




Software Engineering

Course's Code: CSE 305



Chapter 2. Software Process Models

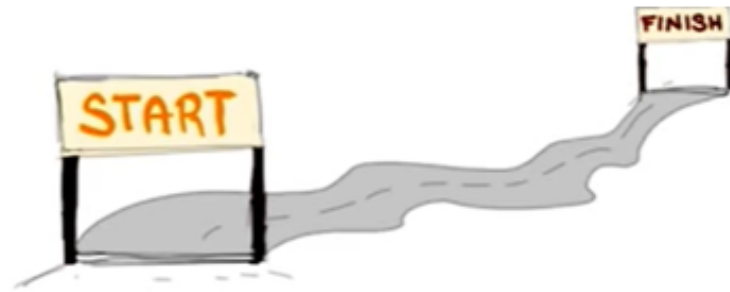
2.1 Classification of Software Process Models

2.2 Predictive Process Models

2.2. Adaptive Process Models

Software Process Models/ SDLC Models

- A software process model is an abstract representation of a process.
- It presents a description of a process from some particular perspective.



- What should happen from very beginning to the very end of a software development process



Software Process Models/ SDLC Models

➤ **Who does it?**

- ☐ Software engineers and their managers adopt a process model to their needs and then follow it.

➤ **Why it is important?**

- ☐ Because it provides stability, control and organization to an activity.

➤ **Can we create our own process model?**

- ☐ Of course, you can create your own, but in this course, we are going to talk about all these different industry standard models that are out there that we can use for a team.



Software Process Models/ SDLC Models

➤ Why do we need so many models? Couldn't you just have one model and then everybody follows it?

- ❑ Every team, every project, every organization is different and the constraints/conditions that are on any project are different.
- ❑ One project, one model, may fit a particular project, whereas the same model may not fit for another particular project.
- ❑ So that's why it's important to learn about different models so that when you are in a situation to select a model for your project, you can choose the right model..



Classification of Software Process Models/ SDLC Models

There are a number of criteria that can be used to classify SDLC models. Some of the most common criteria include:

- 1. Predictive vs. adaptive**
- 2. Sequential vs. iterative:**

Classification of Software Process Models/ SDLC Models

- Predictive means that you have a pretty good understanding of the requirements of the software or the product that you are building
- So in this case, the client or the analyst or the customer knows what exactly they want and so they have a very high confidence of the requirements or what they're looking for.
- Once the requirements are defined completely, then the team goes through this design, implementation, testing and so on
- Then finally produces the product that the client is looking for and they get exactly what they are looking for, what they had originally in mind

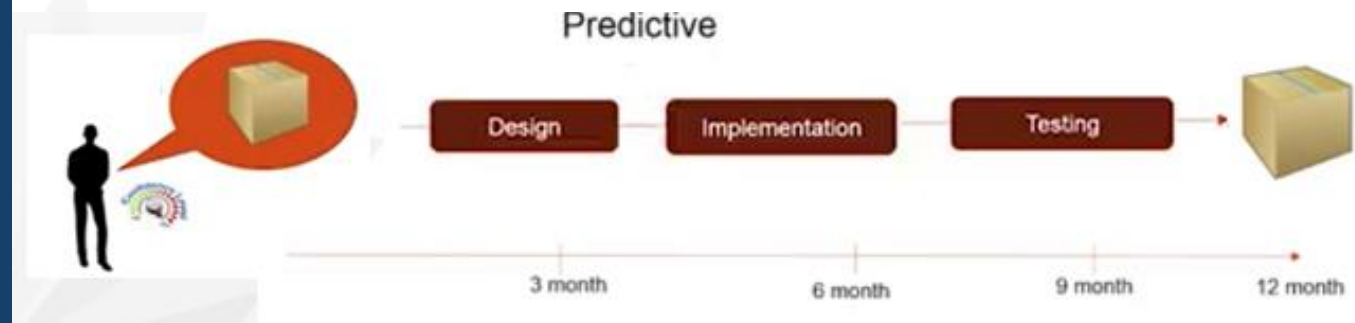


Classification of Software Process Models/ SDLC Models

- Predictive
Vs Adaptive

Predictive

- Since the requirements are known in the beginning, there is generally a desire not to allow changes during the development phase.
- So generally in this case the changes are not desired and so it's whatever we decided to implement, we continue with that during the development process.

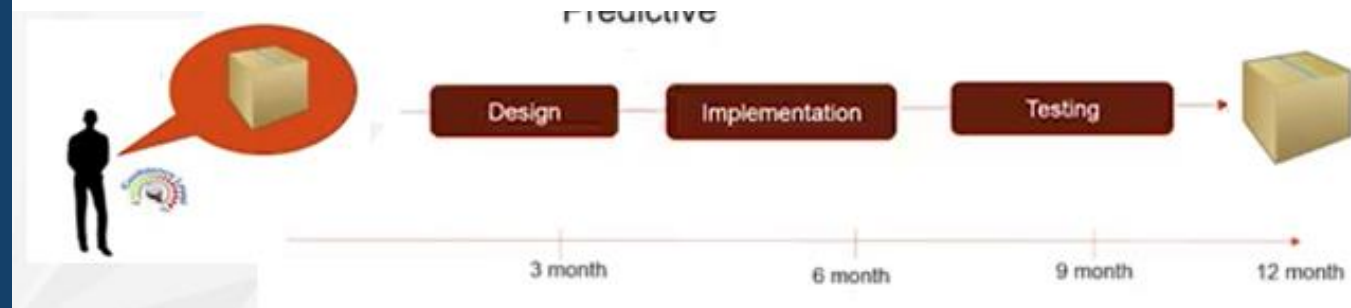


Classification of Software Process Models/ SDLC Models

• Predictive Vs Adaptive

Predictive

- Follow **sequential** approach
- Have **well-defined requirements**
- Have **low risk of change**
- *Waterfall model
- *V-model
- *Spiral model

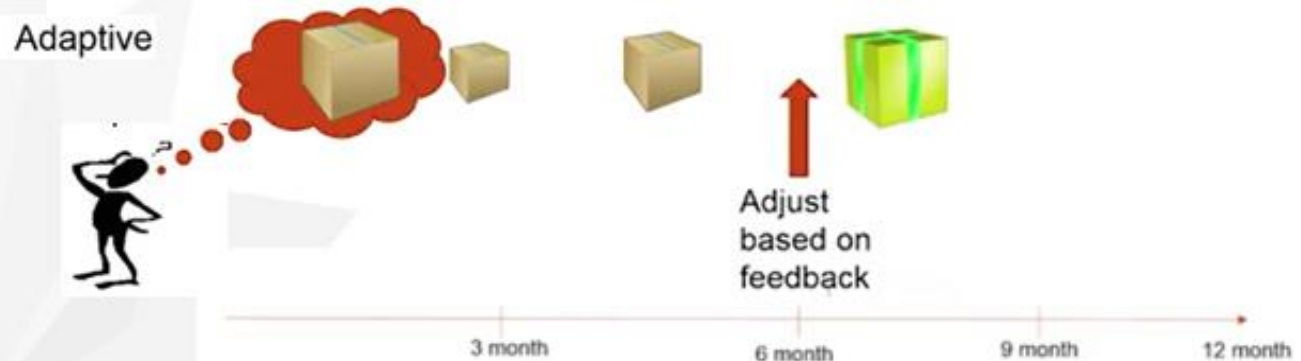


Classification of Software Process Models/ SDLC Models

• Predictive Vs Adaptive

Adaptive

- in adaptive models, the client or the customer generally has an idea of what they want to build, but not quite there, right?
- So they have an idea but they are not 100% sure what they want to build. So in this case, they start with an idea.
- The team that is working on it, they actually build something like a really small version of it or like a low fidelity version of it.
- then they show it to the customer or the actual users using it.
- then they build the next version based on the feedback. And so they change as they are getting the feedback.



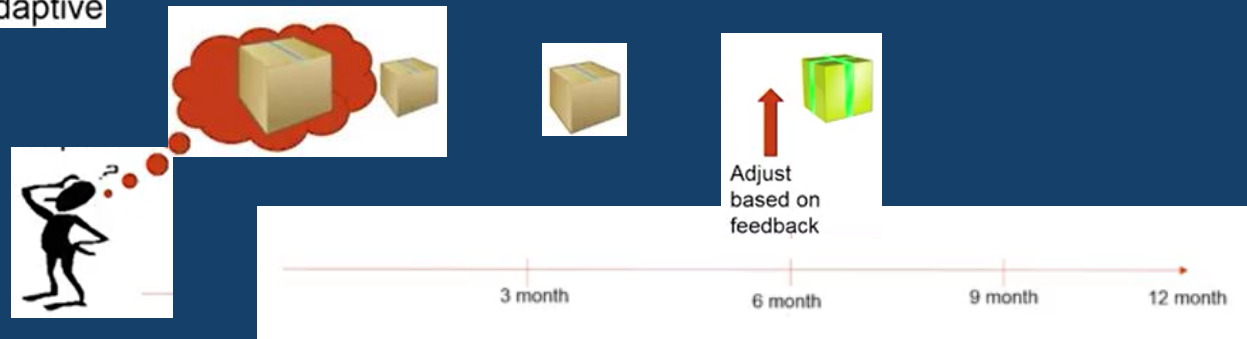
Classification of Software Process Models/ SDLC Models

• Predictive Vs Adaptive

Adaptive

- in adaptive models, the client or the customer generally has an idea of what they want to build, but not quite there, right?
- So they have an idea but they are not 100% sure what they want to build. So in this case, they start with an idea.
- The team that is working on it, they actually build something like a really small version of it or like a low fidelity version of it.
- then they show it to the customer or the actual users using it.
- then they build the next version based on the feedback. And so they change as they are getting the feedback.

Adaptive



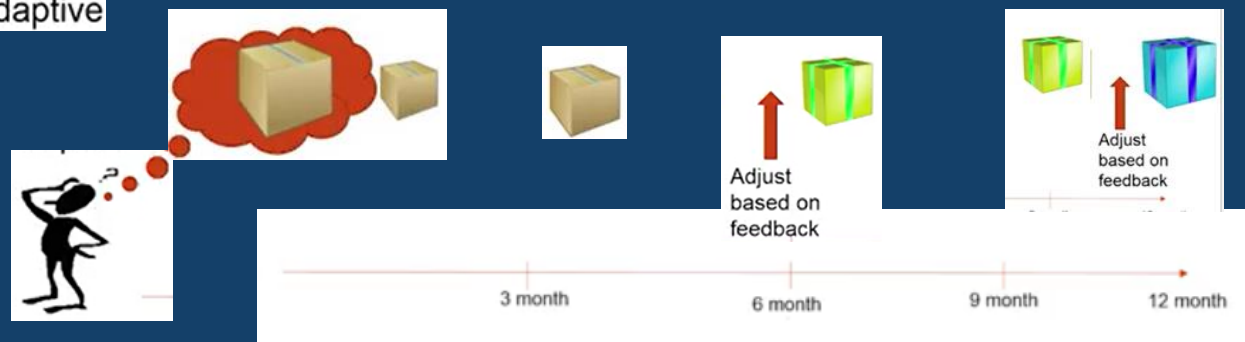
Classification of Software Process Models/ SDLC Models

• Predictive Vs Adaptive

Adaptive

- So let's say after the two increments, they got a major feedback that says "no, we need to go in a different direction."
- So again, then the team moves in the different direction and starts morphing the product into something else. And so on.
- In the end, you might end up getting something different than you were originally thinking, but actually what you get is what users need because all along they were getting feedback as to what will work or not work and they build a product

Adaptive



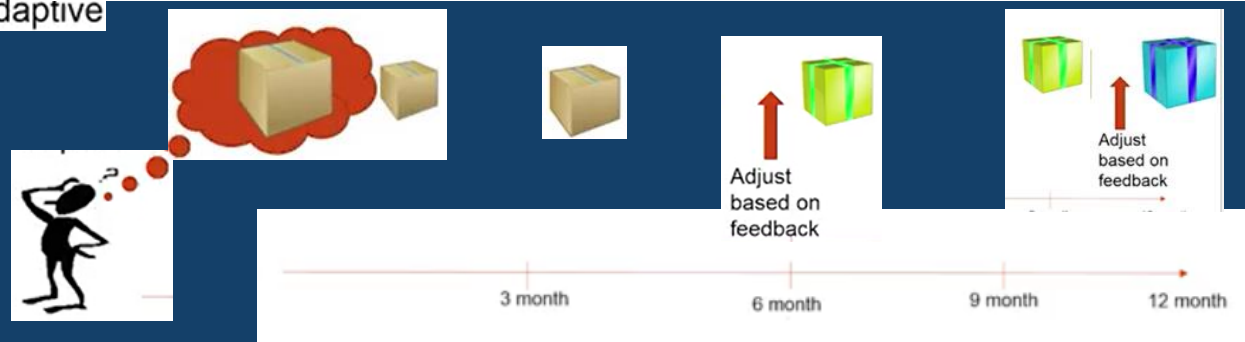
Classification of Software Process Models/ SDLC Models

• Predictive Vs Adaptive

Adaptive

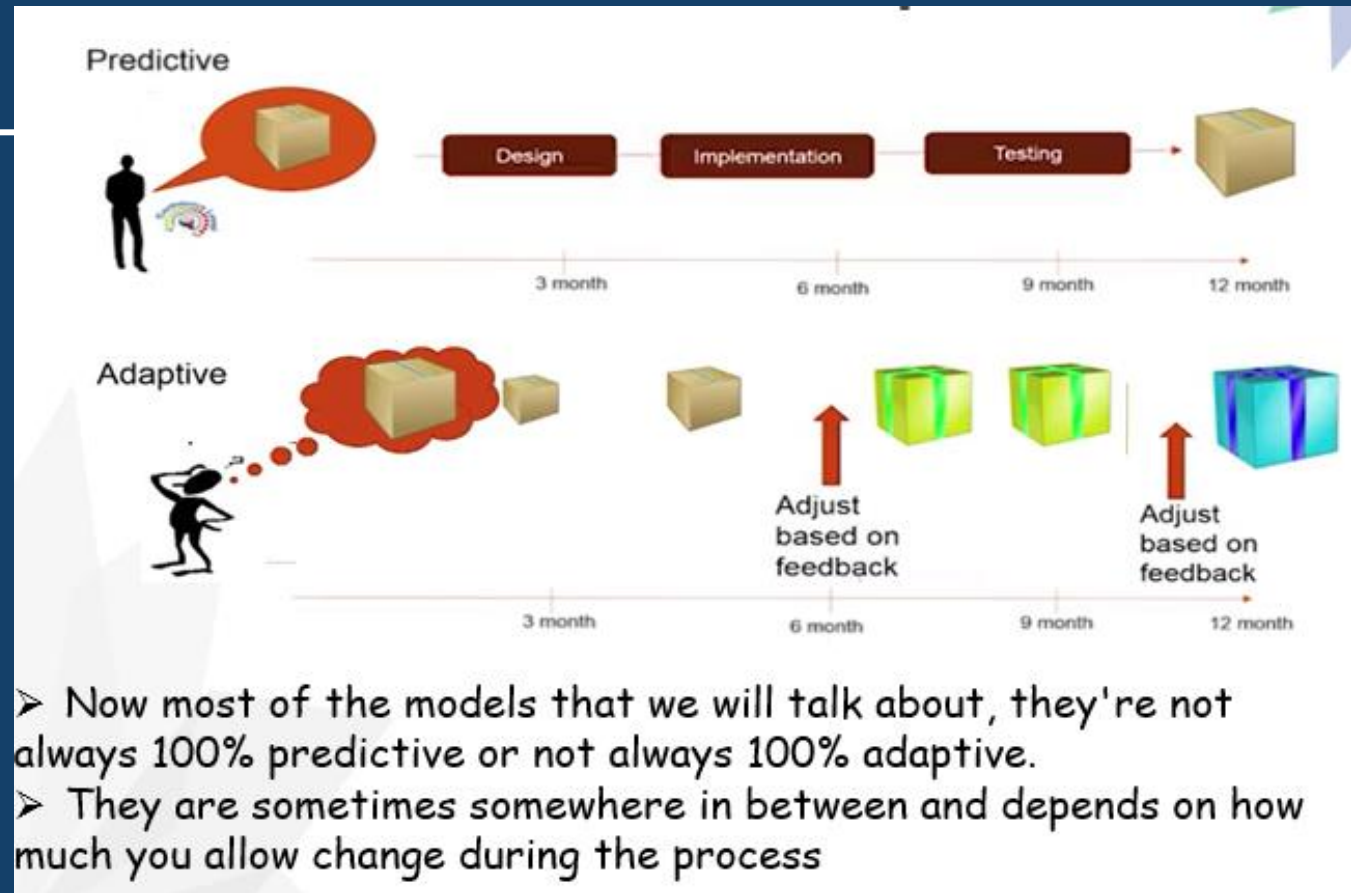
- Can be break down into **smaller, more manageable** iterations
- have **complex** requirements
- have **high risk** of change.
- *Agile models (Scrum, Kanban, Extreme Programming)
- *Incremental model

Adaptive



Classification of Software Process Models/ SDLC Models

- **Predictive
Vs Adaptive**



Incremental

Classification of Software Process Models/ SDLC Models

- Predictive Vs Adaptive
- Iterative Vs Incremental

- in an incremental model, you have a fairly good idea of what you want to build,
- but instead of building it in one shot, like we saw in the predictive model, you build in increments.
- So for example, let's say I wanted to build a car.
- So in this case, the first step, I can just build all the tires, I can create all the tires.
- Once I create the tires, then maybe in the next increment, I create the chassis for the automobile or the car.

Incremental



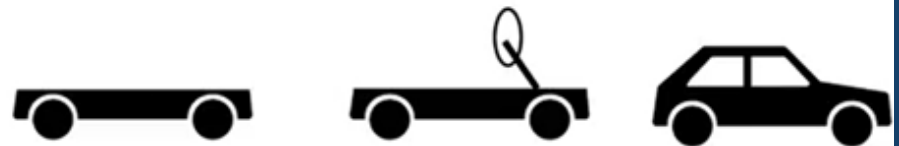
Incremental

Classification of Software Process Models/ SDLC Models

- Predictive Vs Adaptive
- Iterative Vs Incremental

- Then maybe in the next increment I can put the steering, maybe the engine and some more pieces and so on.
- And then finally in the end, I build my car.
- So as you can see that in the increment you still have a good idea of what you want to build but you're building in increments.

Incremental



Classification of Software Process Models/ SDLC Models

- Predictive Vs Adaptive
- Iterative Vs Incremental

Iterative

- In iterative model, again you don't have that clear of an idea, but some idea.
- So for example, let's say your idea was to go from place A to place B. So in an iterative model, you will think about, you need something to go from place A to B.
- we're not sure if this will be really needed or not needed.
- So instead of kind of building the car, why don't we just build a bicycle first which is less expensive or less effort and see if we need that or we need to morph it.
- And so you build a bike and then say, well, it takes a lot of time. So the next iteration you might build a bike, and put an engine to your bike, and now it becomes a motor bike. And so on.

