Software Process

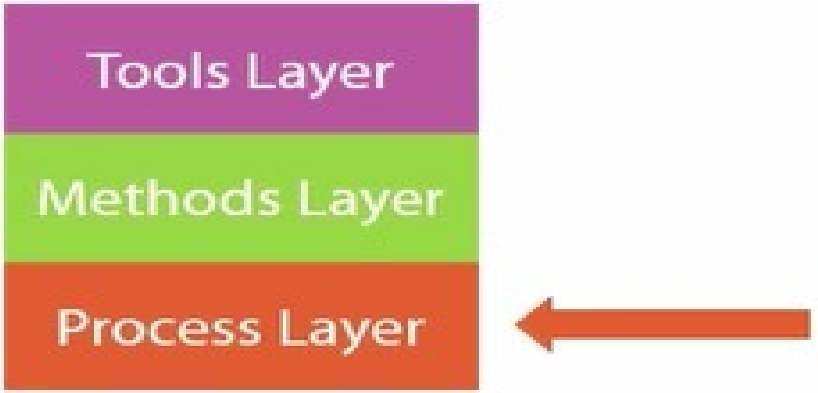
By
Sunil Kumar(Master of Sc.)
Bangalore, India

Agenda

- Process Layer Recap
- Most Common Process Models followed in Software Industries
  1. Waterfall(Linear)
  2. Iterative/Incremental
  3. Prototyping
  4. Spiral
  5. Agile
  6. RUP
  7. Specialized Models
- Summary

Process Layer Recap

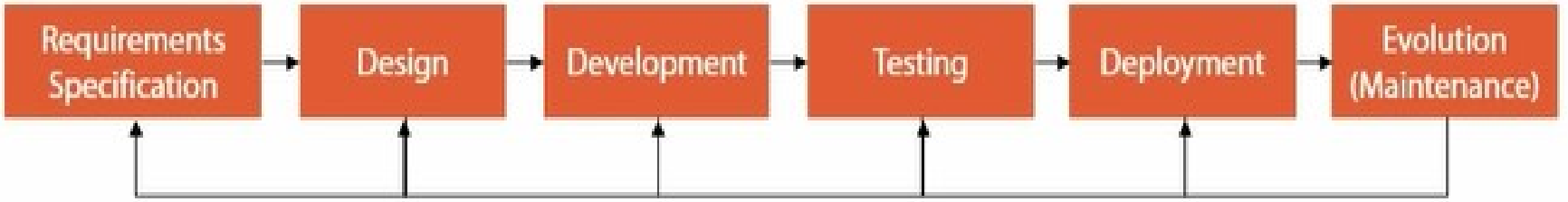
- A Process consists of generic activities: requirements specification, design, development, testing, deployment, evolution.



- A process model indicates activities flow, organization, and artifacts level detail.

1. Waterfall (Linear)

- Systematic and Sequential approach to the software lifecycle.



- This model suggests that each activity finishes completely before the second activity starts without any interaction between these activities.

Waterfall (Linear) contd..

- This model is rarely usable and doesn’t not really describe how practical things are being done, therefore a slightly modified model is practically used. This model uses feedback loops from each phase back the previous phase.
- Even if the feedback loops- changes are difficult to implement
- Customer have to clearly define all requirements at the start
- Customer will wait until the very end to see the first software workable version.

Waterfall : Is it used ?

- When requirement are clear upfront and changes are not expected .
- However even if requirements are fairly stable, still there is a problem with this model because it asks the customer to wait until the very end to see a one-shot workable version of the system.
- Incremental delivery can be very useful to solve this shortcoming of the waterfall model.

2. Iterative/Incremental

- Created in response to the weakness of the waterfall.
- The essence of evolutionary models such as Agile and RUP
- What do iterative and incremental mean? How do they differ ?

Incremental Delivery

- Steps of incremental delivery.
  - Requirements are assigned more priority.
  - Increments are defined- each containing portion of the requirements.
  - Each increment are then analyzed in detail.
  - Increment then goes through design, development, testing, deployment and possibly evolution.

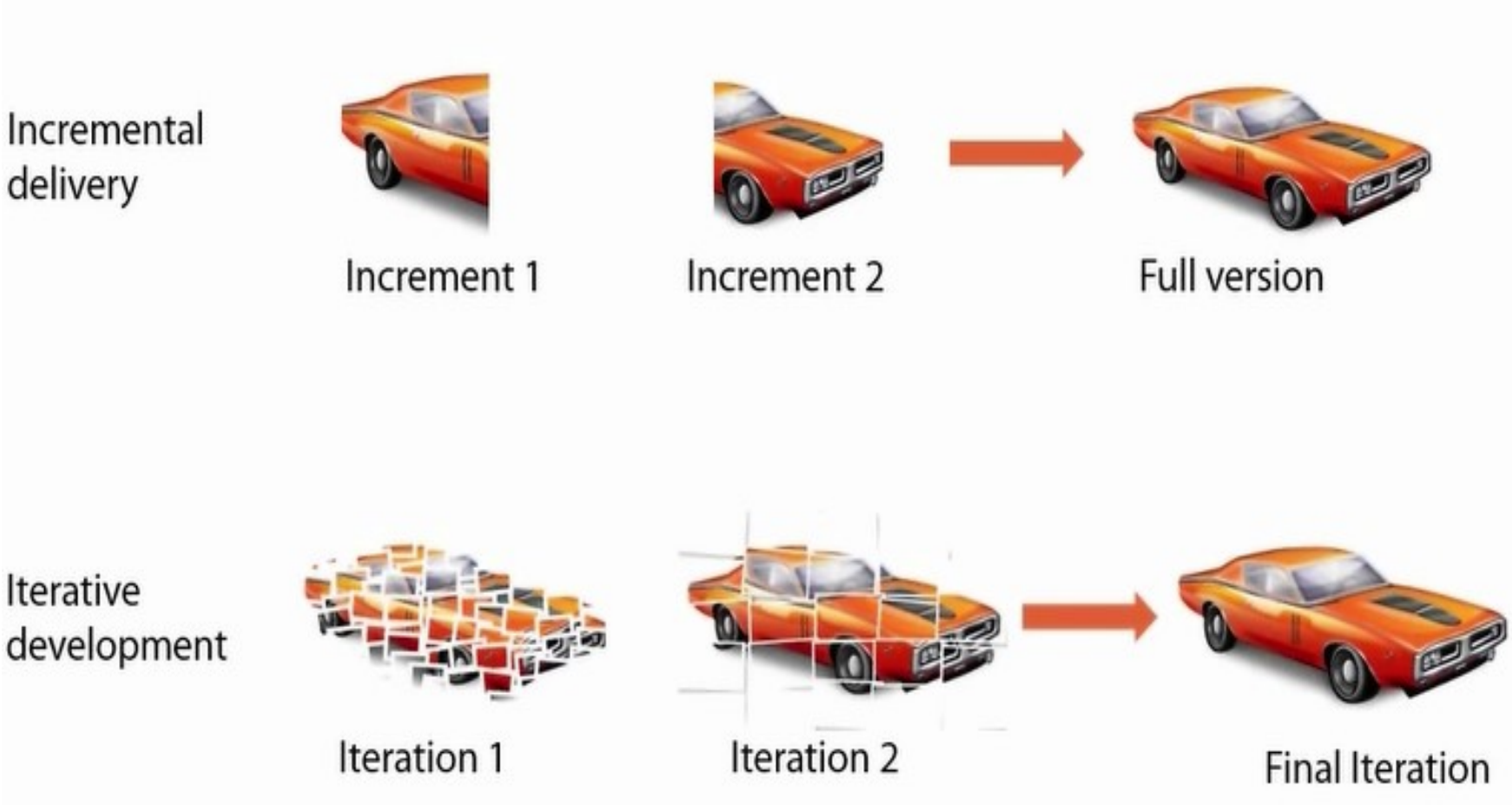
Incremental Delivery Contd..

