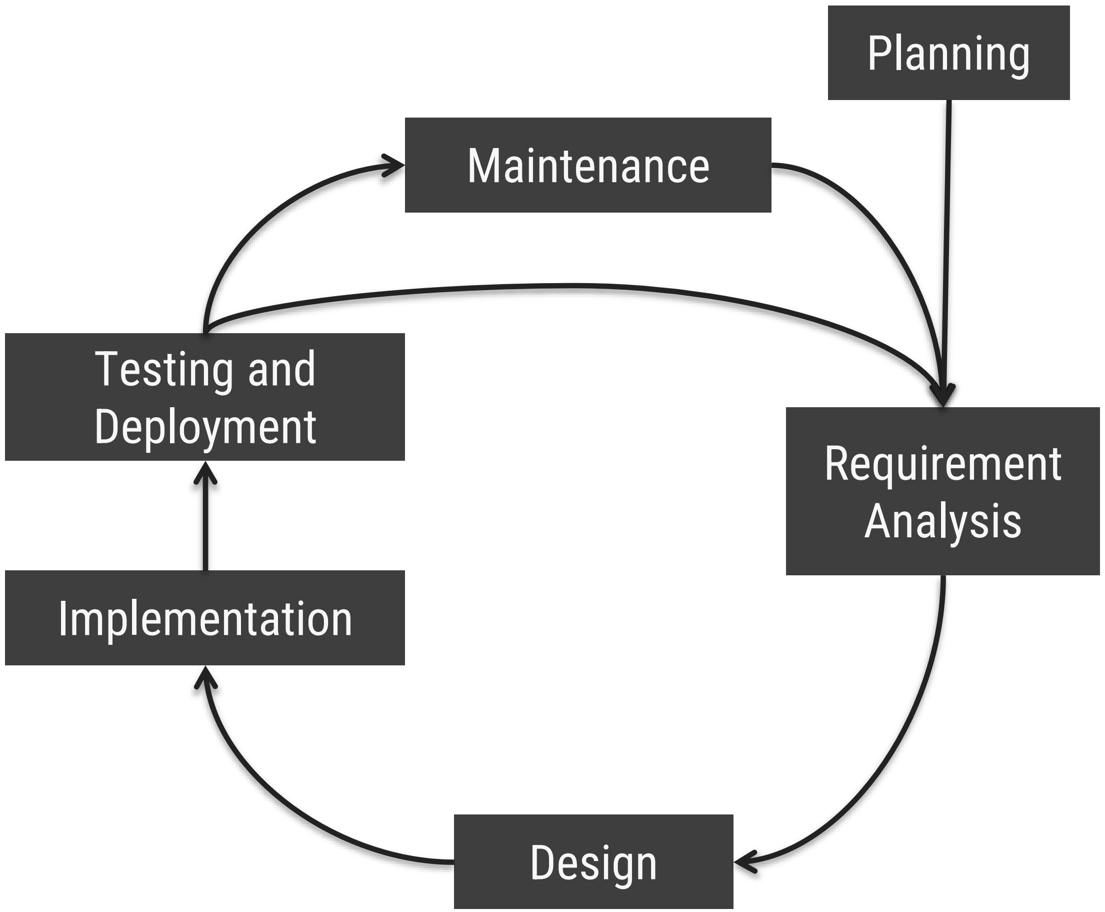
SOFTWARE DEVELOPMENT PROCESS

LECTURE 2:
SEQUENTIAL AND
ITERATIVE AND INCREMENTAL MODELS

Asst. Prof. Dr. ISARA ANANTAVRASILP

SOFTWARE DEVELOPMENT **PROCESS**



How do we **schedule** the phases?