Iterative



Iterative

Classification of Software Process Models/ SDLC Models

- Predictive Vs Adaptive
- Iterative Vs Incremental

- And at some point you might build a car, or you might make a semi-truck, or something like that.
- But in an iterative model, you sometimes are actually replacing what you've built with something different.

Iterative



...



Classification of Software Process Models/ SDLC Models

- Predictive Vs Adaptive
- Iterative Vs Incremental

Incremental



Iterative



- In incremental, you're just breaking the product in smaller pieces. In iterative model, enhancements are happening
- some of the models actually have both going together
- And there are some models which have neither increment nor iterative, they are purely one-shot, big bang development

Classification of Software Process Models/ SDLC Models

- **Predictive Vs Adaptive**
- **Iterative Vs Incremental**

Questions?

Q. In predictive models, change during the development is expected.

a. True

Answer: False

b. False

Q. Incremental models are always predictive models.

a. True

b. False

Answer: False. It can be both predictive and adaptive



Chapter 2. Software Process Models

2.1 Classification of Software Process Models

2.2 Predictive Process Models

2.2. Adaptive Process Models

Predictive Process Models

SDLC Models

<MODEL NAME>



- Information, Characteristics of the model

USE

- Situation in which you can/should use this model



- Pros / Advantages of this approach



- Cons / Disadvantages of this model, approach

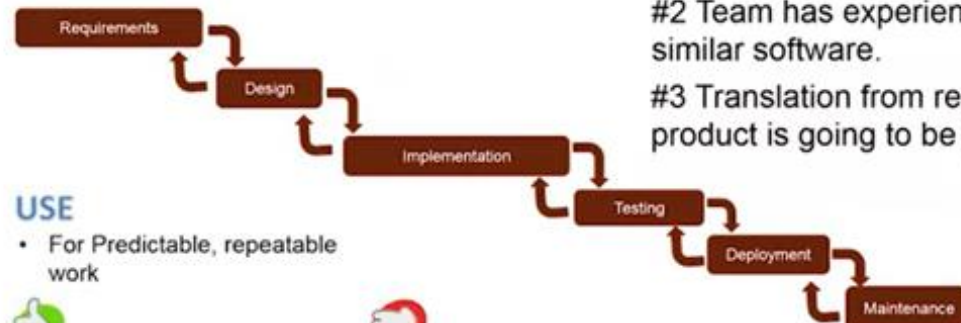
Where does this model stand in predictive vs adaptive scale.



Waterfall Model

Waterfall Model

WATERFALL MODEL



USE

- For Predictable, repeatable work



- Simple, Easy to understand
- Predictability
- Efficient



- Not flexible for change
- First release takes a long time



#1 We know requirements very well. They won't change.

#2 Team has experience building similar software.

#3 Translation from requirement to product is going to be perfect



Predictive Adaptive

3 month

6 month

9 month

12 month

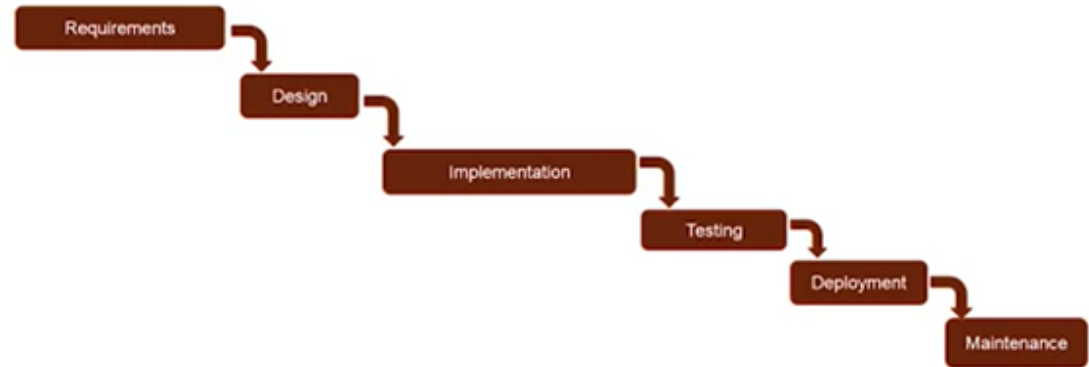
Waterfall Model

Waterfall Model



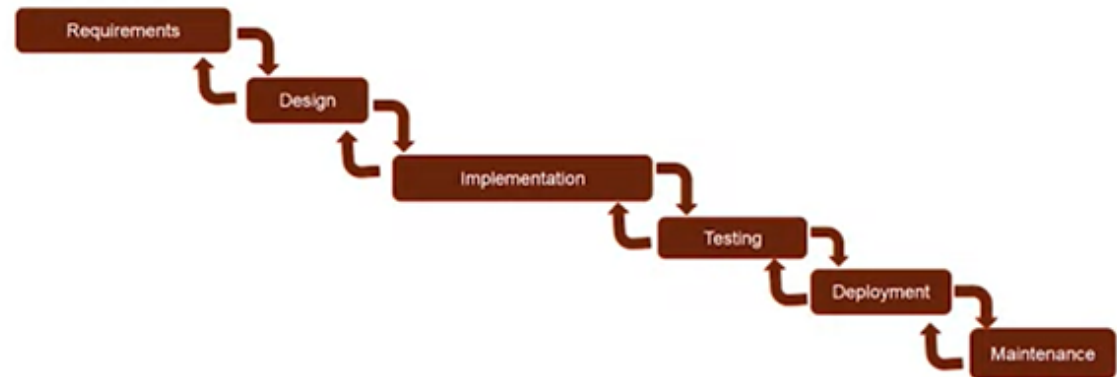
- One Phase finishes before another phase can begin

Waterfall Model



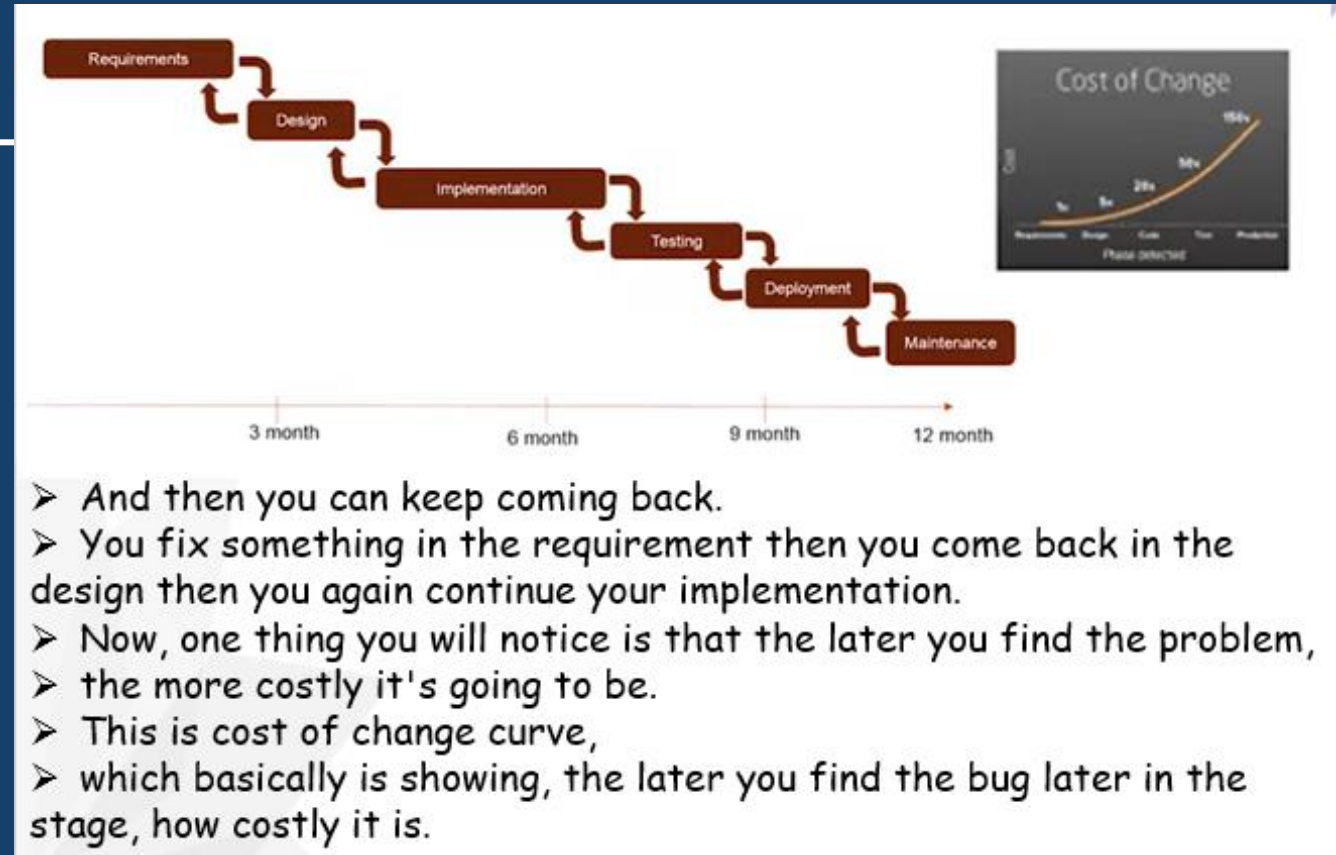
- In the waterfall model, these software engineering processes are happened one after the other in a logical sequence.
- Now, if something goes wrong, or let's say you are doing the implementation and you found out that, this is not what we need?

Waterfall Model

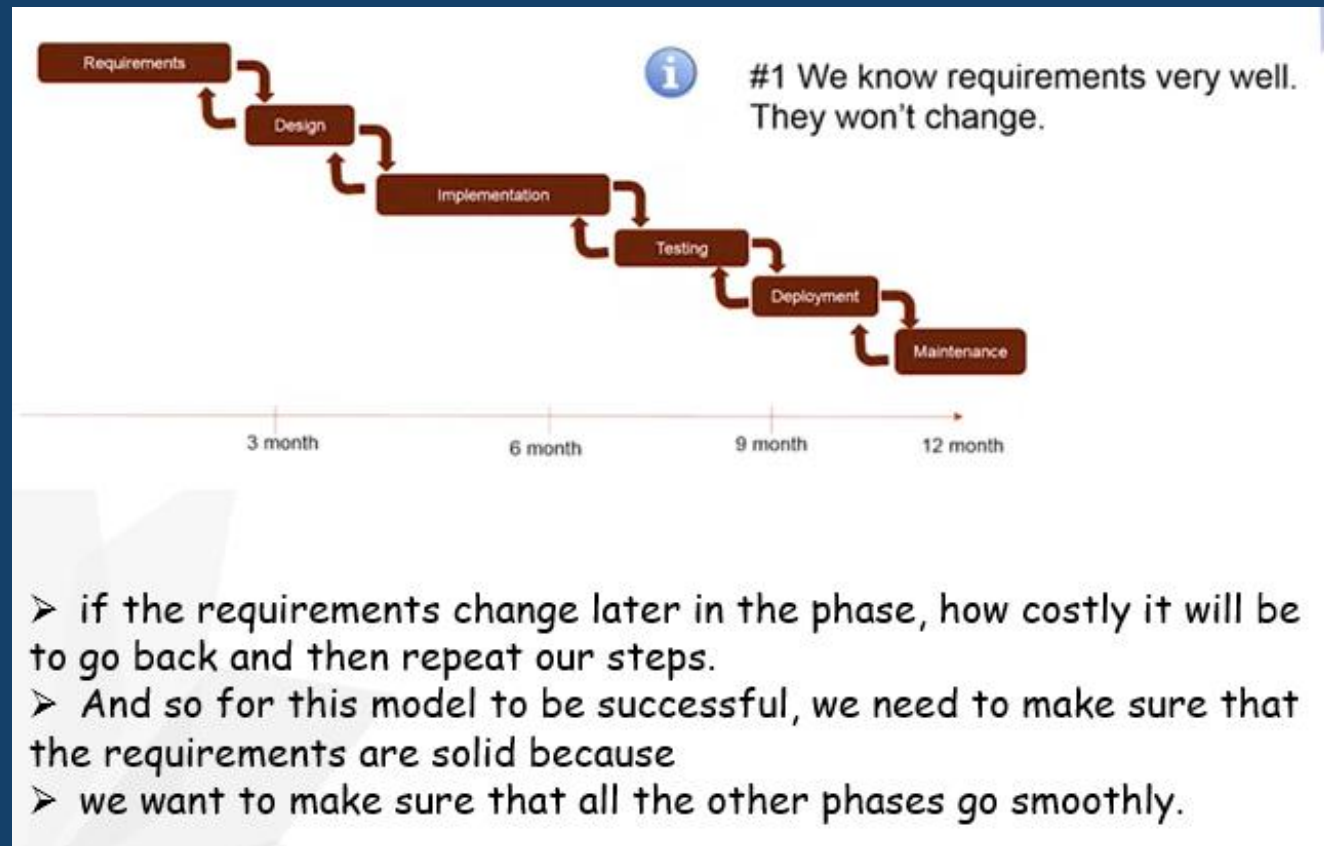


- we sometimes add this feedback loop from each of the phases to the previous phase,
- which means that if you find something wrong in the implementation phase,
- you will go back to the design phase if the problem was in the design phase,
- or you can go into the requirement phase if the problem was in the requirement phase.

