Software Development Life Cycle (SDLC)

SDLC

A framework that describes the activities performed at each stage of a software development project.

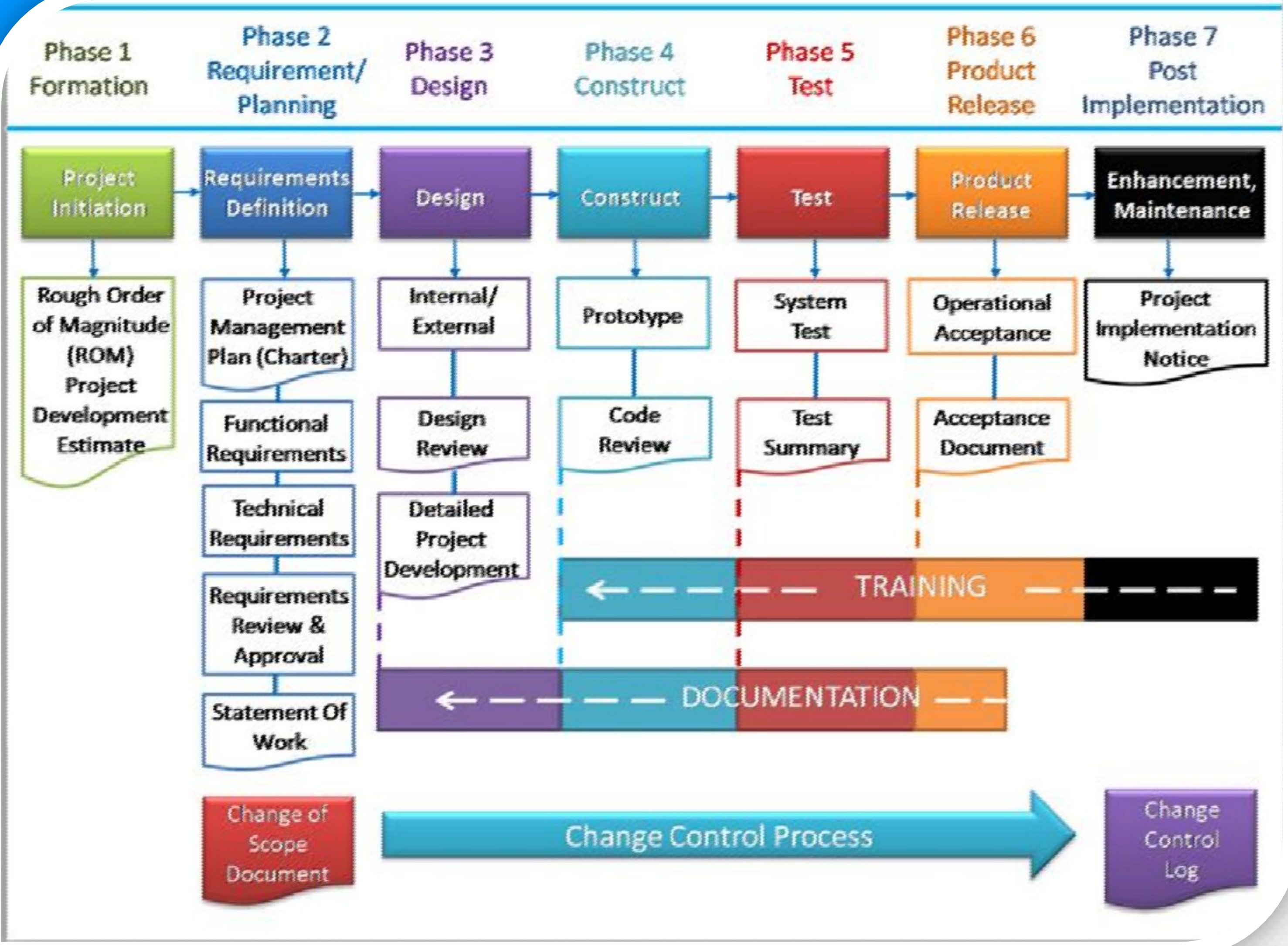
AIM:

Software Development Life Cycle (SDLC) is a process used by the software industry to **design, develop and test high quality software's.**

