Software Engineering

Unit 1: 1 Software Engineering, Process models an Introduction

Software is Complex

- Complex $\neq$ complicated
- Complex = composed of many simple parts

  related to one another
- Complicated = not well understood, or explained

More Complexity

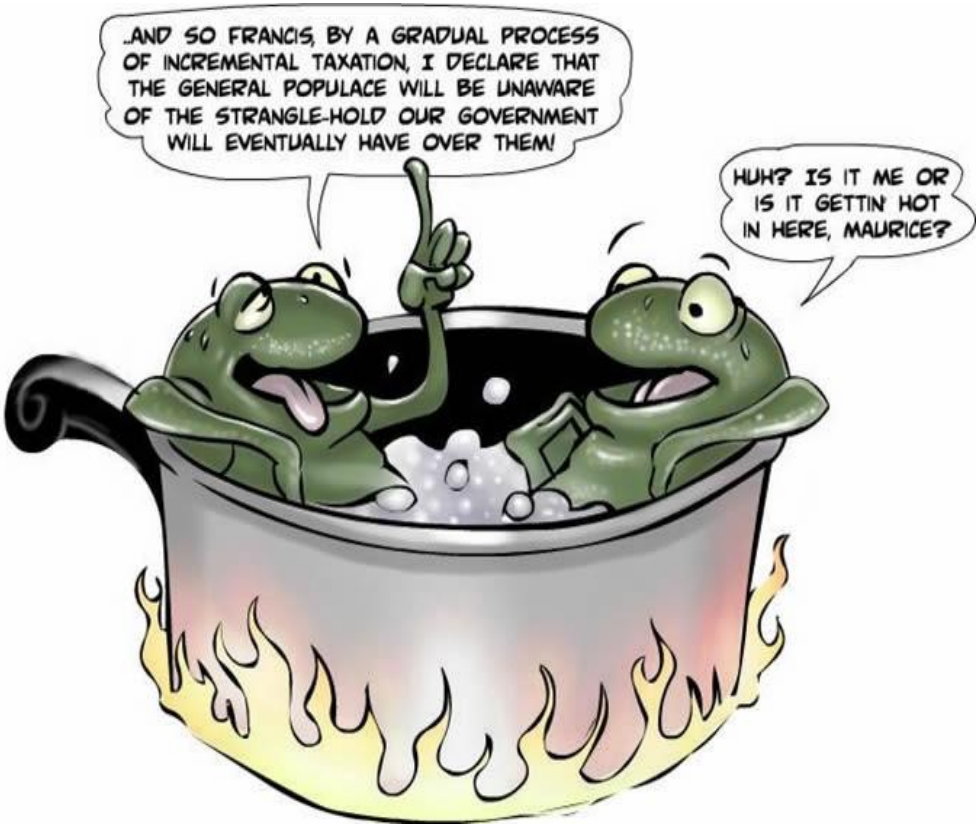
Suppose today is Tuesday, November 29

What day will be on January 3?

[**To answer, we need to bring the day names and the day numbers into coordination, and for that we may need again a pen and paper]**

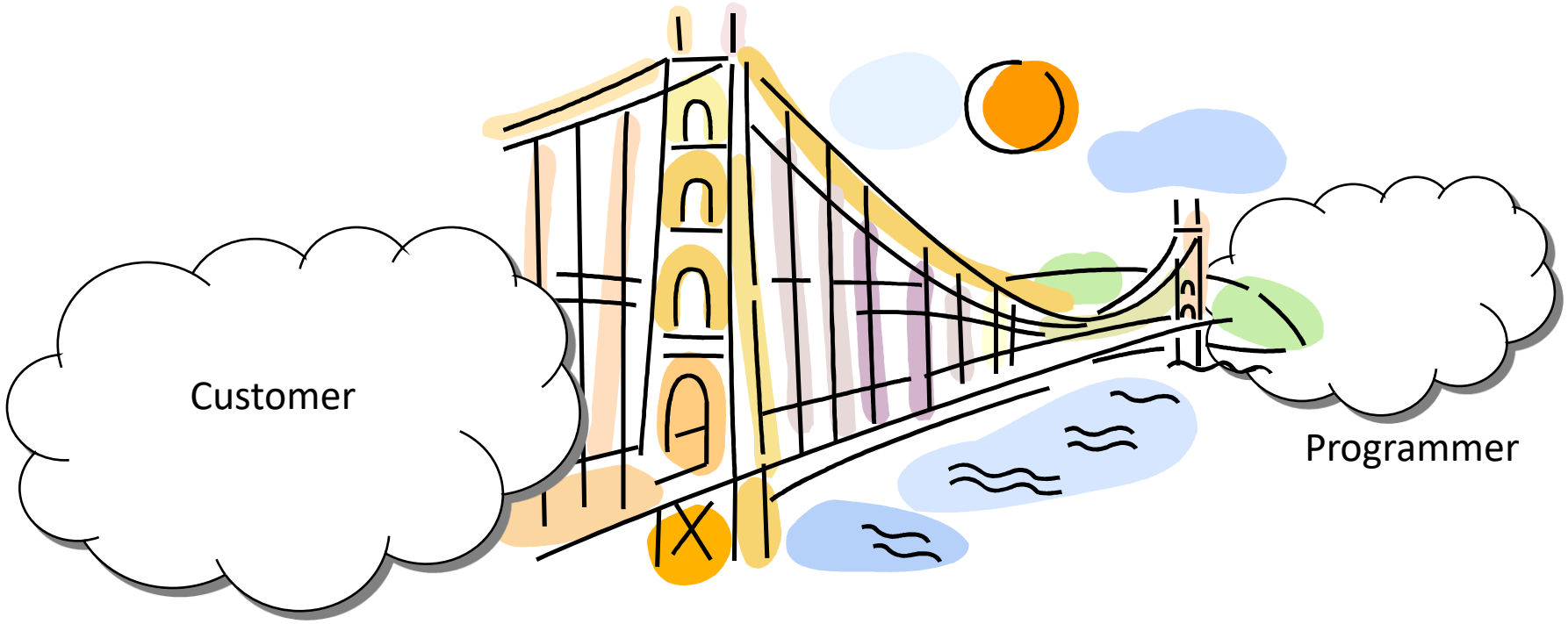
The Frog in Boiling Water

- Small problems tolerate complacency—lack of immediate penalty leads to inaction.
- Negative feedback accumulates subtly and by the time it becomes painful, the problem is too big to address.
- Frog in gradually heated water analogy:
  - The problem with little things is that none of them is big enough to scare you into action, but they keep creeping up and by the time you get alarmed the problem is too difficult to handle



The Role of Software Engg.

