Software Process Models

Lecture 1

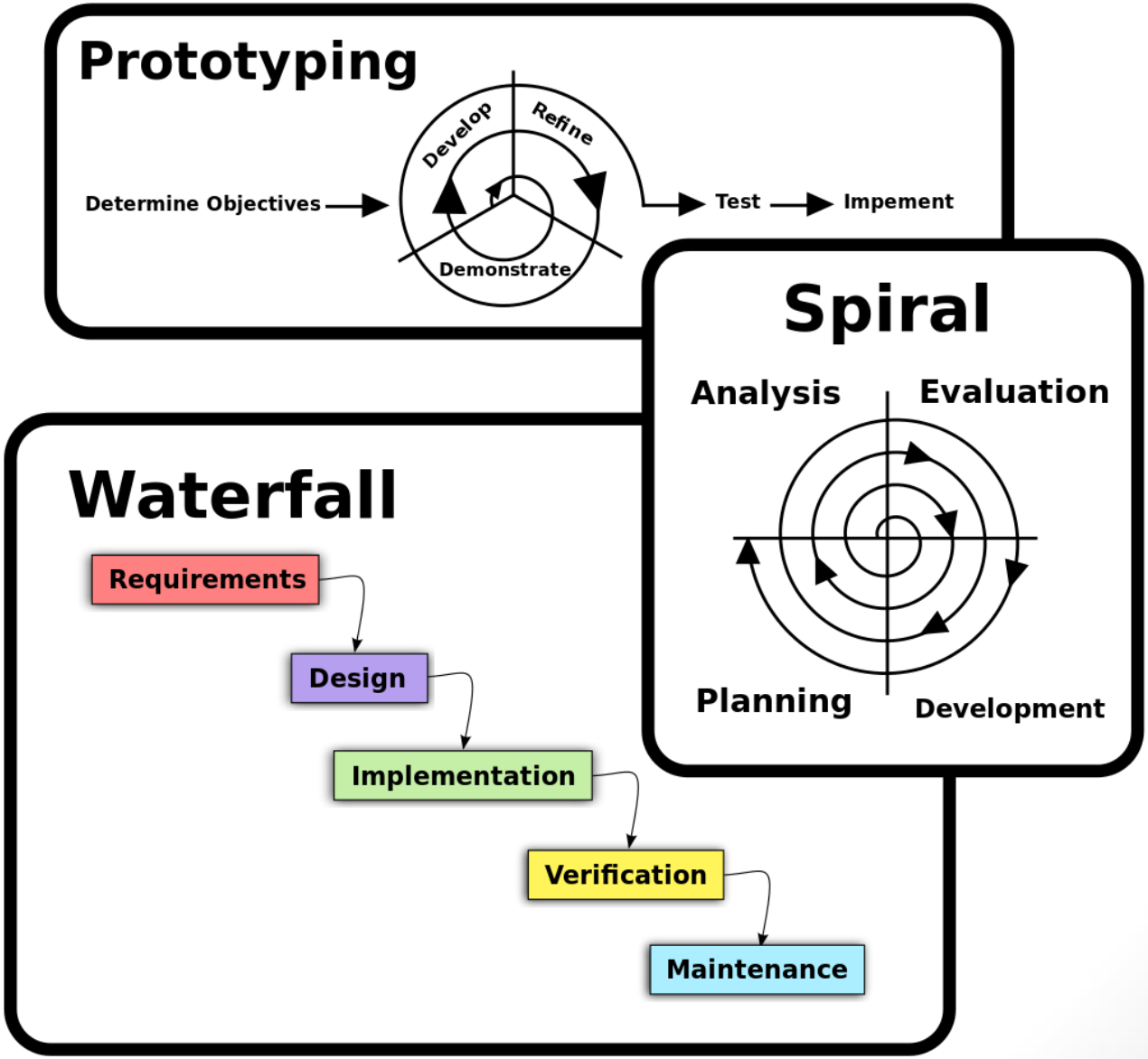
Overview

- Different process models
  - Build-and-fix model
  - Waterfall model
  - Incremental model
  - Evolutionary process models
    - Rapid prototyping model
    - Spiral model
  - Agile process models
    - Extreme programming
  - Object-oriented life-cycle models
    - Unified Process
- Criteria for deciding on a model

Software Process Models

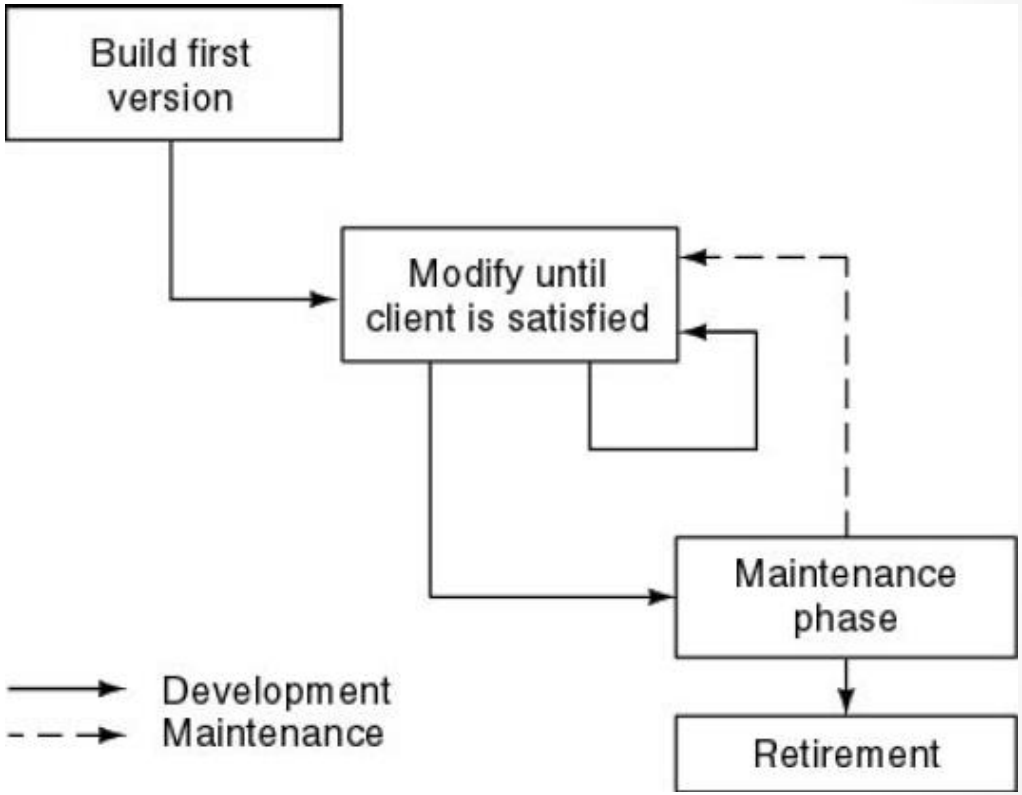
- Process model (Life-cycle model) -steps through which the product progresses
  - Requirements phase
  - Specification phase
  - Design phase
  - Implementation phase
  - Integration phase
  - Maintenance phase
  - Retirement