SDLC PHASES

- Planning and Requirements Gathering and Analysis
- Design
- Development
- Testing
- Implementation
- Maintenance

SDLC


Phase 1 Formation
Phase 2 Requirement/ Planning
Phase 3 Design
Phase 4 Construct
Phase 5 Test
Phase 6 Product Release
Phase 7 Post Implementation
Project Initiation
Requirements Definition
Design
Construct
Test
Product Release
Enhancement, Maintenance
Rough Order of Magnitude (ROM) Project Development Estimate
Project Management Plan (Charter)
Functional Requirements
Technical Requirements
Requirements Review & Approval
Statement Of Work
Internal/ External
Design Review
Detailed Project Development
Prototype
Code Review
System Test
Test Summary
Operational Acceptance
Acceptance Document
Project Implementation Notice
TRAINING
DOCUMENTATION
Change of Scope Document
Change Control Process
Change Control Log

STAGE 1: Planning and Requirements Gathering and Analysis

- This phase defines the **importance of system development** life cycle.
- This is also called the **SDLC planning phase**.
- This is where you do your **market research, conduct customer interviews, research your competition and conduct surveys.**
- The feedback that you gather **helps you analyze your product market fit.**

STAGE 2: SDLC Design Phase

- This phase is crucial because this is **where you design your product.**
- In the previous phase, **you defined what needs to be in the product and what needs to be out.**
- In the design phase in SDLC, you need to be as **visual as possible about your vision.**
- Use the design phase to really **communicate your plan for the product to all your developers.**

STAGE 3:Coding/ development phase

- The coding phase in SDLC is **not handled directly by Product Managers.**
- It is, of course, handled by **developers**.
- The development phase in SDLC is crucial because this is **where your product ultimately gets built.**

STAGE 4:Testing phase

- The software development life cycle models **all hinge on the testing phase.**
- In the software development life cycle **agile models**, testing can happen more frequently.
- But, in the waterfall methodology, testing is not very frequent.

STAGE 5:Deployment

- In the deployment phase, **the application is made available to users.**
- Many companies prefer to **automate** the deployment phase.
- This can be as simple as a **payment portal and download link on the company website.**
- **Example:** It could also be downloading an application on a smartphone.

STAGE 6:Operations and Maintenance

- At this point, the development cycle is **almost finished.**
- The application is **done** and **being used** in the field.
- The Operation and Maintenance phase is still **important**, though.
- In this phase, **users discover bugs** that weren’t found during testing.
- These errors need to be **resolved**, which can spawn new development cycles.

SDLC MODELS

To help understand and implement the SDLC phases various SDLC models have been created by software development experts, universities, and standards organizations.

Reasons forUsing SDLC Models

- Provides basis for project planning, estimating & scheduling
- Provides framework for standard set of terminologies, activities & deliverables
- Provides mechanism for project tracking & control
- Increases visibility of project progress to all stakeholders