- A bridge from customer needs to programming implementation

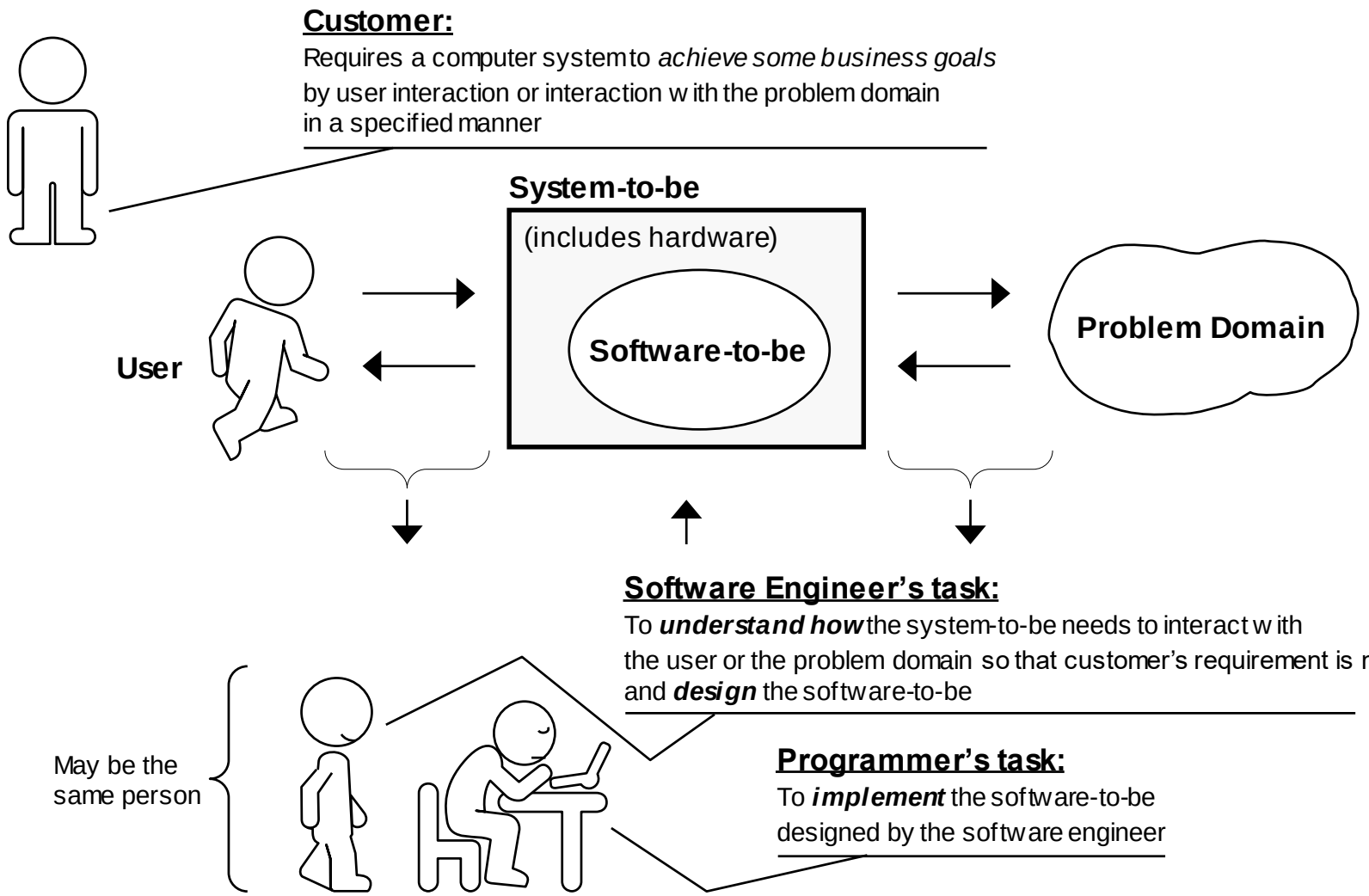


First law of software engineering

Software engineer is willing to learn the problem domain

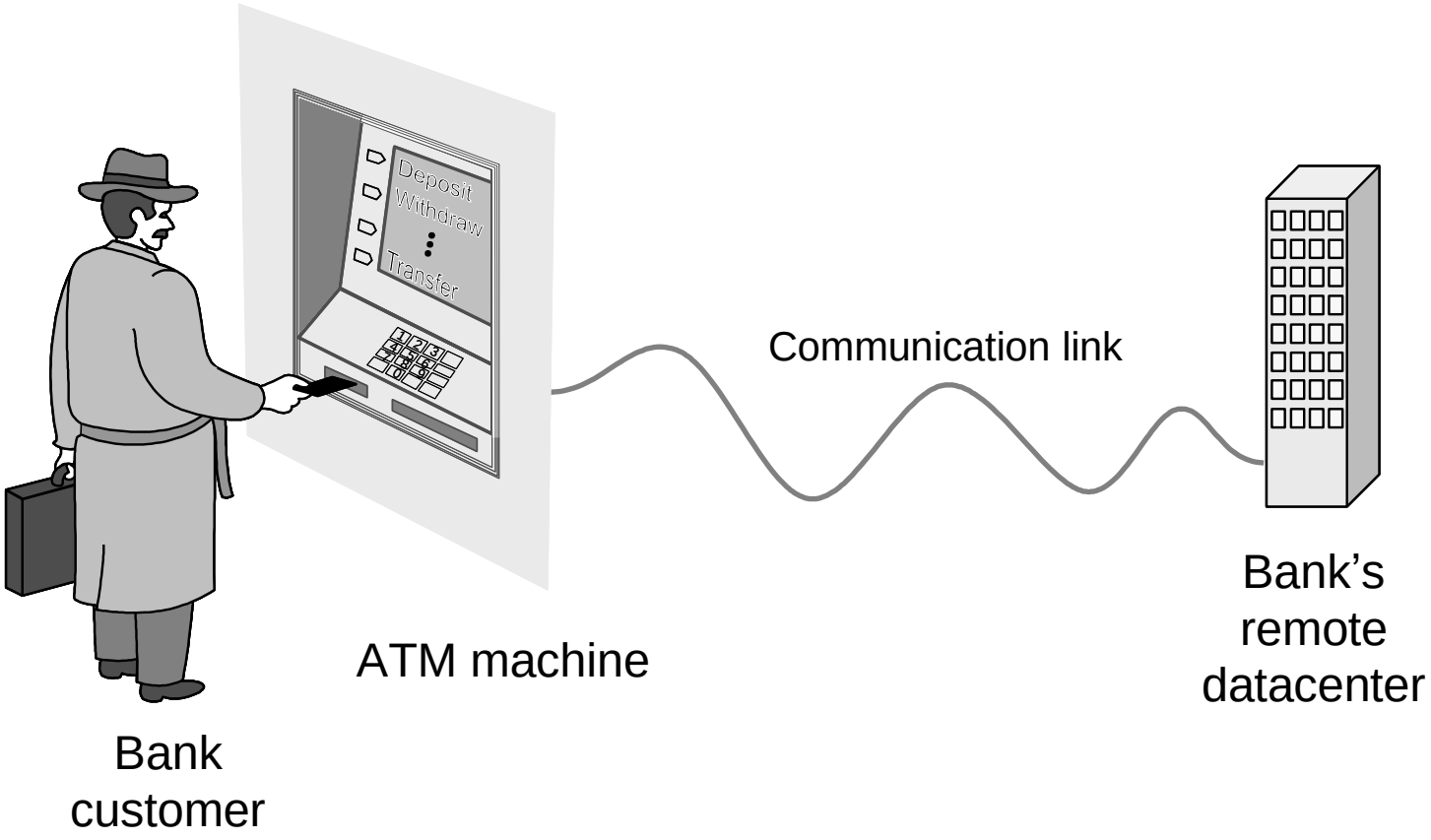
(problem cannot be solved without understanding it first)

The Role of Software Engg. (2)