Software Process Models



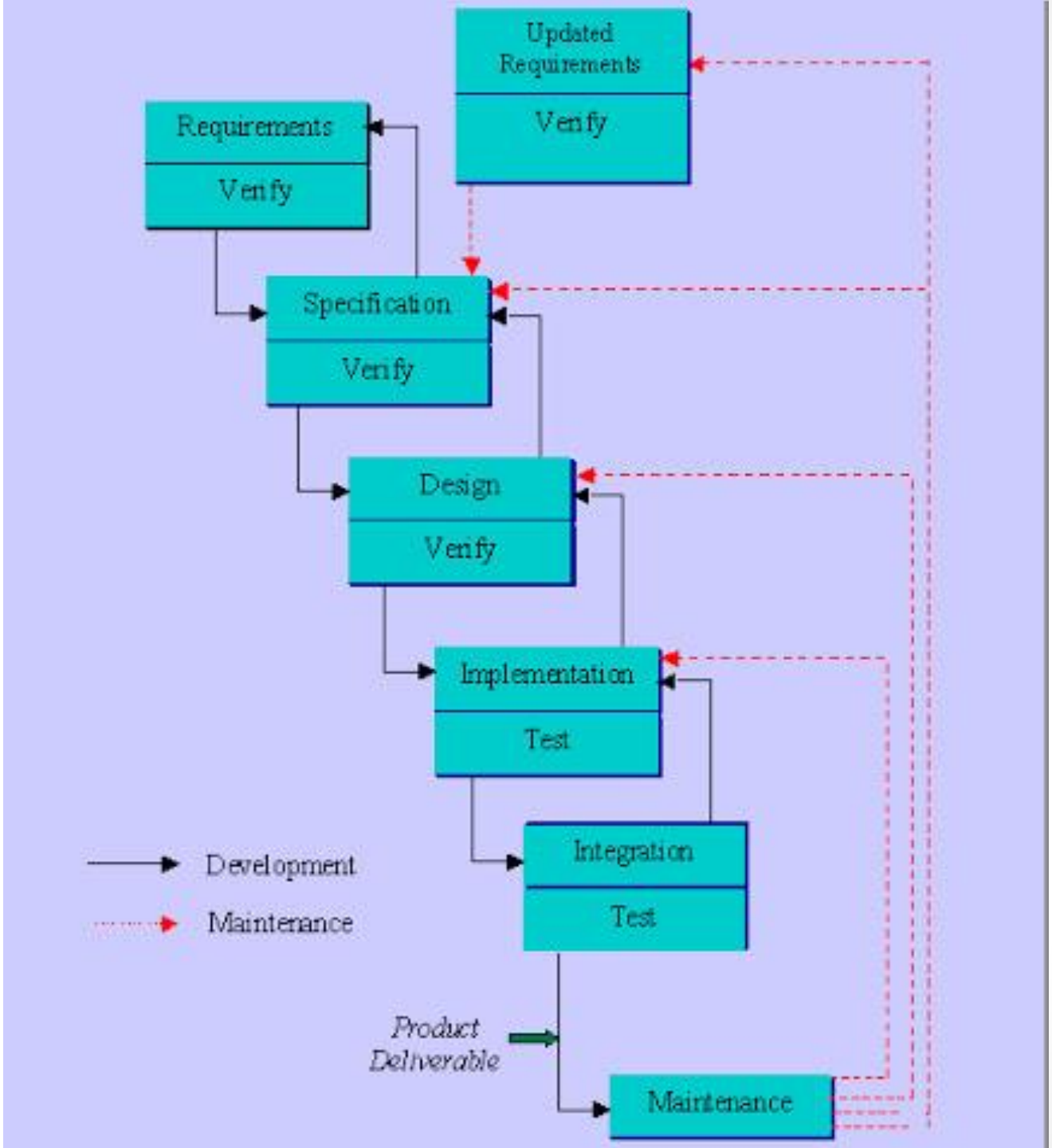
Build-and-Fix Model

- Problems
  - No specifications
  - No design
- Totally unsatisfactory
  - High cost
  - Difficult maintenance



Waterfall Model

- The **waterfall model** is a sequential design **process** in which progress is seen as flowing steadily downwards (like a **waterfall**) through the phases of SDLC.
- **Waterfall model** is an **example** of a Sequential **model**. In this **model**, the software **development** activity is divided into different phases and each phase consists of a series of tasks and has different objectives.
- **Waterfall model** is the pioneer of the SDLC processes.
- **Characterized by:**
  - Feedback loops
  - Documentation-driven

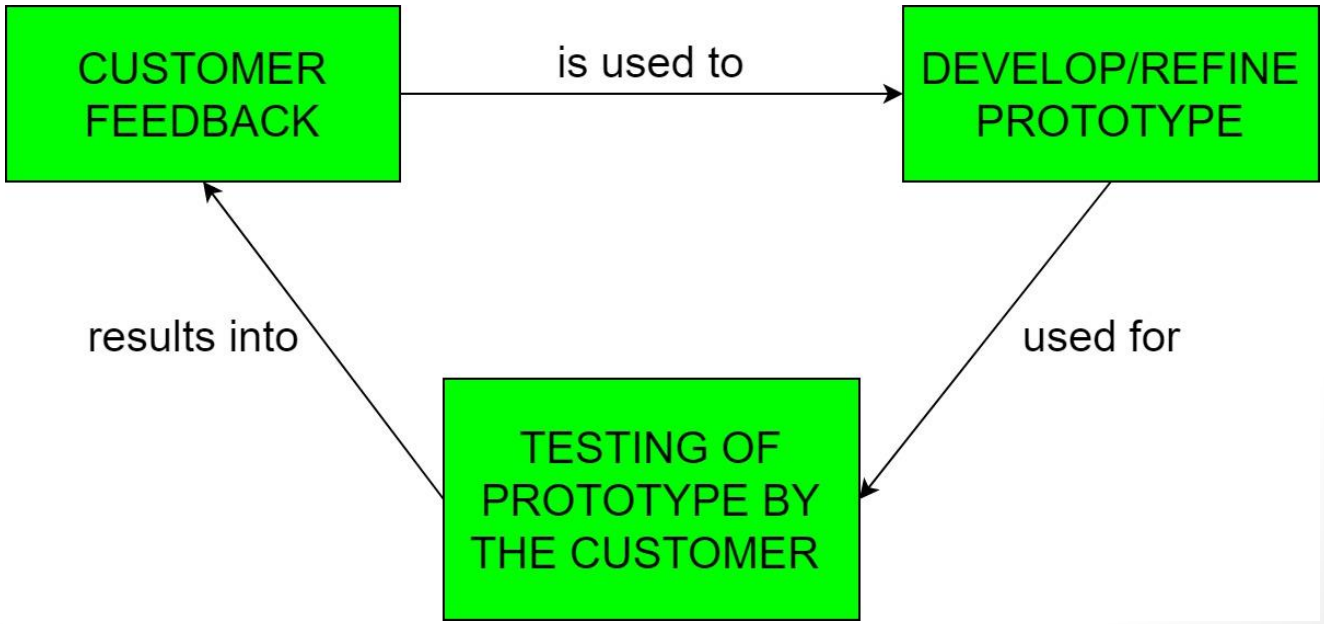

Updated Requirements
Verify
Requirements
Verify
Specification
Verify
Design
Verify
Implementation
Test
Integration
Test
Development
Maintenance
Product Deliverable
Maintenance

Waterfall Model (contd.)