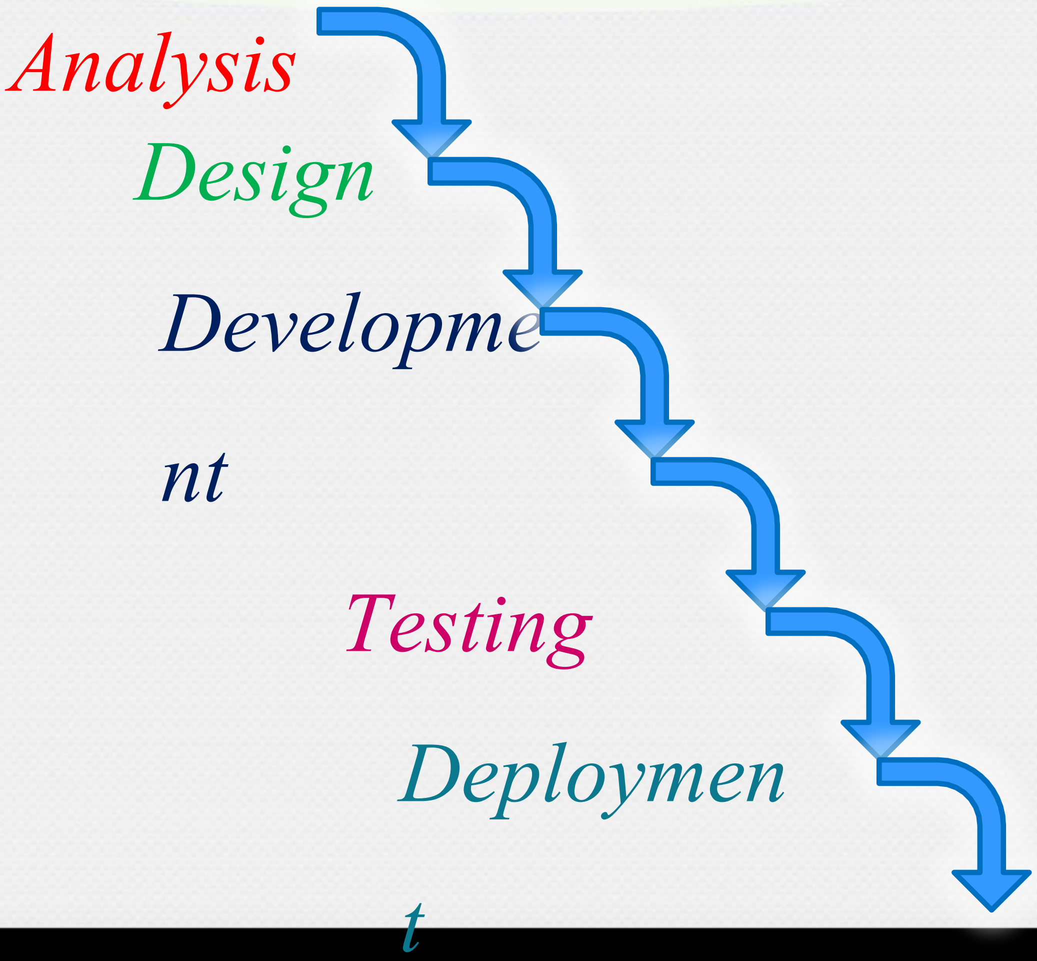
Advantages of Choosing an Appropriate SDLC

- Increased development speed
- Increased product quality
- Improved tracking & control
- Improved client relations
- Decreased project risk
- Decreased project management overhead

Common Life Cycle Models

- Waterfall
- Spiral/Iterative
- Agile

Waterfall Model



Waterfall Model

- Oldest and most well-known SDLC model
- Follows a sequential step-by-step process from requirements analysis to maintenance.
- We can use this, In the manufacturing of a physical product, requirements do not, and cannot, change everyday.

For example, take the manufacturing of a bicycle.

- What model should be followed to manufacture a bicycle?
- Are requirements expected to change frequently on a weekly basis?
- The answer is clearly no. The requirements are not expected to change weekly for a bicycle.