- Increments are timed-boxed(fixed predefined maximum execution time) which cannot be extended
  - If the time box turned out to be not enough to finish the planned requirement, then requirements are allowed to be shifted to later increments.
  - if the increment requirements are completed before reaching the end of the time box, then requirements from future planned increments can be withdrawn and implemented in the current increment.
- As an increment is finished, it is integrated with the previously completed increments and deployed for customers to see. This way customers can give early feedback, which could derive requirements for future increments.

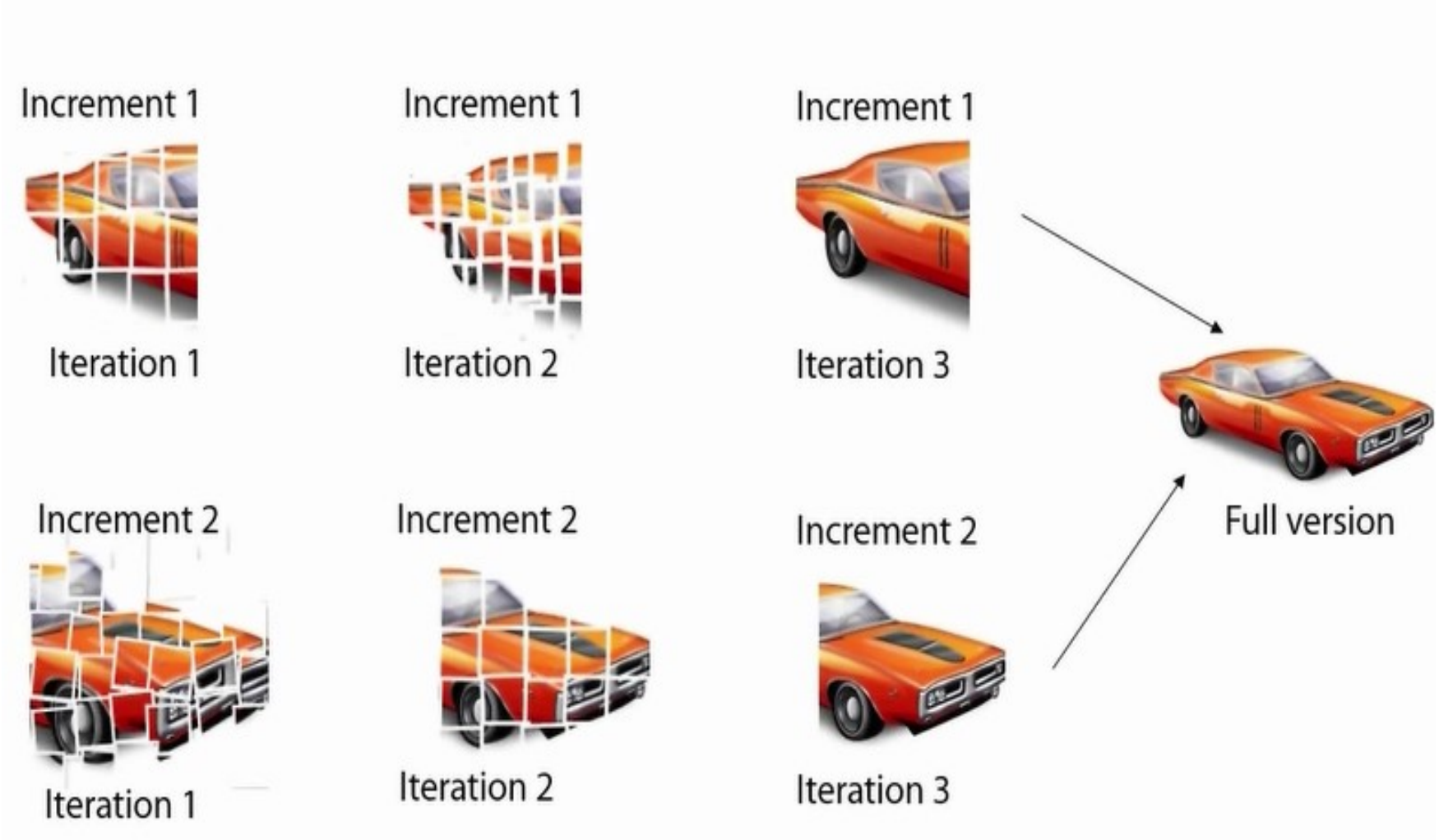
Iterative Development

- Incremental delivery is a scheduling method to deliver requirements
- Iterative development is a method to refine the work done
  - Refinement can be done on a specific increment
  - Refinement can also be done on a full process model(ex. Waterfall).
- Iterative development does not mandate nor it is attached to incremental delivery(although frequently used together)

Ex. Incremental vs. Iterative



Ex. Incremental/Iterative Combined



Incremental/Iterative Combined

- Incremental delivery and iterative development are essential aspects of agile methodologies, such as Scrum with its notion of Sprint.
- The unified process is also built around the concept of incremental delivery and iterative development, although the unified process uses the terms iteration and increment interchangeably.

3. Prototyping

- The prototype is an early sample, model, or release of a product built to test a concept or a process. A prototype is typically used in one of the following cases. When the requirements are not clearly defined, a prototype can be built to derive requirements form the customer.