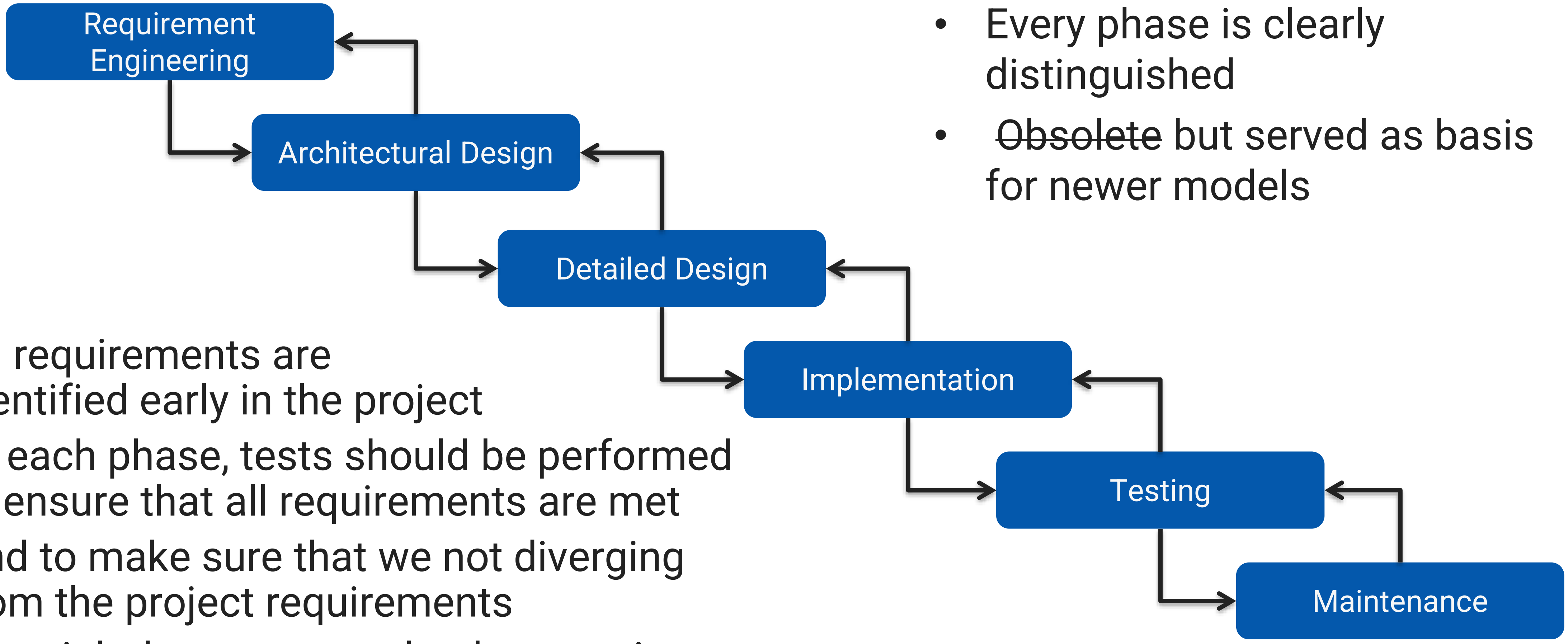
SOFTWARE DEVELOPMENT PROCESS GOALS

- The whole development process must be controlled
  - A software project is usually large and a lot of people are involved
  - Development time could also be very long
- Software process acts as a guideline to control the software development activities
- Example of different models:
  - Sequential Models
  - Iterative and Incremental Models
  - Agile Processes
  - Open Source Process

SEQUENTIAL **MODELS**

- Sequential models execute each phase in sequence
- We will not proceed to the next phase unless the current one is done
- Clear and precise procedure
- Easy to manage and understand
- Also, easy to quote prices and cash-out

