Process Models

Week 3 – Session 1

Y. Raghu Reddy

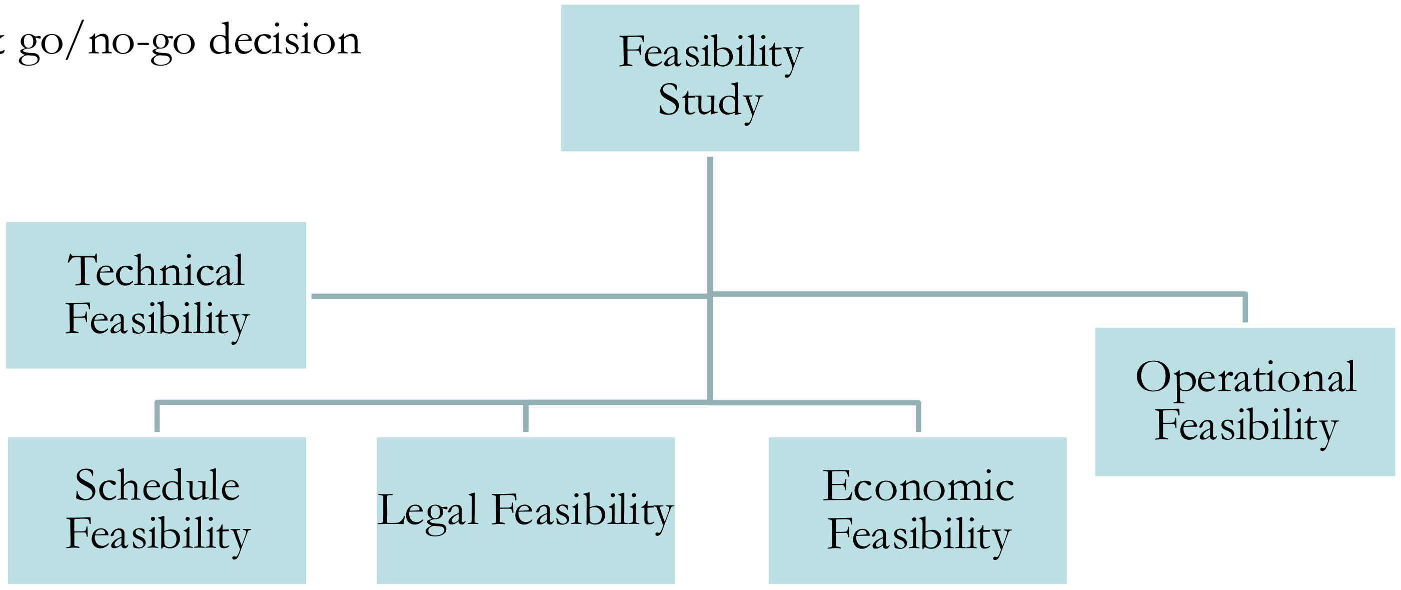
Software Engineering Research Center
IIIT Hyderabad, India

SDLC - Feasibility Study

Aim of feasibility study: Determine whether the proposed system is viable and worth pursuing

Key Activities:

- Technical feasibility
- Economic (cost–benefit) analysis
- Schedule feasibility & preliminary estimation
- Operational feasibility
- Legal/compliance & risk assessment
- Recommendation & go/no-go decision



Requirements Engineering

Elicit, analyze, specify, validate, and manage stakeholder needs so the system built meets intended purpose and constraints

Consists of distinct activities:

- **requirements gathering and analysis**
- **requirements specification**
- verification & validation
- requirements management & change control

Design

Transform validated requirements into a robust, maintainable, and implementable architecture and detailed designs that guide development and testing

High-level design:

- decompose the system into modules/components
- represent invocation relationships among the modules/components
- Specify constraints

Detailed design (Low-level design):

- data structures, class diagrams, interfaces and algorithms for each module are designed;
- API contracts, sequence/state diagrams, and database schemas

Implementation – Building the system

Convert design modules into codified components/classes following coding standards/conventions

Coding Standards:

- Enforce team-specific coding standards (naming, layout, function size) for all programmers.
- Consistency boosts readability, maintainability, and good practices across the codebase.