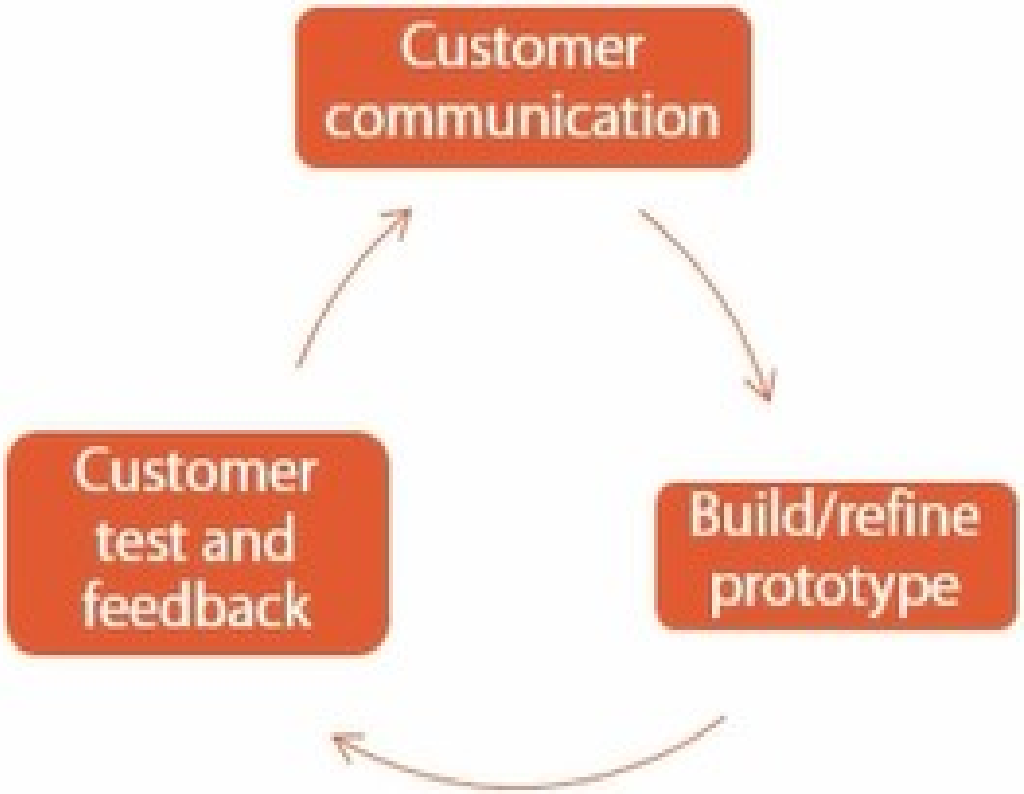
When a prototype used?

- The requirement analysis phase might reveal conflicting or ambiguous requirements
- To derive customer requirements
  - Prototypes derive feedback which helps clarify vague, conflicting and unambiguous requirements
- Mitigate technical and architectural risks
  - Complex architectures are tested
  - New technology are tried

Process Model vs. Technique

- When we talk about prototypes, we need to differentiate between prototyping as a process model versus prototyping as a technique.
- When talking about prototyping as a process model, this means the entire software is built and delivered using prototypes.
- First, requirements are collected from the customer. Next, a mockup is built or revised and the customer performs acceptance testing on the mockup. The cycle continues until a final version of the mockup is put into use.

Prototype-As a process Model



Prototype - As a technique

- When we talk about prototyping as a technique, prototypes are used as a technique within other process models.
- RUP and Spiral model both utilize prototyping technique to mitigate risks known as throwaway prototypes.
  - This means that after the prototype fulfils its purpose, for example to clarify requirements or to mitigate a risk, then it's discarded.