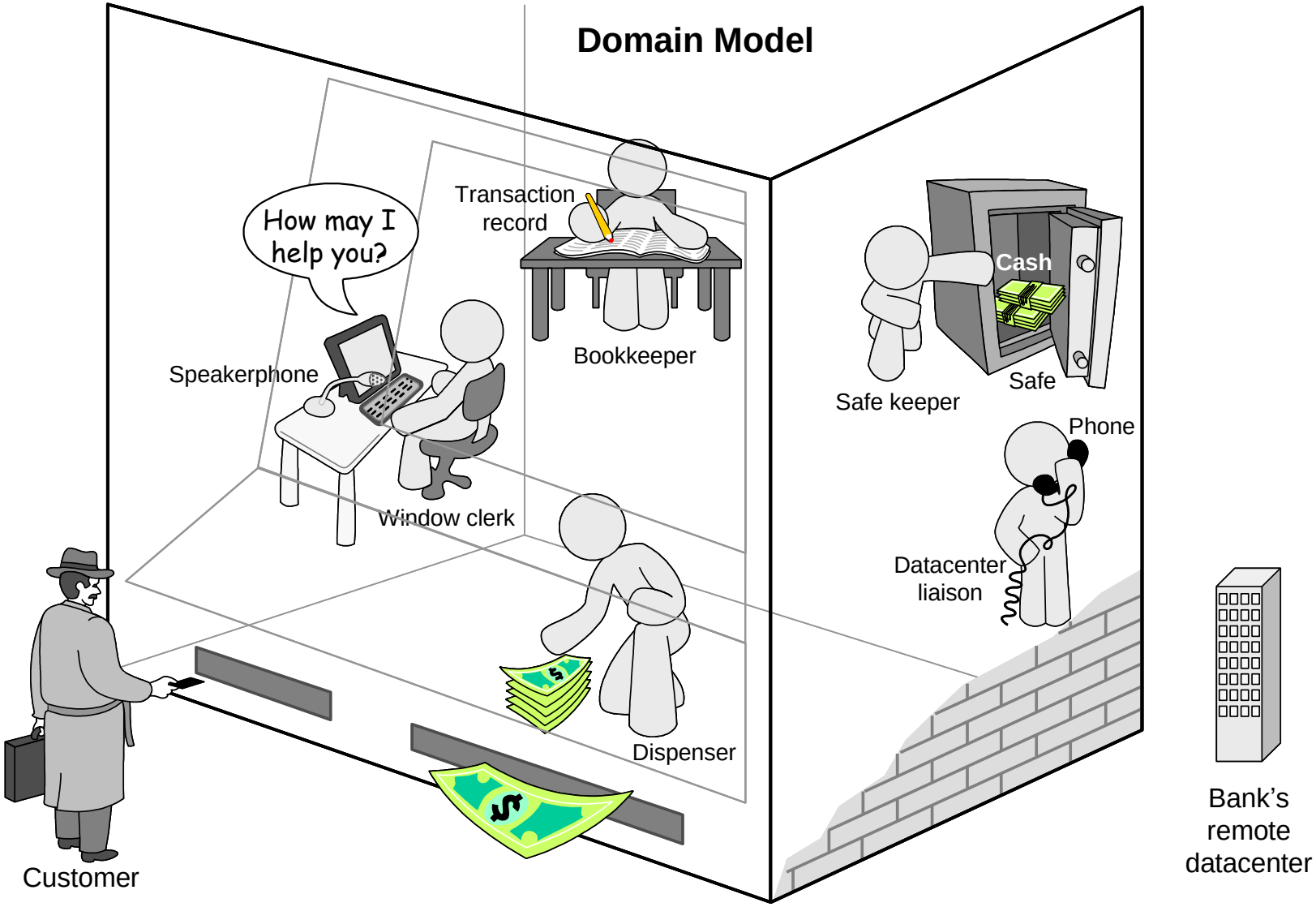


Example: ATM Machine

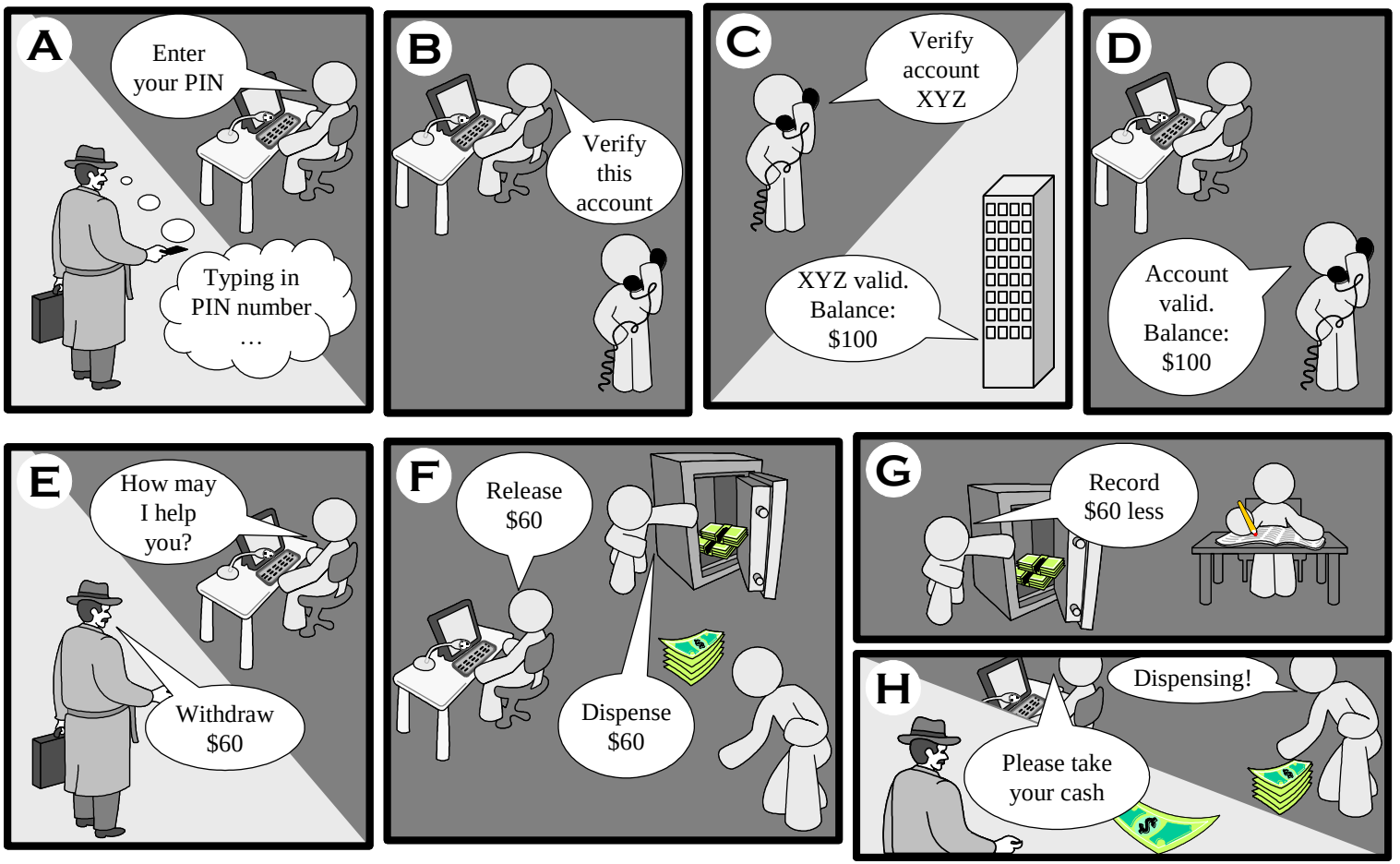
Understanding the money-machine problem:



How ATM Machine Might Work



How ATM Machine Works



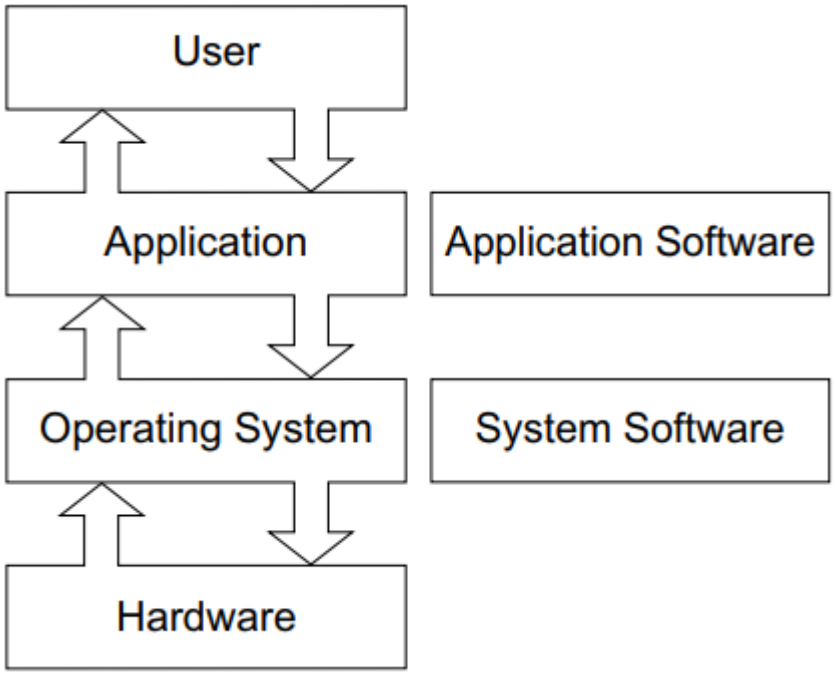
- **Software should be written for people first**
  - (Computers run software, but hardware quickly becomes outdated).
  - Useful + good software lives long
  - To nurture software, people must be able to understand it.

What Is Software?

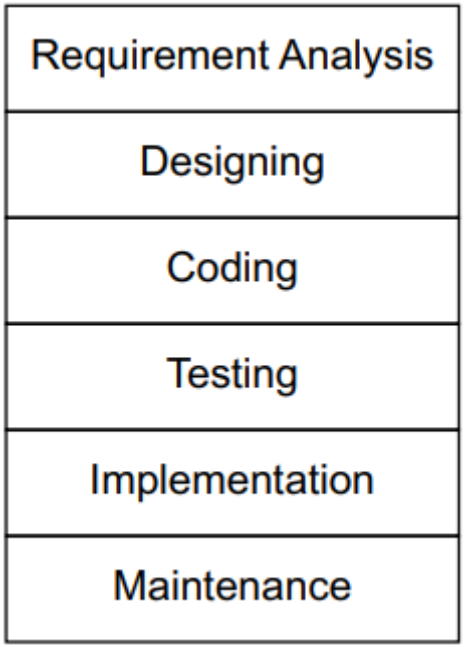
- Software is defined as the tool with designated order of instructions, performing tasks to provide the desired results after obtaining the requirements of the users.
- Software = Computer Programs (Instructions) + Procedures + Rules + Associated documents + Data
- Software is classified into two types based on the areas they command, namely:

  1). Application software and 2). System software
- ***System Software*** comprises programs that control the operations of the hardware components.
- The technical functioning of the computer is dependent on the operating system, built in with a number of adequate system software.
- System software controls the transfer of messages handling the internal processes, which are mostly abstracted from the external users of the computer.
- ***Application Software*** consists of programs purely for the tasks of the users, obtaining the inputs and processing them in the instructions already defined.
- They have no access to the hardware and the prerecorded system software of the device.

What Is Software?



Phases in Software Development



Characteristics of Software

- ***It should be complete:*** The software product being developed should meet all the requirements of the customers no matter how complex the product is, and the requirements agreed upon should be present in the product. The final product should not miss out any of the required functions.
- ***It should be exclusive and reliable:*** The solution obtained after using the software should be distinct. The software needs to be stable for a considerable time and usage, providing an unfailing service
- ***It should be precise:*** Unwanted steps and operations should be eliminated and shorter operations with the right solution should be preferred. Unwanted steps lead to wastage of resources, manpower and hence the cost.
- ***It should be efficient:*** A well organized software produces an optimal result at every moment of usage. The structure of the software needs to obey a strict and predefined order to be effi cient. It should also be fl exible in nature so that it can be extended at a later point of time.

What is Software Engineering?

- Software engineering comprises strict disciplines that need to be followed to develop a programmable solution to solve customer's problems.
- Strategies are defined to obtain a viable (વ્યવહારુ), efficient (કાર્યક્ષમ) and maintainable (જાળવી શકાય તેવું) software solution, which takes care of the costs (ખર્ચ), quality (ગુણવત્તા), resources (સંસાધનો) and other expenditures (અન્ય ખર્ચ).
- These strategies have been proposed by various engineers after implementing and analyzing the merits (ગુણો) and demerits (ખામી) of each strategy.

SOFTWARE PROCESS

- A set of interrelated actions (આંતરસંબંધિત ક્રિયાઓ) and activities to achieve a predefined result is called as process.
- Process has its own set of inputs and produces the output(s).
- Tools and techniques acts on the process to achieve the desired outcome.
- The processes have to be planned with great care and the knowledge about the effects of each implementation and activity on the product should be understood clearly by all the stakeholders.

Software Development Process: The Linear Sequential Model

- The process of development is divided into sequence of actions (steps) from requirements analysis, designing, coding and testing, implementation and finally maintenance.
- Each activity is carried out in a linear order (in sequential fashion) and no other order of execution is followed other than the flow mentioned.
- The result of the first activity is forwarded to the next stage and thus to the final stage. Without the completion of the previous step and forwarding the result, the next activity cannot be executed.
- It follows a strict sequence and it is strongly believed that no step can be overlapped.
- Prototyping Model and Rapid Application Development Models can be the best examples of this model.

Advantages and Disadvantages

Advantages

1. Easy to understand
2. Easy for implementation
3. Widely used and known; hence all stakeholders understand this
4. Identifies deliverables and milestones upfront.