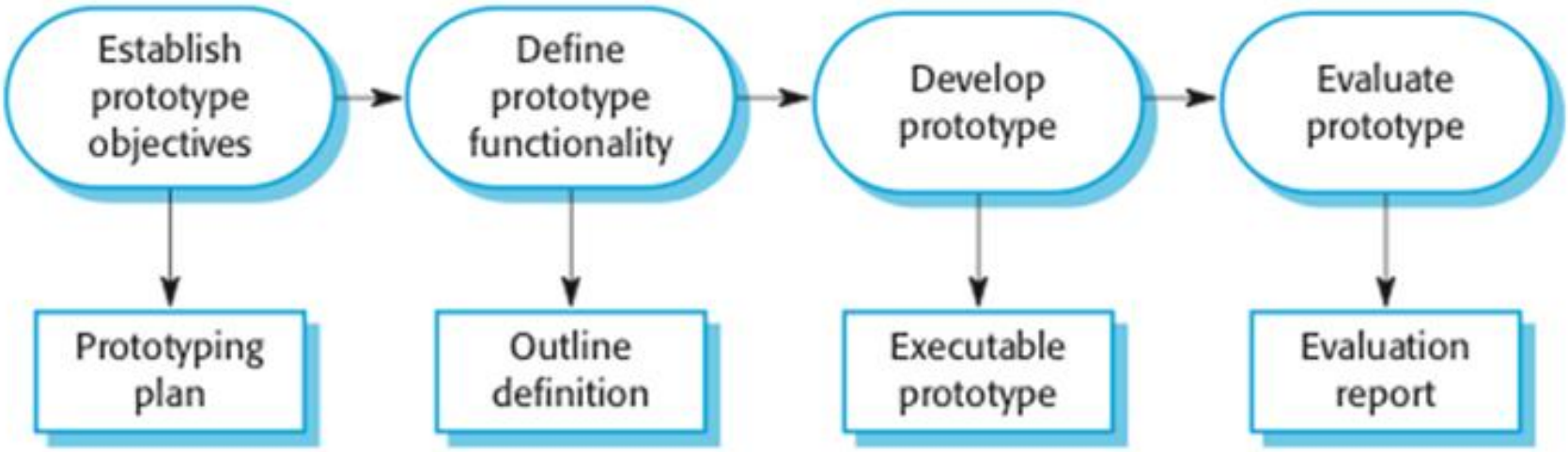
- Advantages
  - Enforces disciplined approach
    - Documentation for each phase
    - Products of each phase checked by SQA group
  - Maintenance is easier
    - Every change reflected in the relevant documentation
- Disadvantages
  - Working version of the software will not be available until late in the project time-span
  - Specifications are long, detailed, written in a style unfamiliar to the client
  - “Blocking states” –some project team members must wait for other team members to complete dependent tasks

Rapid Prototyping Model

- Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered.
- It offers a small scale replica of the end product and is used for obtaining customer feedback as described below:



Rapid Prototyping Model (contd.)



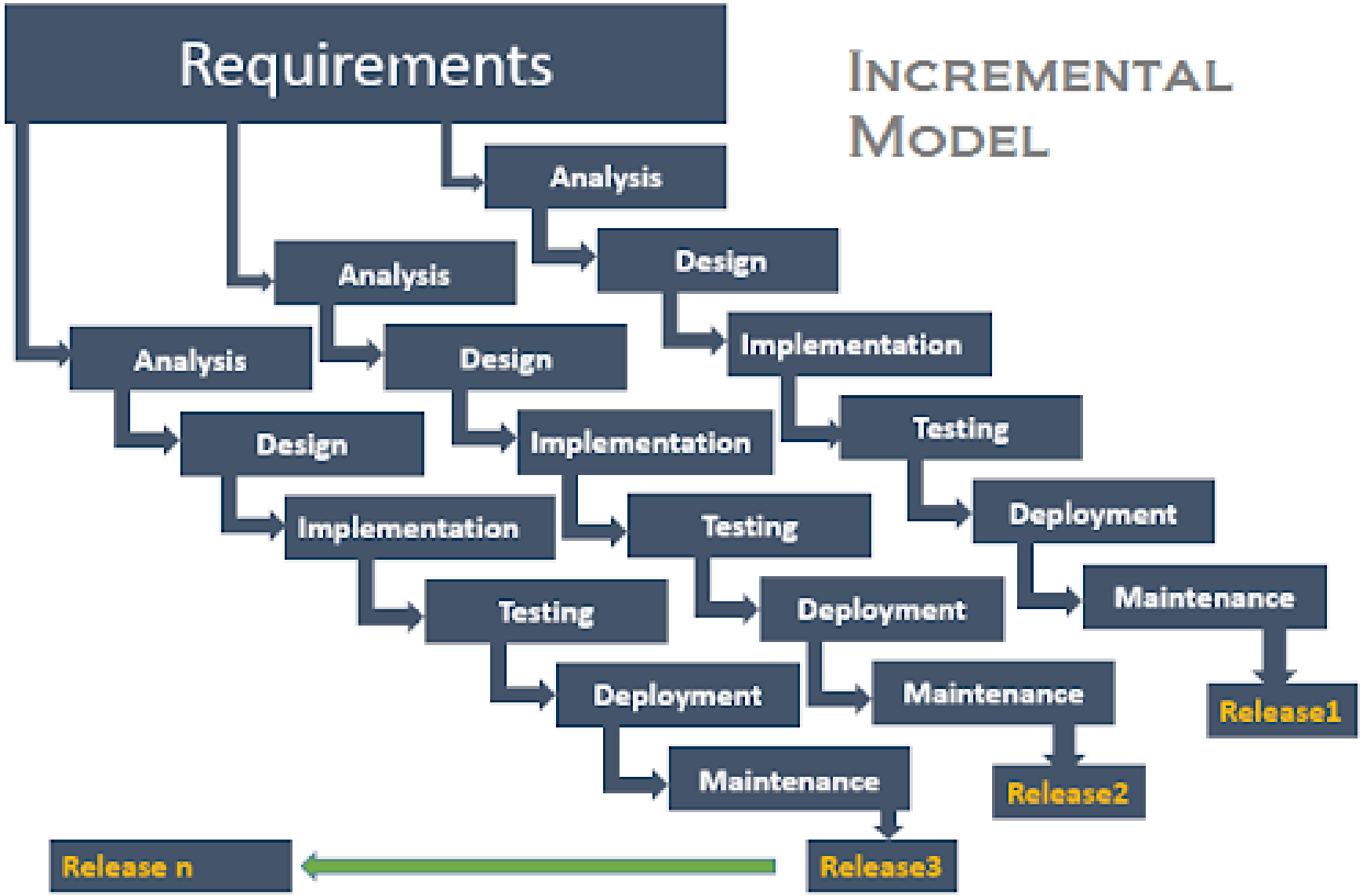
Rapid Prototyping Model (contd.)

