Chapter 2 : The Process

Books Referred : Roger Pressman 5th Edition Chapter 2

Topics Covered

- Layered Technology
- Software Process Framework
- Generic Process Framework Activities
- Umbrella Activities
- CMMI Level
- Software process model
- Build and Fix Model
- Why Models are needed?
- Different types of Process Models

Overview

- **What? A software process** – as a framework for the tasks that are required to build high-quality software.
- **Who?** Managers, software engineers, and customers.
- **Why?** Provides stability, control, and organization to an otherwise chaotic activity.
- **Steps?** A handful of activities are common to all software processes, details vary.
- **Work product?** Software Programs, documents.
- **How do I ensure that I've done right?** A number of assessment mechanisms to determine the "maturity" of a software process.

What is Software Engineering?

Definition :

- (1) The application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1)

Its a discipline that is concerned with all aspects of software production.

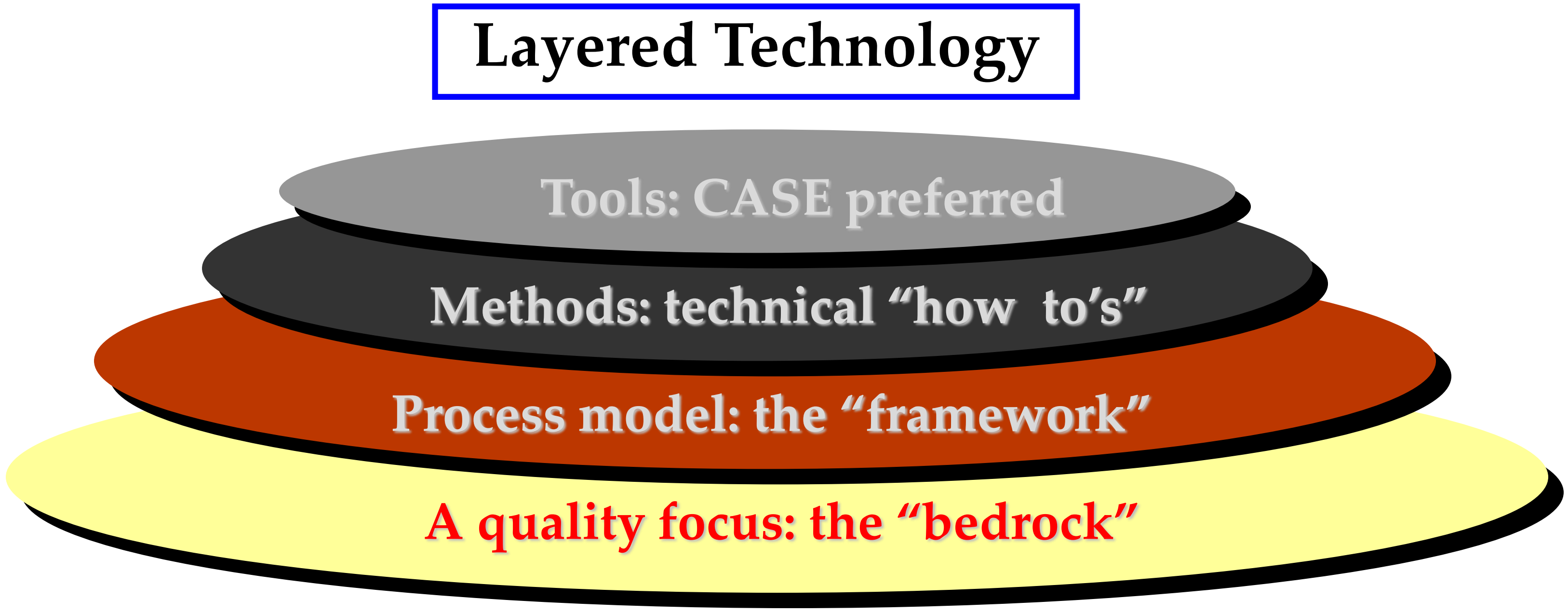
Software engineers should adopt

- Systematic and organized approach to their work
- Use appropriate tools and techniques depending on the problem to be solved
- The development constraints and the resources available

Apply Engineering Concepts to developing Software

Challenge for Software Engineers is to produce high quality software with finite amount of resources & within a predicted schedule

Software Engineering – Layered Technology


Layered Technology
Tools: CASE preferred
Methods: technical "how to's"
Process model: the "framework"
A quality focus: the "bedrock"

Layered Technology

A quality Focus:

- Every organization is rest on its commitment to quality.
- Total quality management, Six Sigma, or similar continuous improvement.
- The bedrock that supports software engineering is a quality focus.