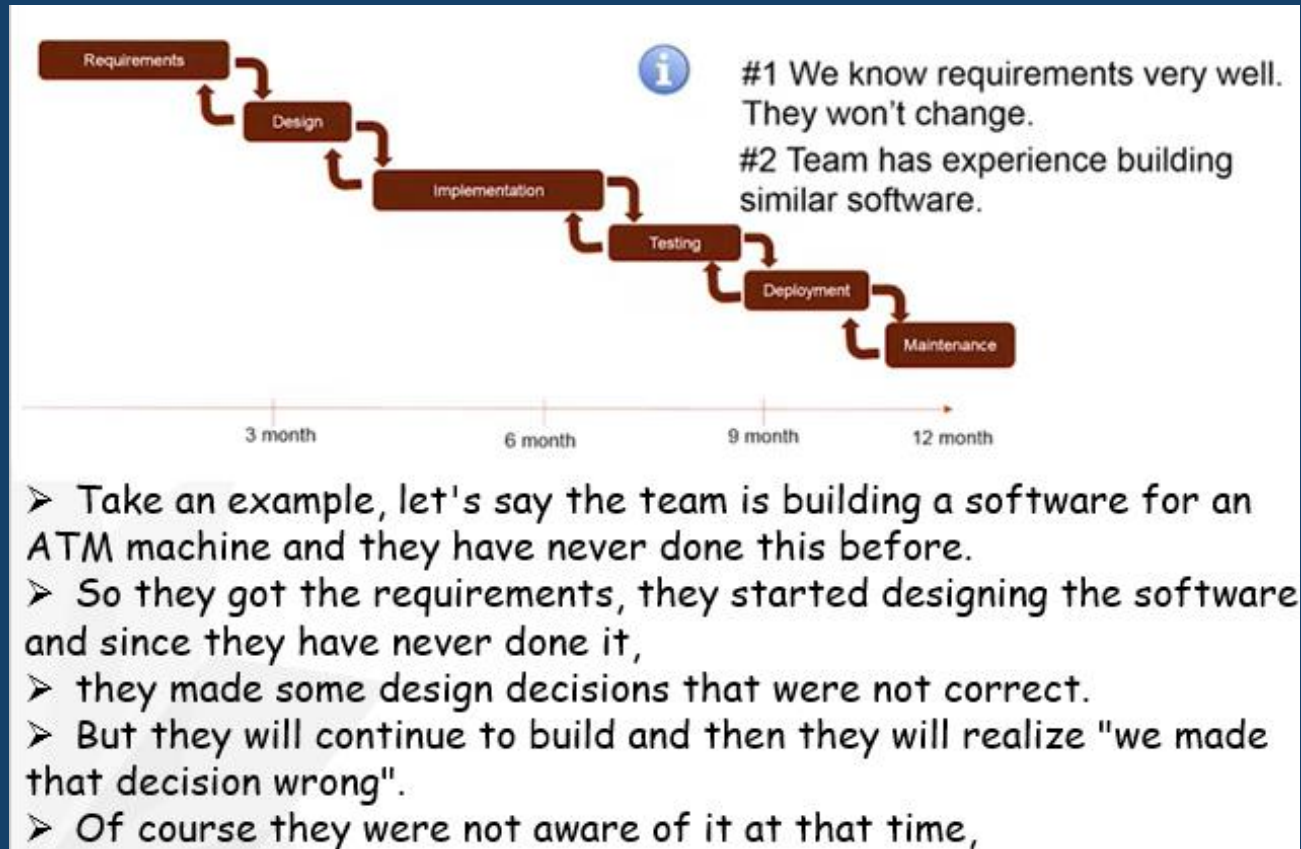
Waterfall Model



Waterfall Model



Waterfall Model

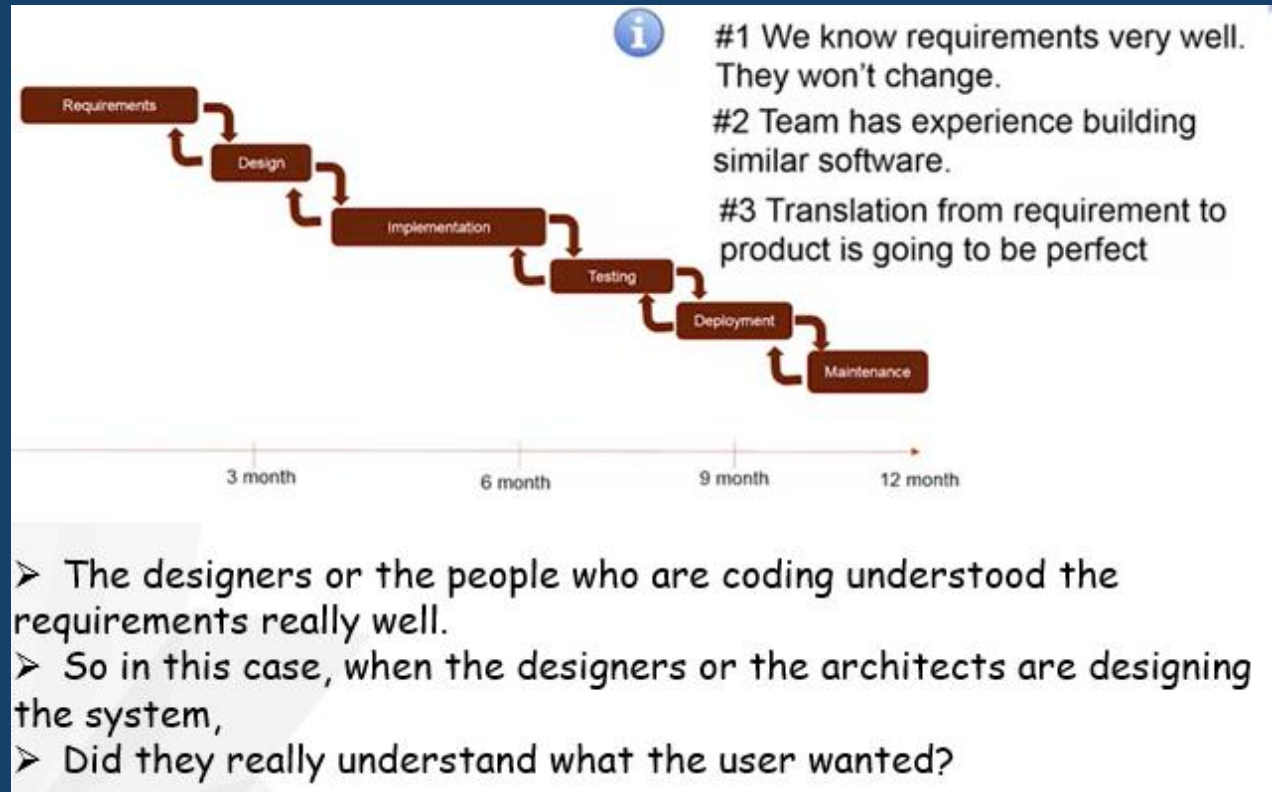


Waterfall Model

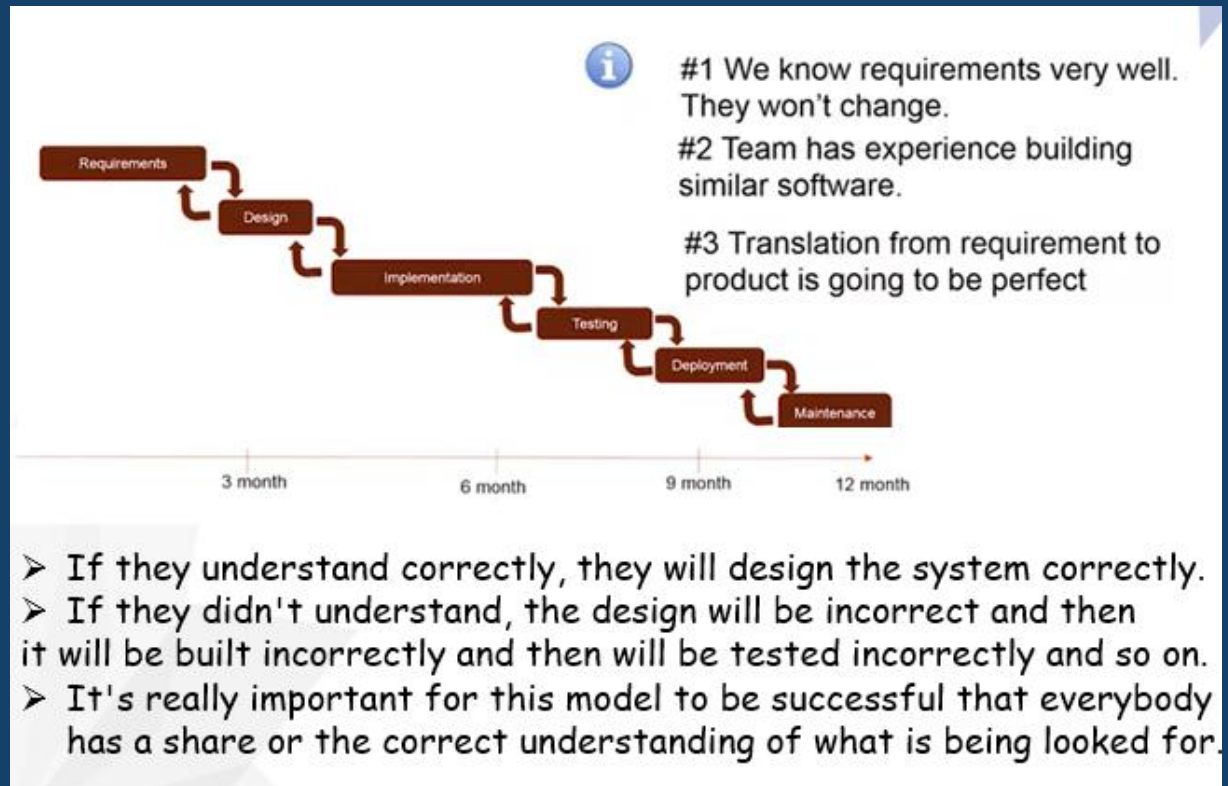


- they haven't built this kind of software before so that's why the mistake was made.
- But the mistake was made and now they have to do some rework.
- So for waterfall model to be successful,
- the team has to experience building similar software.

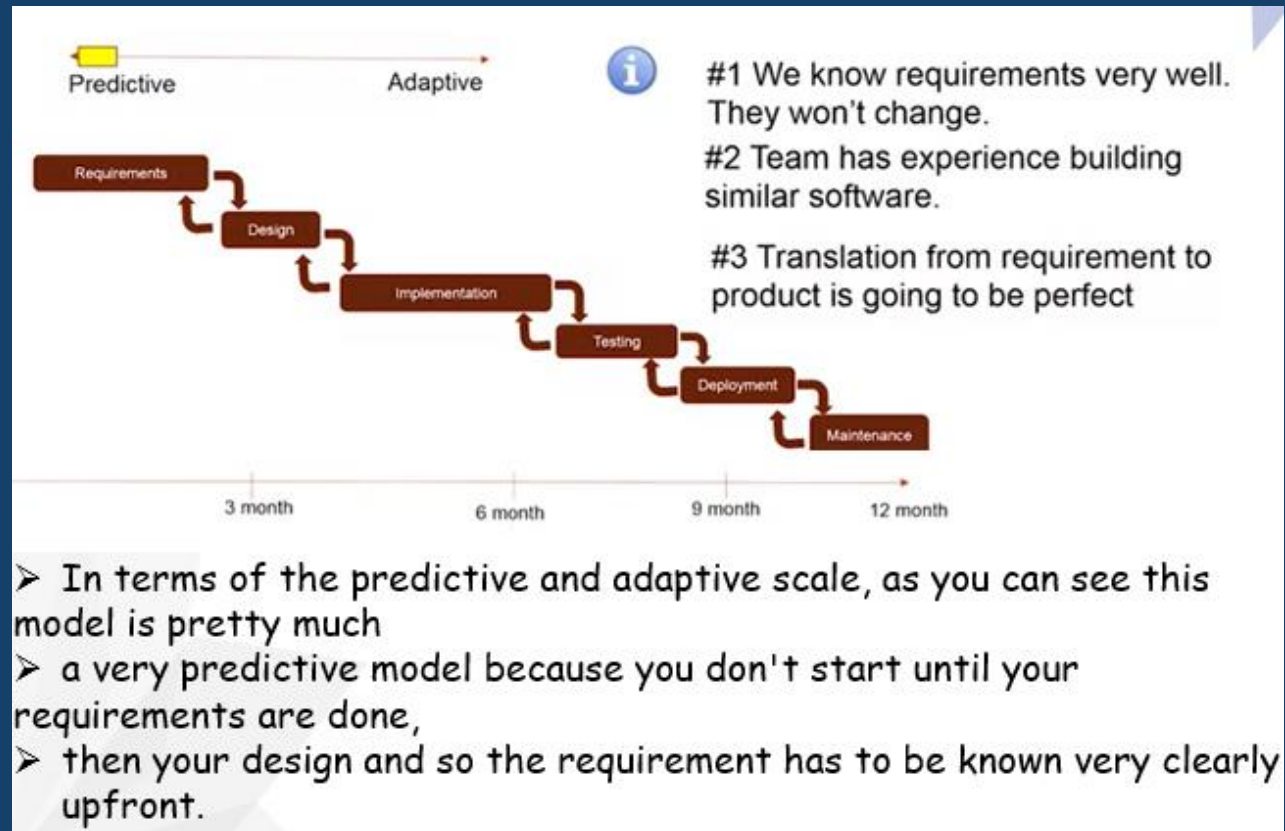
Waterfall Model



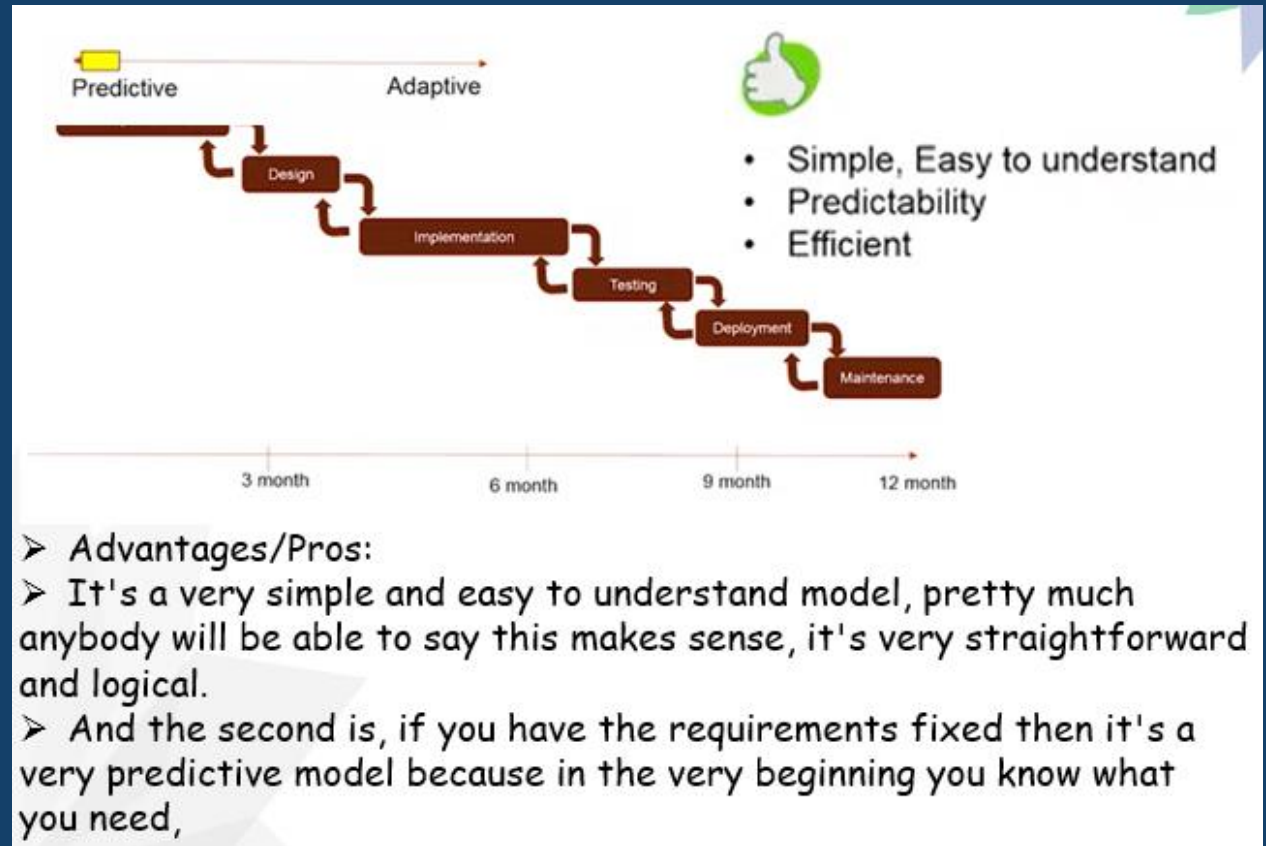
Waterfall Model



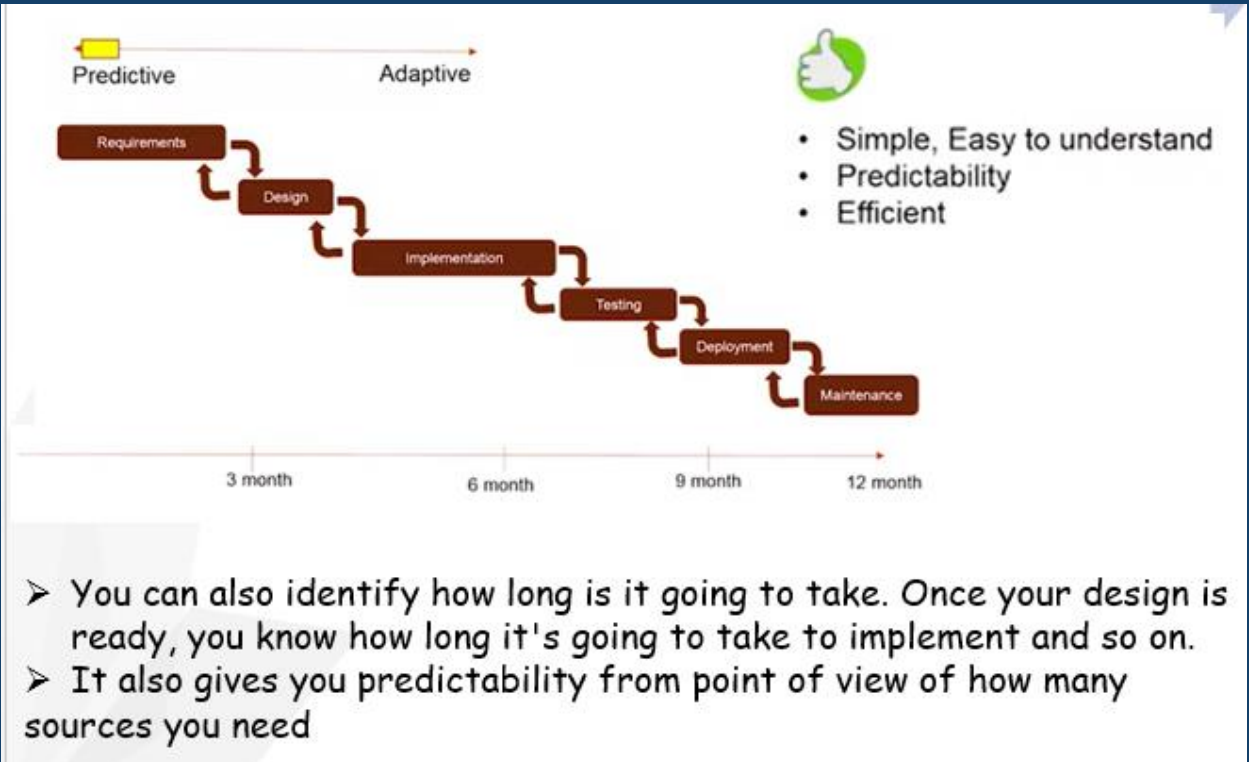
Waterfall Model



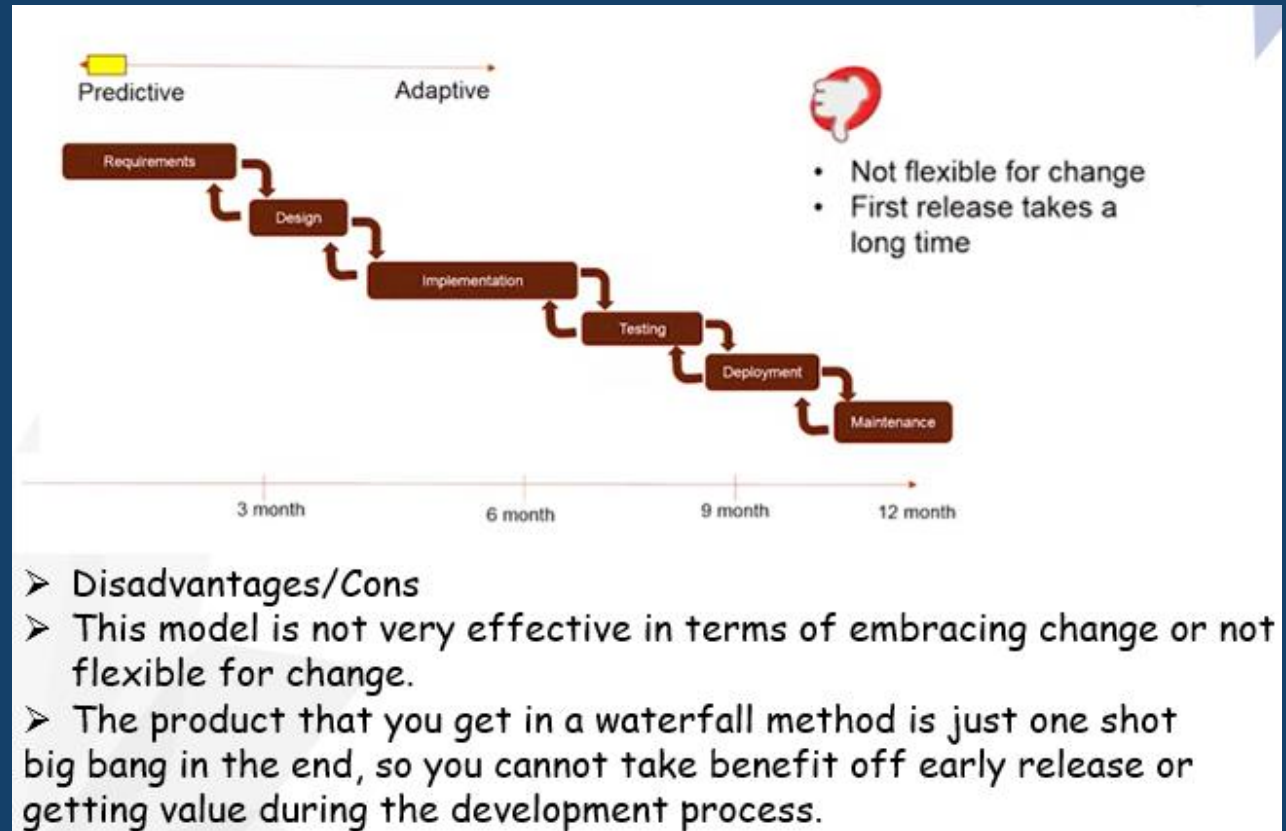
Waterfall Model



Waterfall Model

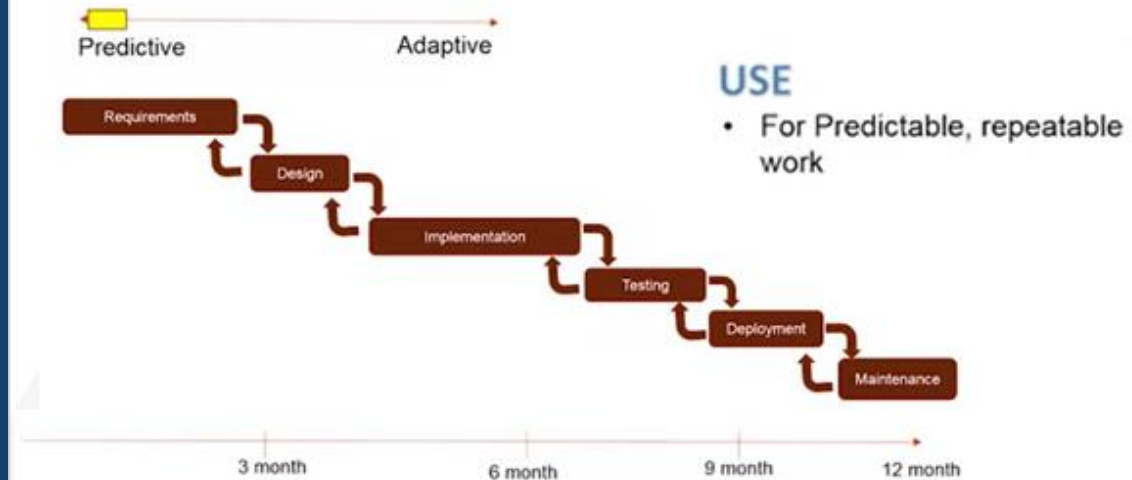


Waterfall Model



Waterfall Model

Waterfall Model



- When to use this software?
- If the project or the product that you're working on is very predictable and it's a repeatable work, then waterfall model will be a decent fit for this kind of work.