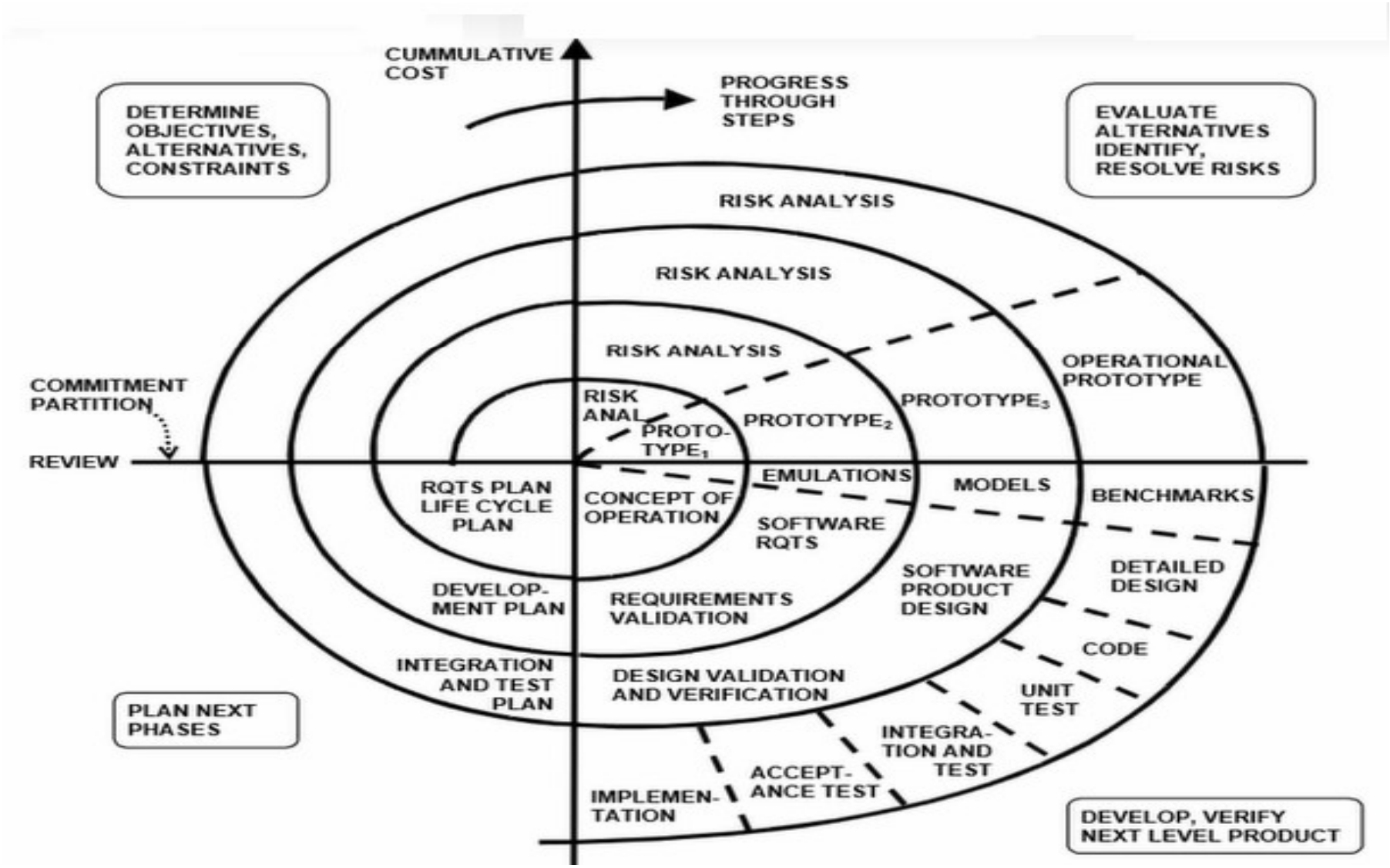
Prototype: Usage

- Prototype process models are not that common
  - Unlikely to have projects with 100% unclear requirements or 100% risky architecture in use cases.
  - There are cases where almost 50% is clear and in this case prototypes are utilized as a tool within other process models such as spiral and RUP.

4. Spiral

- The spiral model is risk-driven
- Combines advantages of incremental and iterative development with the systematic and controlled approach of the waterfall model.
- The spiral model adopts a cyclic approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk.

Spiral Diagram

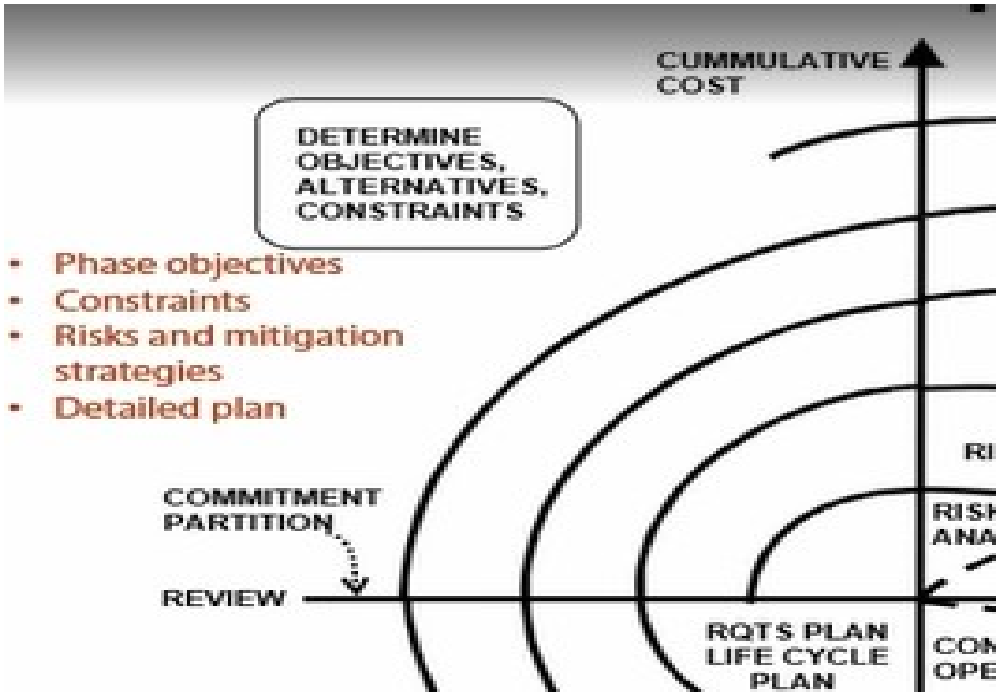


Spiral Contd..

- In spiral model, software is developed using series of iterations where each iterations represents a loop of the spiral and represents a phase of the overall software process.
- Each loop in the spiral model is built into four sections.

Spiral Contd..

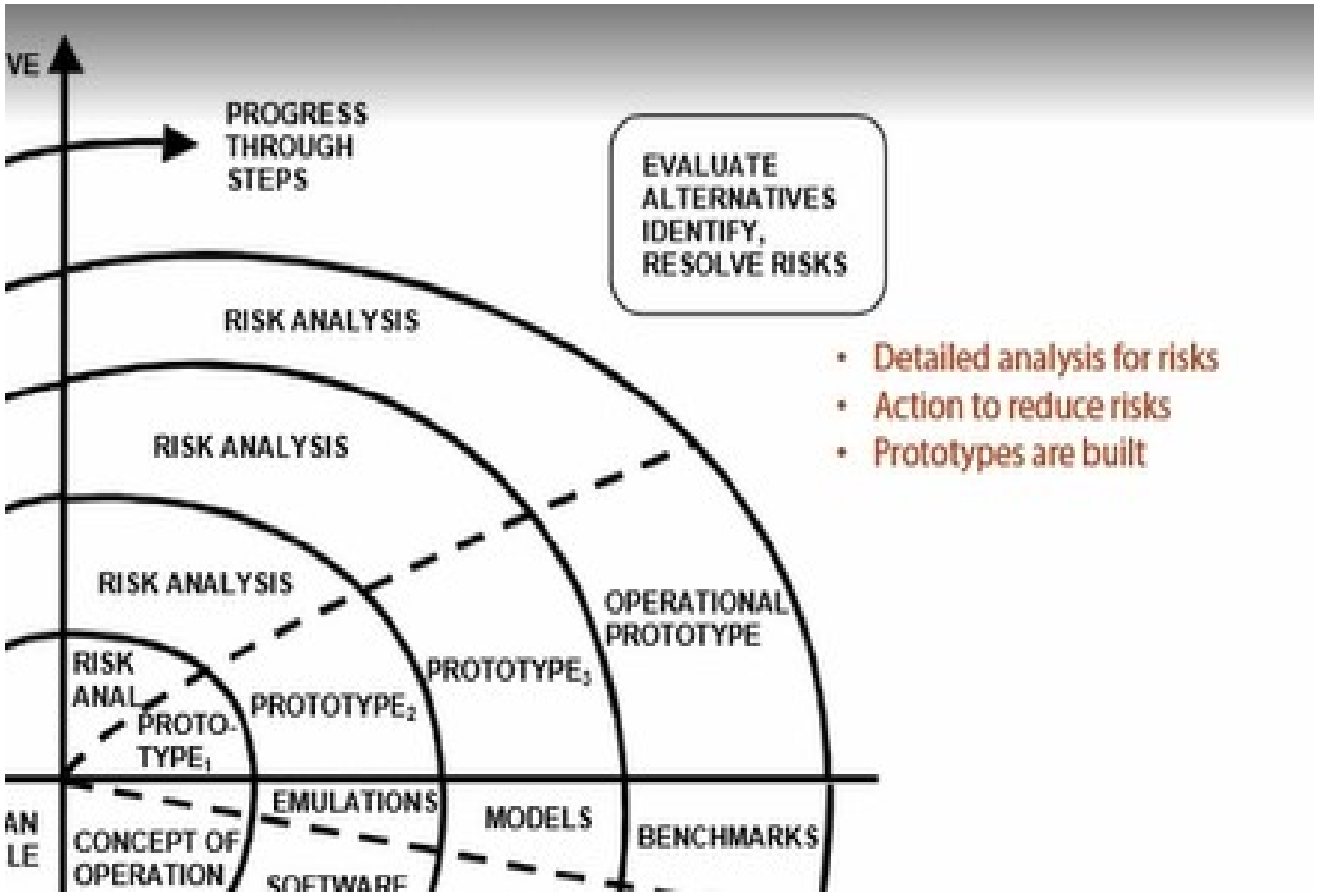
- 1st section is objective setting. Here the objectives of the phases are defined, constraints, risks and corresponding strategies are identified and a detailed plan is created.