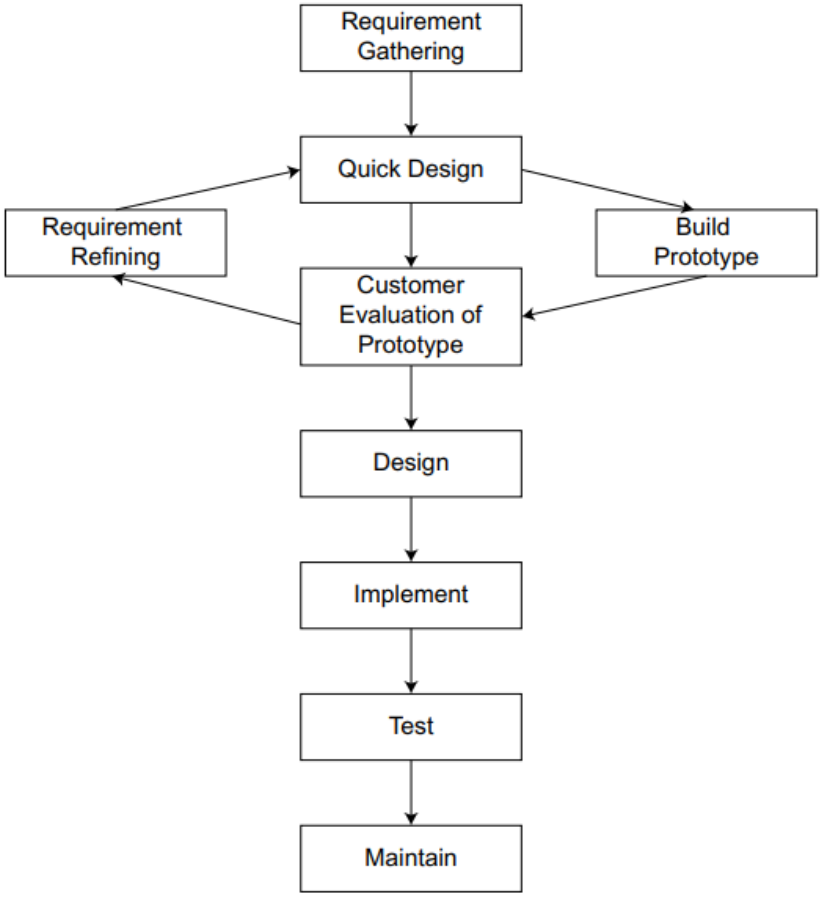
Disadvantages

1. This model assumes that requirements can be frozen before starting the Design Phase, which is not true in most of the situations.
2. Requirements are elaborated over a period of time.
3. Strictly following this model takes longer time to finish the project as each step takes its own time.
4. Specialists are required to run each step, which is difficult in long-term projects.

Software Development Process: Prototyping Model

- The prototyping model was proposed with a completely different strategy.
- Every product to be developed initially receives the requirements from the customers.
- With the given requirements, a sample or test model called the prototype is developed and displayed to the customer.
- The customer would respond by giving their opinion and rectifications needed over the current prototype.
- The prototype is altered until the customer is satisfied on the style, design and execution of the processes.
- Otherwise the development team continues to consider different options for creating the prototype. ***(See the image on next slide)***

Prototyping Model…



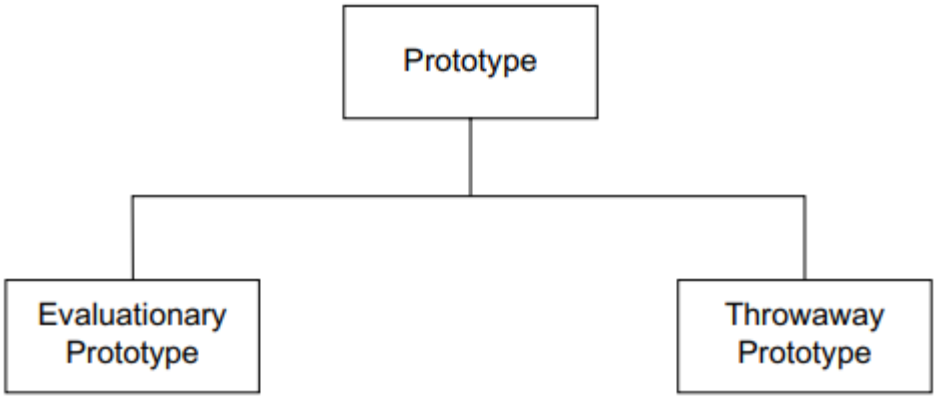
Prototyping Model…

- There are two types of prototypes developed in this model, namely:

  1). Evolutionary prototypes and 2). Throwaway prototypes

1). ***In the Evolutionary prototypes*** type wrong prototype will be reused and the changes are implemented in that particular prototype. The new changes and corrections are updated in the existing prototype until the final list of requirements is obtained, without wasting the used resources.

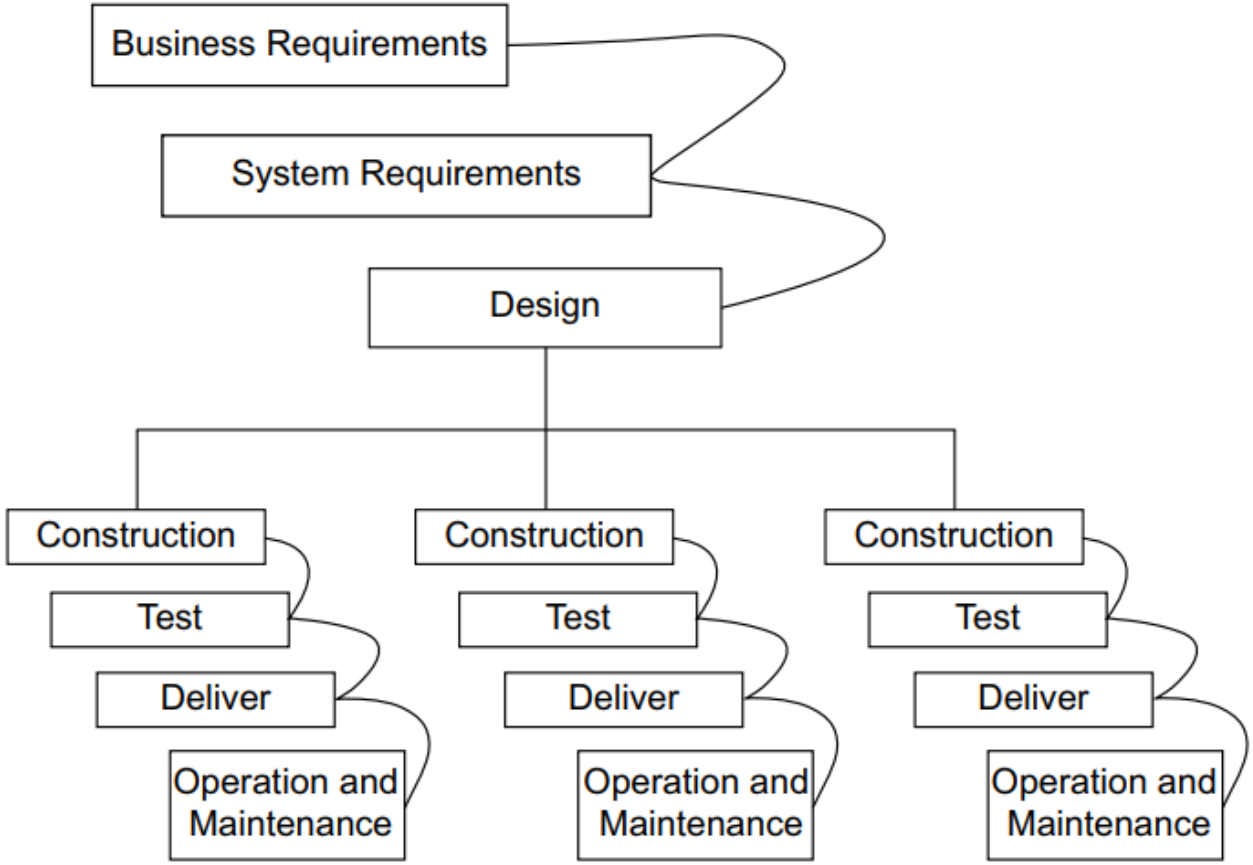
2). ***The Throwaway prototypes***, as the name suggests, destroys the wrong prototypes, and creates the new prototype from the scratch every time the user rejects it.



Software Development Process: RAD Model

- Rapid Application Development (RAD) model is the methodology proposed for quicker design and deployment of a product.
- RAD model requires a very short time span of 60–90 days for completing the software and delivering it to the customer.
- RAD model proves to be the most (વધારે સારું) solution to quick needs and implementation of web based applications.
- Already present with numerous applications in web-based concepts, the customer may have to just refer the available applications and select from the same.
- Reuse of existing prototypes in the referred application facilitates faster deployment of the new application.
- RAD model includes the following five phases for development:
  1. Business modeling
  2. Data modeling
  3. Process modeling
  4. Application generation
  5. Testing and turnover

RAD Model…


Business Requirements
System Requirements
Design
Construction
Test
Deliver
Operation and Maintenance
Construction
Test
Deliver
Operation and Maintenance
Construction
Test
Deliver
Operation and Maintenance

RAD Model…

1). ***Business modeling*** : Business modeling is the stage in which the types of information that are prime (ખુબજ જરૂરી) factors of the application are defined.

With the details on the information being processed, the area of functionality of the application can be determined (નિર્ધારિત).

2). ***Data modeling*** : The obtained information from the previous phase is filtered into useful and meaningful sets of data. They are refined (શુદ્ધ) into entities of high importance.

The characteristics of the entities and their relationships are also defined in this data modeling phase.

3). ***Process modeling :*** Process modeling defines the activities needed to process the entities. Every instruction is framed in this phase.

The instructions concern the processing of the data from the initial to the final stages of the application.

4). ***Application generation :*** Application generation is a phase that uses automated tools to generate the working model of the designed application.

The automated tools are used for faster analysis of similar components for reuse and save the time for new designs.

Software tools have provided a wide range of amenities for faster deployment of an application with the same accuracy.

5). ***Testing and turnover :*** The final phase is to test the correctness (ચોકસાઇ) and consistency of the developed application. With such a short time and implementation of many reusable components, the accuracy of the application has to be tested with greater care.