- Microsoft .NET/C# Guidelines — **https://learn.microsoft.com/dotnet/csharp/fundamentals/coding-style/**
- Python PEP 8 — **https://peps.python.org/pep-0008/**
- Oracle Java Code Conventions / Google Java Style — **https://google.github.io/styleguide/javaguide.html**
- CERT Secure Coding (C/C++/Java) — **https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards**

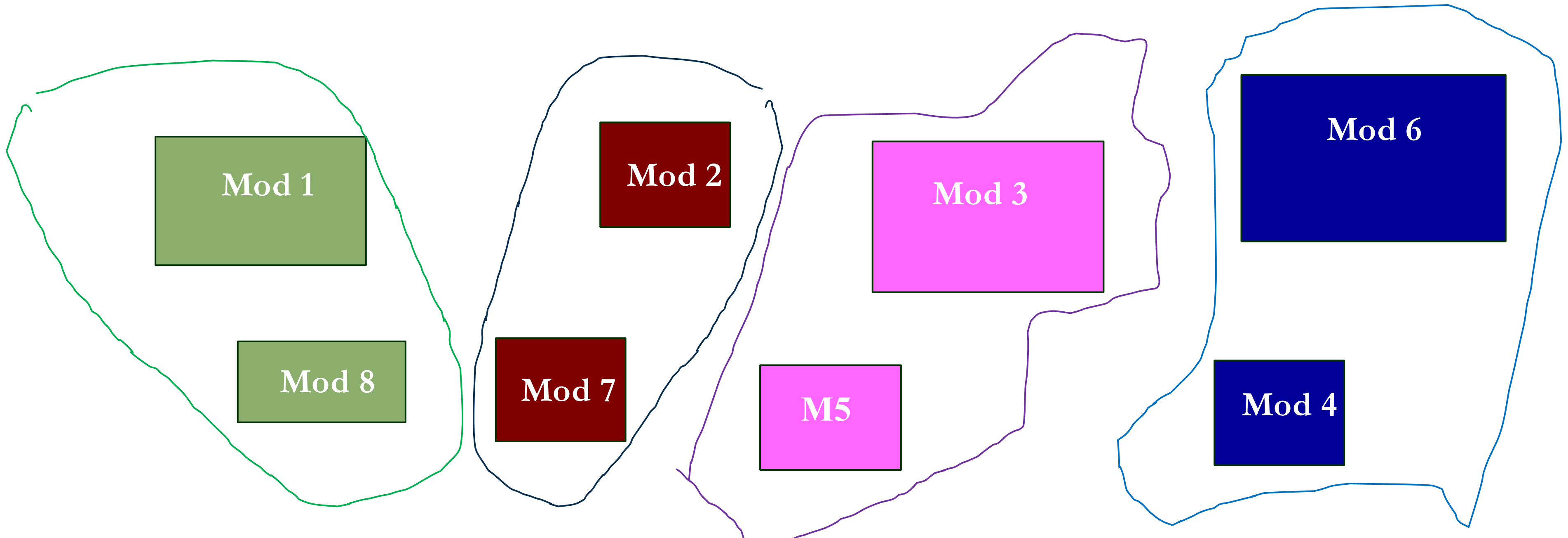
The end product of implementation phase:

- transform design into code
- a set of program modules that can be tested individually (**unit testing**)