Spiral Contd..

- 2nd section, i.e. in the risk assessment and reduction section, a detailed analysis is carried out for each of the risks identified in the objective setting section, and consequently, appropriate steps are taken to reduce these risks.
  - For example, if in the objective setting phase a risk was identified that some requirements are not clearly defined or understood, then in this section a prototype could be built.

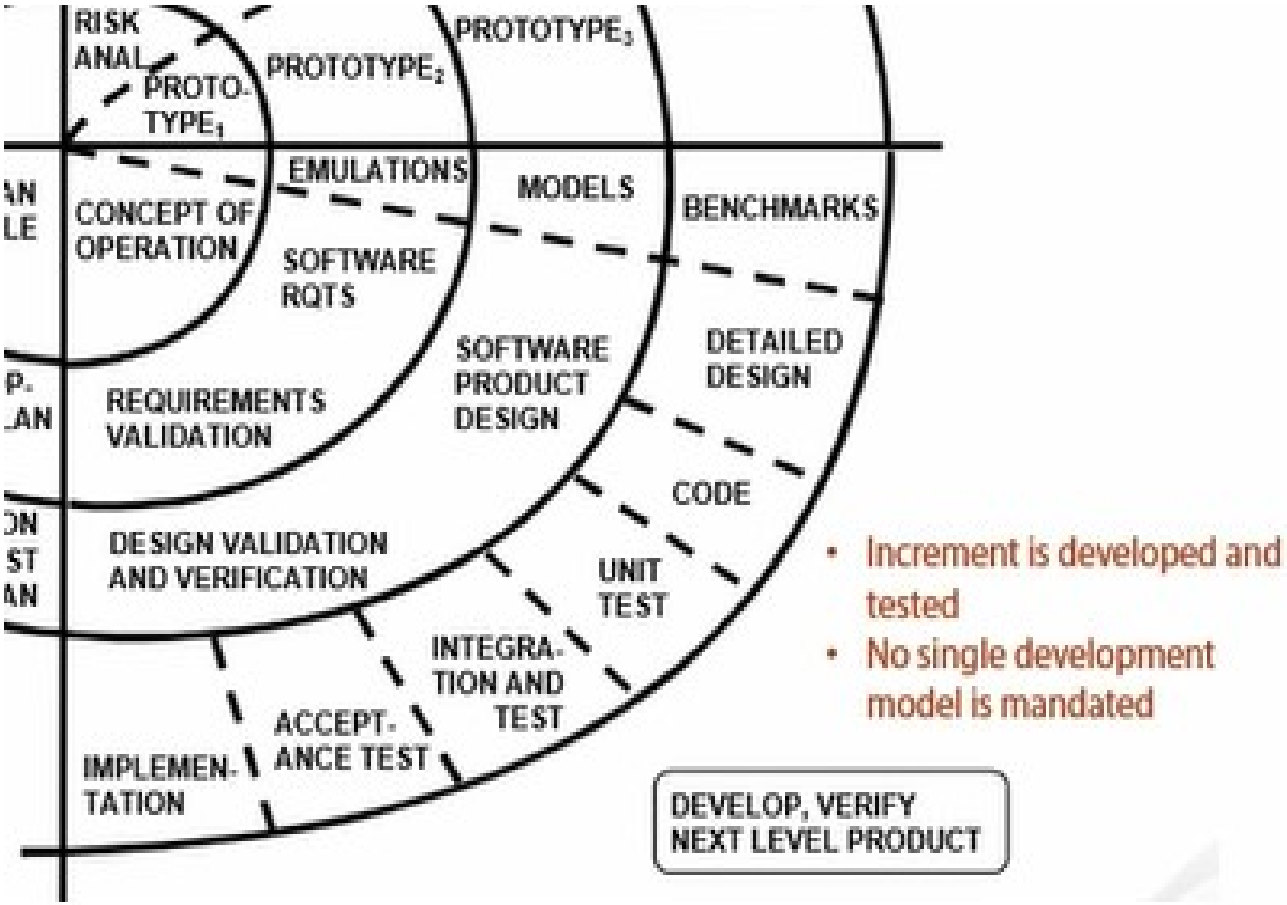
Spiral Contd..



Spiral Contd..

- After having the risk resolved, in the 3rd section, the development and validation, the next increment of the development take place and the result is validated.
- No single development model is mandated and any model can be used based on the current problem statement.
- This could be waterfall, prototyping, agile or any other model.