Software Process Model

- Software process models are systematic methods for controlling (નિયંત્રણ) and coordinating (સંકલન) the development of a software product achieving all the stated objectives (જણાવેલ ઉદ્દેશ્યો).
- Methodical (પદ્ધતિસરની) processes are followed from the initial requirements analysis process to maintenance phase.
- Software process models help the developer team to recognize the following:

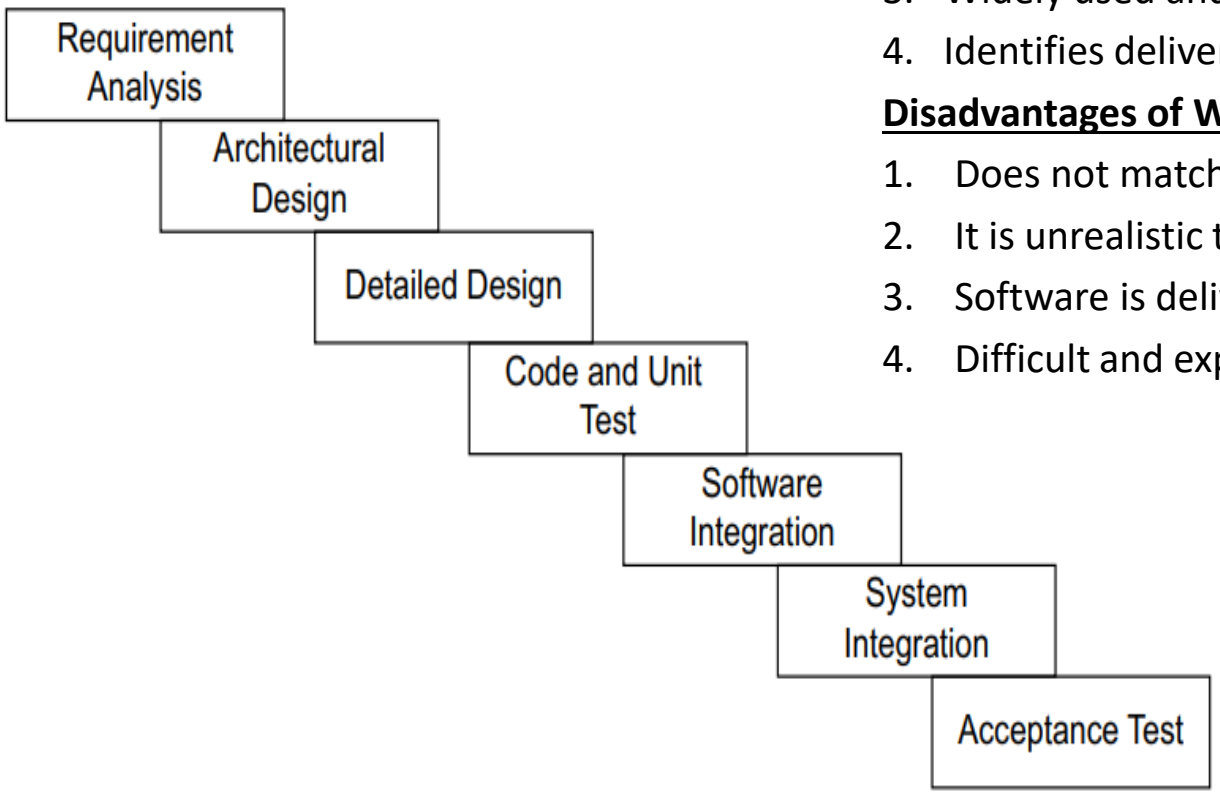
  1). Set of tasks, subtasks and interfaces 2). Resources needed for every process 3). Adequate time span

Traditional Software Process model

1). Waterfall model:

- Waterfall model is the first and foremost methodology of Software Development Life Cycle (SDLC).
- All the processes of the waterfall model follow sequential order.
- The requirements and conditions are obtained from the customer in the first step.
- This specification (સ્પષ્ટીકરણ) and analysis is obviously the first process that leads the development team.
- The development team will start designing a framework for achieving the objectives.
- Following the efficient design, the coding begins.
- Testing is the phase that ensures correctness of the product being developed.
- Without testing, no product will be allowed to be implemented in the real-world environments.
- The maintenance process cannot be predicted and assumed.

Software Process Model: Waterfall Model…



Advantages of Waterfall Model

1. Easy to understand
2. Easy for implementation
3. Widely used and known
4. Identifies deliverables and milestones upfront

Disadvantages of Waterfall Model

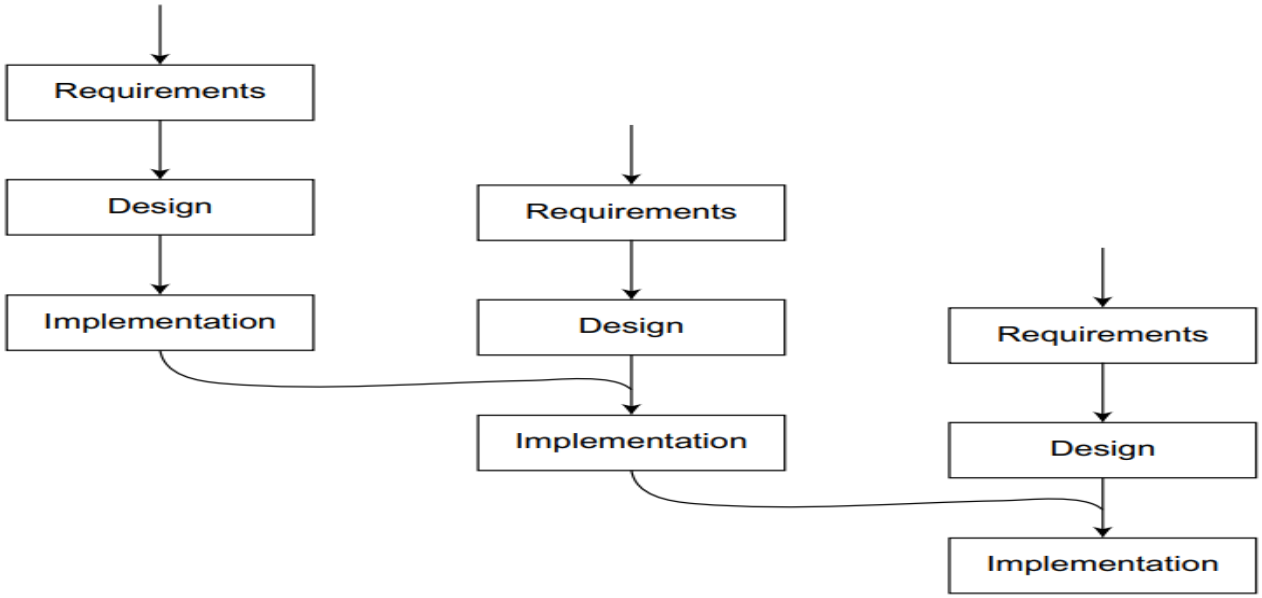
1. Does not match well with reality
2. It is unrealistic to freeze the requirement upfront
3. Software is delivered only at the last phase
4. Difficult and expensive to make changes

Software Process Model: Incremental Process Model

2). Incremental process model:

- The incremental model is one of the evolutionary (ઉત્ક્રાંતિવાદી) models, which overcame the drawback of the linear execution style of the waterfall model.
- Incremental model has the capability of multiple streams being progressed independently and simultaneously.
- Once the requirements are gathered, separate teams can work individually to produce a product assigned.
- The whole product is achieved by integrating all the individual components developed.
- The major advantage of the incremental model is that the consecutive steps do not have to wait until the preceding function completes its activity.
- The large and complex problem can be divided into a number of smaller and unique tasks, providing an easier chance to solve.
- Incremental model is suitable for applications that need additions and alterations in its contents after certain time period.
- Incremental model also facilitates the isolation of errors and faulty components at a faster rate.
- Versions and higher configurations of an existing product can be developed with incremental process model.

Software Process Model: Incremental Process Model…



Advantages of the Incremental Process Model

1. Quick feedback loop from business stakeholders
2. Focus on customer feedback
3. Customer feels the product early

Disadvantages of the Incremental Process Model

1. Scheduling of development is difficult with incremental model
2. Although the final product is given in pieces, tracking and monitoring is difficult
3. Testing needs to be exhaustive for each part