Waterfall Model Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

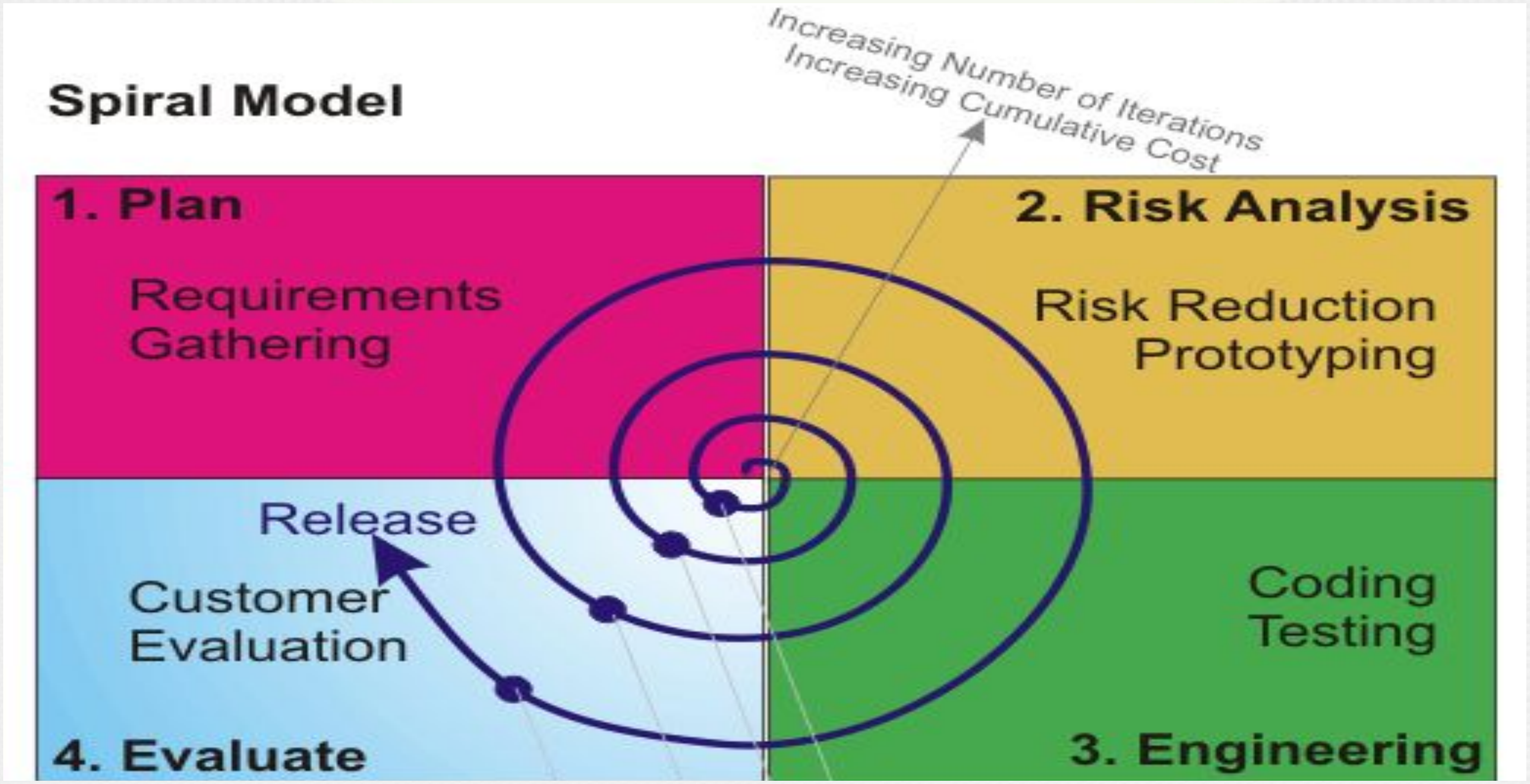
Waterfall Model Weaknesses

- All requirements must be fully specified upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Little opportunity for customer to preview the system (until it may be too late)

When to use the Waterfall Model

- 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.