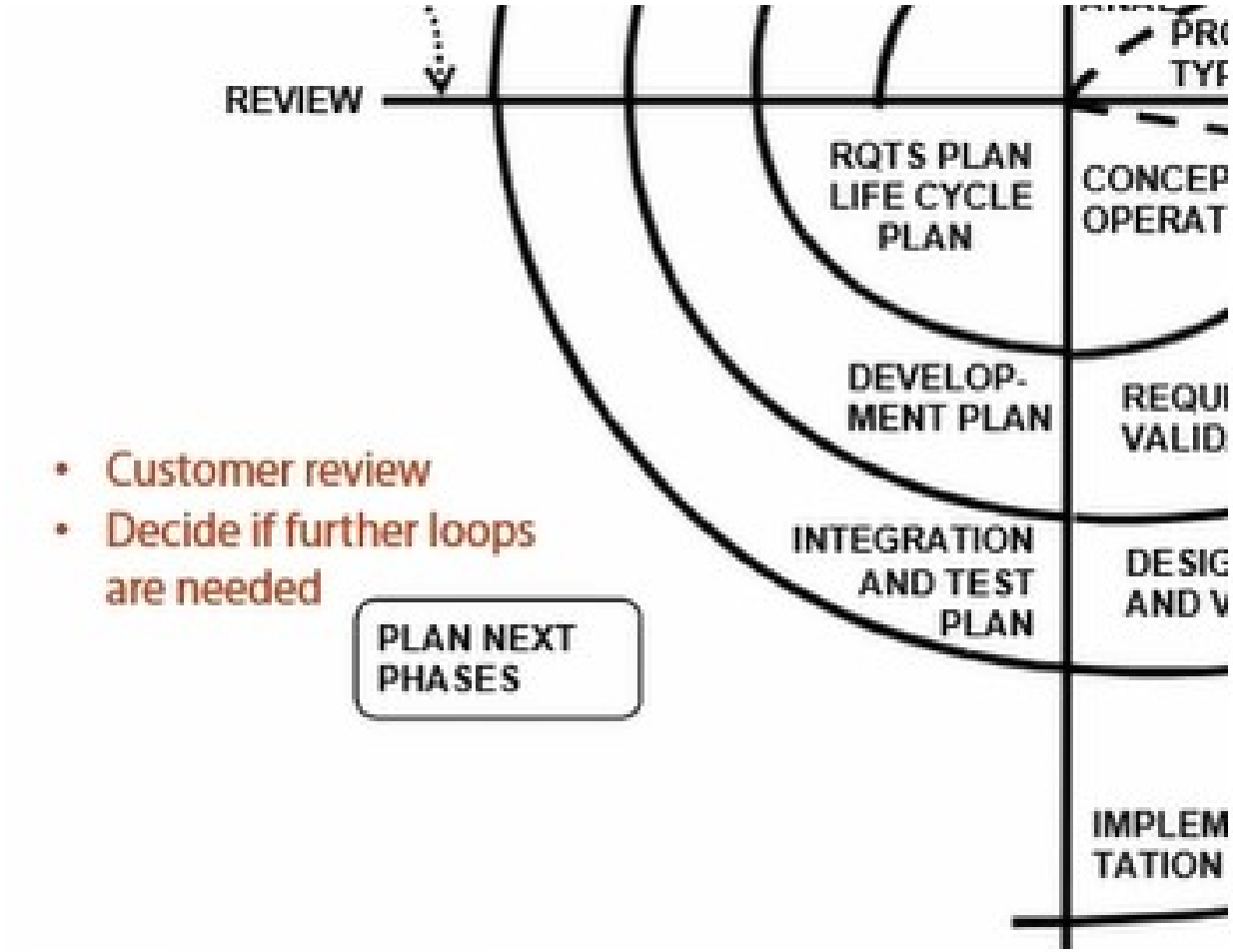
Spiral Contd..



Spiral Contd..

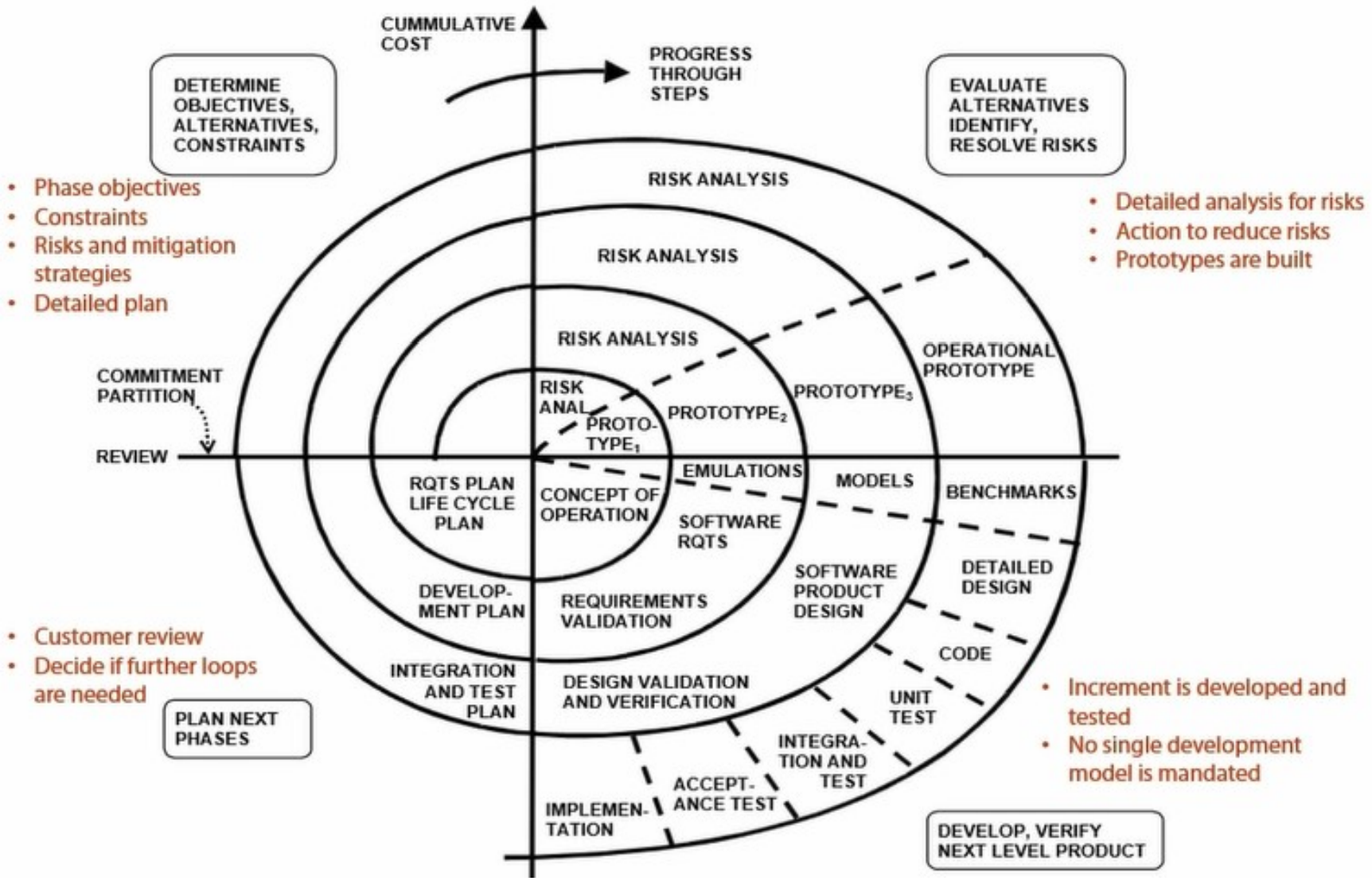
- 4th Section is planning. Customer sees the result and review is conducted.
- Based on review, a decision is then taken whether another loop or iteration of the spiral is needed.