Software Process Model: Spiral Model

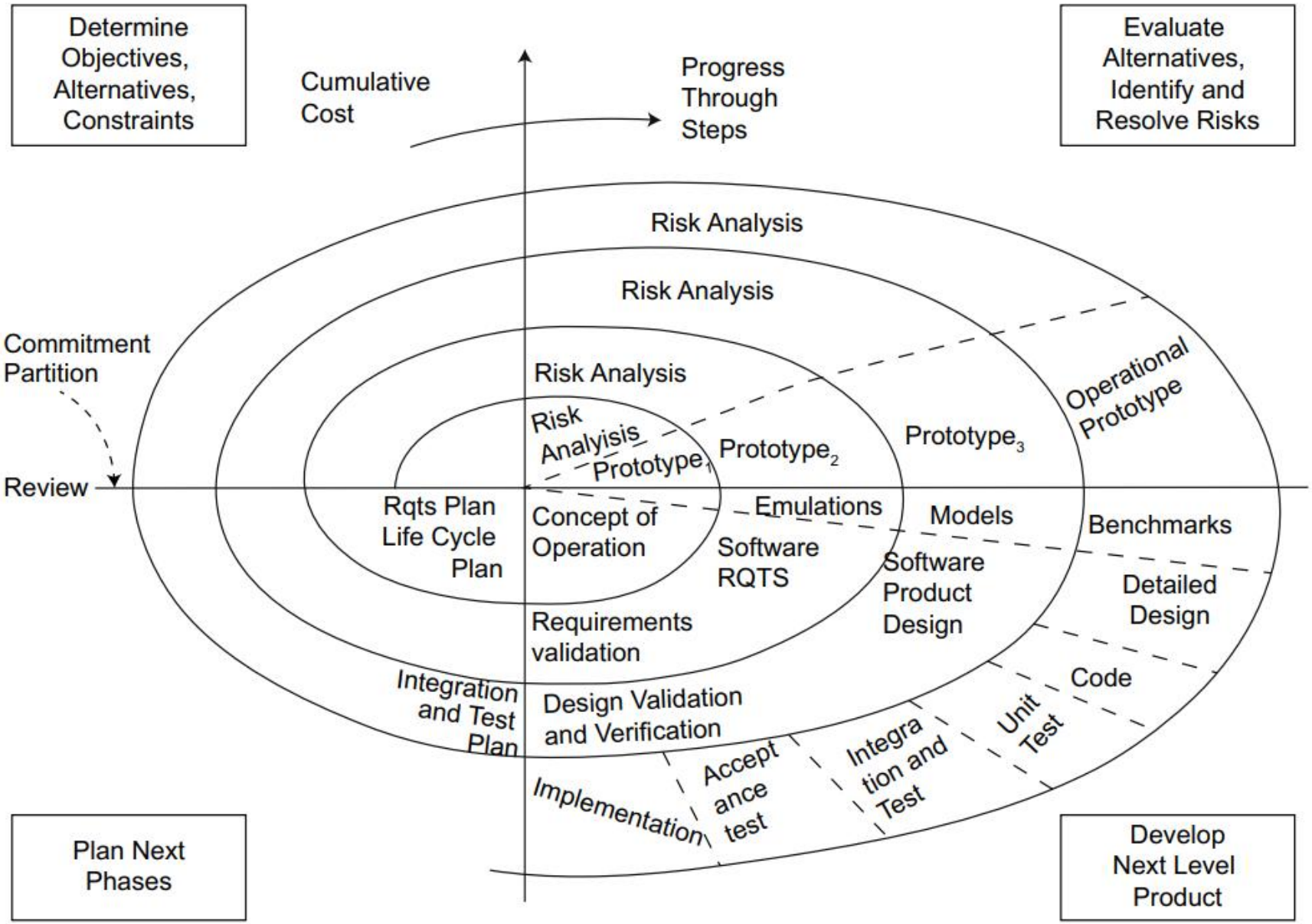
Evolutionary Software Process model

1). Sprial model:

- This method introduced the concept of risk analysis in the earlier stages of development process.
- Spiral model is also called Risk Spiral Model.
- This model combines the software development life cycle with Risk Management principles to control the risks at early stage of the project.
- A spiral model encompasses the iterative steps in a spiral representation.
- Each circle in the model denotes a succeeding level of the previous state.
- These are divided task regions, which represent the selective process.
- The task regions may vary from the complexity level of different products.
- Life Cycle Plan is prepared first; Development Plan is prepared in the next level; and Integration Plan is in the last level.
- In the Risk Analysis Phase the prototype is being developed.
- The risk analysis would report the identification and effects of the present risks in the prototype.
- Assessments of risks at every rotation of the spiral evolution reduce the problems and the time required for solving at the final stage

Software Process Model: Spiral Model…

- Construction and release also happens in iterative fashion based on the base prototype developed.
- At every level it is also being strengthened and the final level is the final code to be delivered to the customer.
- The number of iterations is determined by the team manager depending upon the rectifications made and completion of the product.
- The cost and schedule cannot be assumed right at the beginning phase as the prototypes are altered at every rotation.
- The team manager adjusts cost and time metrics after every circle. The spiral model is selected for software of higher level organizations.


Determine Objectives, Alternatives, Constraints
Evaluate Alternatives, Identify and Resolve Risks
Cumulative Cost
Progress Through Steps
Risk Analysis
Risk Analysis
Risk Analysis
Commitment Partition
Review
Risk Analysis
Risk Analysis
Prototype1
Prototype2
Prototype3
Operational Prototype
Rqts Plan
Life Cycle Plan
Concept of Operation
Emulations
Models
Benchmarks
Software RQTS
Software Product Design
Detailed Design
Requirements validation
Integration and Test Plan
Design Validation and Verification
Code
Unit Test
Integration and Test
Acceptance test
Implementation
Plan Next Phases
Develop Next Level Product

Software Process Model: Component-based Development Model

2). Component-based Development Model.

- This development model is one of the evolutionary practices for developing a software product.
- Component-based development introduced the concept of reusability (પુનઃઉપયોગીતા) to the full extreme.
- After analysis of the requirements, instead of entering into the design phase, components or modules of the existing similar products are checked for their match.

a). Requirements Comparison:

The requirements needed for a new product are compared with the recorded list of requirements from the existing products.

The percentage of match between the new and existing requirements is evaluated.

The matching level should be considerably high enough to be a perfect match.

b). Modification and Implementation:

After a suitable match or a near-matching component is perceived, they are implemented in its place.

But the question is whether a 100% match can be obtained or not.

The result is a NO in most cases. No component can be directly implemented into a new product. The near-matching product needs some modifications to meet the stated requirements.

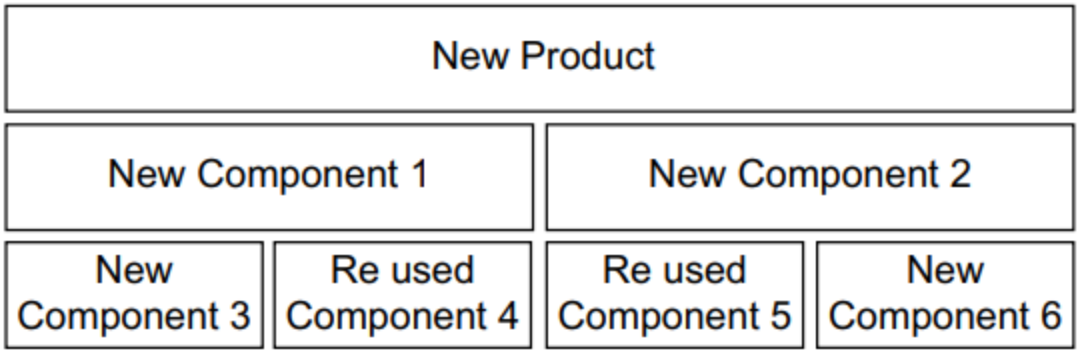
Software Process Model: Component-based Development Model

c). Risk Analysis

Integration of existing modules has a greater level of risk than the newly developed components.

The outcome of the product may be similar but the preceding and succeeding activities may disturb the new component.

Hence, a more sincere risk analysis methodology is required for eradication of the risks.



Agile Software Development: An Introduction, Characteristics

- Agile is a type of software development methodology that expect (અપેક્ષા રાખે છે) the need for flexibility and applies a level of practicality to the delivery of the finished product.
- It focuses on the clean delivery of individual pieces or parts of the software and not on the entire application.
- It provides the opportunity to make changes as needed and alert teams to any potential issues.
- Agile software development refers to a group of software development methodologies based on iterative development

Characteristics of Agile Projects

There are mainly four characteristics of Agile projects. They are:

1). Adaptability in Agile projects

2). Visibility in Agile projects

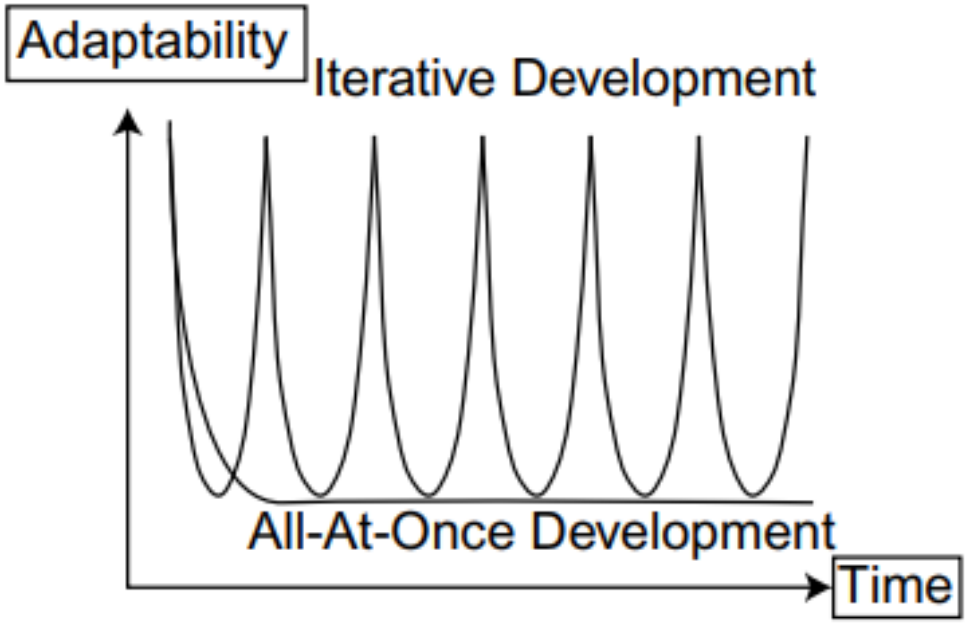
3). Value in Agile projects

4). Risks in Agile projects

Characteristics of Agile Projects...

1). ***Adaptability in Agile projects:*** (Adaptability = અનુકૂલનક્ષમતા)

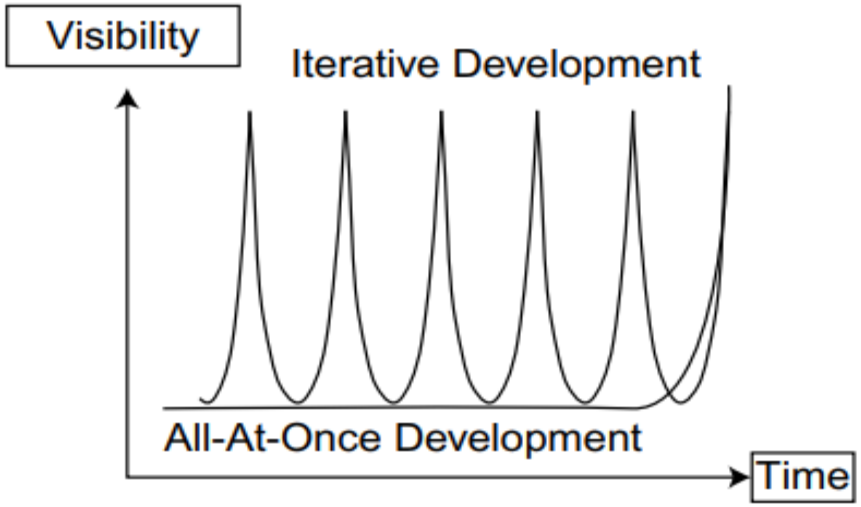
- In traditional projects (all at once development), the adaptability is only in the beginning of the project and it goes down drastically once the project starts.
- There is no adaptability in the traditional projects once the execution of the project starts.
- In agile project, because it follows an iterative development approach, the adaptability is maintained throughout the life cycle of the project.