Spiral/Iterative Model

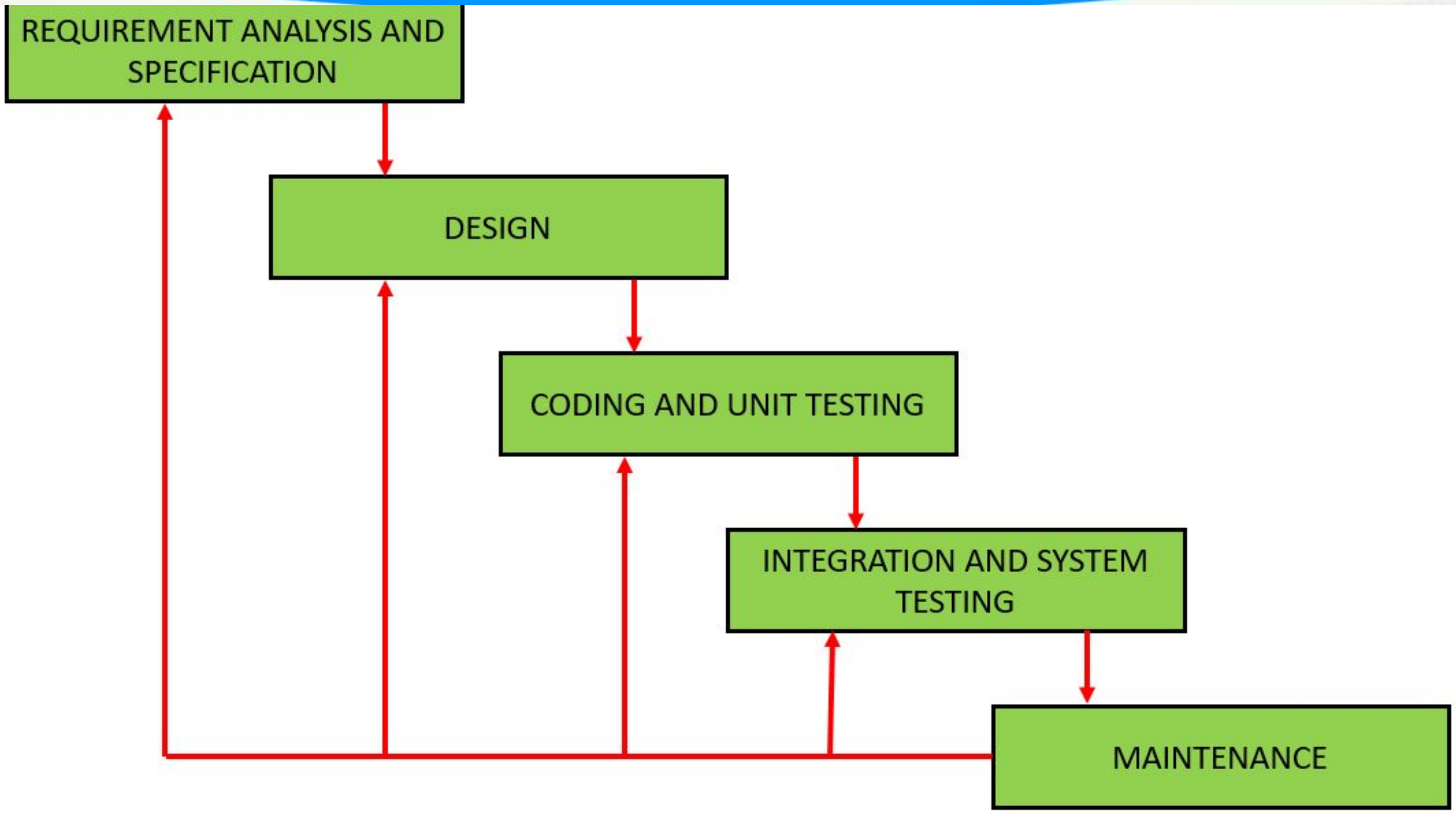


Spiral Model

- Spiral Model is a "risk-driven"(prioritize your **risks**) iterative model
- It take 6 months to 2 years to complete the project
- Each iteration starts with small set of requirements and goes through development phase (except Installation and Maintenance) for those set of requirements.

Spiral Model

- Iterate until all major risks addressed and the application is ready for the Installation and Maintenance phase (production)
- Last iteration is a waterfall process


REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

Spiral Model Strengths

- Critical high-risk functions are developed first

  The design does not have to be perfect
- Users see the system early because of rapid prototyping tools
- Users can be closely tied to all lifecycle steps