Spiral Contd..



- Customer review
- Decide if further loops are needed

Reading Spiral Diagram



Spiral: Pros & Cons

- Pros
  - Suitable for large and complex projects(risk driven)
  - Adopt incremental delivery
  - Uses prototyping technique to mitigate risks
- Cons
  - Complex to manage
  - Incur high cost (risk analysis and prototyping)
- This model is not suitable for small and medium size projects and/or where Agility is important.

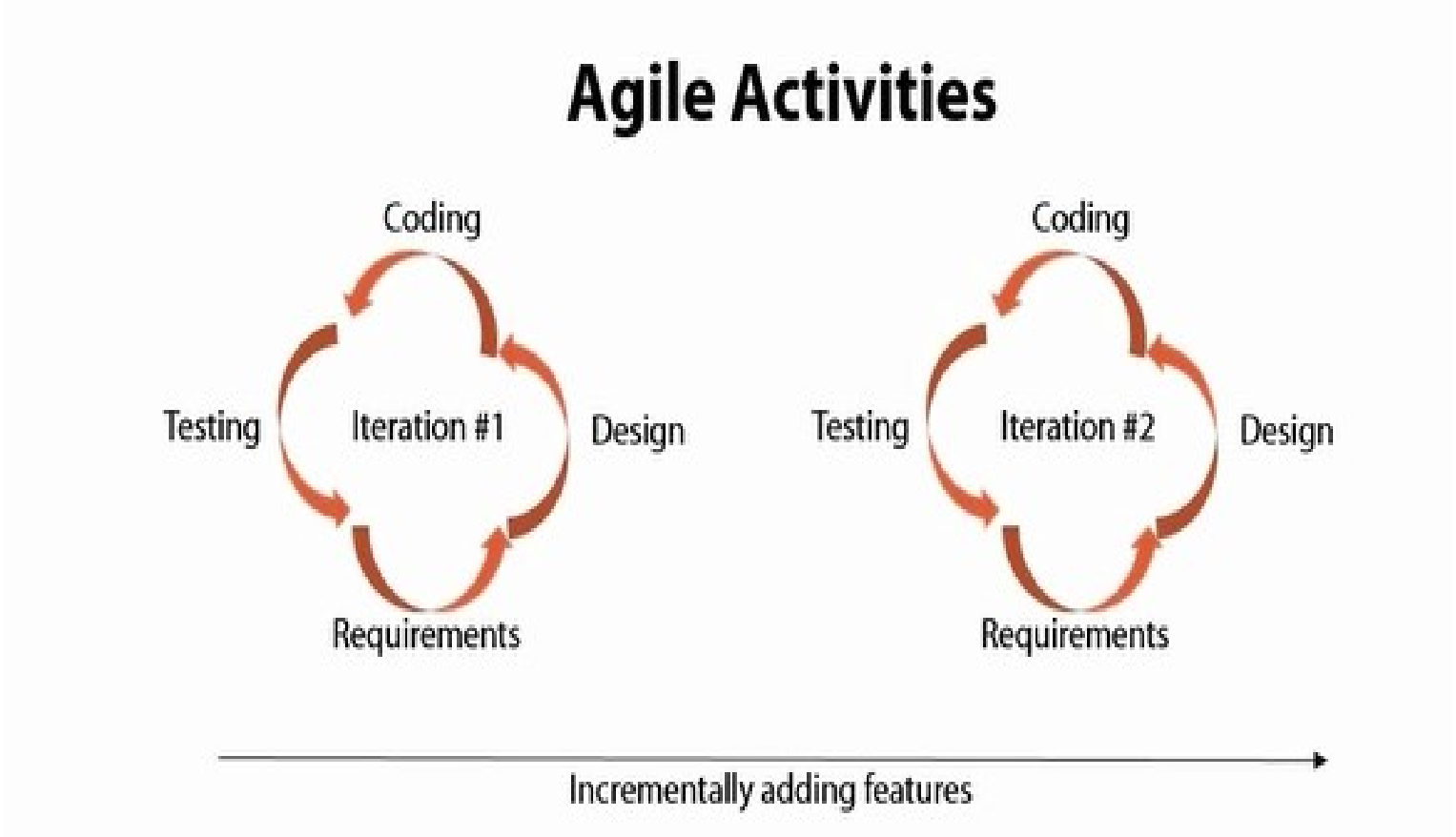
Agile

- Based on incremental/iterative delivery
- Promise of Agile:
  - Create a reliable software
  - Quickly
  - Eliminate waste and overhead
- In agile, software is delivered in a series of iterations, where each iteration implements a subset of the features.

Agile Contd..

- Key difference between Agile and other iterative/spiral model is that it doesn't put clear boundaries between the software engineering activities.
- In each iteration, requirements analysis, design, coding, and testing take place by close collaboration between teams. There is no emphasis on any single activity in any given iteration.
- The focus is always to create reliable software quickly and with minimal overhead.

Agile Contd..


Agile Activities
Coding
Testing
Iteration #1
Design
Requirements
Coding
Testing
Iteration #2
Design
Requirements
Incrementally adding features

Agile Manifesto

- Created by a group of Agile leaders
- The philosophies that underline Agile methods.
- Manifesto has values and derived principles.

Agile Manifesto Values

Values	Explanation
Individual and interactions over processes and tools	• Strong collaborating team > process tools • Coherent team > Individual gurus • Put more effort in building coherent teams
Working software over comprehensive documentation	• Documentation is important • Too much documentation is costly and documents become outdated • Generate just enough high-level structure/dynamics documents.
Customer collaboration over contract negotiation	• Continuous customer feedback is essential • Customer work closely with team • Feedback derive future iterations

Respond to change over following a plan	• Plans must be flexible to change
	• Change can and will happen
	• Multi-level planning.
	  • Weekly detailed plans
	  • Iteration plan
	  • Overall plan

Agile Manifesto Values

More Values Items	Valued-but less than left items
Individual and interactions	Over processes and tools
Working software	Over comprehensive documentation
Customer collaboration	Over contract negotiation
Respond to change	Over following a plan

- While there is value in the items on the right, Agile values the items on the left more.
- Qs. How much planning we do? To what extent we follow a process? And how much analysis and design documentation and models we generate?
- Ans. Do just enough planning and documentation, and follow just enough of a process that helps you achieve the Agile values listed on the left.