Process:

- It's a foundation layer for software engineering.
- It's define **framework** for a set of *key process areas* (KPA) for effectively manage and deliver quality software in a cost effective manner
- A Key Process Area identifies a cluster of related activities that, when performed together, achieve a set of goals considered important.
- The processes define the tasks to be performed and the order in which they are to be performed

Layered Technology

Methods:

- It provide the technical **how-to's** for building software.
- Methods encompass a broad array of tasks that include requirements analysis, design, program construction, testing, and support.

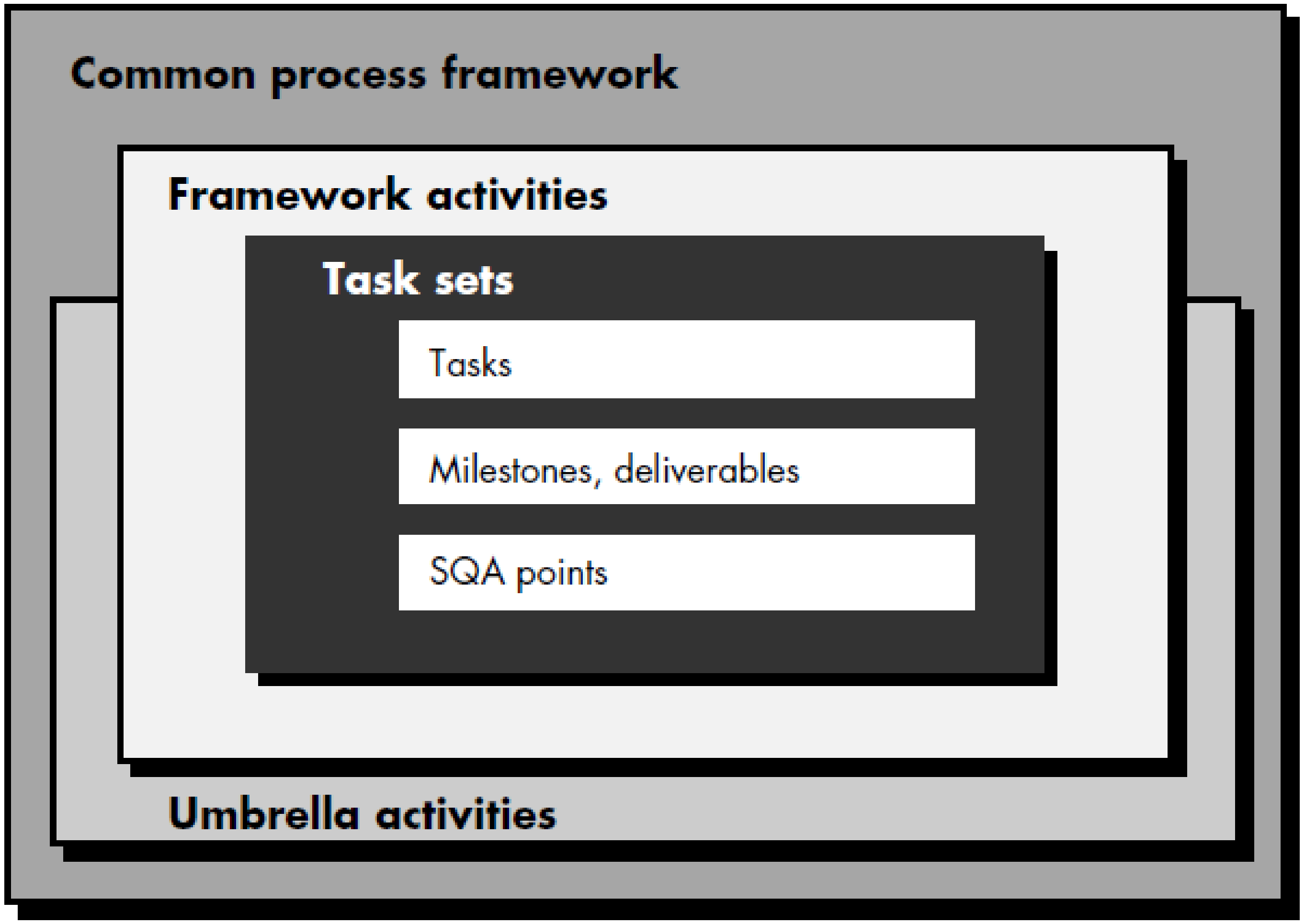
Tools:

