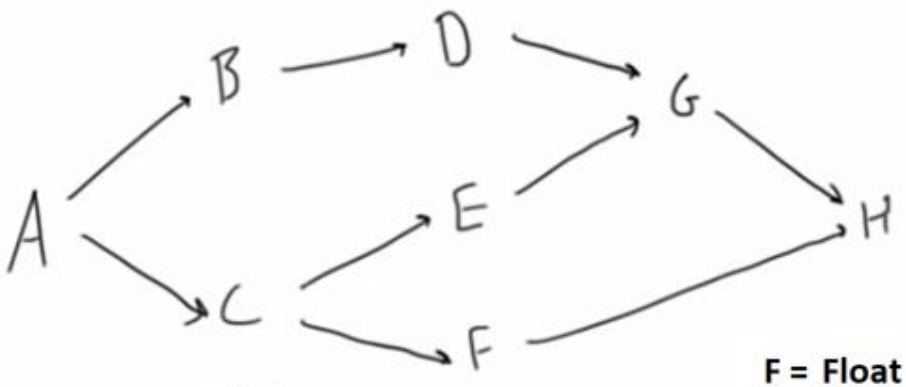
Gantt chart example



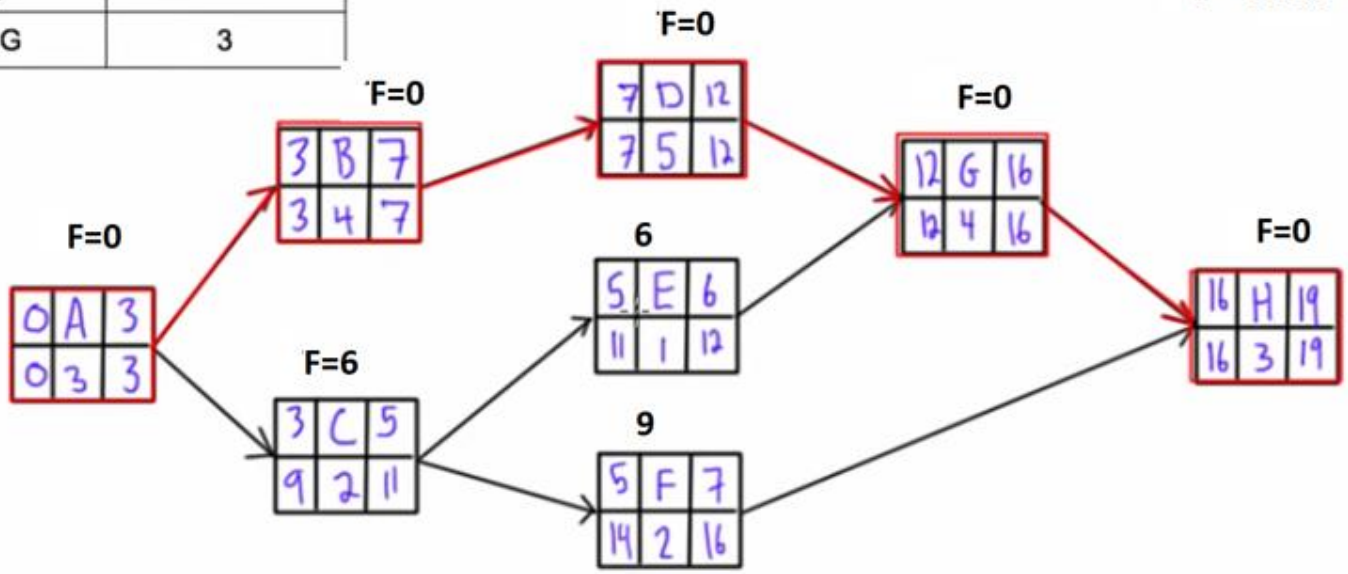
Critical path analysis

Using program evaluation and review technique (PERT)

Activity	Predecessor	Duration (days)
A	-	3
B	A	4
C	A	2
D	B	5
E	C	1
F	C	2
G	D,E	4
H	F,G	3



F = Float =LS-ES



ES	Ac	EF
LS	Du	LF

ES and EF are the Earliest Start and Earliest Finish (calculated using forward pass)

LS and LF are the latest Start and Latest Finish times (calculated using bakward pass)

Why is project planning important?