Agile Manifesto Principles

- The agile values derives a set of agile principles that characterize agile methods.
- These principles are
  - The highest priority is to satisfy the customer through early and continuous delivery of working software.
  - Welcome changing requirements, even late in the development process. Agile processes harness change for the customer's competitive advantage.
  - Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Agile Manifesto Principles

- Business people and developers must work together daily throughout the project. They give early and continuous feedback, which drives future iterations.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face communication.
- Progress is measured by working software, not by the level of adherence to processes and practices.

Agile Manifesto Principles

- Sustainable development is important as an Agile team maintains a constant pace. This means that a team's energy is not burnt out trying to over-deliver.
- Continuous attention to technical excellence and good design enhance agility.
- Simplicity- the art of maximizing the amount of work not done – essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

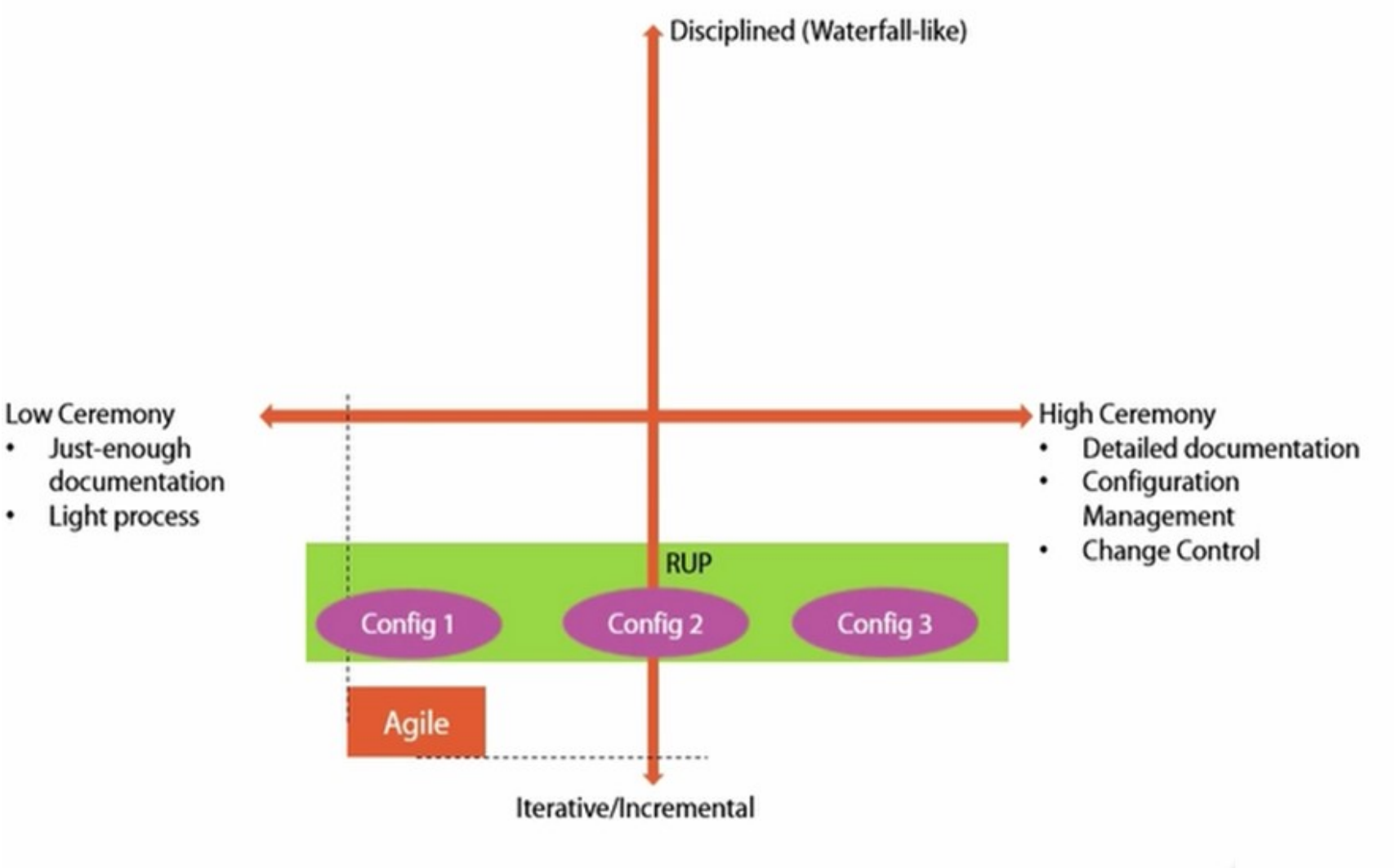
Agile Methodologies

- Agile methodologies share manifesto values and principles- although they might differ in activities. Some of the most known methods are
  - Scrum
  - XP (Extreme Programming)
  - Lean Software Development
  - Feature-Driven Development
  - Dynamic System Development

RUP

- Rational Unified Process(RUP)
  - Based on incremental/iterative delivery
  - Driven by risk
  - Development approach is use-case driven and architecture centric
- RUP is also a process framework
  - Process configuration are created via process customization and authoring which allows the creation of process configuration.
  - These configurations can support different team sizes and either disciplined or less formal development methods.

RUP Configuration



RUP Configuration Analysis

- On the y axis, we measure the process delivery method by two extremes, at one end, a waterfall-like delivery, and at the other end, an iterative and incremental delivery.
- On the x axis, we measure the process level of ceremony, which indicates the level of documentation, configuration management, and the change control.
- At one end we have high ceremony processes, which generate a lot of documentation and exert a high level of configuration and a change control. On the other end, we have low ceremony processes where documentation is minimal and flexibility is high.

RUP Configuration Analysis

- As we know, agile processes are highly iterative and sacrifice ceremony and strictness in favor of high flexibility. Therefore, Agile processes fall onto the bottom left corner of this diagram.
- As it was mentioned in previous slide, RUP is incremental and iterative, so it must be placed somewhere in the bottom portion of the diagram. However the special thing about RUP is that it's a process framework, which means it can be adapted to whatever level of ceremony required. Therefore, we can have various configurations ranging from low, to medium, to high ceremony.

Phases & Milestone

- At the start of this section, it was mentioned that RUP is risk-driven, and its development approach is both use-case driven and architecture-centric.
- Let's explore this statement more by understanding a main differentiator in RUP, phases and their associated milestones.
- The RUP lifecycle is made up of four phases, inception, elaboration, construction, and transition. These phases occur in a linear fashion so that each phase must end before the next one starts; however, each phase is made up of various iterations.

Phases & Milestone