Waterfall Model

Q: Which of the following are limitations of the waterfall model? Select three.

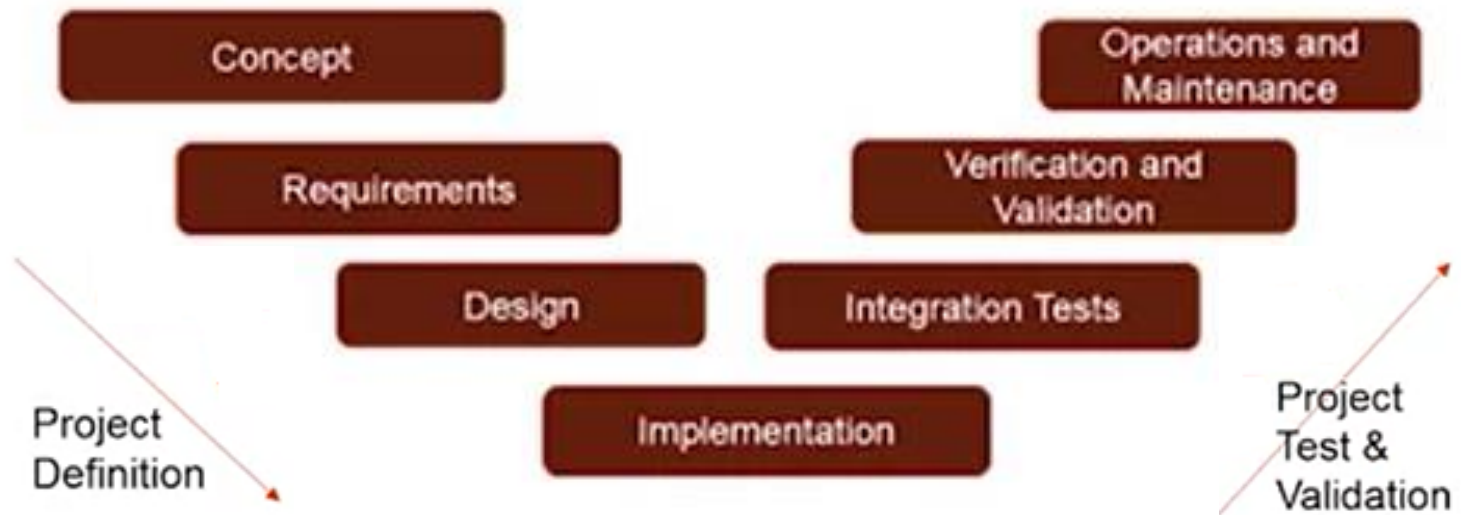
- a) It is not suitable for big projects.
- b) Misinterpretations of requirements or design can remain undetected until the later development phases.
- c) Integration issues may remain undetected until the last phase.
- d) It is difficult to respond to requirements changes.

Correct Answer is : b, c and d

Q: In waterfall method, you get your product in one big bang deployment

True
False

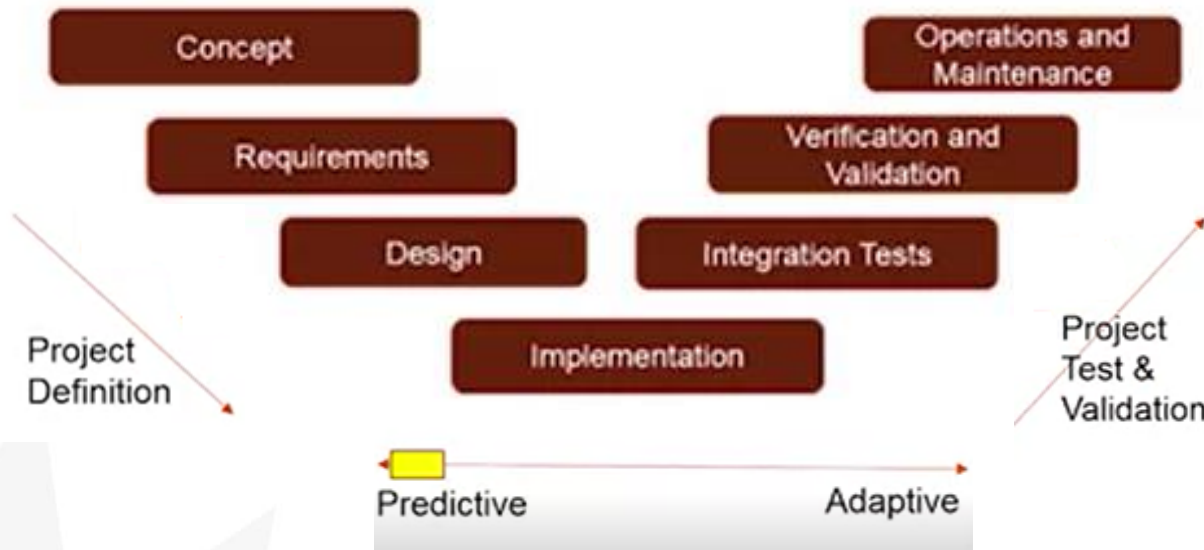
Answer: True



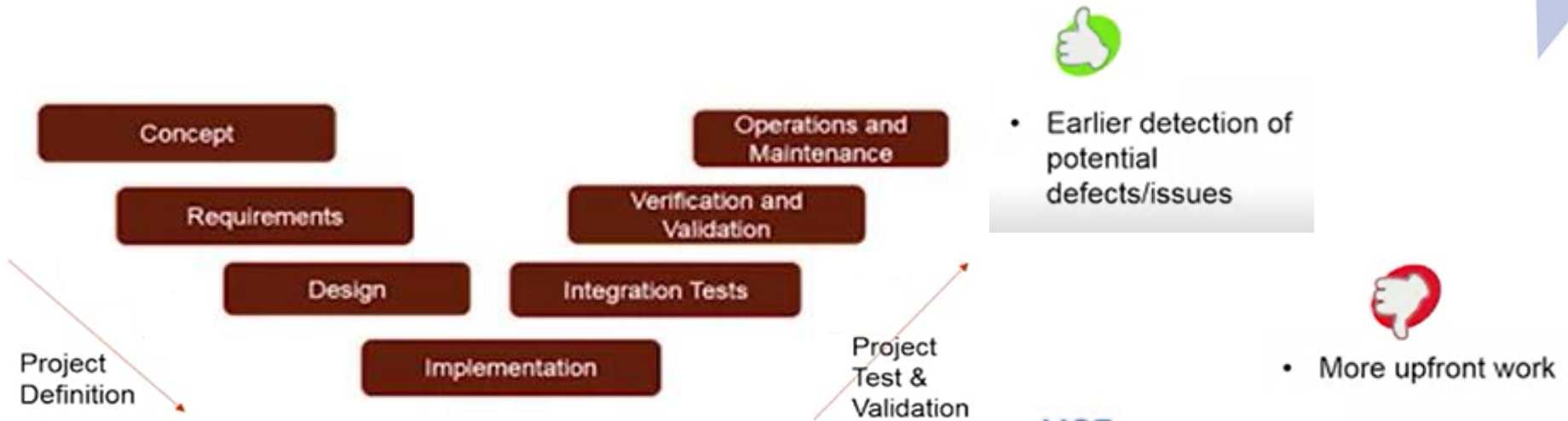
- This model looks very similar to the waterfall method, but it is just bent into a shape of a V
- The left side of the model is about project definition or the product definition,
- The right side is showing all the validation steps that are corresponding to the steps on the left.



- Emphasis on validation earlier in the process



- The testing activities related to the testing phase are introduced earlier in the model
- The basic idea behind V-model is that there is a lot of emphasis on the validation earlier in the process
- V-model is very much a predictive model. Because it doesn't allow any kind of feedback or doesn't allow any kind of changes during the phases.

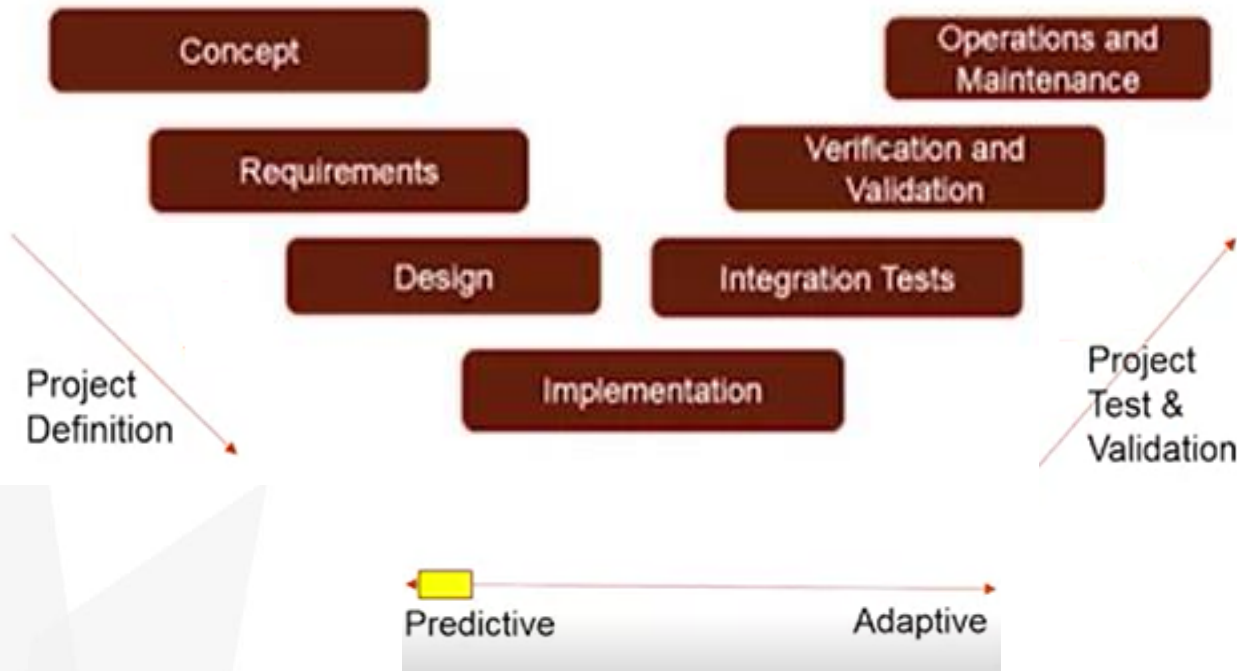


USE

- Ambiguity in requirements and early validation would be useful.

- **Advantages/Pros**
- it provides earlier detection of potential defects and the issues.
- **Disadvantages/Cons**
- It requires extra work upfront, because during the requirement phase, you are also going to talk about the validation of those requirements
- If there is ambiguity in the requirements, and then you want some kind of early validation, would help, then you can use V-model.

V Model



- Emphasis on validation earlier in the process

USE

- Ambiguity in requirements and early validation would be useful.



- Earlier detection of potential defects/issues



- More upfront work

Question?



Q. Which of the following are true for the V-model? Select two.

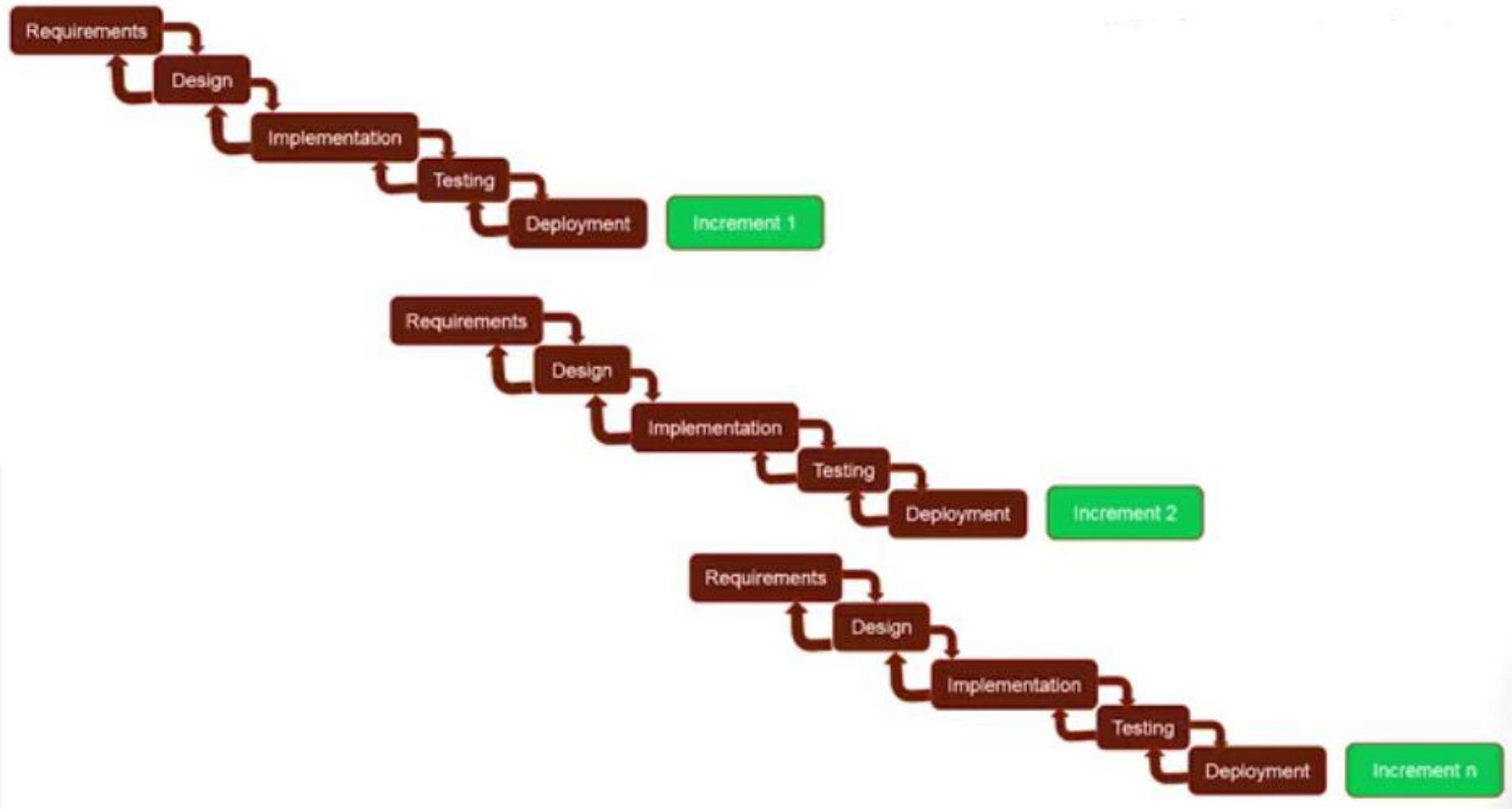
- a) Testing-related activities are started earlier in the process.
- b) It is a predictive model.
- c) Requirement changes are welcomed in all phases of this project.





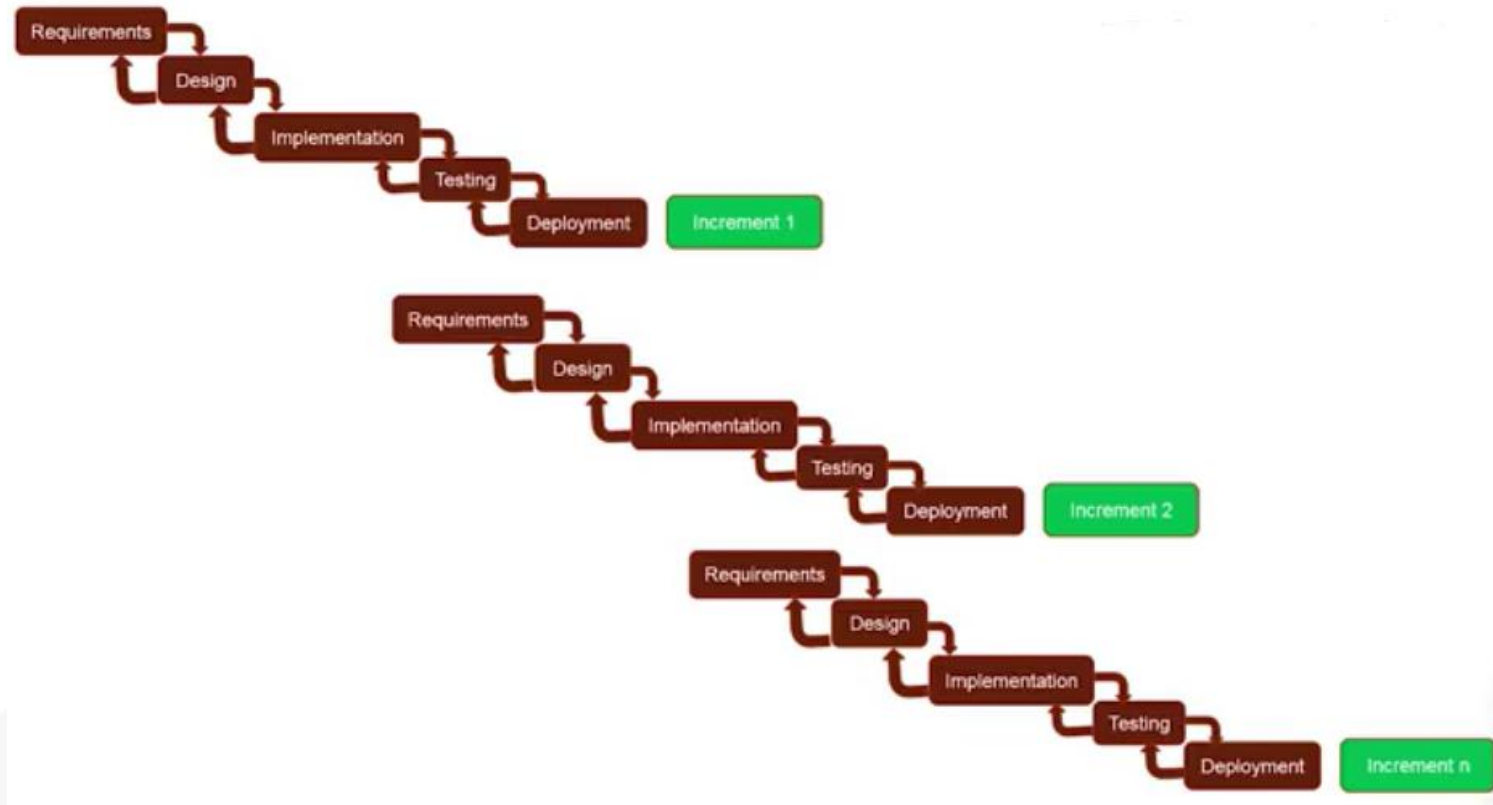
- In incremental model instead of building the whole application in one shot, you build in increments
- So you can still use your waterfall steps, but only for that particular increment.
- you will do your requirements, design, implementation, testing and deployment but only for part of the application.
- And if that increment is useful for users, you can potentially deploy this and have users use it.