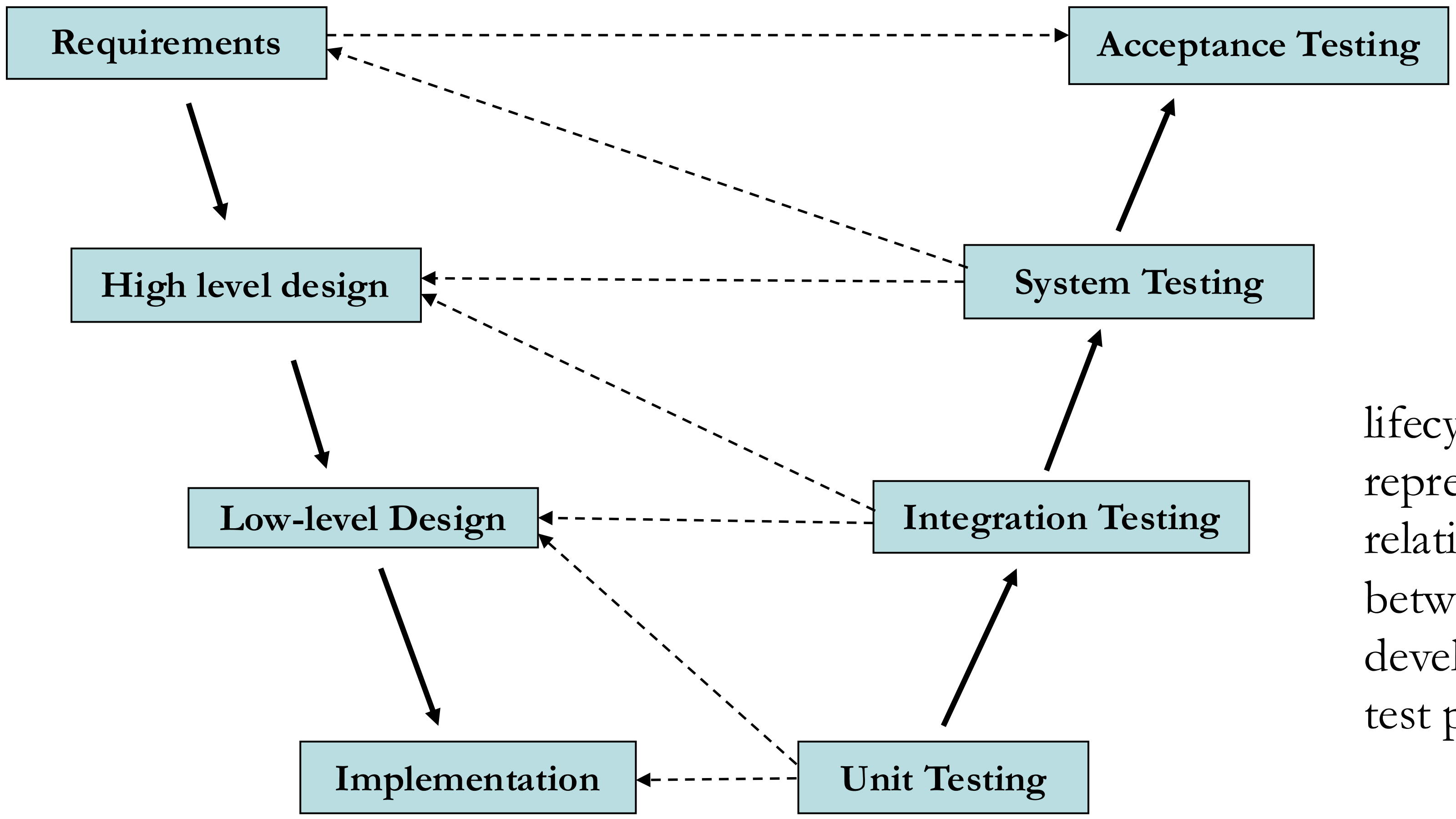
Integration and Testing

- Different modules are integrated in a planned manner:
  - modules are almost never integrated in one shot.
  - Normally integration is carried out through a number of steps.

- During each integration step,
  - the integrated system is tested.



V- Model



lifecycle view representing relationships between development and test phases

Maintenenace

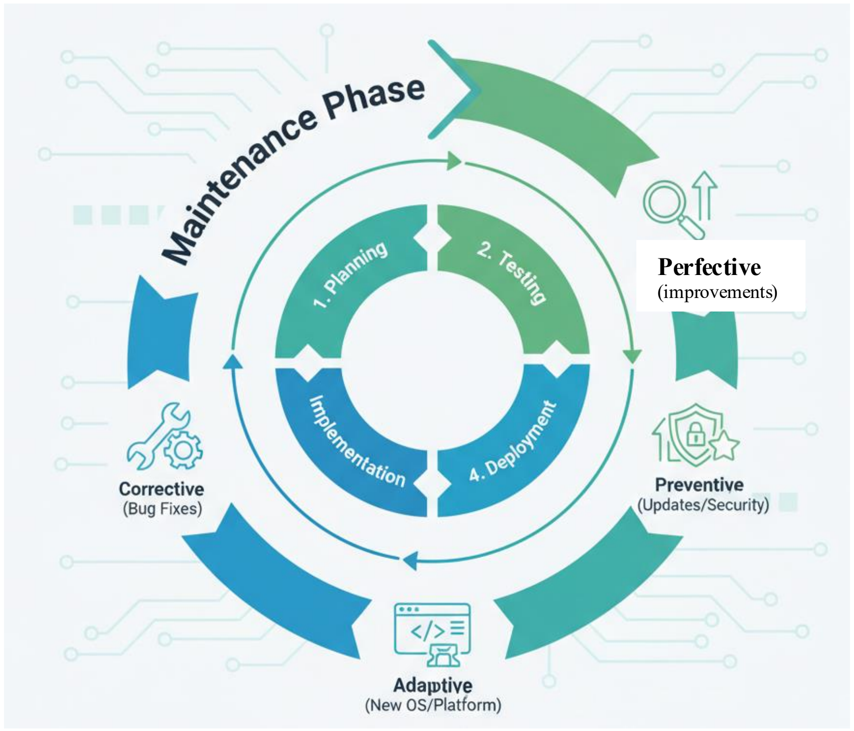
- Maintenance of any software product:
  - requires much more effort than the effort to develop the product itself