  Early and frequent feedback from users

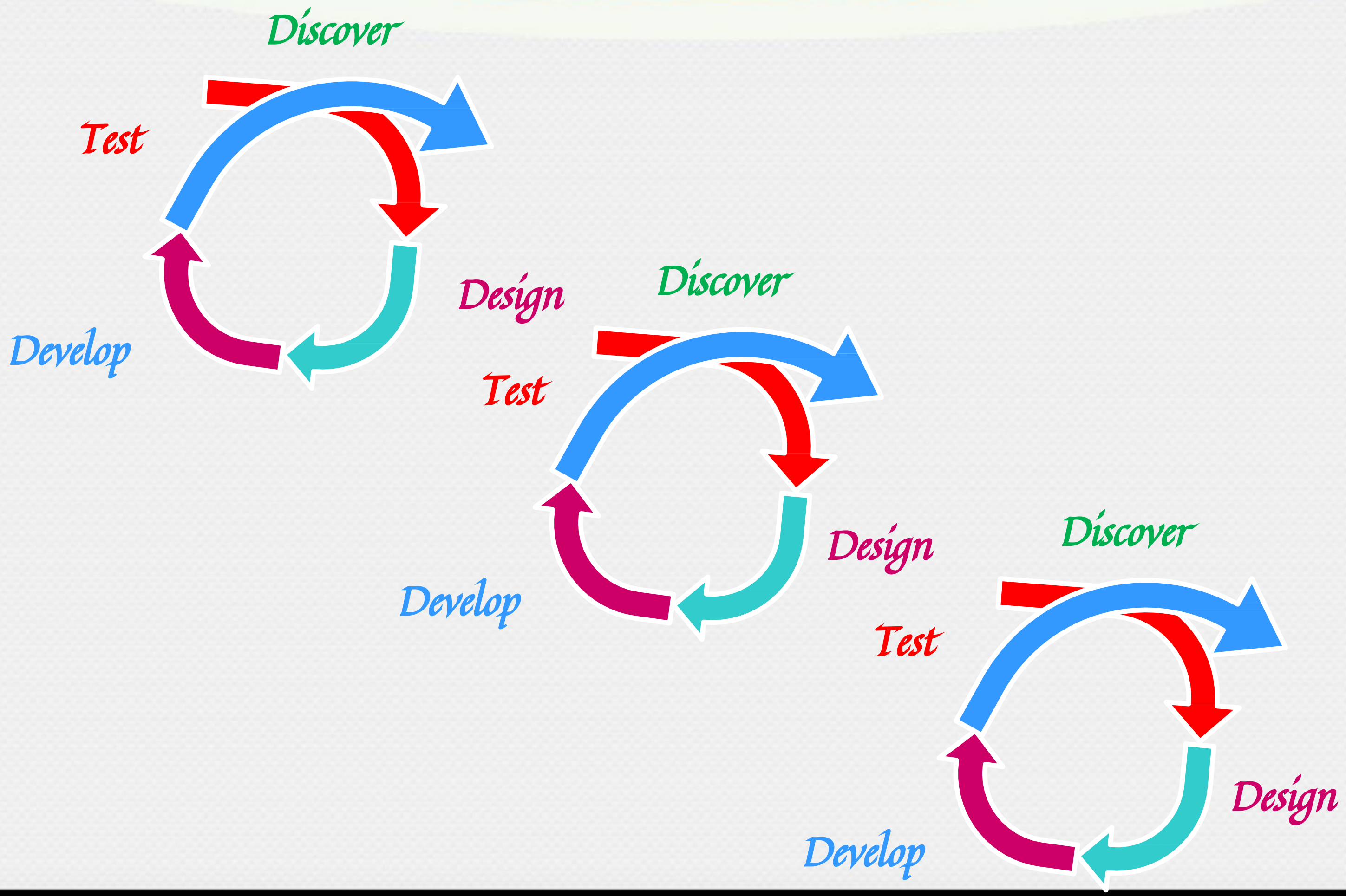
Spiral Model Weaknesses

- Time spent for evaluating risks too large for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- The model is complex
- Risk assessment expertise is required

When to use Spiral Model

- When risk evaluation is important
- For medium to high-risk projects
- Users are unsure of their needs
- Requirements are complex New
- product line

Agile Model


Discover
Test
Design
Develop
Discover
Test
Design
Develop
Discover
Test
Design
Develop

Agile Model

- Customer feedback at every stage
- Used for time-critical applications
- Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from **one to four weeks.**

Agile Model Strengths

- Deliver a working product faster than conventional linear development model
- Customer feedback at every stage ensures that the end deliverable satisfies their expectations
- No guesswork between the development team and the customer, as there is face to face communication and continuous inputs from the client

Agile Model Weaknesses

- For larger projects, it is difficult to judge the efforts and the time required for the project in the SDLC.
- Since the requirements are ever changing, there is hardly any emphasis, which is laid on designing and documentation.