Incremental Model



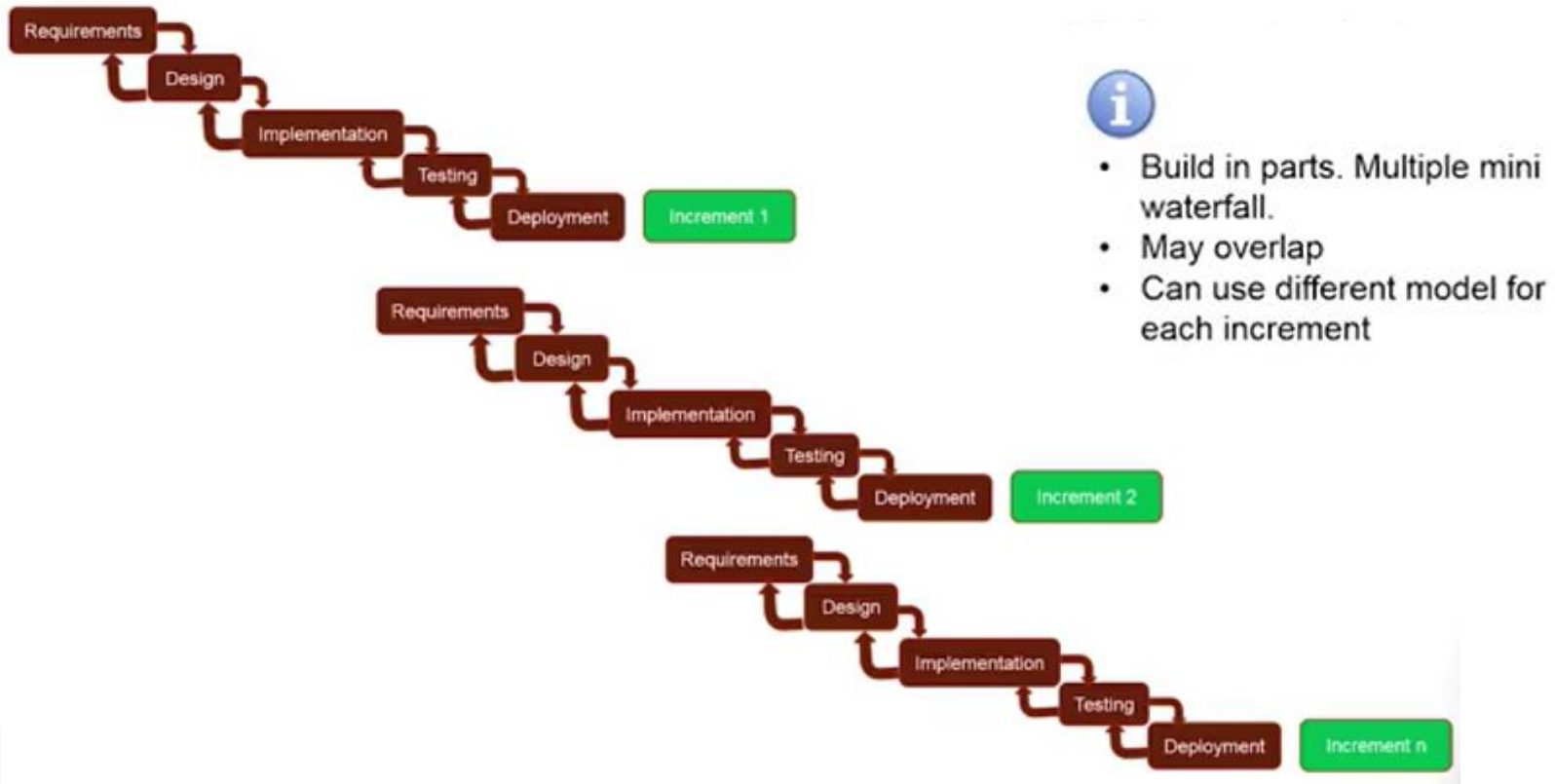
➤ So once the first increment is ready, you can start with the next increment and then once the second increment is done, then you can start with the third increment.

Incremental Model



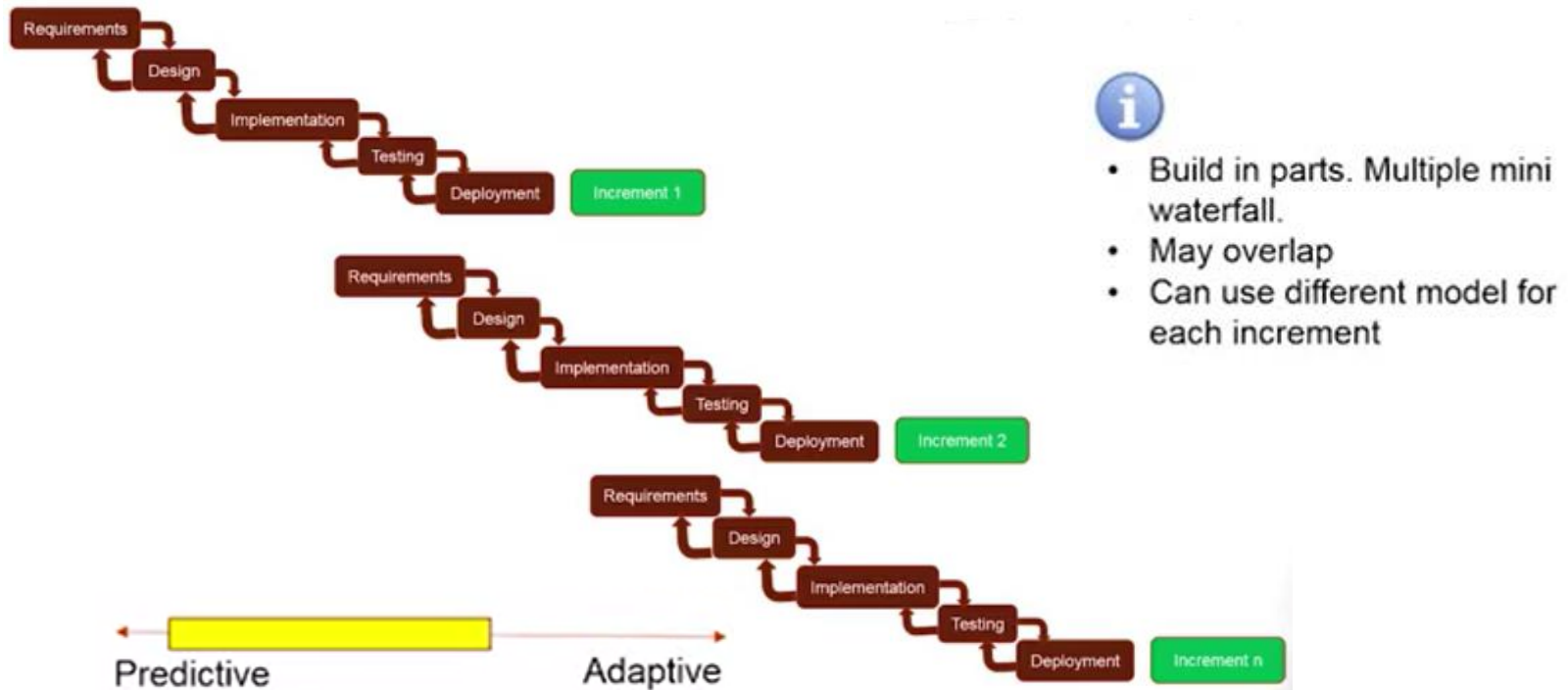
- You can also overlap these different increments as well.
- while the testing was going on for the first increment, we can start the requirements at the same time for the second
- when the implementation was being done or the testing phase was just about to begin we can start the requirement phase for the increment number N.

Incremental Model



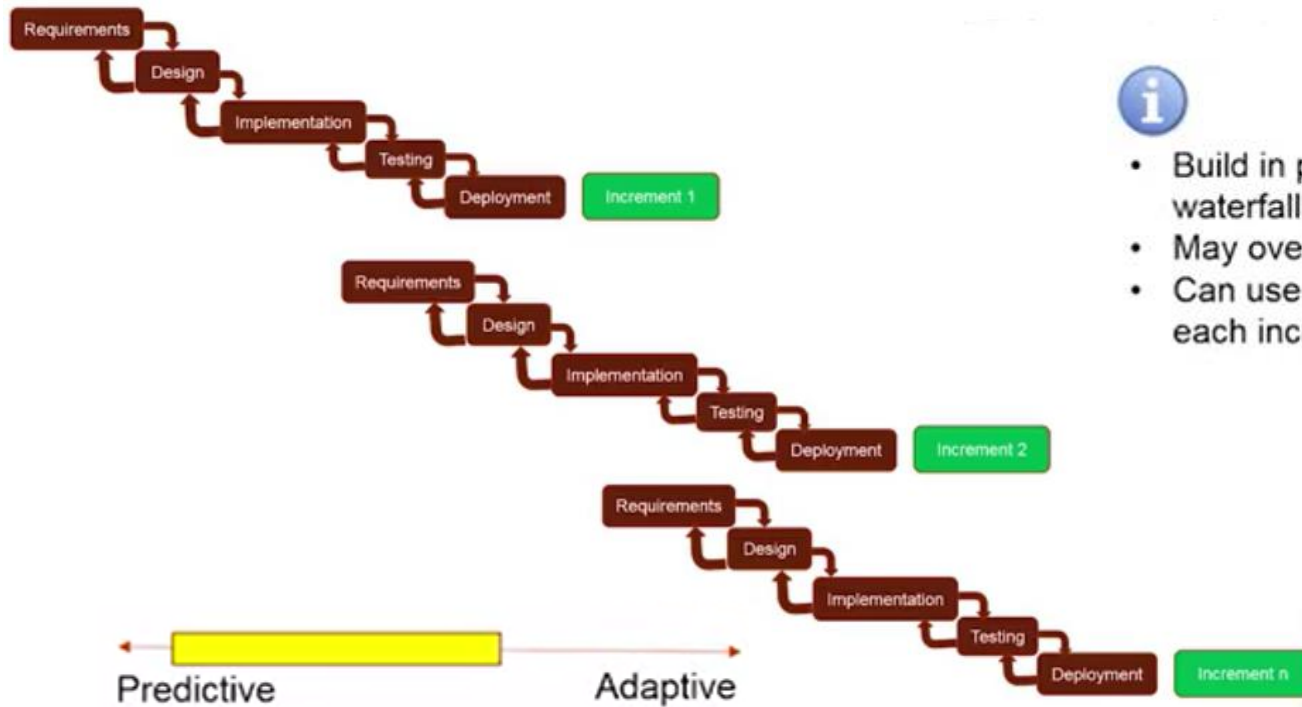
- You can actually use different models for each of the increments. So for example, for increment 1 you could have used V Model, for increment 2 you could have used the Waterfall Model and for increment N one you could have used some other model.

Incremental Model



- In terms of the predictive and adaptive scale, this depends quite a bit of how much overlap is there and how much feedback from one increment you apply to the next increment.
- So, if you don't apply any feedback from one increment to another increment and you just keep building, then it is more towards the predictive model.

Incremental Model

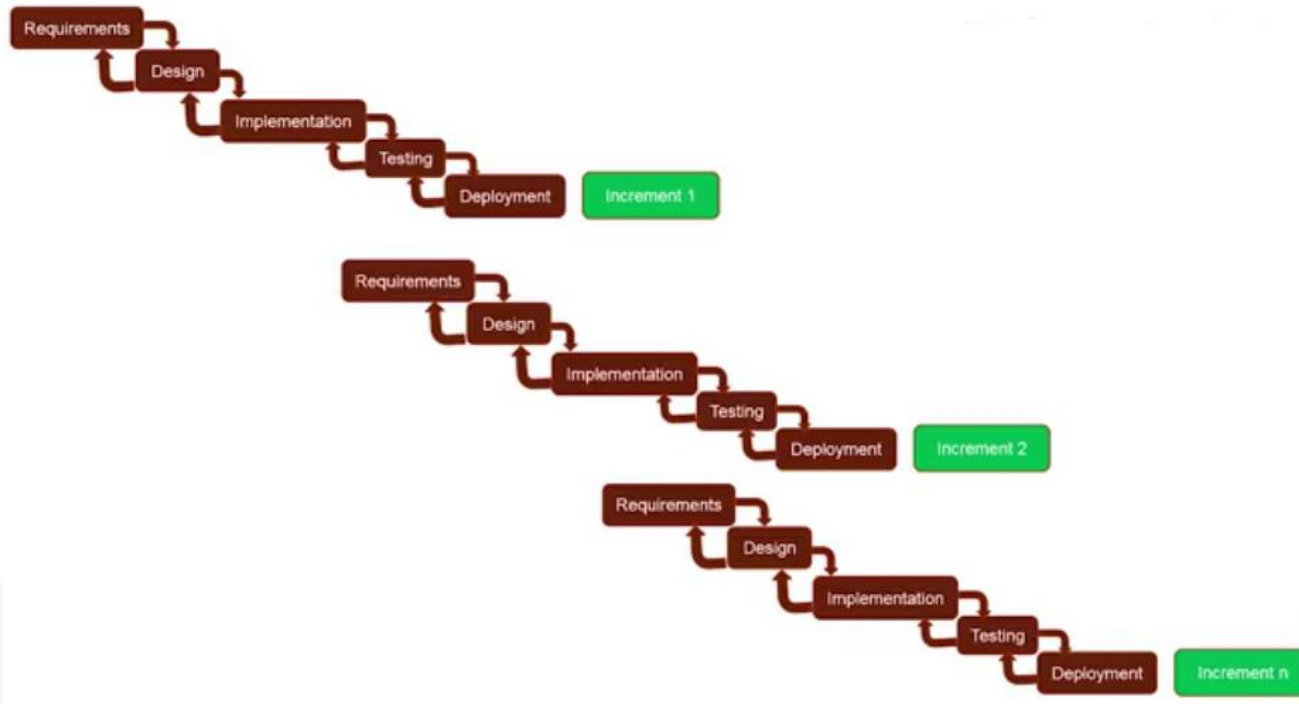


- Build in parts. Multiple mini waterfall.
- May overlap
- Can use different model for each increment

- But if you apply the feedback from one increment towards another one, it goes towards the adaptive model.
- So that's why there is a big range where this model fits between predictive and adaptive model.

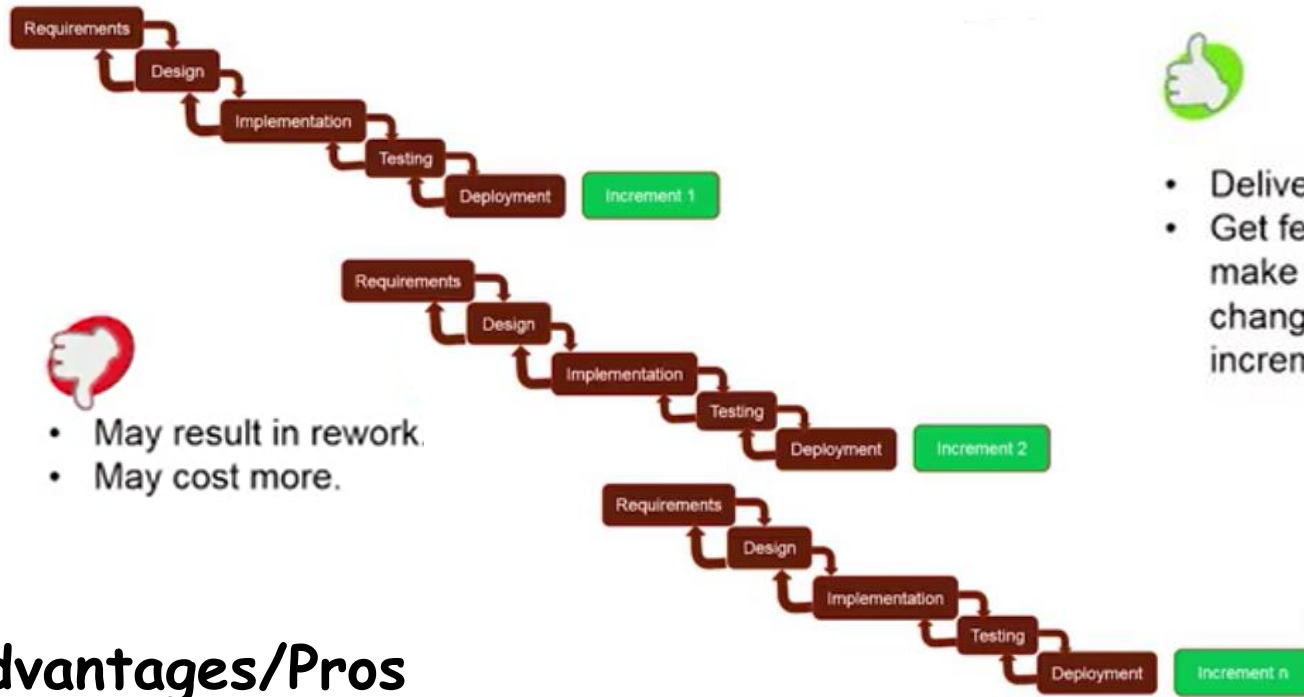


- Deliver value earlier
- Get feedback and make necessary changes between increments



- **Advantages/Pros**
- you deliver value early, in those cases where that increment delivers value to the customers.
- Sometimes if you are building a car and you just build the tires first then maybe it's not that useful but if you build it in such a way that the part of the application that you're building is actually useful for user, then it can deliver value earlier.