- Rapid prototype characteristics:
  - Used in the requirements phase
  - Evaluated by the customer/user
  - Then, it is discarded -do not turn into product
- Rapid prototyping model is not proven and has its own problems
  - Possible solution
    - Rapid prototyping for defining requirements
    - Waterfall model for rest of life cycle

Incremental Model

- **Incremental Model** is a process of **software development** where requirements are broken down into multiple standalone modules of **software development** cycle.
- Each iteration passes through the requirements, design, coding and **testing** phases.
- Typical product takes from 5 to 25 builds (iterations).

Incremental Model (contd.)



Incremental Model (contd.)

- Waterfall and rapid prototyping models
  - Deliver complete product at the end
- Incremental model
  - Deliver portion of the product at each stage
- Advantages
  - The software will be generated quickly during the software life cycle
  - It is flexible and less expensive to change requirements and scope
  - Throughout the development stages changes can be done
  - This model is less costly compared to others
  - A customer can respond to each building
  - Errors are easy to be identified

Incremental Model (contd.)

- Disadvantages:
  - It requires a good planning designing
  - Problems might arise due to system architecture as not all requirements collected up front for the entire software lifecycle
  - Each iteration phase is rigid and does not overlap each other
  - Correcting a problem in one unit requires correction in all the units and consumes a lot of time

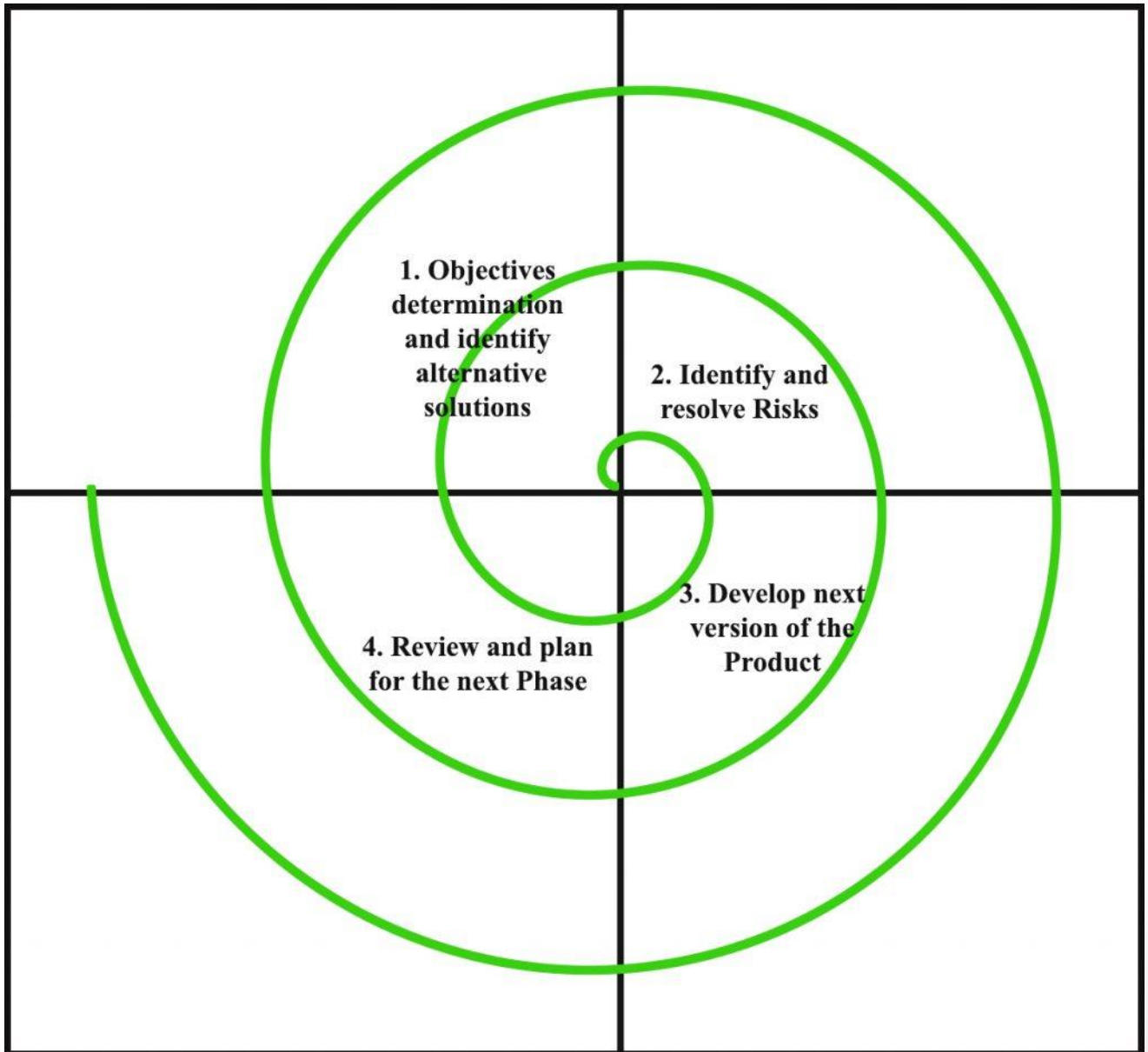
When to use Incremental models?

- Requirements of the system are clearly understood
- When **demand for an early release** of a product arises
- When software engineering **team are not very well skilled** or trained
- When high-risk features and goals are involved
- **Such methodology is more in use for web application and product based companies**