Preventive maintenance: Defect prevention

Corrective maintenance: Bug fixes

Perfective maintenance: Improvements/enhancements

Adaptive maintenance: Port software to a new environment

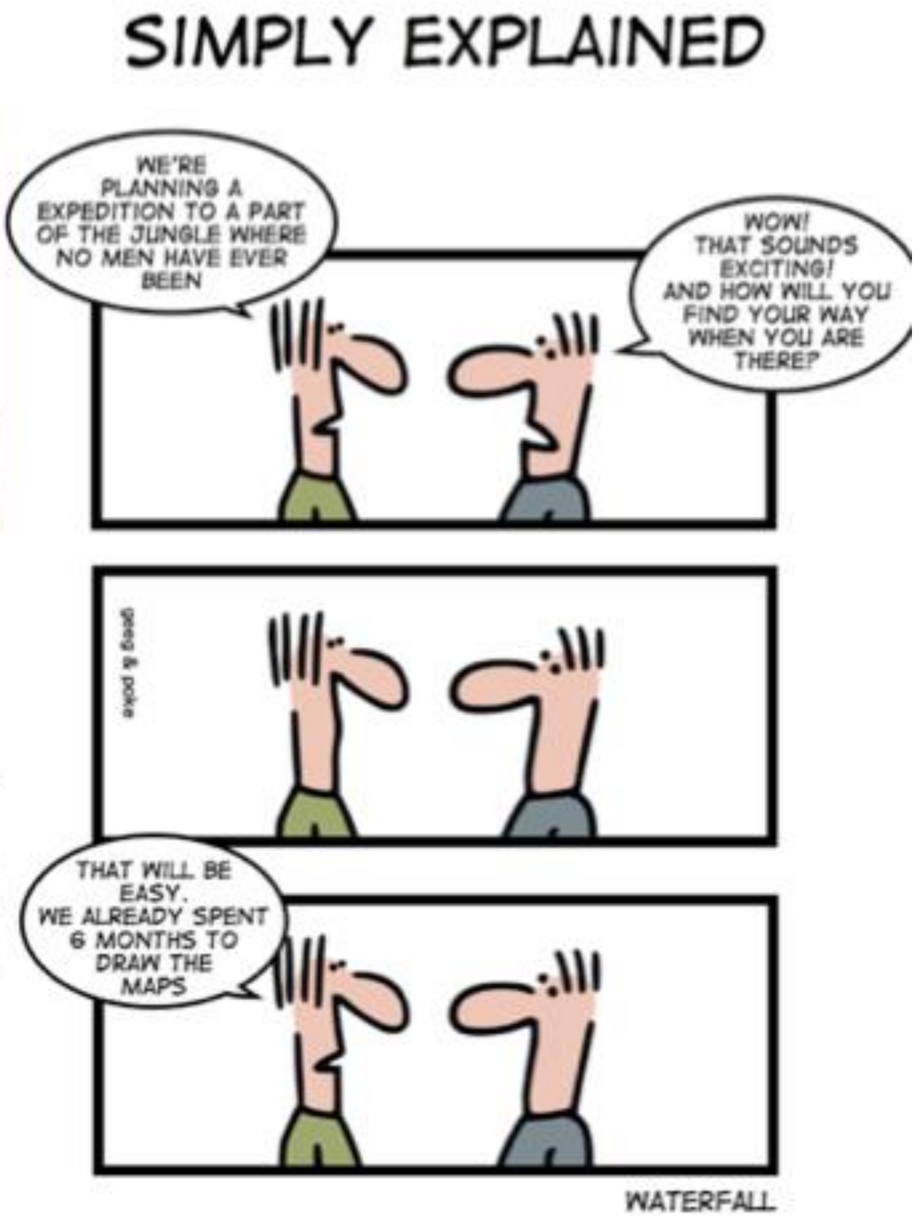


Process Models

Many life cycle models have been proposed

- Traditional Models (plan-driven)
  - Waterfall model
  - Prototyping
  - Evolutionary
  - Spiral model
  - …
- Agile Models
  - eXtreme Programming (XP)
  - Scrum
  - Crystal
  - Feature-Driven Development (FDD)
  - …