- Project planning is important at every phase of a project. It lays out the basics of a project, including the following:
  - scope
  - objectives
  - goals
  - schedule

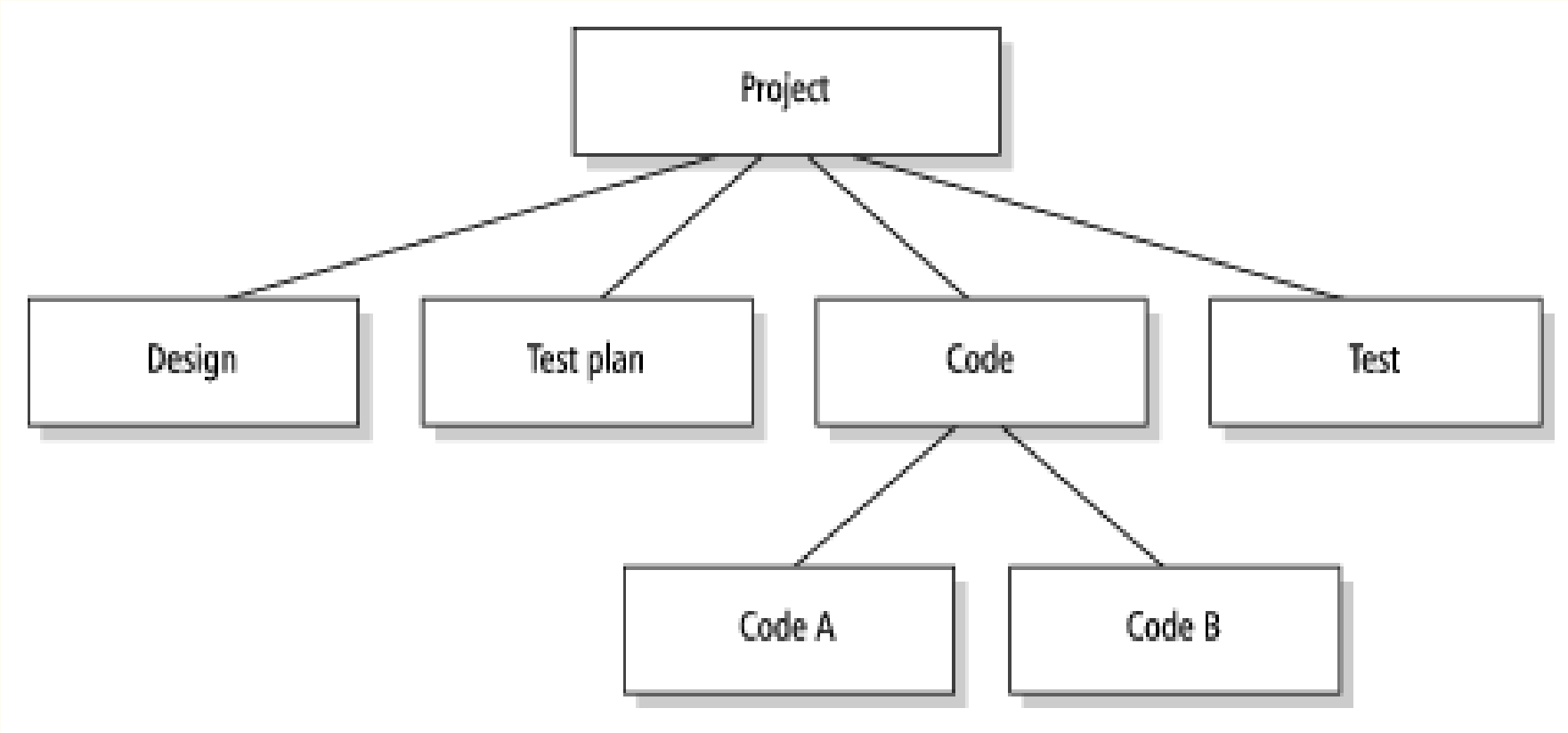
What are the components of a project plan?