- Provide automated or semi-automated support for the process, methods and quality control.
- When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called ***computer-aided software engineering (CASE)***

Phases of software engineering

- The work associated with software engineering can be categorized into three generic phases, regardless of application area, project size, or complexity.
  - **Definition** – focuses on “what”.
  - **Development** – focuses on “how”.
  - **Support** – focuses on “change”.
    - Correction
    - Adaption
    - Enhancement
    - Prevention

Process Framework



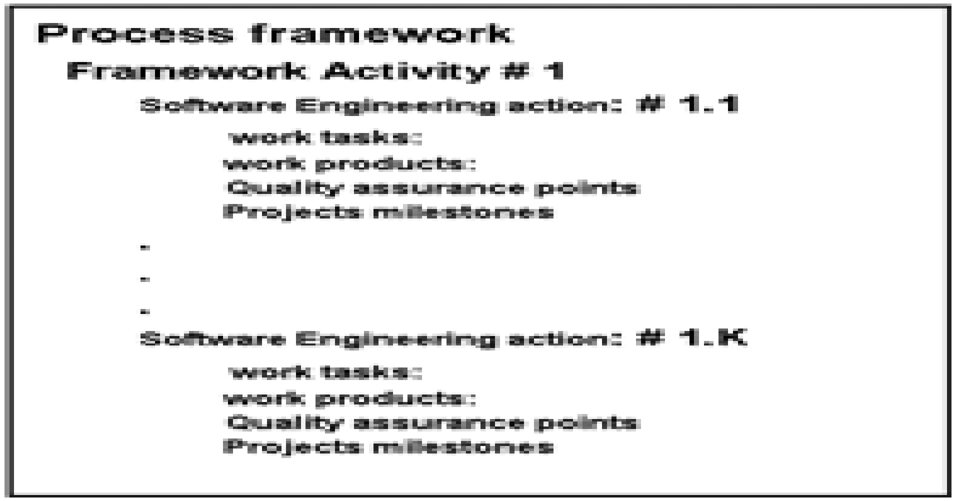
Process framework

Why process :

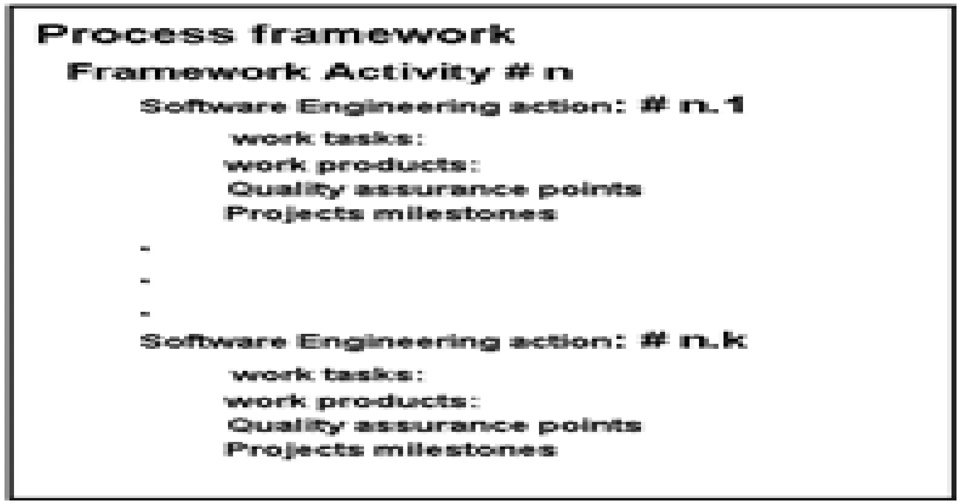
A process defines who is doing what, when and how to reach a certain goal.