Characteristics of Agile Projects...

2). <u>Visibility in Agile Projects:</u> (Visibility = દૃશ્યતા)

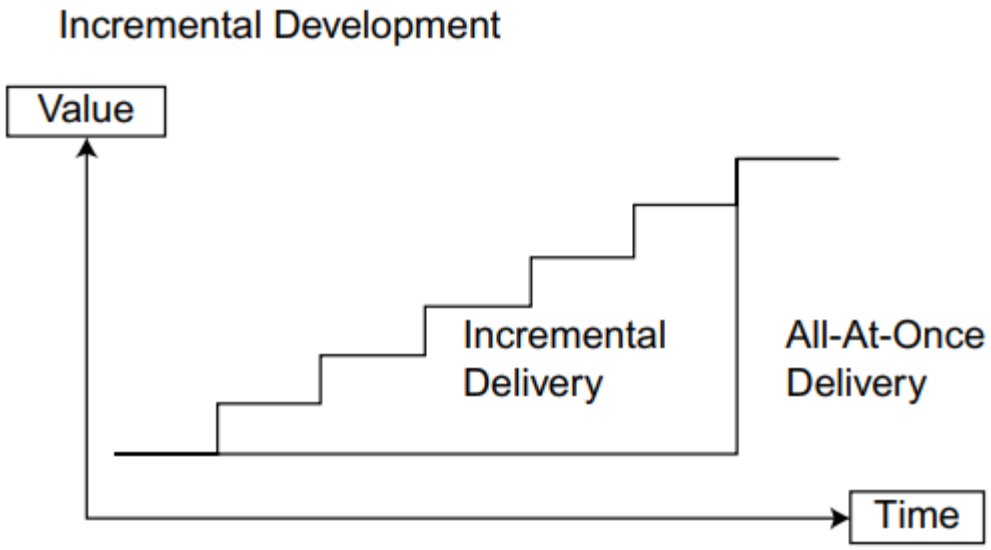
- In traditional projects (all at once development), the visibility of the product is high only at the end of the project, as the product is given to the customer only at the last phase of the project.
- There is no visibility in the traditional projects until the end of the execution of the project.
- In agile project, because it follows an iterative development approach, the visibility is maintained throughout the life cycle of the project.
- Visibility is maintained throughout the project and the customer knows what he wants and what is happening.



Characteristics of Agile Projects...

3). <u>Value in Agile Projects:</u> (Value= મૂલ્ય)

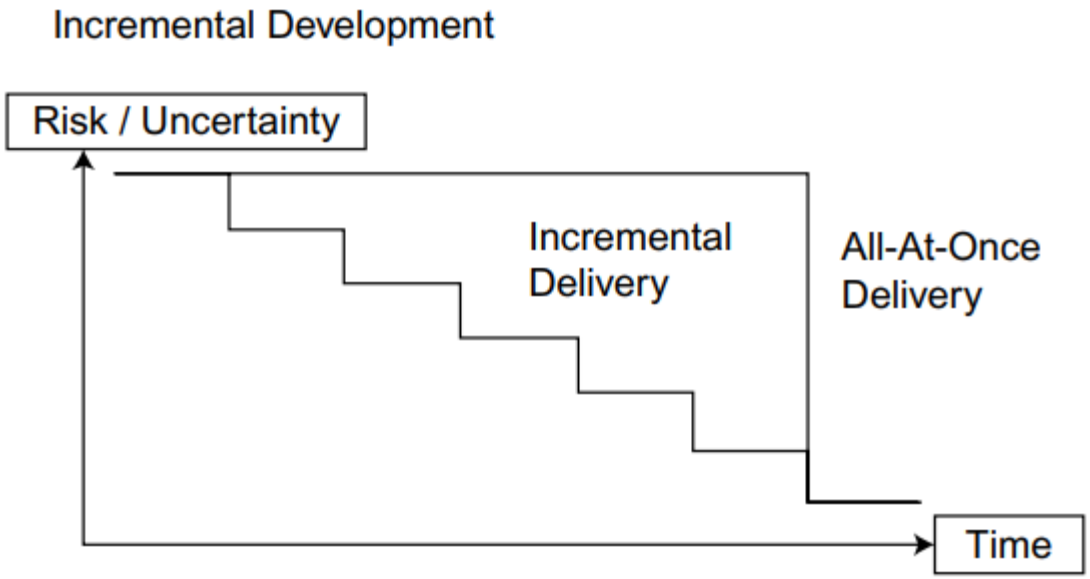
- In traditional projects (all at once development), the value is delivered to the customers and end users only at the end, so the value map increases only at the end.
- In agile project, due to its incremental nature, the value is also delivery incremental in steps.
- Customer gets early value in the agile project than in traditional projects.



Characteristics of Agile Projects...

4). <u>Risks in Agile Projects:</u> (Risk= જોખમ)

- In traditional projects (all at once development), the risks level gets lowered only at the end after the product is delivered to the customer.
- In agile project, the risks gets decreased every iteration as highly risky items are executed in the first few iterations.
- Fail early is the principle of agile.



Limitations of Traditional Software Models

1. Modern projects are constrained by lot of uncertainties more on what is expected/What is anticipated (ધારેલું) in the beginning of the project.
2. Although there is as process to handle difficult situation, people should be empowered to execute it.
3. It is difficult to change the plan once it is base lined. It is difficult to change the plan once it is base lined.
4. It is difficult to change the plan once it is base lined.
5. Once initiated the changes are often very difficult to be inserted by the customer however genuine case it may be.
6. Traditional project management tries to define the timeline and cost based on the scope of the project. But as the project progresses, the scope is getting changed but the timeline is not revised which leads to team frustration.
7. Often the customer is ending up paying more than what was estimated for the project in the beginning.
8. As team is not empowered to take decisions, the execution part is entirely diff erent from the planning (mostly owned by the project manager).
9. Most of the projects are scrapped due to change in technologies and business conditions during the execution of the project.
10. Faces problem at the Acceptance Test.

Advantages of Agile Methodologies

1. Agile project involves the end user throughout the life cycle of the project to avoid conflicts at a later stage.
2. Agile project produces incremental product at the end of every phase of the project.
3. Agile project delivers important features first.
4. Customers are given flexibility to add/modify/delete the requirement until the team actually starts working on it.
5. Customers are also given options to prioritize the requirements.
6. Agile project engages all the stakeholders.
7. Agile project gets the commitment from the team.
8. Agile project does planning at multiple levels of the projects (in traditional project management, only high-level planning is done and once it is fixed execution starts. There are no plans at execution levels.)
9. Agile project encourages proactive risk identification.
10. Agile project allows open and transparent project management.
11. Agile project continuously improves product, process, and people (in traditional project management more concentration is on process.).

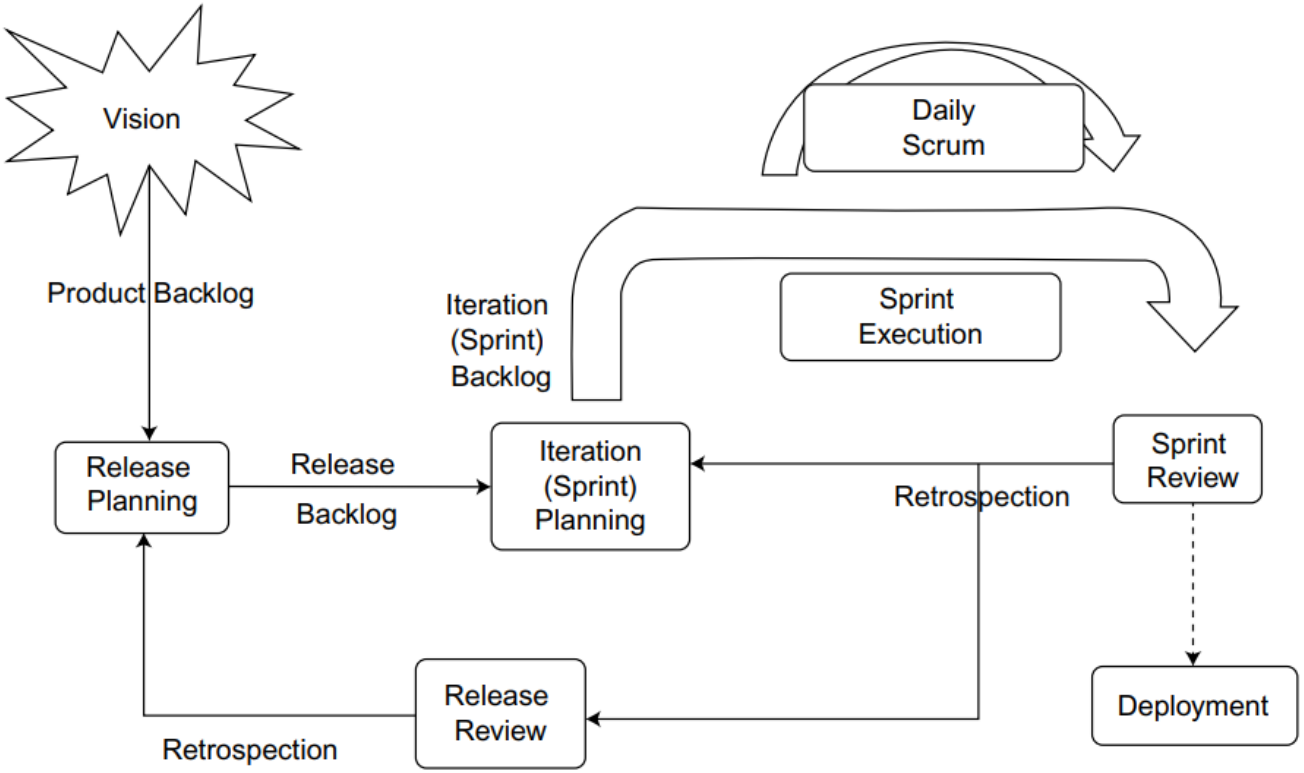
Agile Software Process Model: Scrum

1). Scrum Methodology

- Scrum uses standard project management concepts but with different terminology and best practices.
- Scrum is the most widely used methodology framework of agile.
- The term scrum is related to the Rugby sports.
- Some of the scrum best practices are short fixed iterative cycle, adjustable scope, customer satisfaction through quick turnaround time (કાર્ય પૂર્ણ કરવાનો સમય), responding to change quickly, and customer collaboration throughout the project.
- This methodology consists of three phases namely pregame, development phase, and post-game.
- Iterative cycle is called as sprint.
- Sprint planning is happening in pregame phase.
- Actual sprint execution is happening in development phase.
- Integration, retrospection is happening in the post-game phase
- A project team called a SCRUM team
- A product backlog is a list of all known requirements
- A sprint backlog is a list of all known requirements which team is going to work currently (in the current sprint)
- A period of work is called as sprint (it is actually the current phase of the project)