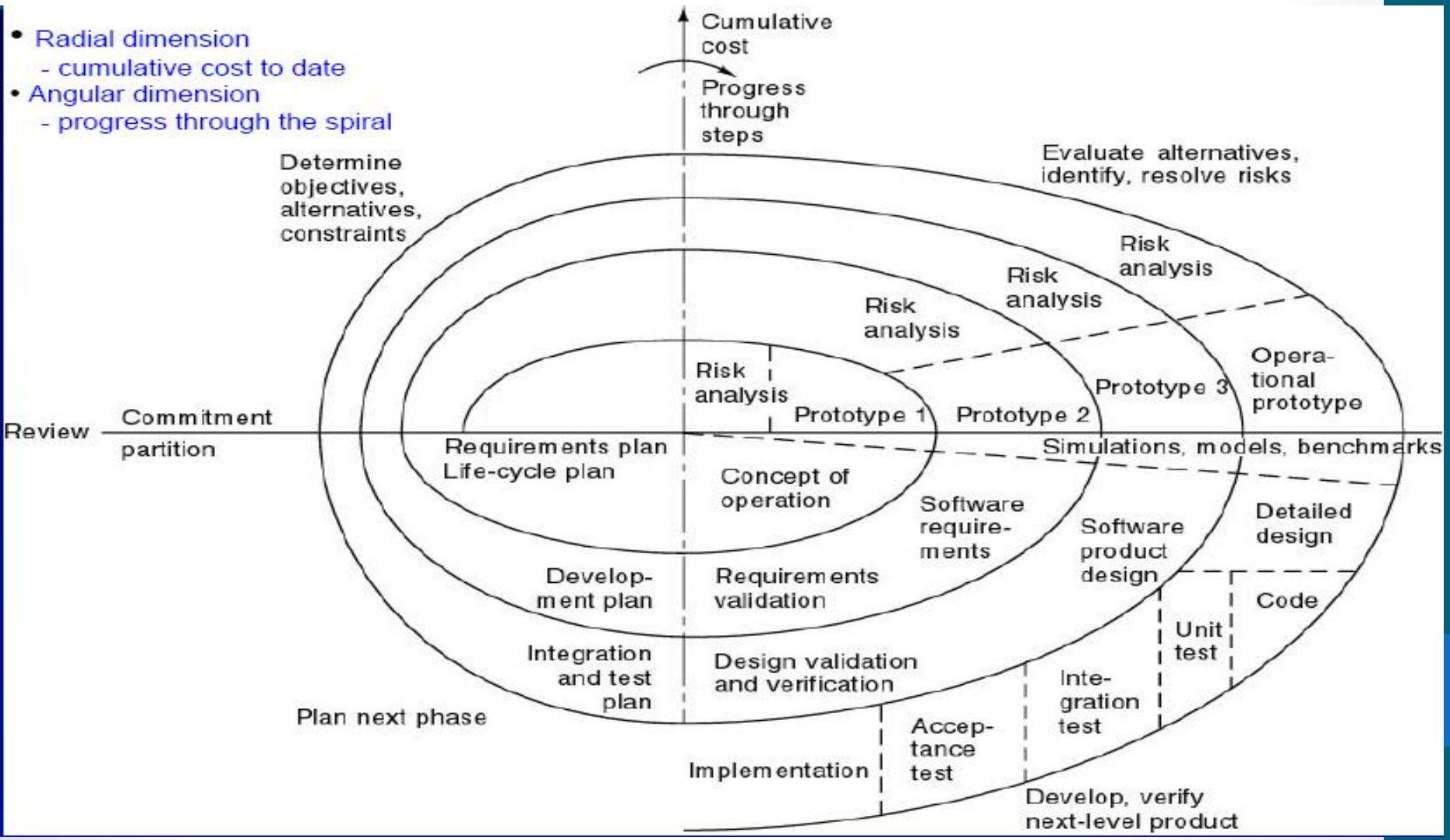
Spiral Model

- The **spiral model** is a risk-driven **software development** process **model**.
- Based on the unique risk patterns of a given project, the **spiral model** guides a team to adopt elements of one or more process **models**, such as incremental, waterfall, or evolutionary prototyping.
- **Risk Analysis:** Identification of potential risk is done while risk mitigation strategy is planned and finalized
- Precede each phase by
  - Alternatives
  - Risk analysis
- Follow each phase by
  - Evaluation
  - Planning of next phase

Simplified Spiral Model



Full Spiral Model



When to use Spiral Methodology?

- When project is large
- When releases are required to be frequent
- When creation of a prototype is applicable
- When risk and costs evaluation is important
- For medium to high-risk projects
- When **requirements are unclear** and complex
- When **changes may require at any time**
- When long term project commitment is not feasible due to changes in economic priorities

Advantages of Spiral Model

- Additional functionality or changes can be done at a later stage
- Cost estimation becomes easy as the prototype building is done in small fragments
- Continuous or repeated development helps in risk management
- Development is fast and features are added in a systematic way
- There is always a space for customer feedback

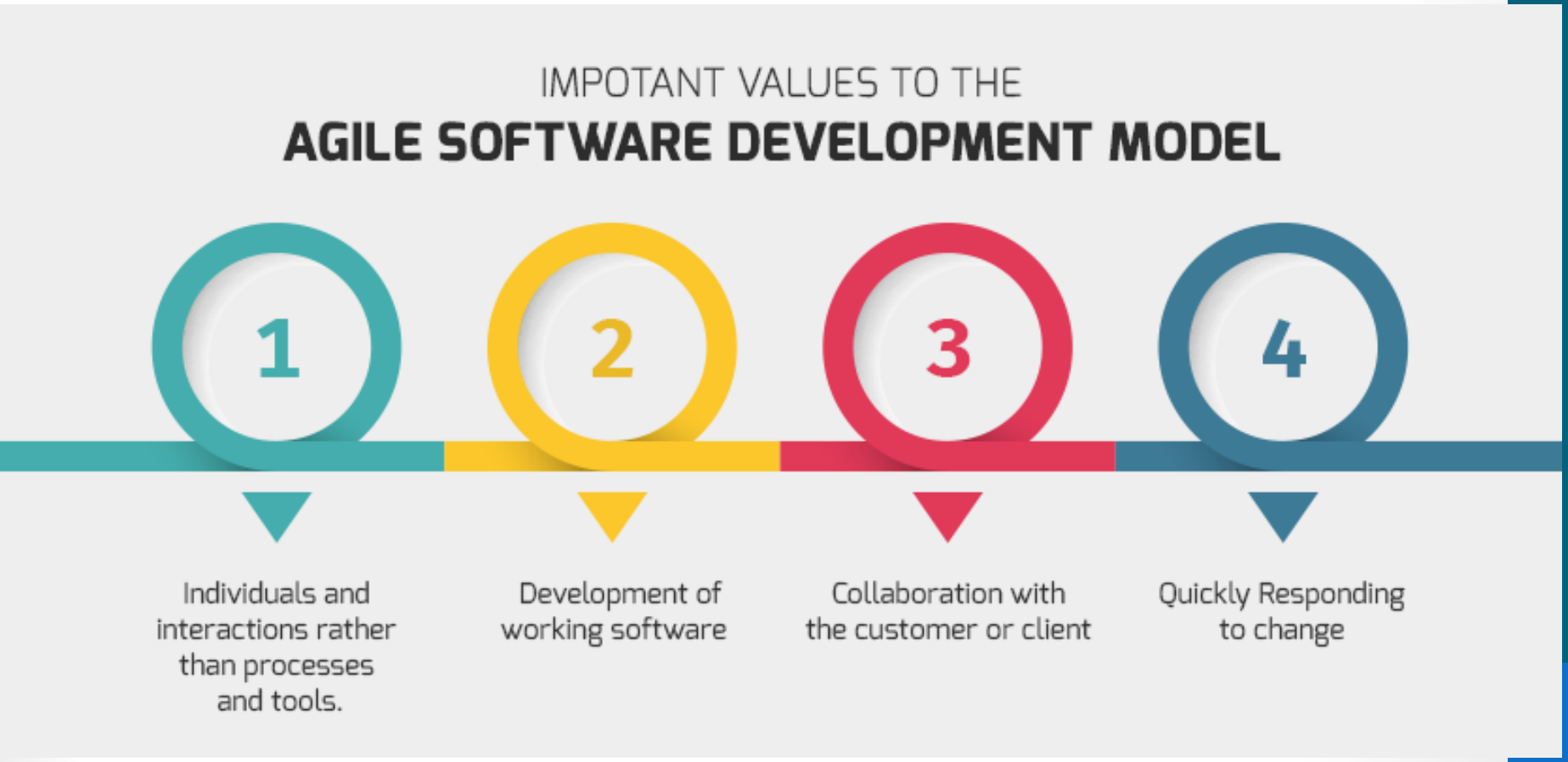
Disadvantages of Spiral Model

- Risk of not meeting the schedule or budget
- It works best for large projects only also demands risk assessment expertise
- For its smooth operation spiral model protocol needs to be followed strictly
- Documentation is more as it has intermediate phases
- It is not advisable for smaller project, it might cost them a lot