- To build complete software process.
- Identified a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.
- It encompasses a set of umbrella activities that are applicable across the entire software process.

Process Framework



Process framework
Framework Activity # 1
Software Engineering action: # 1.1
work tasks:
work products:
Quality assurance points
Projects milestones
.
.
.
Software Engineering action: # 1.K
work tasks:
work products:
Quality assurance points
Projects milestones

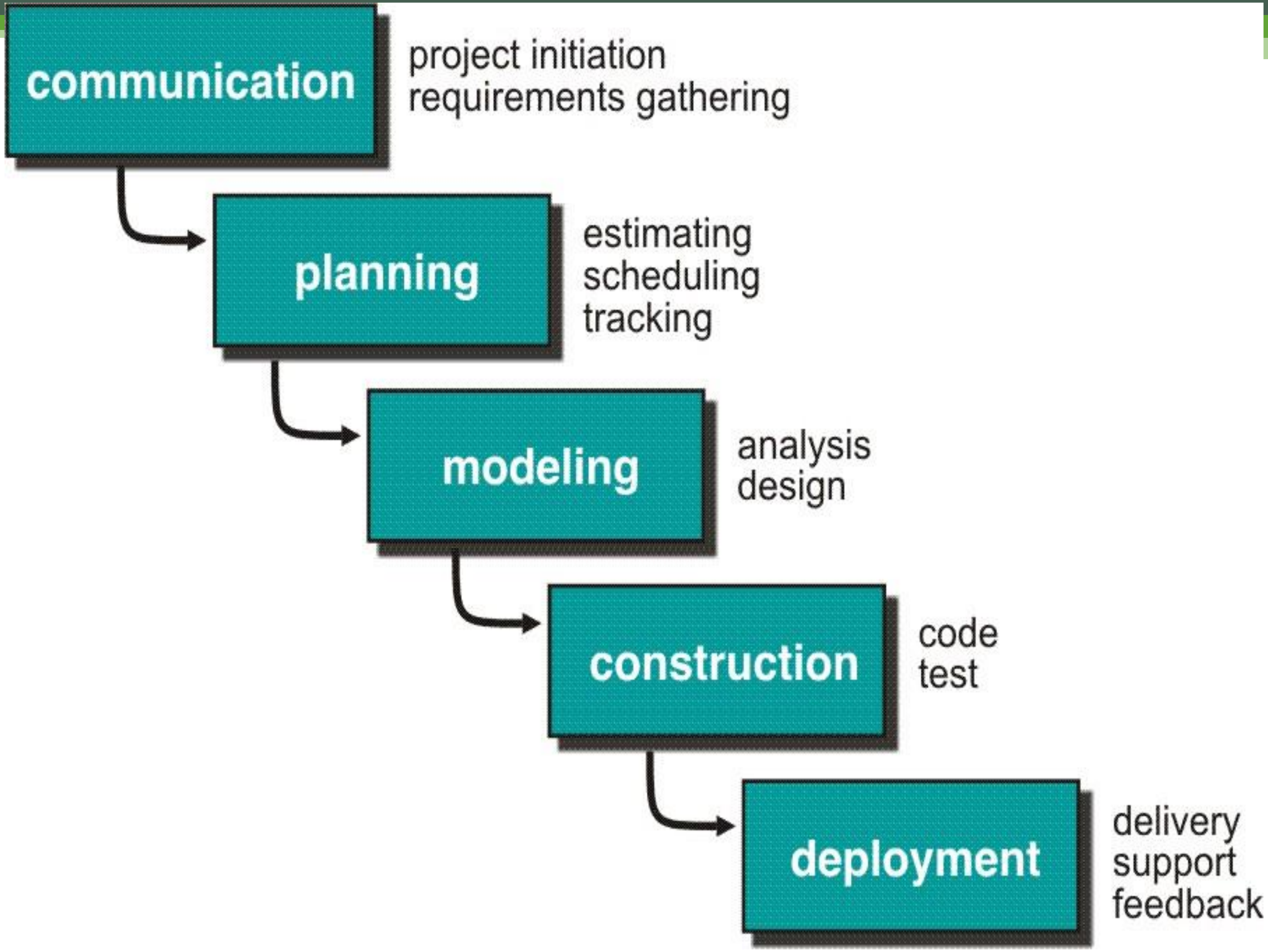


Process framework
Framework Activity # n
Software Engineering action: # n.1
work tasks:
work products:
Quality assurance points
Projects milestones
.
.
.
Software Engineering action: # n.k
work tasks:
work products:
Quality assurance points
Projects milestones

- Each framework activities is populated by a set for *software engineering actions* – a collection of related tasks.
- Each action has individual *work task*.

Generic Process Framework Activities

- Communication:
  - Heavy communication with customers, stakeholders, team
  - Encompasses requirements gathering and related activities
- Planning:
  - Workflow that is to follow
  - Describe technical task, likely risk, resources will require, work products to be produced and a work schedule.
- Modeling:
  - Help developer and customer to understand requirements (Analysis of requirements) & Design of software
- Construction
  - Code generation: either manual or automated or both
  - Testing – to uncover error in the code.
- Deployment:
  - Delivery to the customer for evaluation
  - Customer provide feedback


communication
project initiation
requirements gathering
planning
estimating
scheduling
tracking
modeling
analysis
design
construction
code
test
deployment
delivery
support
feedback

Umbrella Activities

- They occur throughout the process. They focus on project management, tracking and control.
- Software project tracking and control
  - Assessing progress against the project plan.