WATERFALL **MODEL**

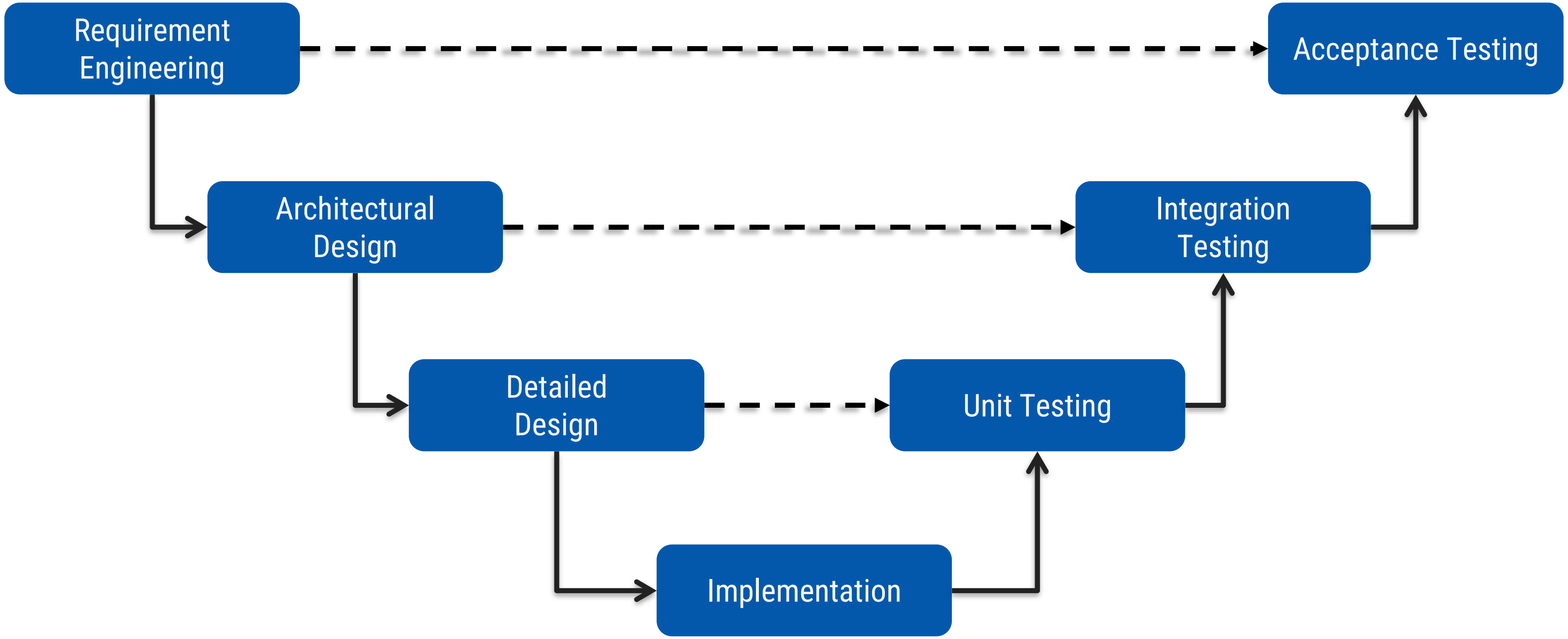


- Sequential
- Every phase is clearly distinguished
- ~~Obsolete~~ but served as basis for newer models

- All requirements are identified early in the project
- At each phase, tests should be performed to ensure that all requirements are met
- And to make sure that we not diverging from the project requirements
- We might have to move back to previous phase

V-Model

- Sequential
- V-Model shows how a software product is validated
- It relates different kinds of testing to corresponding design phases
- Test plans are developed after each phase on the left is done



SEQUENTIAL MODELS **PROS**

- **Simple and easy to use**: Phases are well-defined and executed sequentially
- **Easy to manage**: The model is rigid. Each phase has specific deliverables and review process
- **Facilitates allocation of resources**: Different phases require different personnel with different skills
- Works well for project with **requirements that are well-understood**
  - Short and clear projects are ideal

SEQUENTIAL MODELS **CONS**

- **Requirements must be known beforehand**: Does not work with projects with hazy knowledge
- **No feedback** from stakeholders until testing phases
- Problems with projects might not be discovered until testing phases
- **Lack of parallelism**: Second phase cannot be executed along with the first phase
- **Inefficient use of resource**: Due to lack of parallelism, team members (e.g., developers) must wait until other teams (e.g., designers) finish their work