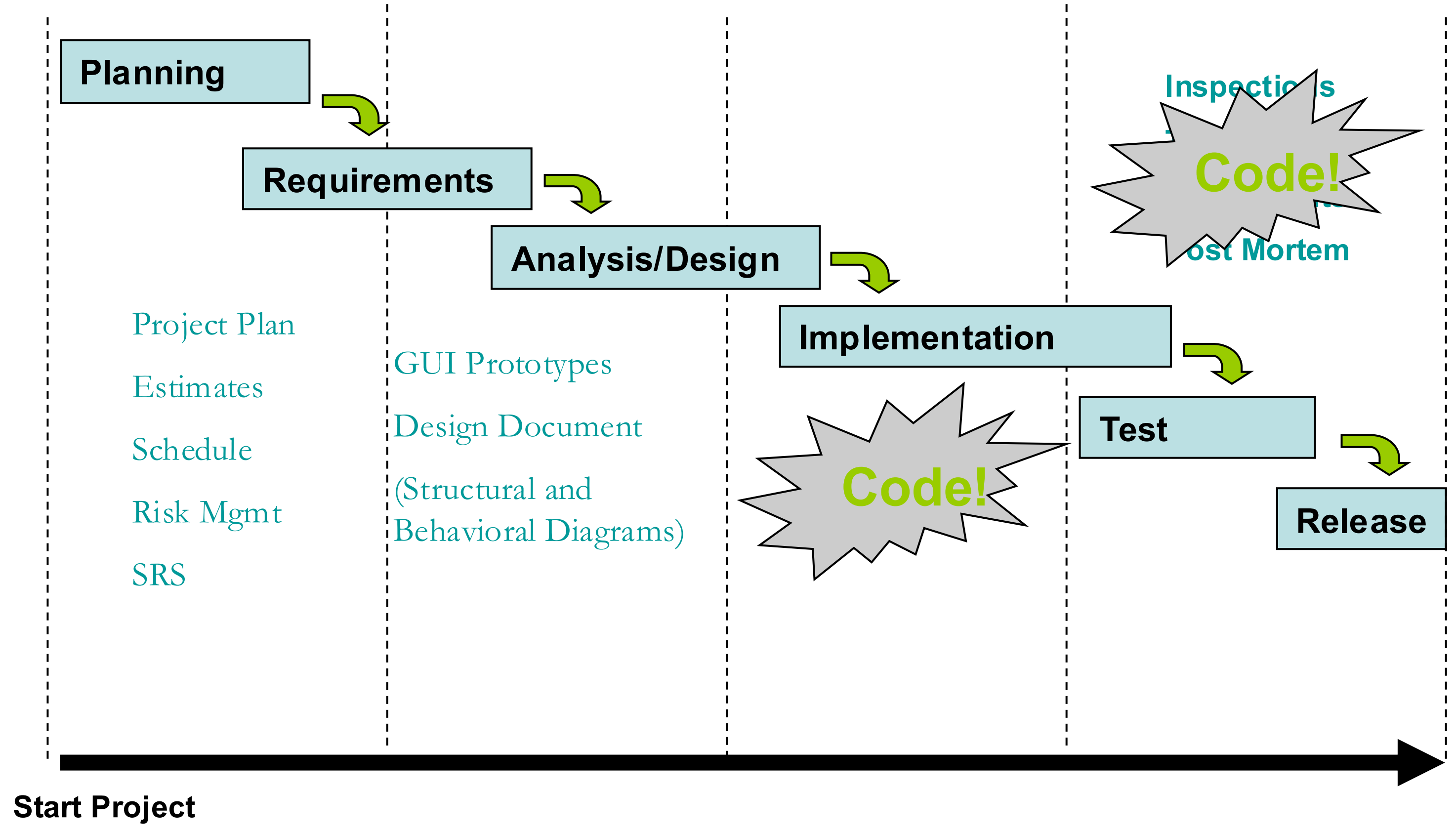
Waterfall Model


SIMPLY EXPLAINED
WE'RE PLANNING A EXPEDITION TO A PART OF THE JUNGLE WHERE NO MEN HAVE EVER BEEN
WOW! THAT SOUNDS EXCITING! AND HOW WILL YOU FIND YOUR WAY WHEN YOU ARE THERE?
geek & poke
THAT WILL BE EASY. WE ALREADY SPENT 6 MONTHS TO DRAW THE MAPS
WATERFALL

O. Widder. (2013). *geek&poke*. Available: http://geek-and-poke.com

- Sequential process
- Requirements, Design, Implementation, Testing, and Maintenance need to be followed in that order
- Each phased needs to be completed before moving to next phase