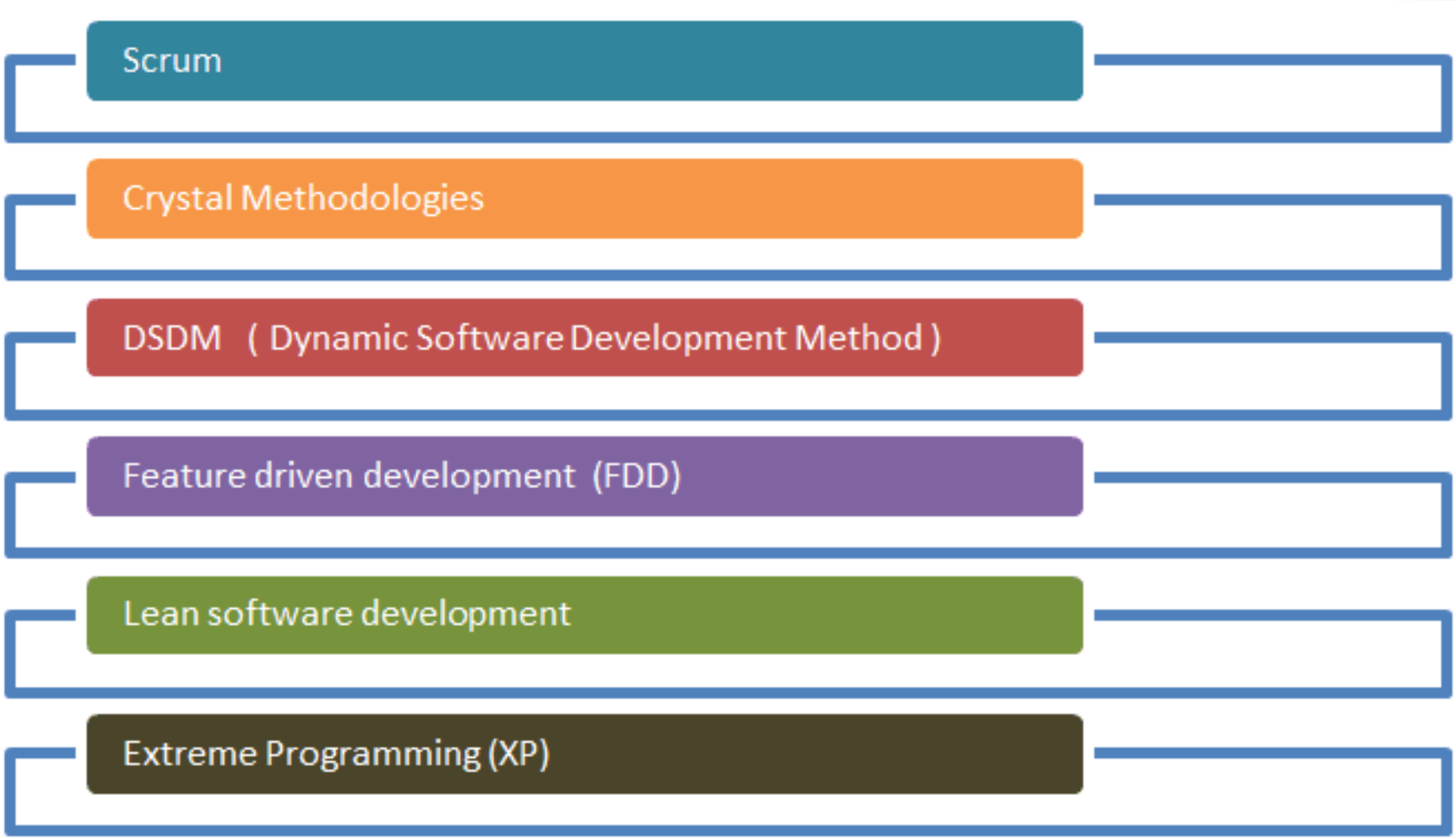
Agile Process Models

- Agile software engineering combines a philosophy and a set of development guidelines
- **Philosophy**
  - Encourages customer satisfaction and early incremental delivery of the software
  - Small highly motivated project teams
  - Informal methods
  - Minimal software engineering work products
  - Overall development simplicity
- **Development guidelines**
  - Stress delivery over analysis and design
  - Active and continuous communication between developers and customers

Agile Process Models (contd.)



Agile Process Models (contd.)



Agile vs. Waterfall Method

Agile Model	Waterfall Model
Agile method proposes incremental and iterative approach to software design	Development of the software flows sequentially from start point to end point.
The **agile process** is broken into individual models that designers work on	The design process is not broken into an individual models
The customer has early and frequent opportunities to look at the product and make decision and changes to the project	The customer can only see the product at the end of the project
Agile model is considered unstructured compared to the waterfall model	Waterfall model are more secure because they are so plan oriented

Agile vs. Waterfall Method (contd.)

Agile Model	Waterfall Model
Small projects can be implemented very quickly. For large projects, it is difficult to estimate the development time.	All sorts of project can be estimated and completed.
Error can be fixed in the middle of the project.	Only at the end, the whole product is tested. If the requirement error is found or any changes have to be made, the project has to start from the beginning
Development process is iterative, and the project is executed in short (2-4) weeks iterations. Planning is very less.	The development process is phased, and the phase is much bigger than iteration. Every phase ends with the detailed description of the next phase.
Documentation attends less priority than software development	Documentation is a top priority and can even use for training staff and upgrade the software with another team