ITERATIVE AND **INCREMENTAL MODELS**

- Start with small portions of a software project
- Repeatedly add portions into the projects
- Iterative vs. incremental
  - **Incremental** means "**add onto something**"
    - Incremental development helps you improve your process
  - **Iterative** means "**re-do**"
    - Iterative development helps you improve your product

Image: https://www.etsy.com/hk-en/listing/484837199/mc-escher-print-escher-art-waterfall

SPIRAL **MODEL**

- Proposed by Boehm in 1988
- One of the earliest iterative and incremental models
- It is a risk-driven approach combined with waterfall model
- Risk management is included in each iteration
- Number of iterations (cycles) depends on the size of the project

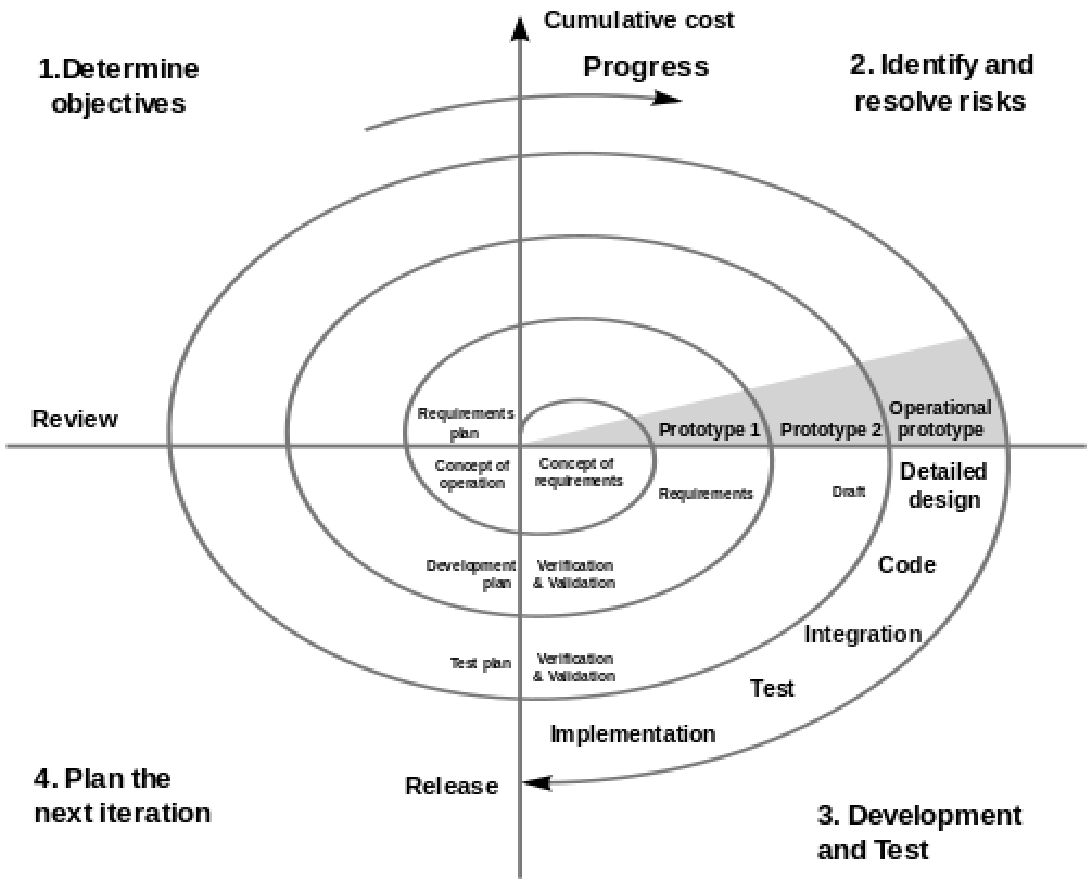


Image: http://en.wikipedia.org/wiki/Spiral_model

SPIRAL MODEL **ACTIVITIES**

1. Determine objectives
   - Requirement elicitations
   - Feasibility studies
2. Identify and resolve risks
   - Risk analysis (cost overruns, wrong calculations, etc.)
   - Evaluate alternatives
   - Planning risk mitigation strategy
   - Develop series of prototypes to identify risks
   - Use a waterfall model for each prototype development
   - Customer can abort the project if the risks are too great