Incremental Model



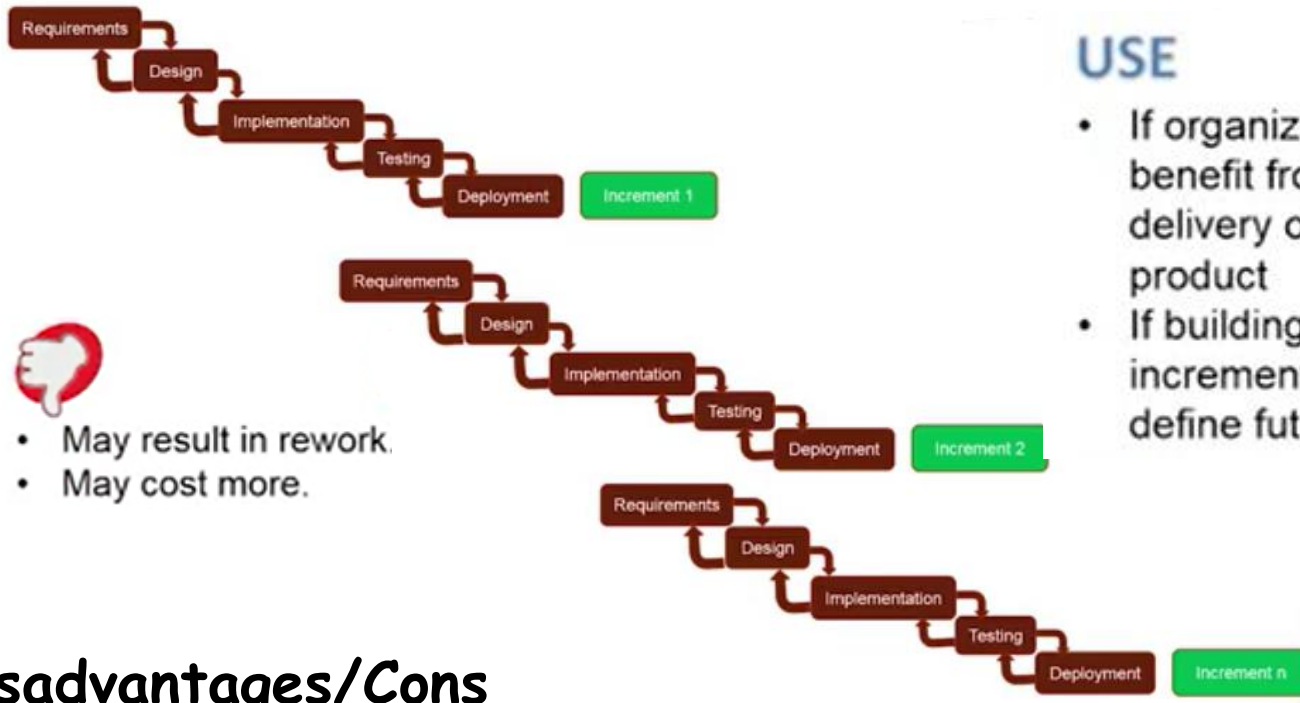
➤ Advantages/Pros

➤ The second thing is you can actually get feedback from one increment and apply that feedback to the future increments.

➤ Disadvantages/Cons

➤ it may result in a rework. Since, you didn't know all the requirements earlier in the phase and you started doing the requirements of the next increment later on, you may end up doing extra work in future increments

Incremental Model

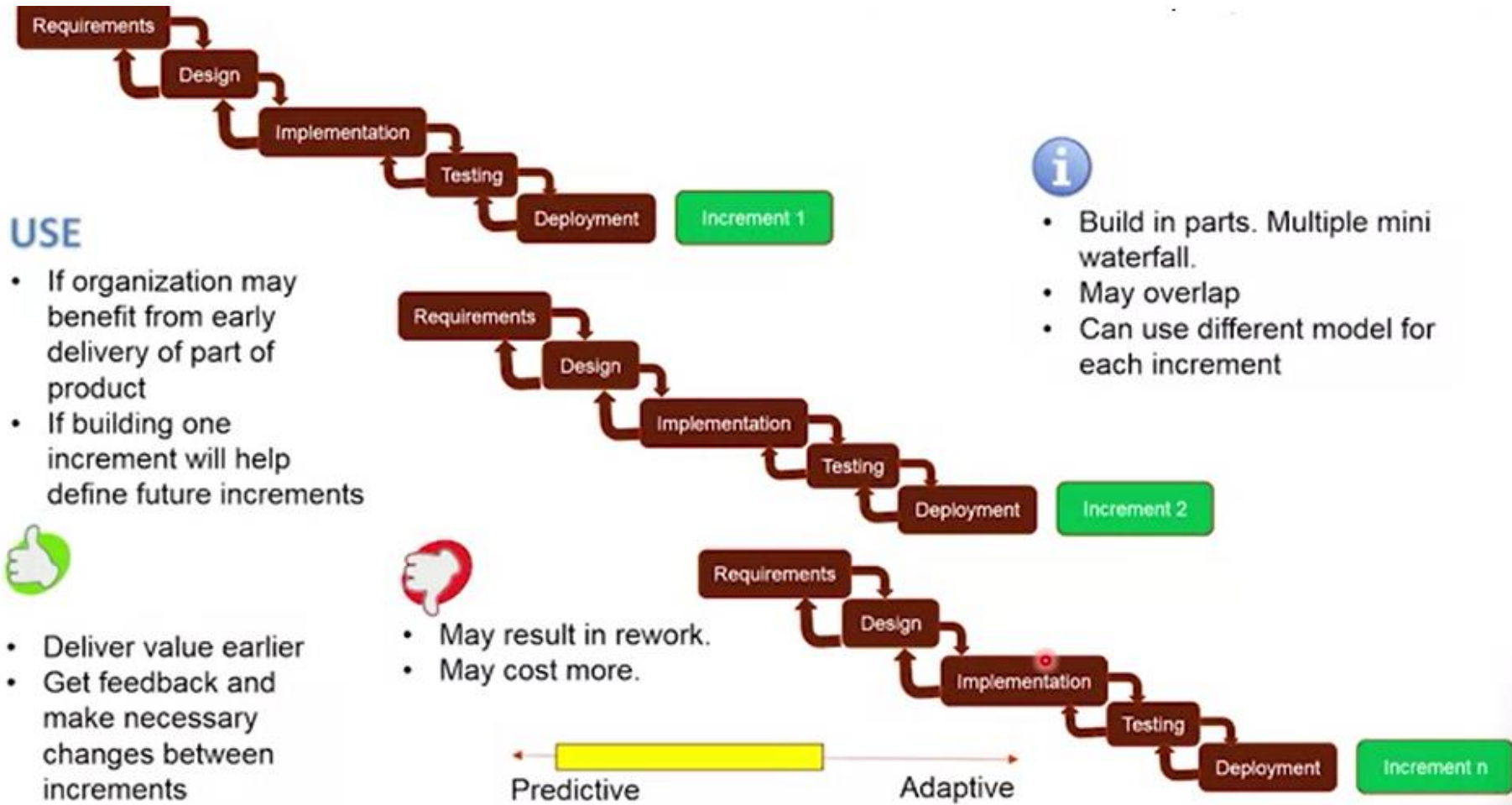


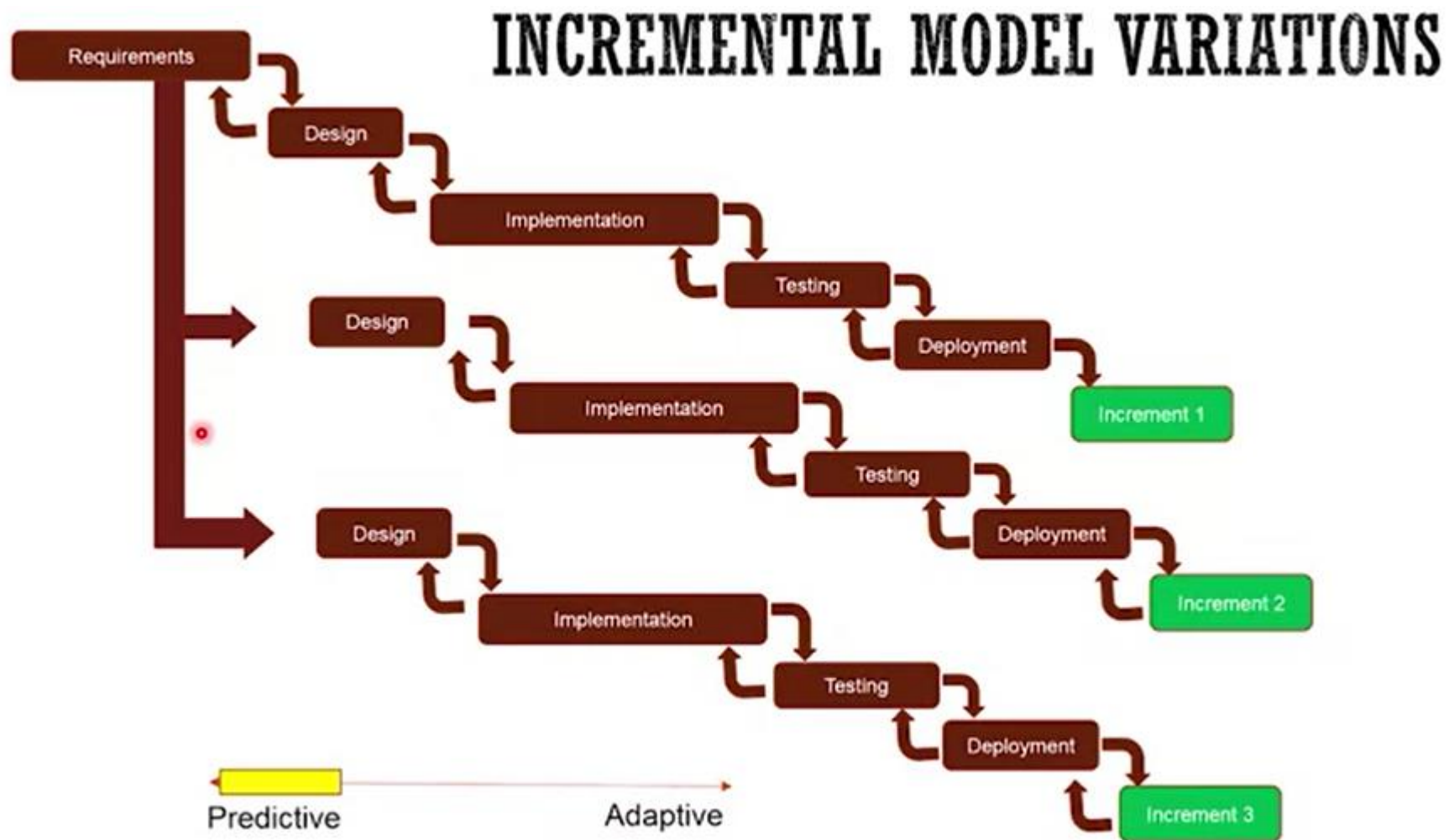
USE

- If organization may benefit from early delivery of part of product
- If building one increment will help define future increments

- **Disadvantages/Cons**
- if there is required changes, then it may cost more as well.
- **Use:**
- If the organization can benefit from early delivery of part of the application or part of the product then you can use this model and also if the feedback from one increment can be applied to the future increments, then again this model will be very useful.

Incremental Model





INCREMENTAL MODEL VARIATION

The diagram illustrates three variations of the incremental model, categorized by a spectrum from Predictive to Adaptive.

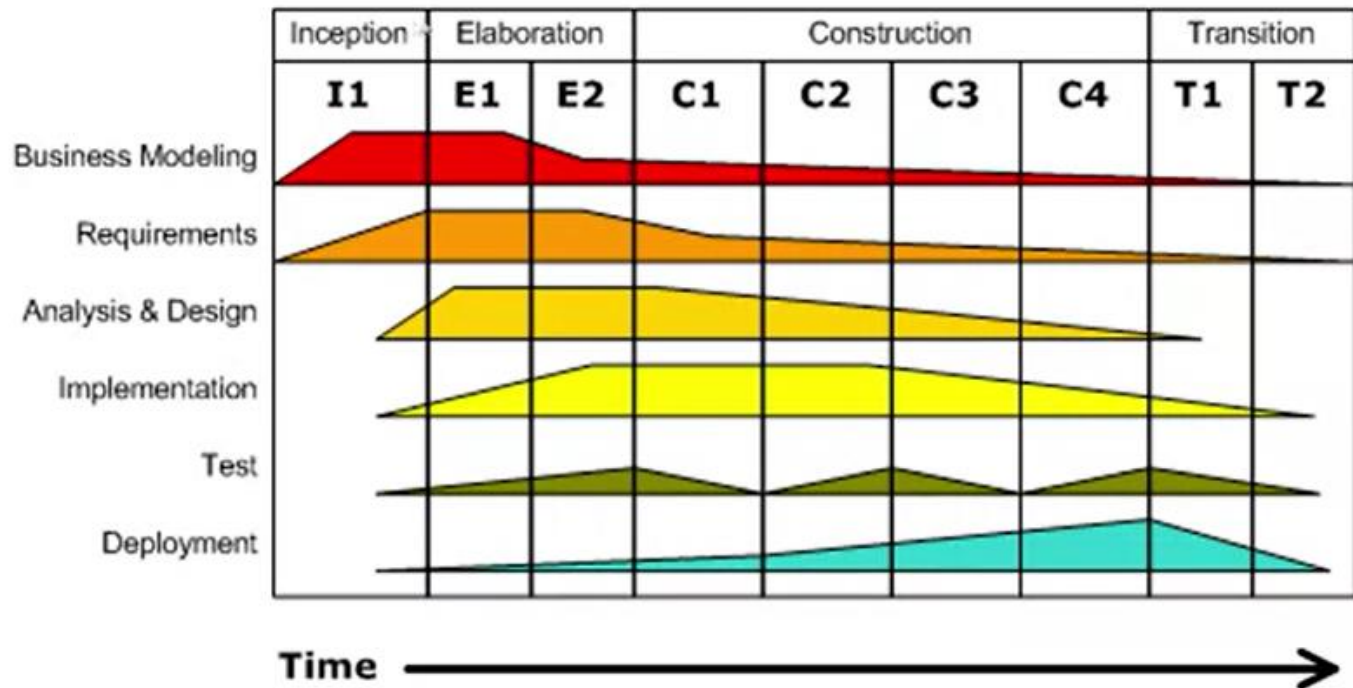
- Predictive:** This variation shows a linear flow from Requirements to Design, Implementation, Testing, and Deployment. A large vertical arrow on the left indicates a direct path from Requirements to Design, bypassing the iterative steps.
- Adaptive:** This variation shows a feedback loop from Testing back to Design, indicating that the design can be updated based on testing results.
- Iterative:** This variation shows multiple cycles of Implementation, Testing, and Deployment, with each cycle leading to a new increment (Increment 1, Increment 2, Increment 3). The flow is: Requirements → Design → Implementation → Testing → Deployment → Increment 1 → Implementation → Testing → Deployment → Increment 2 → Implementation → Testing → Deployment → Increment 3.

Feature	Waterfall model	V-model	Incremental model
Focus	Development	Testing	Development and testing
Structure	Sequential	V-shaped	Iterative
Requirements	Gathered and documented at the beginning of the project.	Gathered and documented at the beginning of the project.	Gathered and documented in iterations , with each iteration focusing on a specific set of requirements.
Design	The complete software system is designed before development begins.	The complete software system is designed before development begins , but the design is updated and refined throughout the project.	The software system is designed in increments , with each increment designed in detail before development begins.
Development	The complete software system is developed before testing begins.	Development and testing are performed concurrently , with each test phase corresponding to a development phase.	Development is performed in increments , with each increment developed before testing begins.
Testing	performed after development is complete.	performed concurrently with development, with each test phase corresponding to a development phase.	performed after each increment is developed.
Deployment	deployed to users after testing is complete.	deployed to users in increments , with each increment deployed after testing is complete.	deployed to users in increments , with each increment deployed after testing is complete.
Strengths	Simpler and easier to understand, well-defined requirements	Focuses on testing early and often, which can help to improve the quality of the software, flexible and adaptable to change, delivers value to users early and often	Flexible and adaptable to change, delivers value to users early and often
Weaknesses	Can be inflexible and difficult to change requirements once development has begun, can be risky as development progresses	Can be complex and difficult to manage, can lead to scope creep	Can be time-consuming and expensive, requires careful planning and coordination

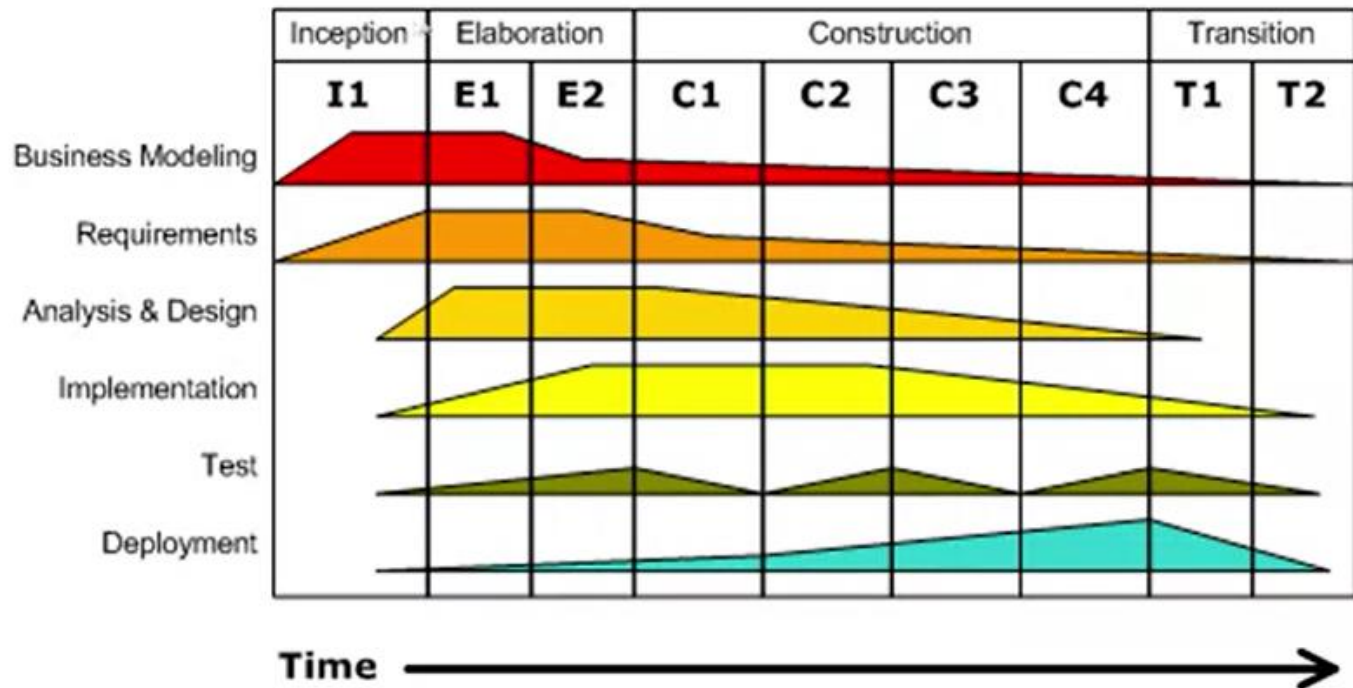


Q. Which of the following is true for Incremental Models? Select three.

- a) If deploying an increment to actual users can benefit the organization, using an incremental model is a potential candidate to consider.
- b) Incremental models are always predictive models
- c) You always have to use the same model for each of the increments
- d) Incremental models may result in rework
- e) You can overlap building of one increment with another