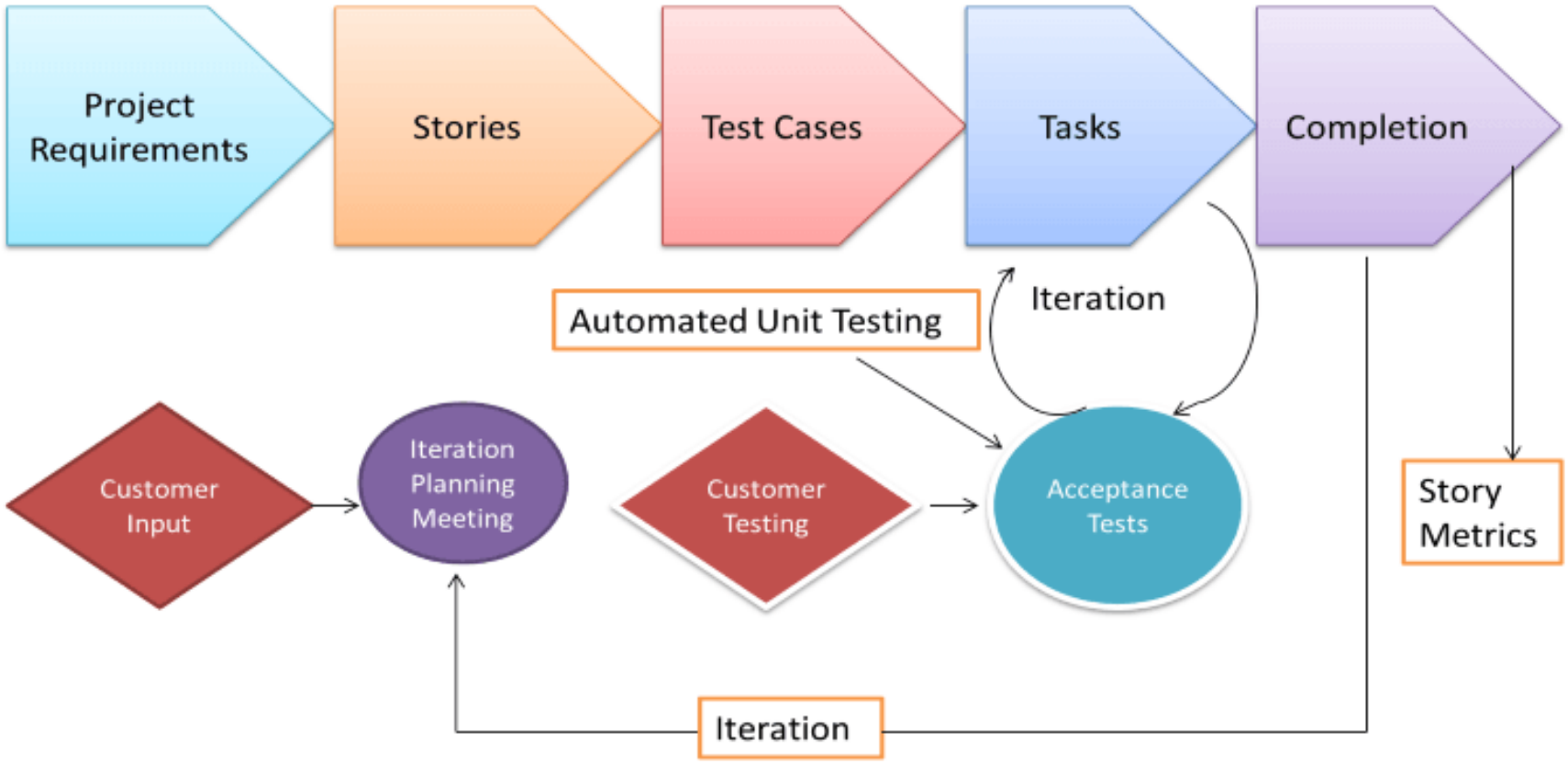
Agile vs. Waterfall Method (contd.)

Agile Model	Waterfall Model
In agile testing when an iteration end, shippable features of the product is delivered to the customer. New features are usable right after shipment. It is useful when you have good contact with customers.	All features developed are delivered at once after the long implementation phase.
Testers and developers work together	Testers work separately from developers
At the end of every sprint, user acceptance is performed	User acceptance is **performed** at the end of the project.
It requires close communication with developers and together analyze requirements and planning	Developer does not involve in requirement and planning process. Usually, time delays between tests and coding

Extreme Programming (XP)

- Somewhat controversial new approach; variation of the incremental model
- First step
  - Determine features that client wants (stories)
  - Estimate duration and cost of each feature
- Client selects stories for each successive build
- Each build is divided into tasks
- Test cases for a task are drawn up
- Pair programming –working with a partner on one screen
- Continuous integration of tasks

Extreme Programming (contd.)



Extreme Programming (XP)

Features of XP

- Computers are put in center of large room lined with cubicles
- Client representative works with the XP team at all the times
- Individual cannot work overtime for 2 successive weeks
- There is no specialization
  - all members of the XP team work on specification, design, code, and testing
- There is no overall design phase before various builds are constructed – Refactoring

Advantages of Agile Model

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of Agile model

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

When to use Agile model

- When **new changes need to be implemented**. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way.
- Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

Unified Process

- Unified process is a framework for OO software engineering using UML (Unified Modeling Language)
- Unified process (UP) is an attempt to draw on the **best features and characteristics of conventional software process models**, but characterize them in a way that implements **many of the best principles of agile software development.**

Unified Process Characteristics

- It is an iterative and incremental development framework
- It is architecture-centric with major work being done to define and validate an architectural design for most coding is done
- It is risk-focused and emphasizes that highest-risk factors be addressed in the earliest deliverables possible
- It is use-case and UML model driven with nearly all requirements being documented in one of those forms