SPIRAL MODEL **ACTIVITIES**

3. Development and test
   - Implementation
   - Conduct testing
4. Evaluation
   - Customer evaluates the product

PROTOTYPING

- Risk-management technique
- A partial implementation of the target product
- Can be used for:
  - Identifying ***risky*** parts of the project
  - Determine the idea about customer's requirements
  - Gather look-and-feel in GUI
- However, prototypes could also be expensive and complex
- We shall build a prototype if the development cost is low and the yielded value is high

TYPES OF PROTOTYPES

- **Illustrative Prototype**
  - Develop the user interface with a set of storyboards
  - Implement them on a napkin or with a user interface builder
  - Good for (early) client discussion
- **Functional Prototype**
  - Implement and deliver an operational system with minimum functionality
  - Then add more functionality
- **Exploratory Prototype ("*Hack*")**
  - Implement part of the system to learn more about the requirements.
  - Good for paradigm breaks.

PROTOTYPING

- Risk-management technique
- A partial implementation of the target product
- Can be used for:
  - Identifying ***risky*** parts of the project
  - Determine the idea about customer’s requirements
  - Gather look-and-feel in GUI
- However, prototypes could also be expensive and complex
- We shall build a prototype if the development cost is low and the yielded value is high

SPIRAL MODEL **PROS AND CONS**

- Advantages:
  - Fast development
  - Risks are managed throughout the process
  - Software evolves as the project progress
    - Good for large-scale project
  - Planning is integrated into the process
    - Planning is included in each cycle, keeping the process on track
- Disadvantages:
  - Risk analysis requires an expert
  - Risk management is expensive and may not be necessary for small projects
  - Cannot handle ***changes*** very well
  - Tedious documentation

SOFTWARE DEVELOPMENT RISKS

- **Software Development Risks** refer to uncertain future events that may cause loss to the software project
- Thus, identifying risks is very important
- Software development risks include
  - Schedule risks
  - Budget risks
  - Operational risks
  - Technical risks

SCHEDULE RISKS

- **Schedule risks** refers to time-related risks that may cause delay to the project
- That is, the project is not completed in the estimated time
- Delayed development could cause financial (and reputational) loss to both the customers and the developers
  - Missing target date: missing opportunity
  - Fine (as in being fined)