Agile Software Process Model: Scrum…

- Daily standup meetings happens with the SCRUM team
- An incremental product is delivered to the customer at the end of each sprint
- SCRUM master is a facilitator and leader and also responsible for teaching others how to use the scrum process to deal with every new complexity encountered during a project



Agile Software Process Model: Extreme Programming (XP)

- Extreme programming is a method that is based upon agile concepts and the supporting XP principals of rapid development, flexibility, team empowerment, and customer-based quality management.
- This can be used when customer changing the requirements more often (વધુ વખત) than expected, almost daily.
- Customer changes the requirement even after the entire coding gets over.
- And the worst part, customer wanted everything quickly.
- Where XP is good to use?:
  - Requirements changing almost every day
  - Total chaos in the project environment
  - Customer wants everything fast
  - Handling with new domain/technology
  - Total uncertainty in everything
- XP sometimes make use of Pair Programming, where two people sitting together and writing the same code.

Agile Software Process Model: Extreme Programming (XP)…

XP		
4 Accelerators	**4 Business Questions**	**5 Critical Success Factors**
1. Make change your friend	1. Who needs what and why?	1. Self-Mastery
2. Build on people's desire to make a difference	2. What will it take to do it?	2. Leadership by Commitment
3. Create ownership for results.	3. Can we get what it takes?	3. Flexible Project Model
4. Keep it simple	4. Is it worth it?	4. Real-Time Communication
		5. Agile Organization

XP Values

10 Shared values	
1. Client Collaboration	6. Fast Failures
2. People First	7. Early Value
3. Clarity of Purpose	8. Visibility
4. Result Orientation	9. Quality of life
5. Honest Communication	10. Courage

Agile Software Process Model: Adaptive Project Framework (APF)

- Adaptive project framework uses five phases:

  1). Version scope 2). Cycle plan 3). Cycle build 4). Client checkpoint 5). Post-version review

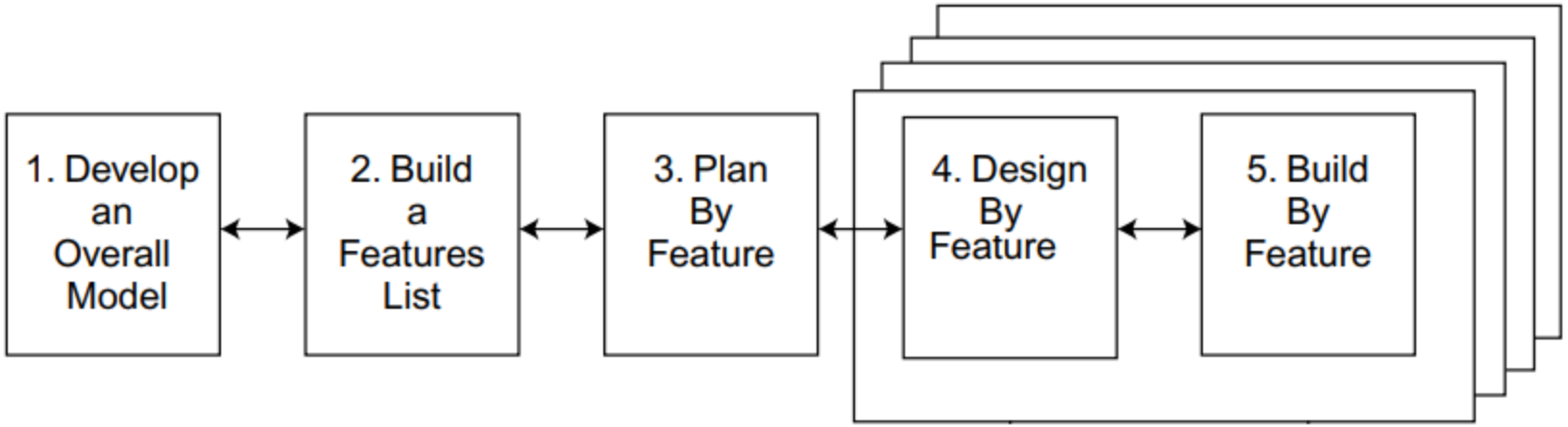
Phases	Main Output
Version Scope	Prioritized Functionality, number of cycle
Cycle Plan	Dependencies and Schedule
Cycle Build	Build Functionality
Client Check Point	Quality review of completed functionality
Post Version	Check on Business Outcome, Lessons Learnt

- Below is the difference between Traditional approach, Adaptive approach and Extreme Programming:

Traditional	Adaptive	Extreme
No Chaos	Chaos	Total Chaos
One Iteration	Known iteration	Unknown Iteration
Fixed Scope	Variable Scope	Unknown Scope
Change Intolerant	Change part of the System	Change necessary

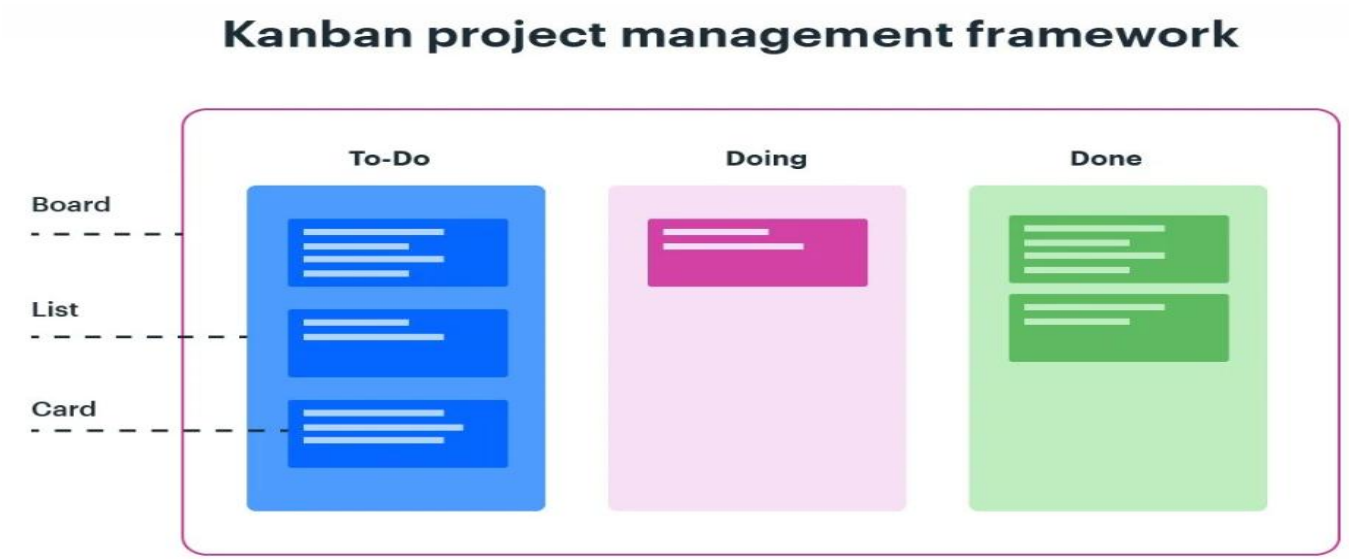
Agile Software Process Model: Feature Driven Development(FDD)

- FDD is very simple methodology and consists of only five processes namely, 1). develop an overall model of the system, 2). build the features list, 3). plan by feature, 4). design by feature, and 5). build by feature.



Agile Software Process Model: Kanban Method

- Kanban methodology is an agile method that aims at continuous improvement, flexibility in task management, and enhanced workflow.
- With this illustrative approach, the progress of the whole project can be easily understood in a glance.
- Kanban method revolves around the Kanban board.
- It is a tool that visualizes the entire project to track the flow of their project.
- **Kanban board indicates**
  - the current tasks that are being performed
  - the tasks to do in the future
  - the tasks that are completed



Agile Software Process Model: Kanban Method…

Core principles of kanban methodology

1. Initiate with the existing workflow:

- Kanban framework emphasizes on making small and gradual changes. Therefore, the team must start with the existing workflow and continuously improve the process.

2. Limit the existing tasks:

- It is important for the team to realize its own limits and cap the WIP accordingly. Taking on more than you can handle will only waste time and negatively affect the project.

3. Respect existing roles and responsibilities:

- An important reason for Kanban's success is that it does not require organizations to completely overhaul the existing work culture.
- Many organizations resist modern methodologies because they don't feel comfortable with change.
- With Kanban, efficiency is improved while staying in the confines of the existing setup.

4. Encourage leadership at all levels:

- Kanban gives the freedom of making decisions to the individual working on the task.
- This grooms future leaders who continuously learn from their mistakes and improve their work.