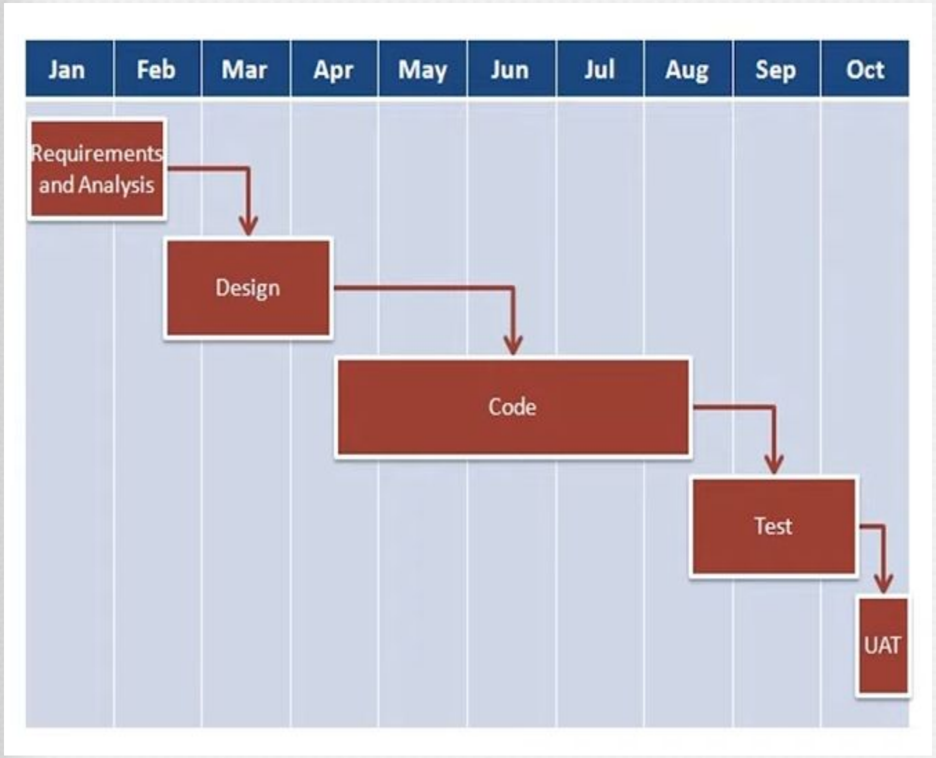


Fig: Waterfall model

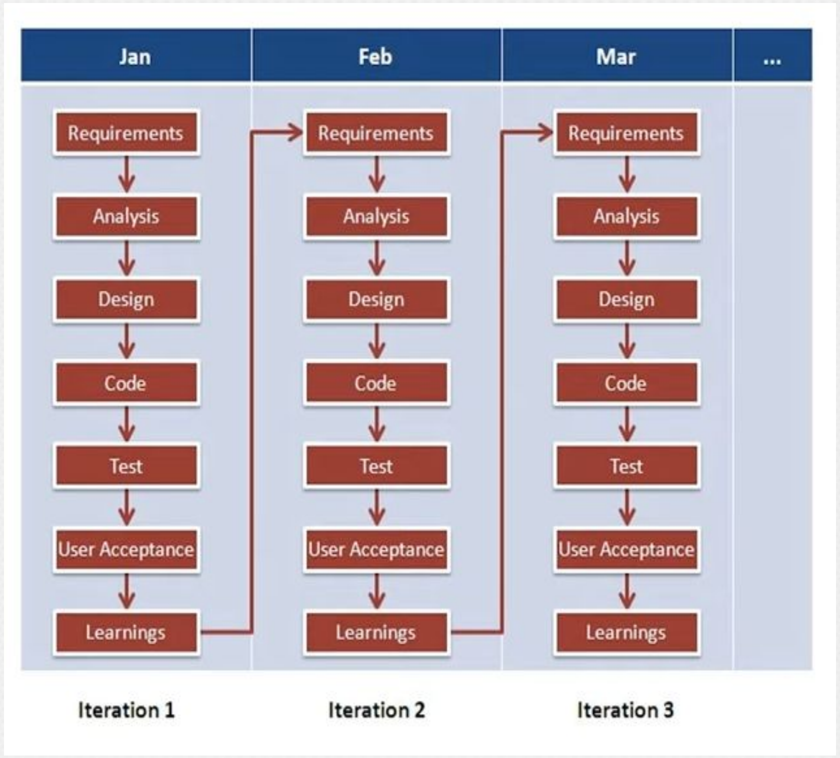