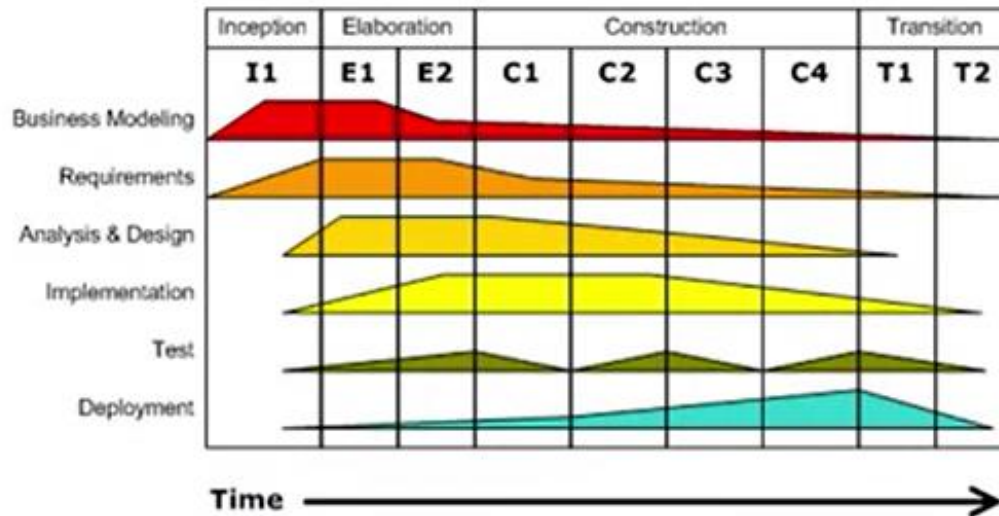
- The unified process has two dimensions
- the x axis, it defines certain phases or the steps like Inception, Elaboration, Construction and Transition
- The y axis displayed software development activities - Requirements, Analysis, Design, Implementation and Test



- Highlighted parts basically defines how much effort do you put for that software development activity in this step
- Each of these steps are also divided into multiple iterations.
- Number of columns in each step represents number of iterations

Iteration Model: The Unified Process

UNIFIED PROCESS



Predictive  Adaptive

USE

- Bigger and riskier projects
- All requirements not known early in the project
- Desire to deliver value earlier



- Adaptive
- Quality and Reuse
- Focus on risk mitigation increases chances of success
- Flexible to incorporate other s/w dev models

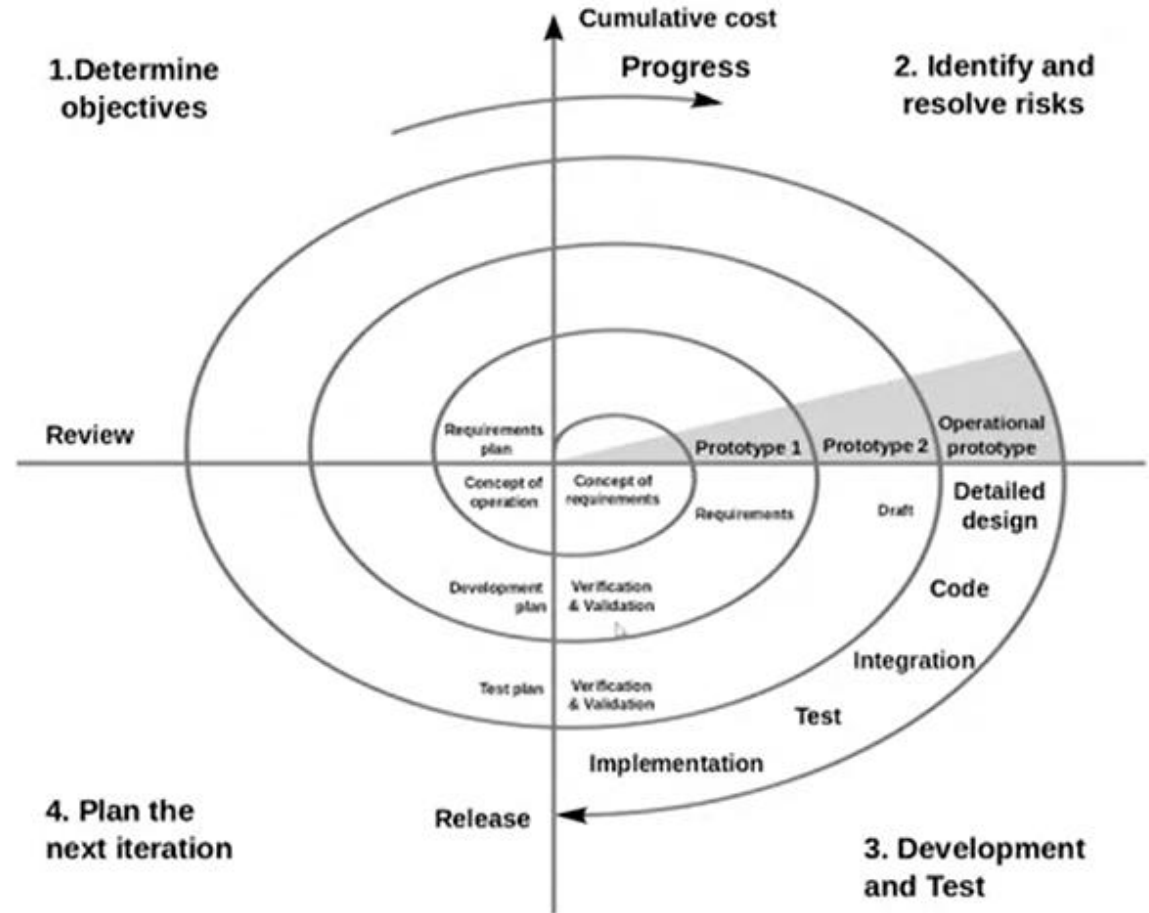


- Complicated
- Need more resources
- Too much overhead for smaller project

EIU Iteration Model: Spiral Model

WHAT DOES IT LOOK LIKE?

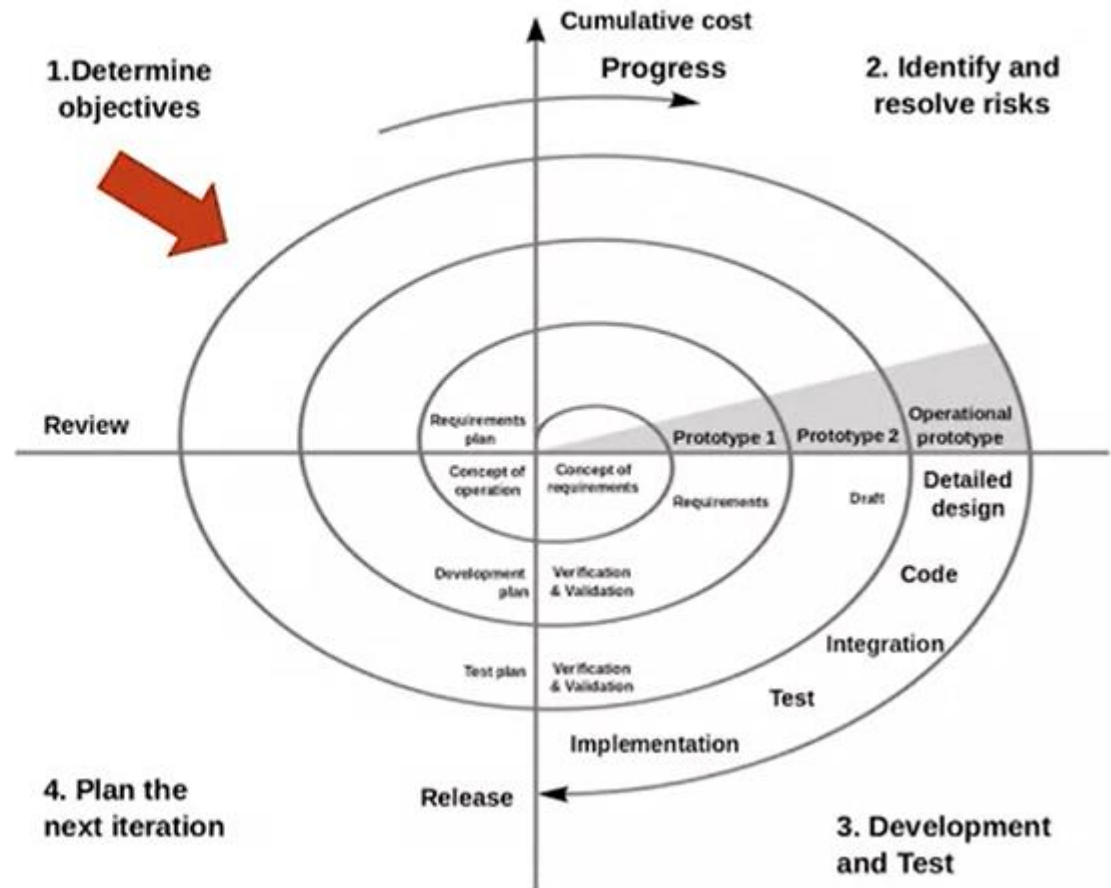
1. Cyclic
2. Four basic steps
3. Not every activity need to be performed



EIU Iteration Model: Spiral Model

DETERMINE OBJECTIVES

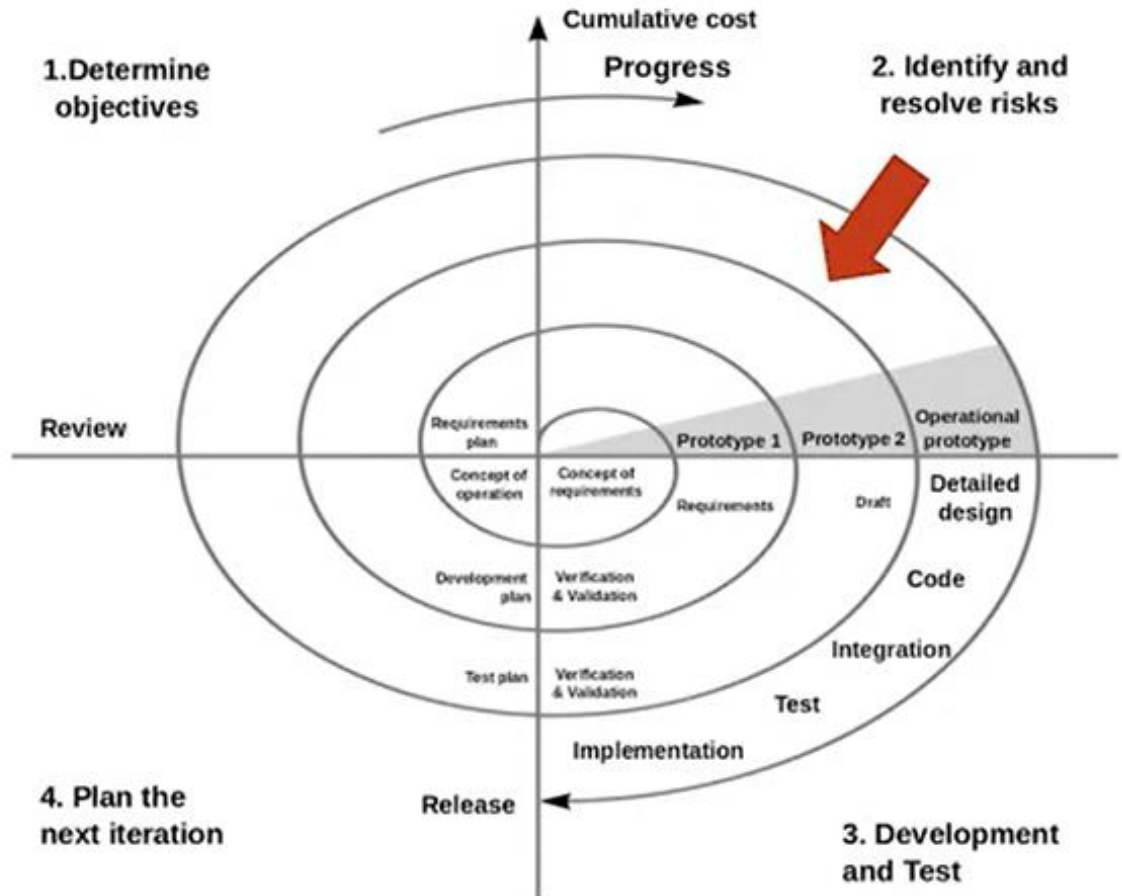
- 1.Objectives
- 2.Constraints
- 3.Alternatives