  - Take adequate action to maintain schedule.
- Formal technical reviews
  - Assessing software work products in an effort to uncover and remove errors before goes into next action or activity.
- Software quality assurance
  - Define and conducts the activities required to ensure software quality.

Umbrella Activities

- Software configuration management
  - Manages the effects of change.
- Risk management
  - Assesses risks that may effect that outcome of project or quality of product
- Document preparation and production
  - Help to create work products such as models, documents, logs, form and list.
- Reusability management
  - Define criteria for work product reuse
  - Mechanisms to achieve reusable components.

Capability Maturity Model (CMM)

- The Software Engineering Institute (SEI) has developed model to measure organization different level of process capability and maturity.
- CMM – developed by SEI
- The CMM defines each process area in terms of “specific goals” and the “specific practices” required to achieve these goals.
- ***Specific goals*** establish the characteristics that must exist if the activities implied by a process area are to be effective.
- ***Specific practices*** refine a goal into a set of process-related activities.
- CMM specifies an increasing series of levels that a software development organization can be at. The higher the level, the better the software development process is.

CMM Level

Level 0 (Incomplete) –

- Process are not perform or not achieve all the goals and objectives defined by the CMMI for Level I capability.

Level 1 (Performed / Initial) – Ad hoc and chaotic, few processes defined and individual efforts made.

Level 2 (Managed / Repeatable) –

- All level 1 criteria have been satisfied
- In addition to Level I;
  - People doing work have access to adequate resources to get job done,
  - Stakeholders are actively involved,
  - Work tasks and products are monitored, controlled, reviewed, and evaluated for conformance to process description.

CMMI Level (cont.)