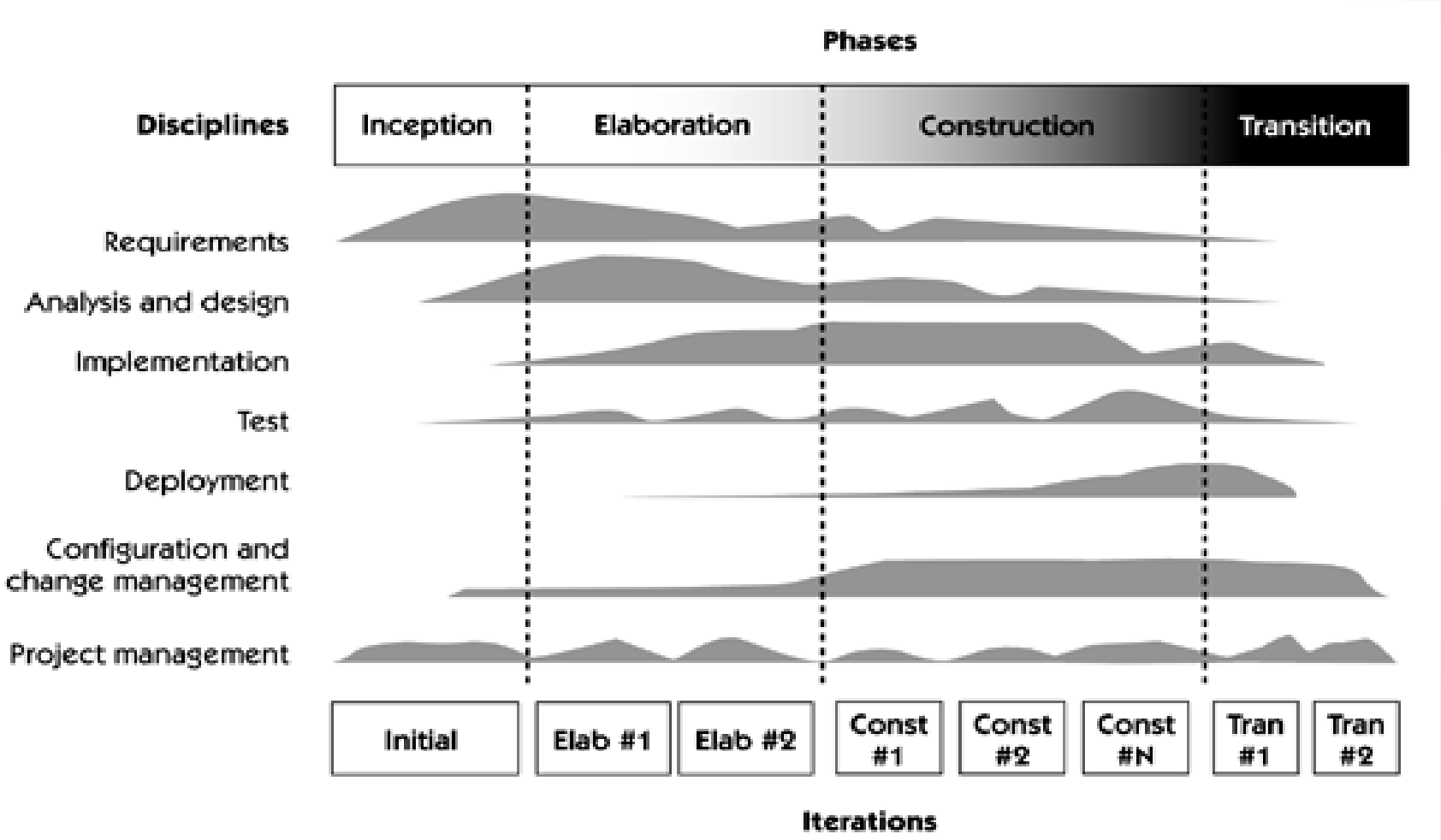
Unified Process: Phases

- **Inception phase**
  - Encompasses the customer communication and planning activities
  - Rough architecture, plan, preliminary use-cases
- **Elaboration phase**
  - Encompasses the customer communication and modeling activities
  - Refines and expands preliminary use-cases
  - Expands architectural representation to include: use-case model, analysis model, design model, implementation model, and deployment model
  - The plan is carefully reviewed and modified if needed

Unified Process: Phases

- **Construction phase**
  - Analysis and design models are completed to reflect the final version of the software increment
  - Using the architectural model as an input develop or acquire the software components, unit tests are designed and executed, integration activities are conducted
  - Use-cases are used to derive acceptance tests
- **Transition phase**
  - Software is given to end-users for beta testing
  - User report both defects and necessary changes
  - Support information is created (e.g., user manuals, installation procedures)
  - Software increment becomes usable software release

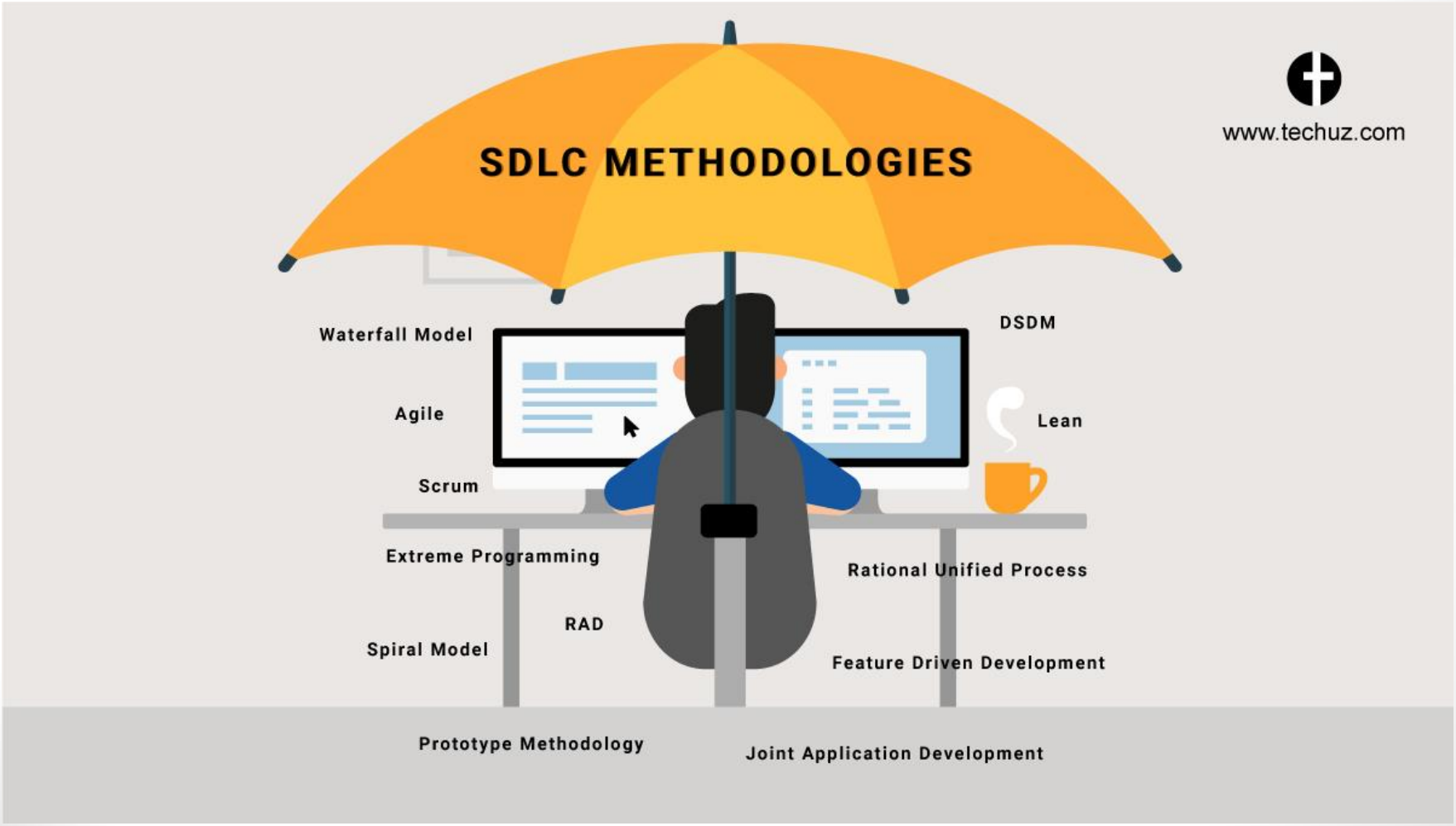
Unified Process Phases



Unified process work products

Inception phase	Elaboration phase	Construction phase	Transition phase
vision document	use-case model	design model	SW increment
initial use-case model	requirements	SW components	beta test reports
initial business case	analysis model	test plan	user feedback
initial risk list	preliminary model	test procedure	...
project plan	revised risk list	test cases	
prototype(s)	preliminary manual	user manual	
...	...	installation manual	
		...	

How to Choose between SDLC Methods?



How to Choose between SDLC Methods?

- To know which is the best model out of all the different types of SDLC models, it **is important to understand that each of these approaches** are suitable for different projects, environments, and requirements.
- For example, if your project is simple and straightforward with set requirements that do not need to be changed, then Waterfall is best suited for it.
- However, if your project is large-scale and consists of multiple components and segments, then choosing Iterative or Spiral methodology would suit your project better.

How to Choose between SDLC Methods?

- To answer the question simply, there is **no ONE model is best from all the SDLC models** discussed.
- A preference of one method over the others cannot be determined.
- However, to select the right SDLC methodologies, you should know all the types of SDLC models, assess the requirements of all the stakeholders and then decide on a method that best fits your needs.

Criteria for deciding on a model include

- Criteria for deciding on a model include
  - Product Complexity
  - Product Size
  - Magnitude of Changes
  - Frequency of Changes
  - Skills of the Dev Team
  - Time constraints
  - Access to Users