- The three major parts of a project plan are the scope, budget and timeline. They involve the following aspects:
  - Scope
  - Budget
  - Timeline

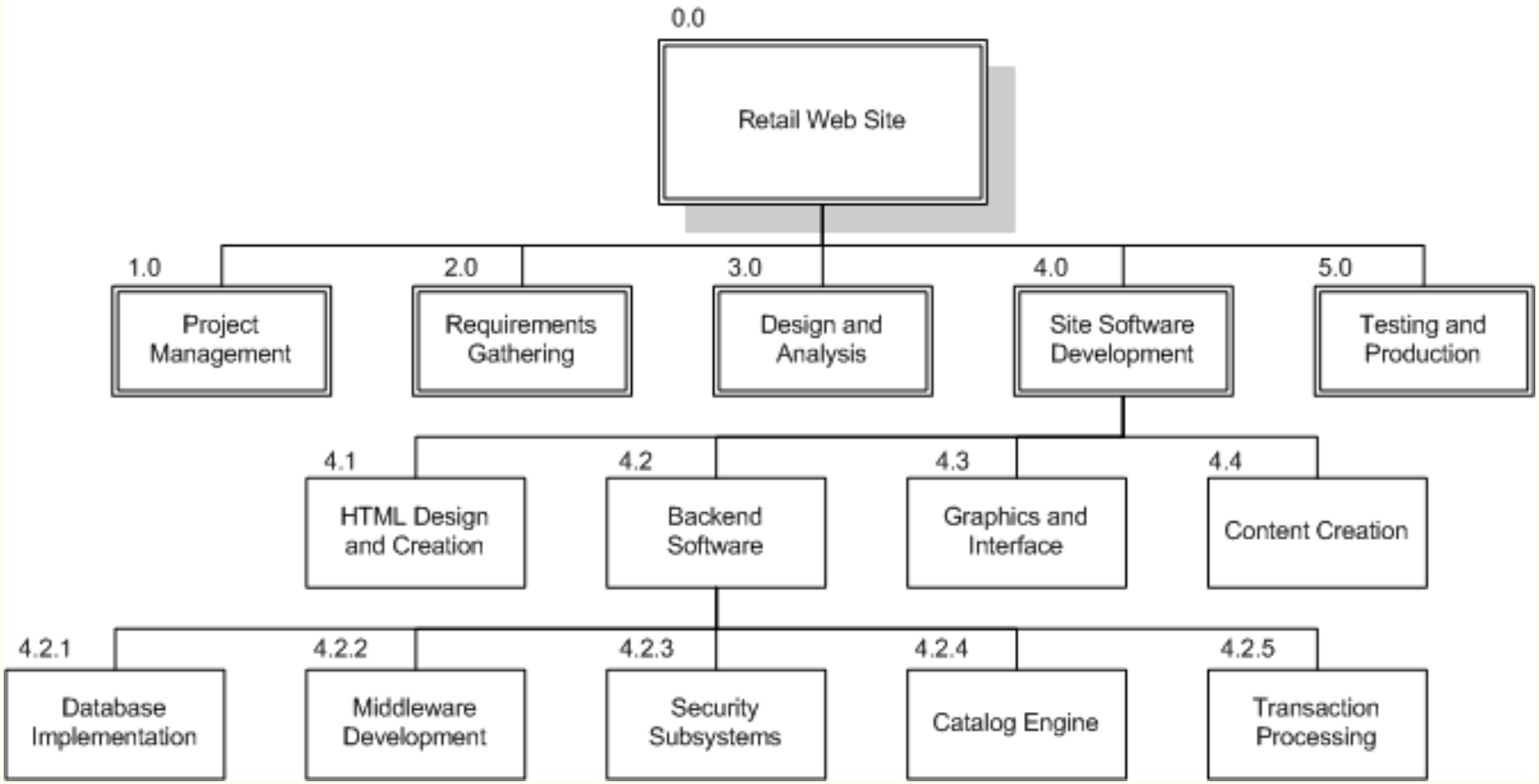
How do you create a project plan?

- Define stakeholders
- Define roles.
- Set goals.
- Prioritize tasks.
- Create a schedule.
- Assess risks.
- Communicate.
- Reassess.
- Final evaluation.

Project estimation: work breakdown structure



WBS Chart Example - Detail



Project scheduling example

Task	Effort (person-days)	Duration (days)	Dependencies
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2, T4 (M3)
T6	10	5	T1, T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3, T6 (M5)
T10	20	15	T7, T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)