Traditional waterfall process model



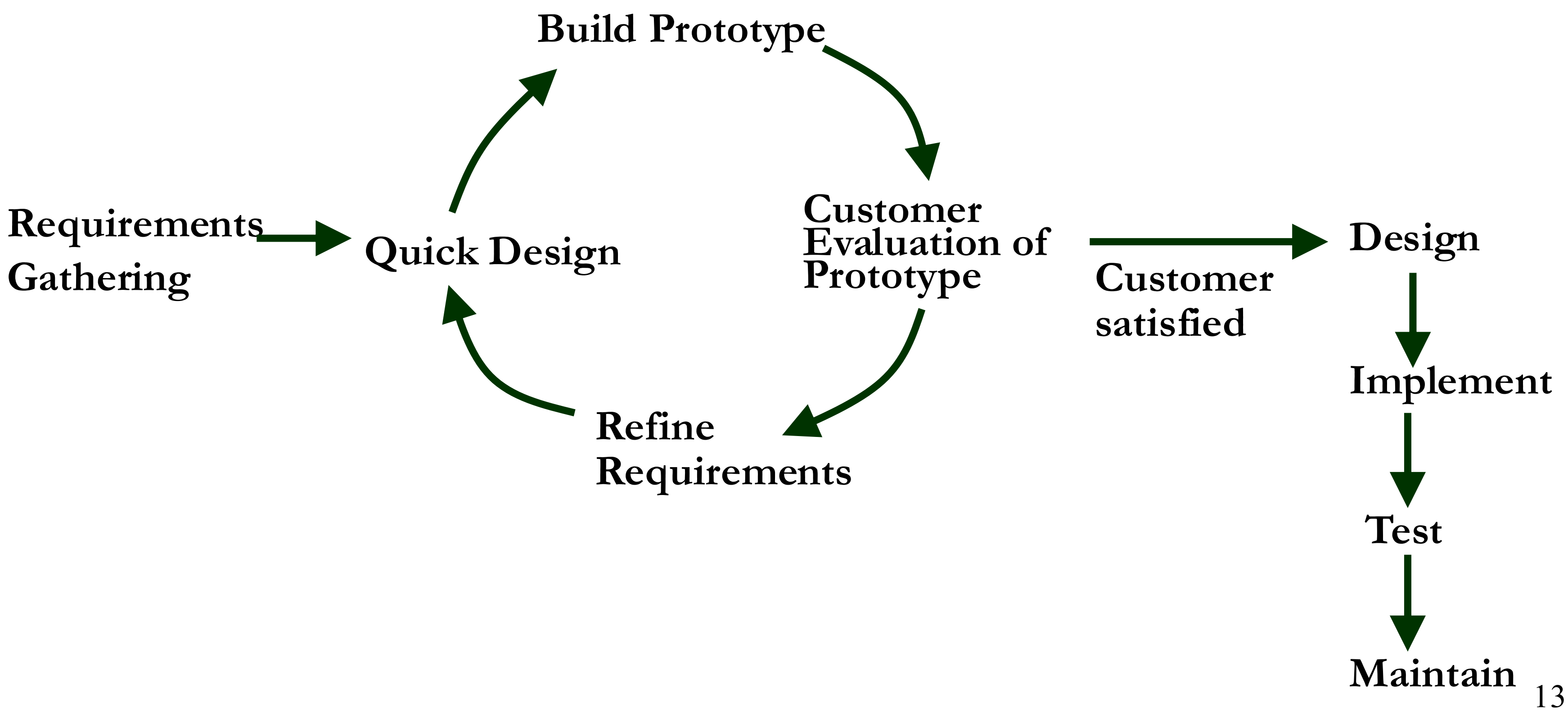
Challenges with waterfall model

- Heavyweight process
- Document intensive
- Minimal customer involvement after requirements phase
- Less flexible design
- Big bang approach to coding/integration
- One-shot delivery opportunity
- Limited opportunity for process improvement

Best suited systems: Aircraft Flight Control Software; Medical Imaging Devices (MRI/CT firmware); Banking Core Transaction Systems

Prototyping process model

- Before starting actual development,
  - a working prototype of the system should first be built.
- A prototype is a toy implementation of a system:
  - limited functional capabilities,
  - low reliability,
  - inefficient performance.