EIU Iteration Model: Spiral Model

IDENTIFY AND RESOLVE RISKS

1. Identify Risks
2. Resolve what you can

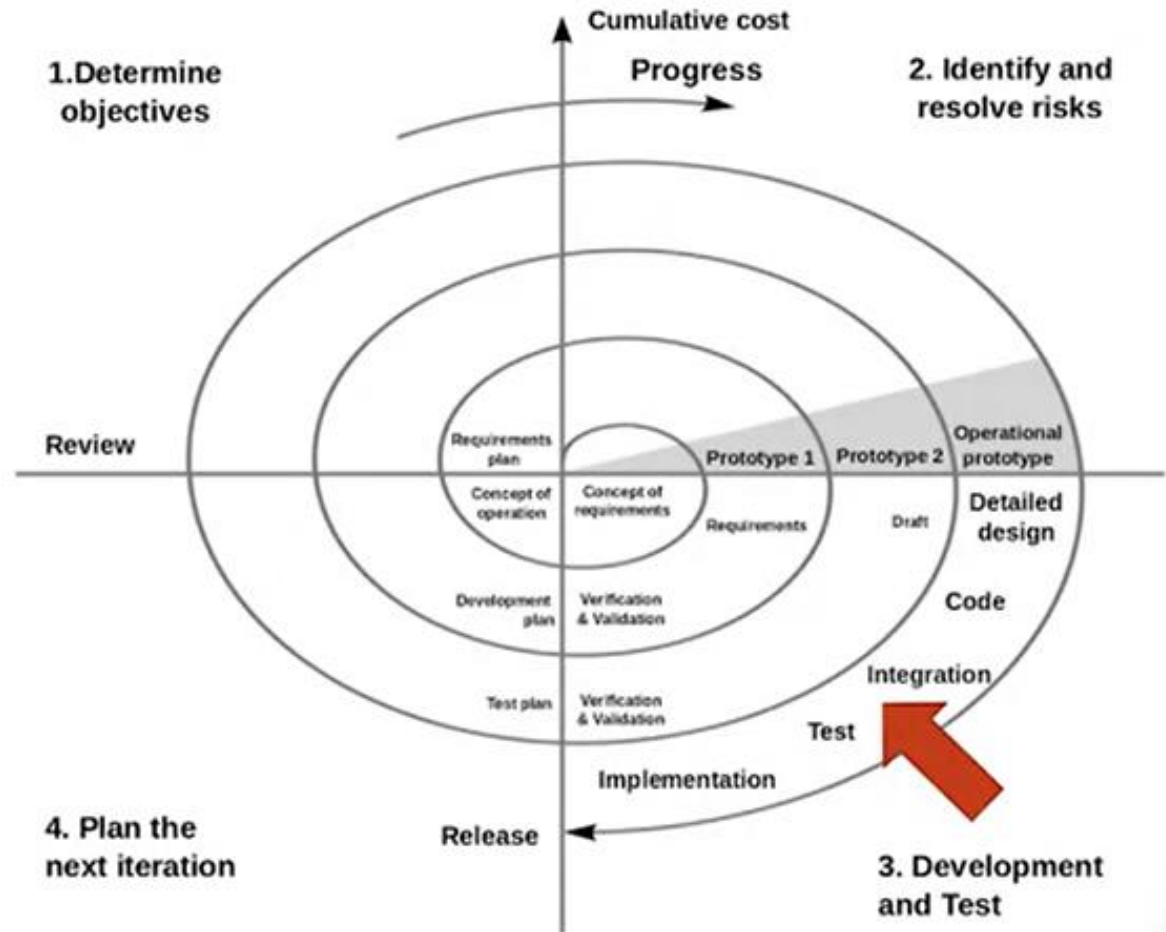


EIU Iteration Model: Spiral Model



**DEV AND
TESTS**

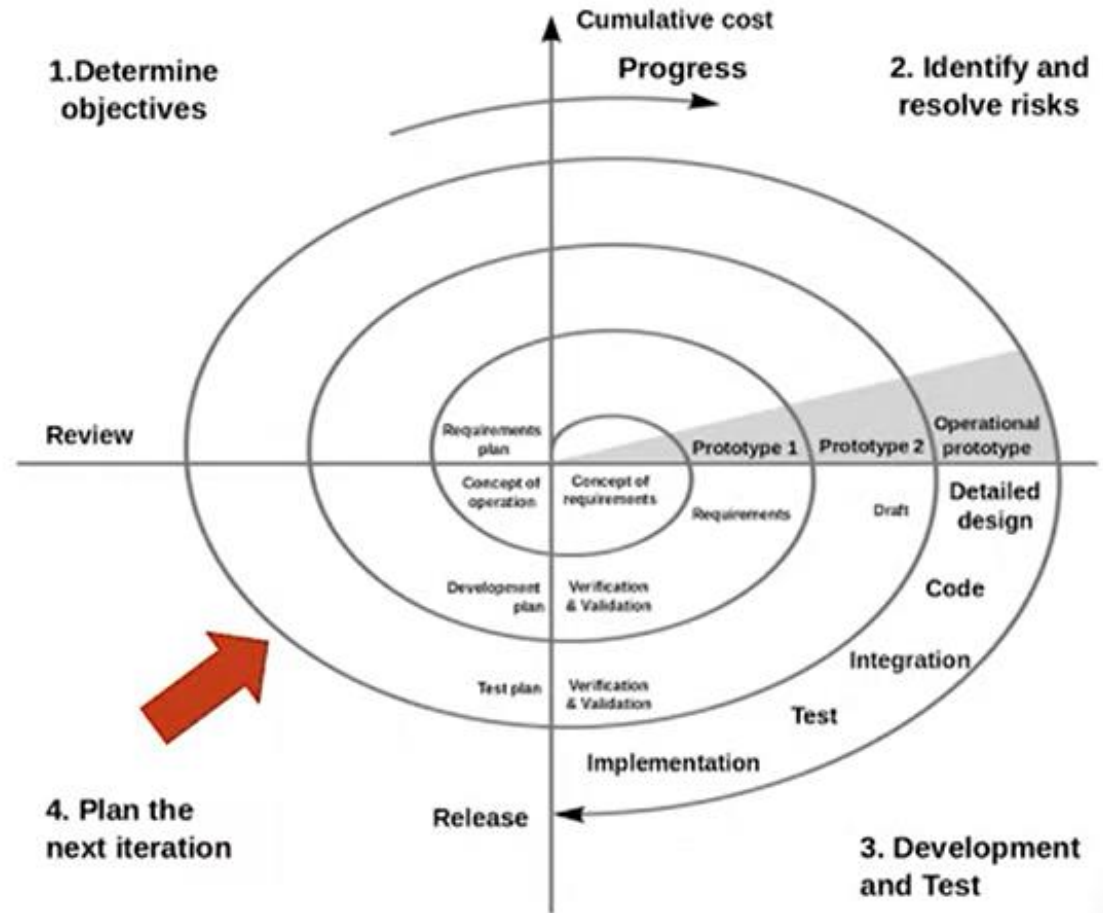
Work done to
meet
objectives



EIU Iteration Model: Spiral Model

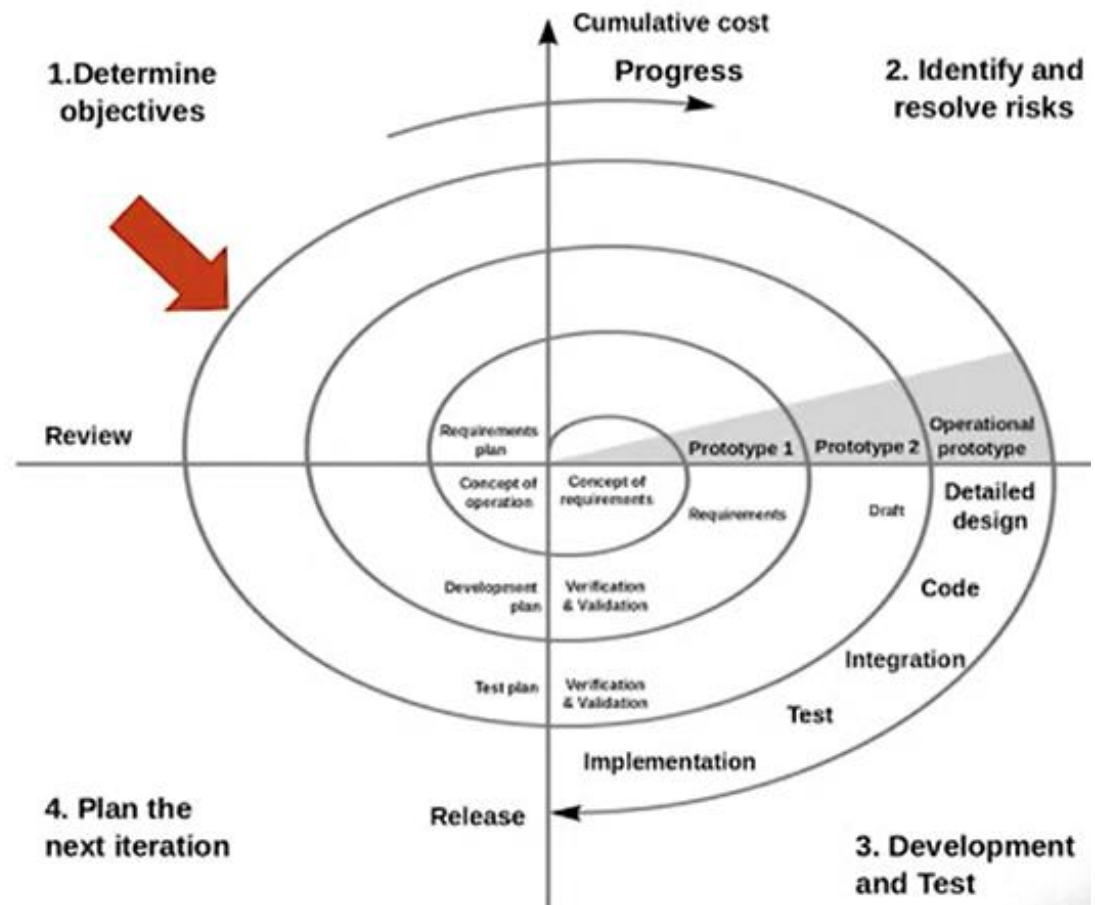
PLAN FOR NEXT ITERATION

Review work done and commitment for next iteration



EIU Iteration Model: Spiral Model

**START THE
NEXT CYCLE**



EIU Iteration Model: Spiral Model

USE

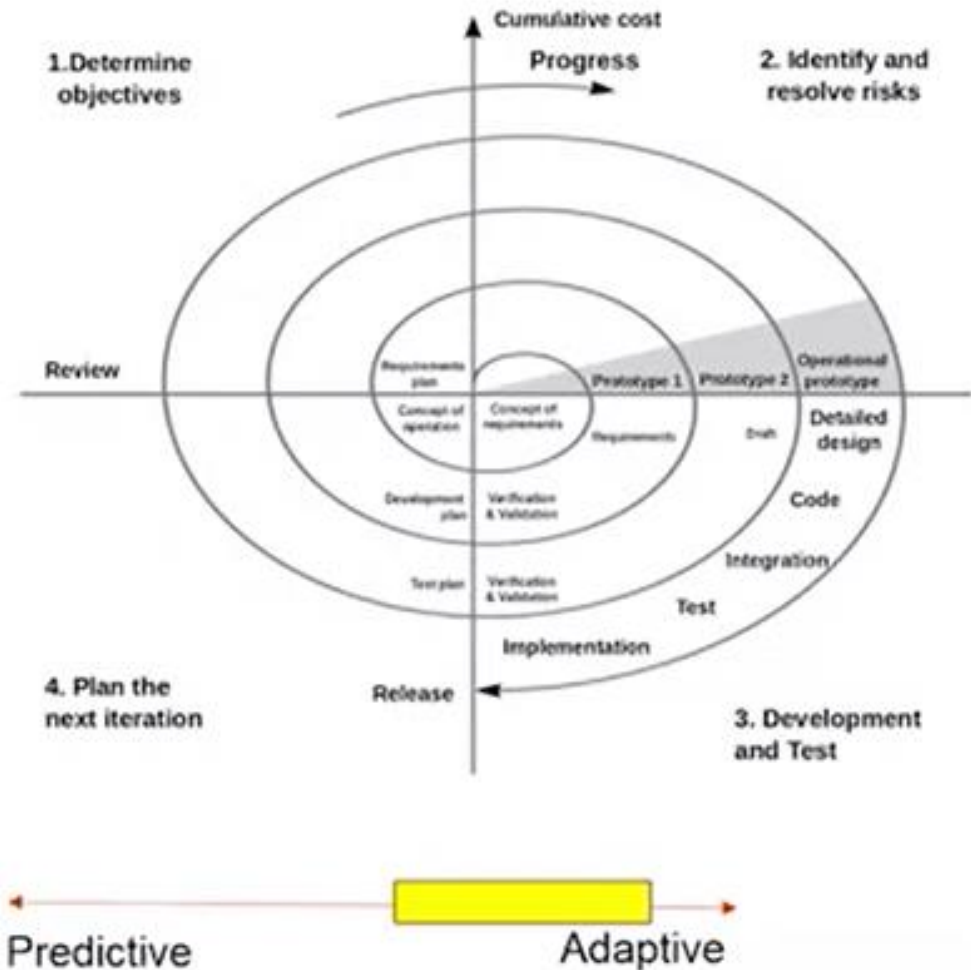
- Very large High risk projects



- Adaptive
- Risk focus increases chances of success
- Flexible for using any model
- Minimizes waste.
- Options for go/no-go



- Complicated
- Cost more to manage
- Need stakeholder engagement





Feature	Unified Process	Spiral model
Focus	Risk-driven, delivering value to users early and often	Risk-driven, iterative and incremental
Structure	Four phases: inception, elaboration, construction, and transition	Four phases per cycle: planning, risk analysis, engineering, and construction and release
Strengths	Comprehensive framework, well-defined phases	Flexible and adaptable to change
Weaknesses	Can be complex and difficult to implement, requires experienced team	Can be time-consuming and expensive



Q. In the Unified Process, all requirements work is done upfront and no requirements work is done in the construction phase.

- a) True
- b) False

Q. Which of the following activities happen in Step 2 of the Spiral Model?
Select two.

- a) Make a decision whether to continue with the next cycle
- b) Resolve risks
- c) Identify risks
- d) Decide objectives and constraints

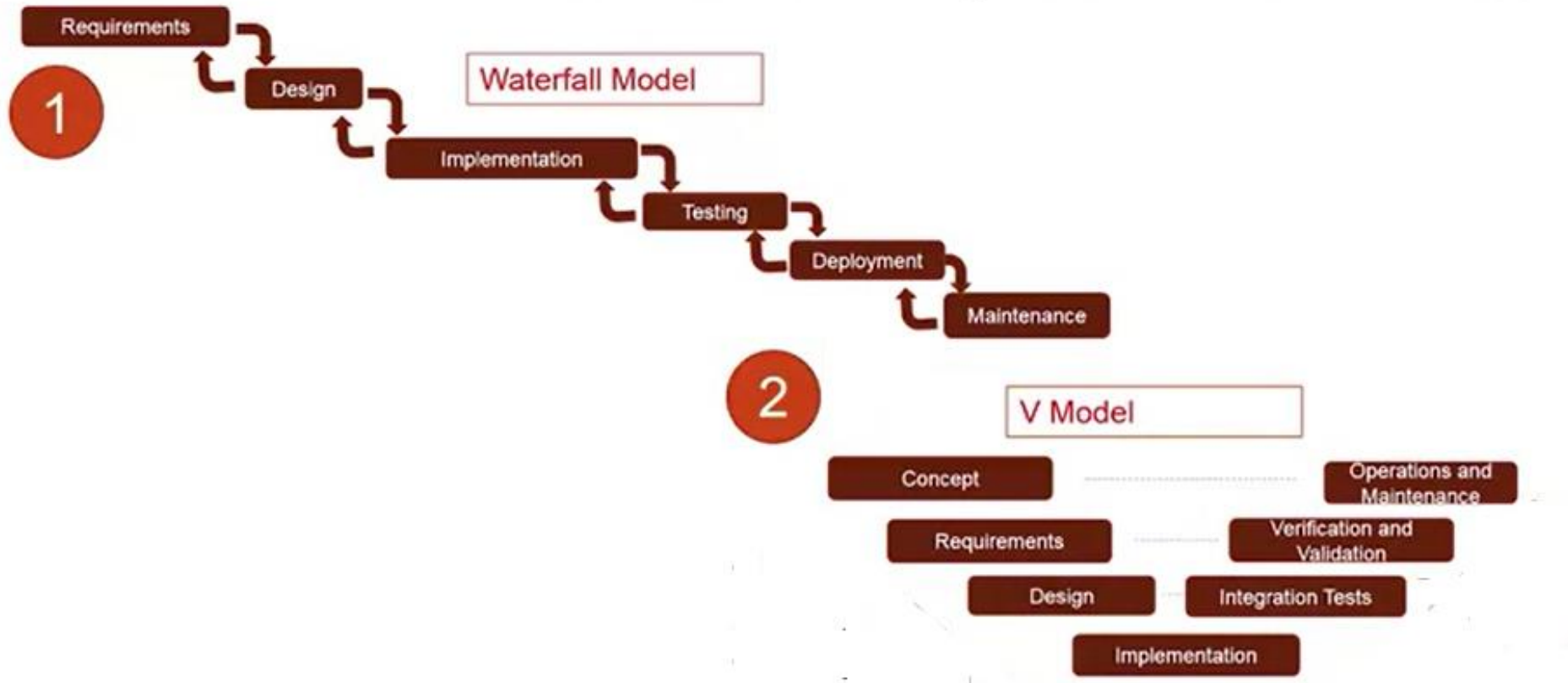


EXAMPLE 1

- Company X need to install a well known HR Management System at a big retailer's HQ.
- Company X has done many such installs on other big retailers before.



KNOWN PROBLEM, KNOWN SOLUTION

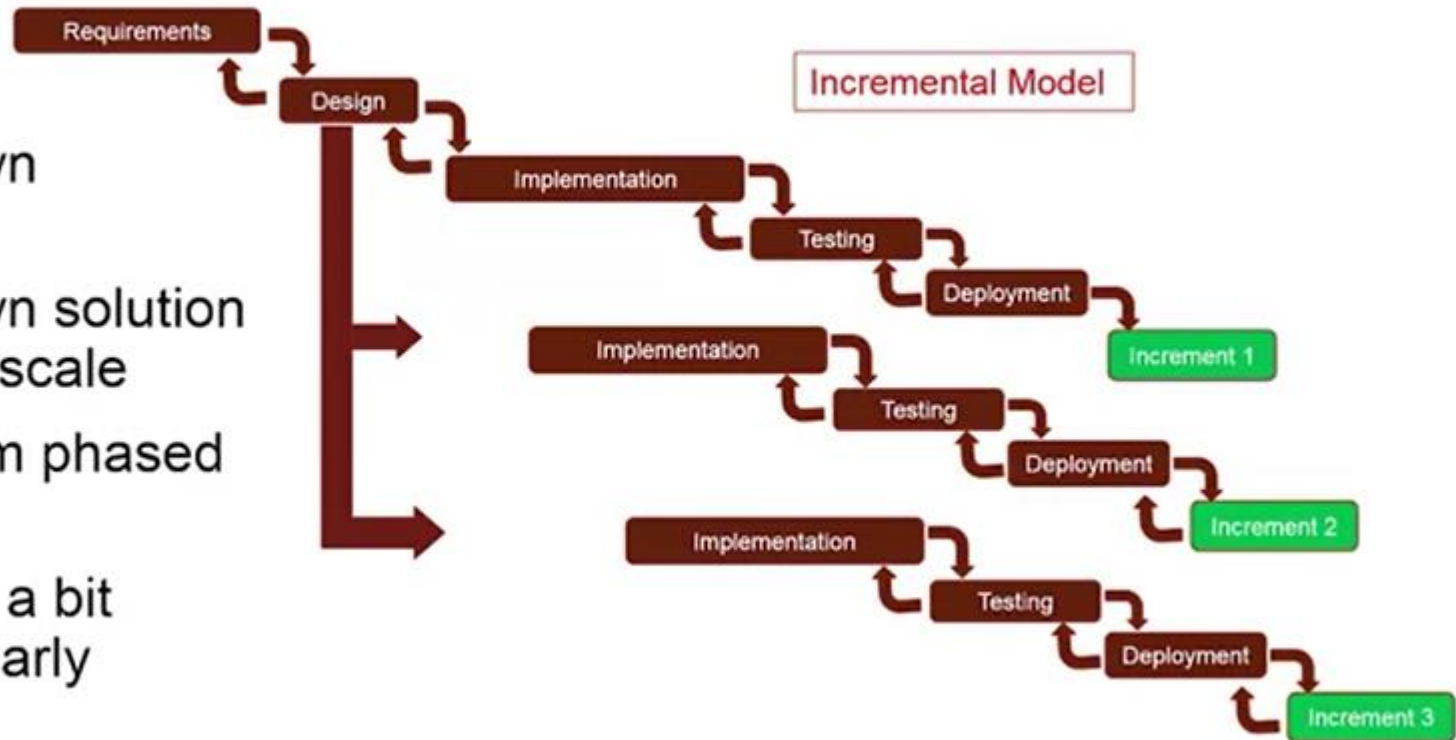




EXAMPLE 2

- Very large hospital with locations all over the world wanted to automate their processes
- Hired a company that is expert in this area and has done similar automation but not at this scale.
- Hospital management want consistency across the globe but not sure about the nuances in different part of the world.
- Few of the places can benefit greatly from the immediate automation

EXAMPLE 2 — ANALYSIS & RECOMMENDATIONS



- Fairly known problem
- Fairly known solution except the scale
- Benefit from phased delivery
- May tweak a bit based on early iterations
- Wants consistency

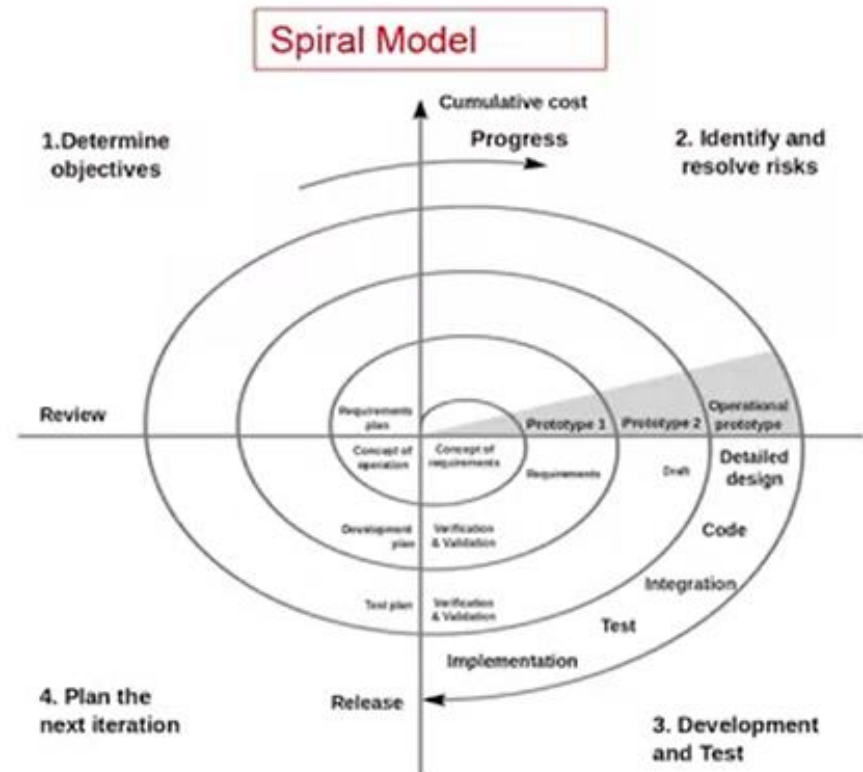


EXAMPLE 3

- Defense organization of a country did a recent study and the research recommends new capability the country should build to keep the country protected from potential conflicts in the region.
- The system required to be build has never been attempted and no literature exists for such system.
- It is fairly big and complex system and potentially can take a decade to build.
- Scientists have vague ideas but no concrete plan exists.
- There are lot of Organization stakeholders and constraints that will impact this initiative.

EXAMPLE 3 – ANALYSIS & RECOMMENDATIONS

- Fairly unknown needs and outcomes
- Very very risky
- Very large and complex project





EXAMPLE 4

- A fairly big organization has a product for processing medical prior authorizations submitted by insurance members. Product has been in use for years. This Organization mostly follow traditional project mgmt.
- The current system has several limitations and has not been keeping up to date with current industry trends. It cannot fulfill functionality expected by clients.
- Organization wants to build a whole new system that will satisfy client needs and setup the organization for future.
- The potential team who will be working on this has fairly good domain knowledge and knows the base technology pretty well. There is still lot of new technology they will be working with.
- The current system is fairly complex and it won't be easy to migrate existing workflows to new system. Expected duration of this migration is around 2 years.
- Lot of stakeholders (including leaders, managers, potential users) will be impacted by this change. This means significant training, change management, stakeholder mgmt is needed to be successful.

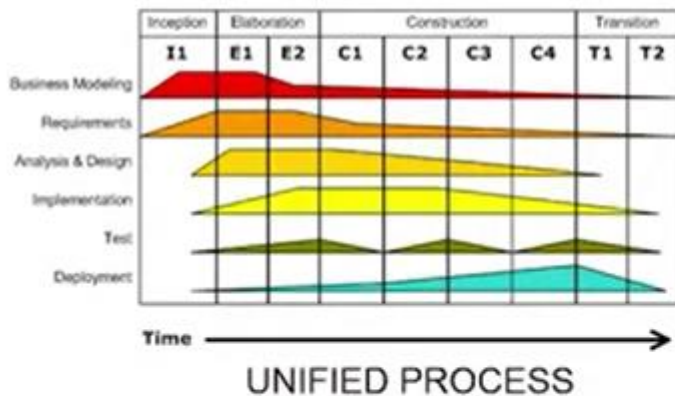


EXAMPLE 4 - ANALYSIS

- Some unknowns and risks
- Since team has good domain knowledge and base technical expertise, it is not totally new space for the team.
- Medium size project
- Complex deployment and change management
- Architecture is key to setup the organization for future. Need some work to get this right.
- Looking at complexity and number of stakeholder, it will be beneficial to roll this out incrementally to get some feedback and tweak the system.
- Most of the projects in the Organization are managed using Traditional approaches and leaders prefer certainty. milestones and solid plans.



EXAMPLE 4 - RECOMMENDATIONS



OR