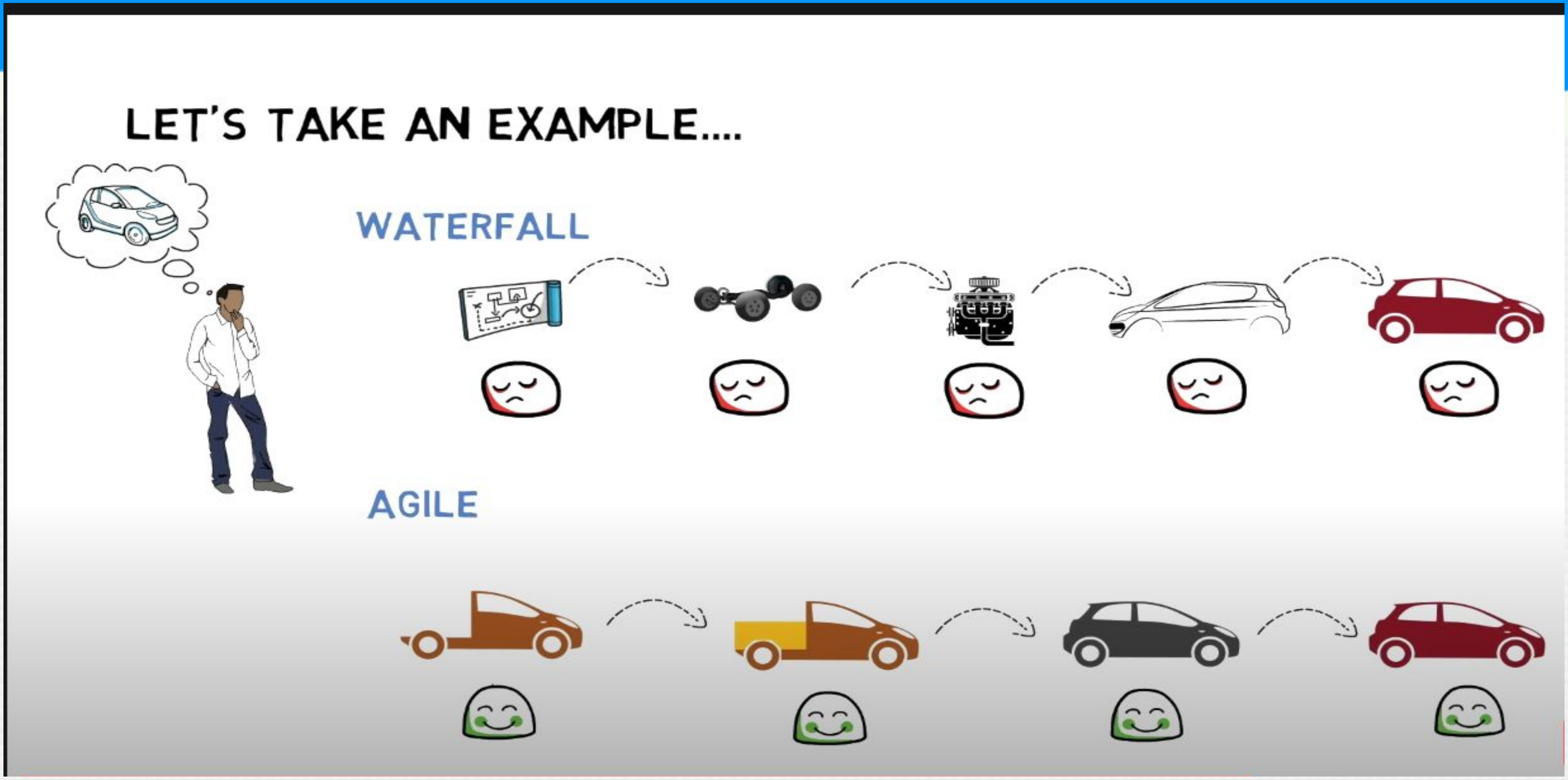


Fig: Agile model


LET'S TAKE AN EXAMPLE....
WATERFALL
AGILE

Thank You!