Prototyping process model

Types

- Throwaway Prototyping
- Evolutionary Prototyping

Key Characteristics

- Requirements are **unclear at the beginning**
- High user involvement
- Focus on **UI/UX and functionality understanding**
- Prototype may or may not become the final system

Best-Suited Example Systems: ATM User Interface Design; Hospital Appointment Booking System; E-commerce Website Front-End

Evolutionary process model

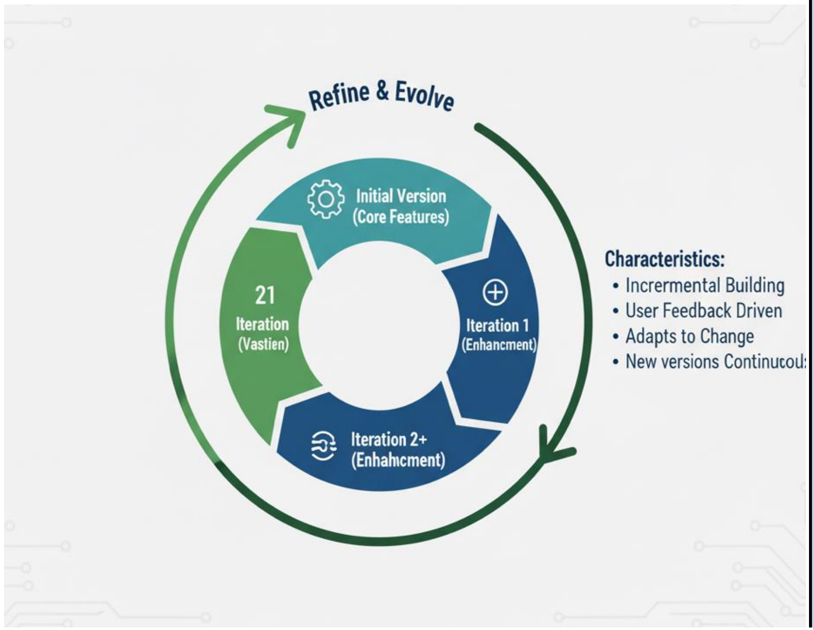
- Evolutionary model (aka successive versions or incremental model):
  - The system is broken down into several modules which can be incrementally implemented and delivered.

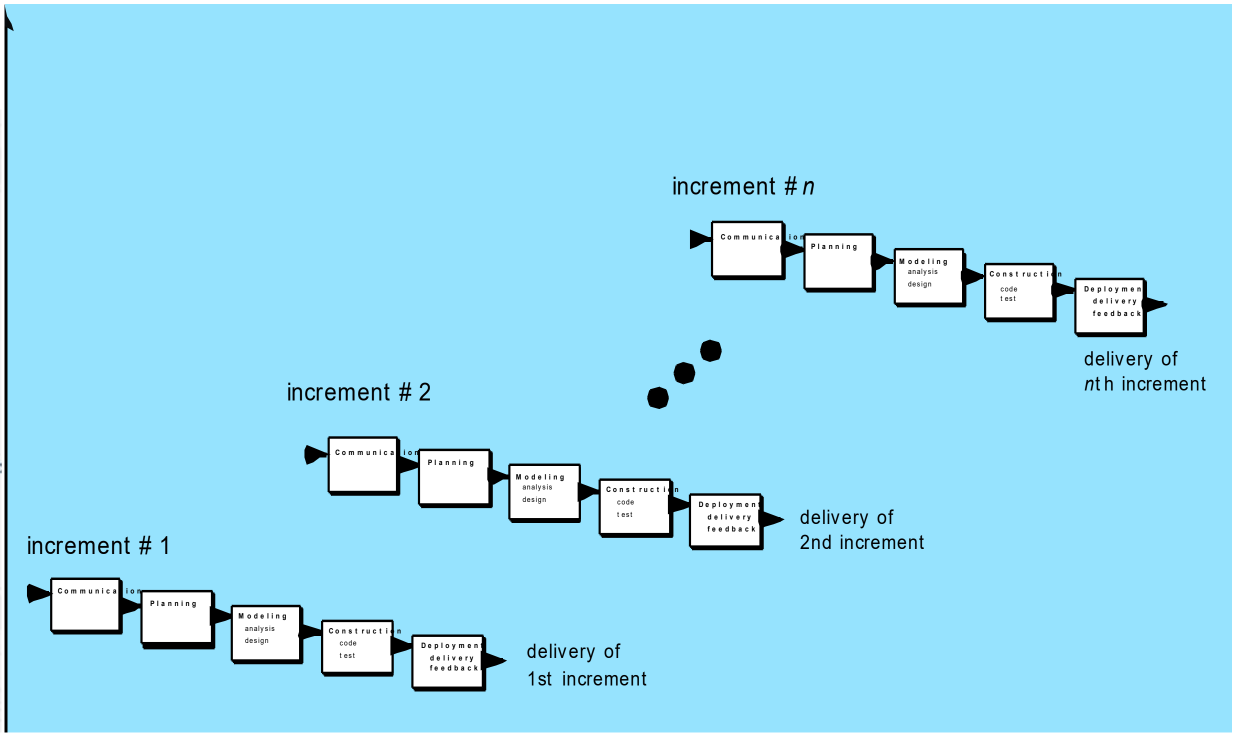
- First develop the core modules of the system.