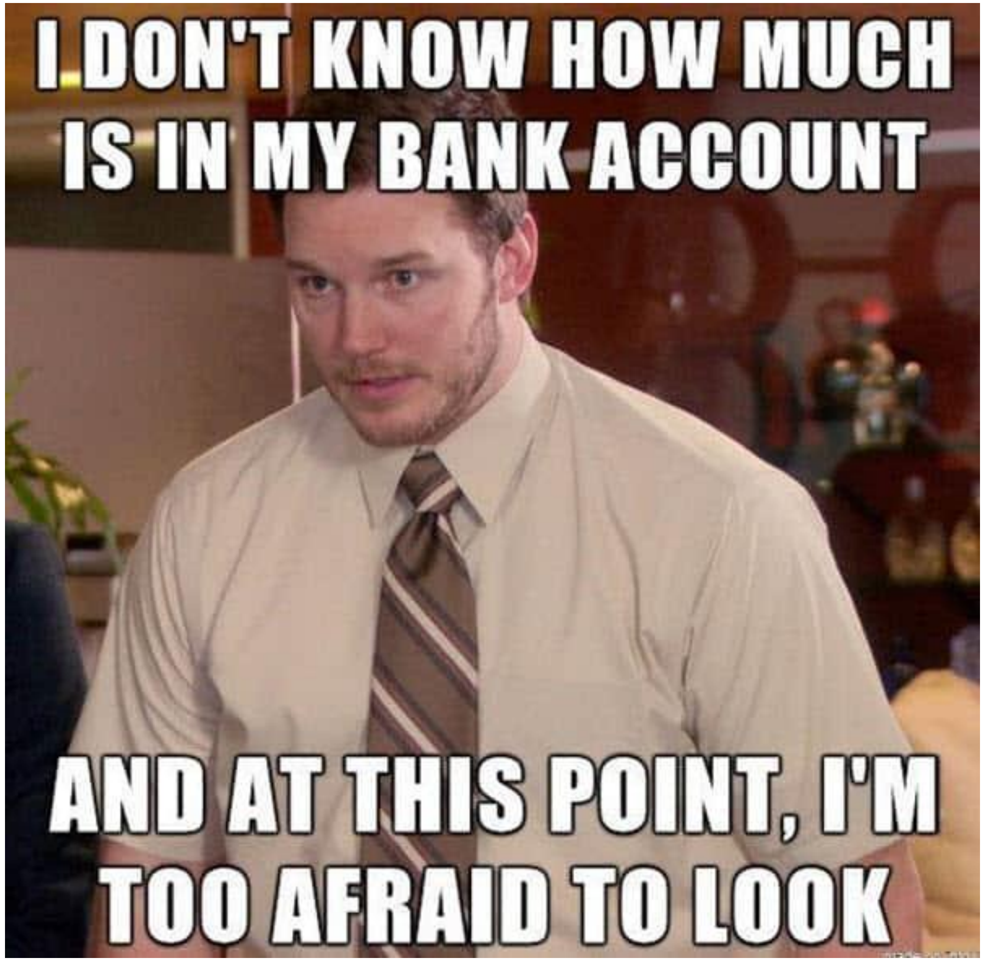
CAUSE OF SCHEDULE RISKS

- Improper effort estimation
- Inadequate requirement elicitation
- Features and their completion are not well-defined
- Emergent requirements / change and feature requests
- Poor resource allocation and tracking
    - Under-staffed
    - Staff quit the project / company
    - Team members are being allocated to other projects
    - Developers’ skill do not match the tasks
    - Employed tools (both hardware and software) are not suitable to the tasks

BUDGET RISKS

- **Budget risks** relates directly to monetary issues
- That is, the actual development cost is more that the estimated one
- Proper budget management is required throughout all development phases
- One must have clear understanding the costs that are need to bring the product to the market or using it (not just to develop)
- Financial issues must be identified and resolved early

CAUSE OF BUDGET RISKS


I DON'T KNOW HOW MUCH IS IN MY BANK ACCOUNT
AND AT THIS POINT, I'M TOO AFRAID TO LOOK

- Inaccurate cost estimation or budget calculation
- Emergent requirements / project expansions or product scaling
- Mismanagement in budget handling
- Improper budget tracking
- Do not have financial backups for budget overruns
- Counting on future income, payments or fundings

OPERATIONAL RISKS

- **Operational risks** are risks related to day-to-day activities during the project
- They are issues that cause inadequate development process
- This include, management, communication, product delivery, etc.
- Such issues are usually overlooked in the requirement elicitation and design phases

CAUSE OF OPERATIONAL RISKS

- Improper tasks management and planning
- Unclear roles and task assignments
- Lack of communication and collaboration
- Conflict between tasks and/or among employees
- Insufficient resources (including budget, human, tools)
- Bureaucratic necessities such as excessive paper works, approvals and meetings
- Lack of experiences or necessary skills

TECHNICAL RISKS

- **Technical risks** refer to functions or performance of the product
- The software does not function properly, or its non-functional requirements are not met
- Such risks are higher when developing new technologies or products or using unfamiliar tools or methods

CAUSE OF TECHNICAL RISKS

- Lack of knowledge or experience in the application domain
- Lack of knowledge or experience in the employed technology
- Employing outdated technology
- Unable to deliver all functionalities with current tech-stack
- Emergent requirements or frequent change requests
- Unseen or underestimated difficulties