- The initial product skeleton is refined into increasing levels of capability:
  - by adding new functionalities in successive versions.

EVOLUTIONARY MODEL WITH ITERATION (Iterative Incremental Model)

- Many organizations use a combination of iterative and incremental development:
  - a new release may include new functionality
  - existing functionality from the current release may also have been modified.



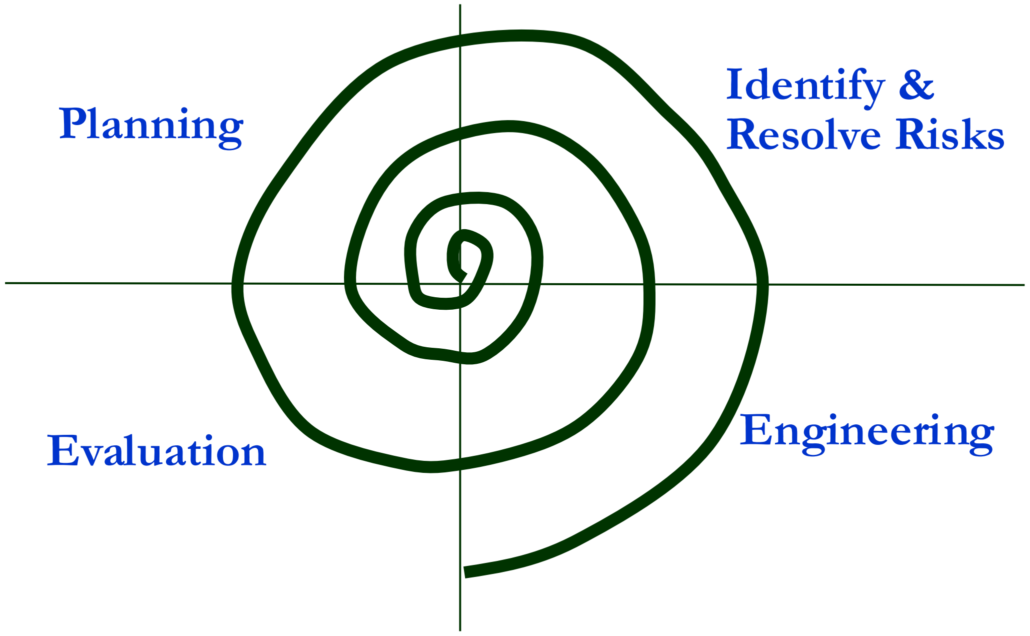


SPIRAL MODEL

Proposed by Boehm in 1988.

- Strong emphasis on **risk identification and mitigation**
- Continuous customer feedback
- Suitable for large, complex, high-risk systems
- Expensive but highly controlled

It's often referred as a meta-model as each loop might be a process model in itself.



Comparison of Plan-driven Process models

- Waterfall model
  - Each phase must be completed before going to next phase
  - Suitable for well-understood problems.
- Prototype model is suitable for projects where:
  - user requirements are not well understood
  - technical aspects are not well known
- Evolutionary model is suitable for large problems:
  - can be decomposed into a set of modules that can be incrementally implemented,
  - incremental delivery of the system is acceptable to the customer.
- The spiral model:
  - suitable for development of technically challenging software products that are subject to several kinds of risks.

Which process model is best suited?

- A company needs a payroll system to be built in 6 months to calculate salaries based on fixed tax laws and fixed internal policies.
- A retail client wants a unique "analytics dashboard" but isn't sure how they want the data visualized
- Building a system where failure could lead to loss of life
- Developing a massive Operating System like Windows.

Selection logic (summary)

Model	Primary goal	Example	Avoid if…
Waterfall	Stability	Payroll/Govt forms/Embedded Hardware systems/Core Banking	Requirements change often
Prototype	Clarity	UI/UX/New concepts	System is backend heavy
Evolutionary	Speed	Startup Apps/SaaS	Safety is critical
Spiral	Safety	ATC/Nuclear/Military	Budget is tight or project is simple

Questions ?