Level 3 (Defined) –

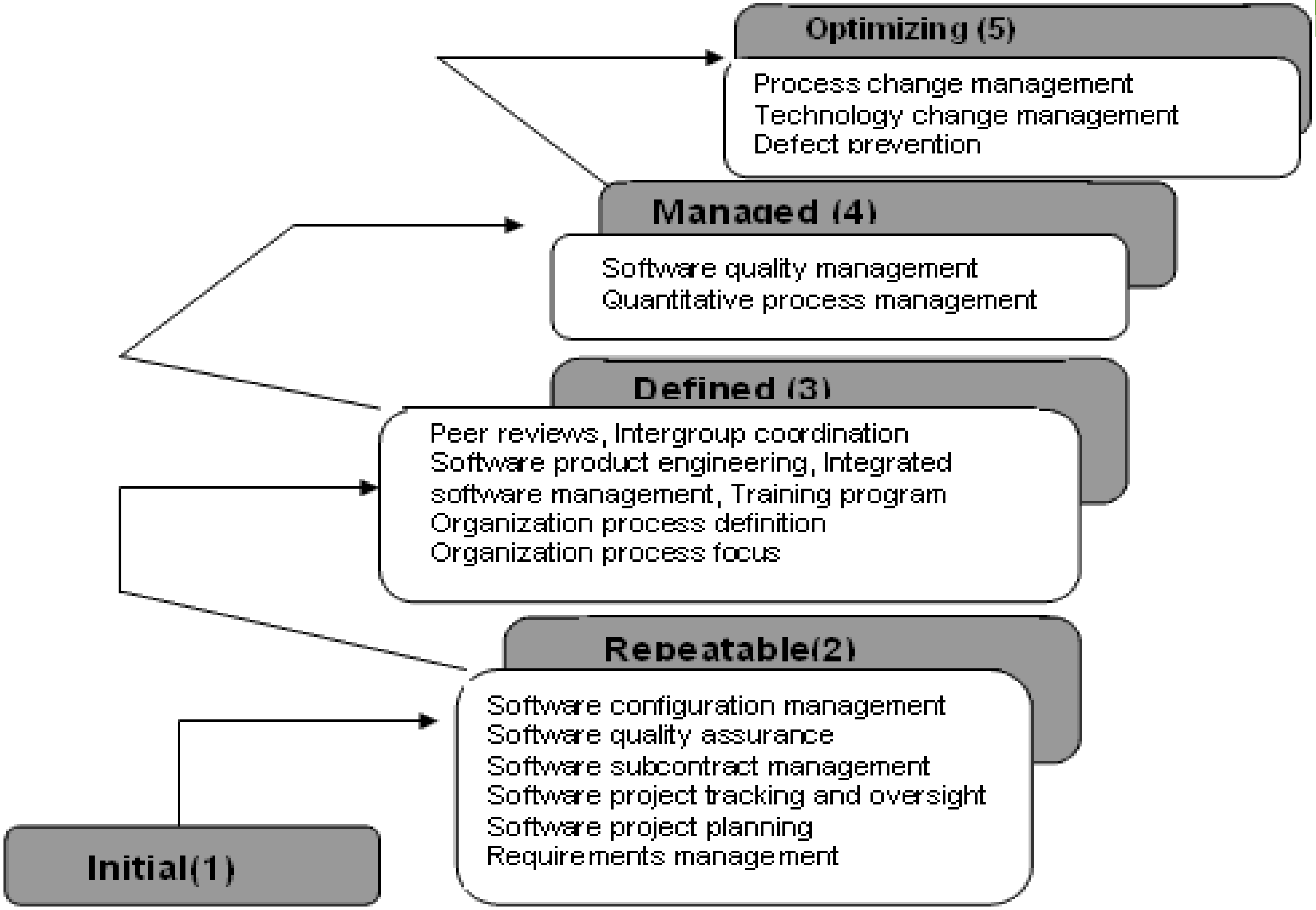
- All level 2 criteria have been achieved.
- In addition;
  - The process is standardized, documented and followed. All the projects use documented and approved version of software process which is useful in developing and supporting software.

Level 4 (Quantitatively Managed) -

- All level 3 criteria have been satisfied.
- Software process and products are quantitatively understood
- Controlled using detailed measures and assessment.

Level 5 (Optimized) –

- Continuous process improvement is enabled by quantitative feedback from the process and testing innovative ideas.


Optimizing (5)
Process change management
Technology change management
Defect prevention
Managed (4)
Software quality management
Quantitative process management
Defined (3)
Peer reviews, Intergroup coordination
Software product engineering, Integrated software management, Training program
Organization process definition
Organization process focus
Repeatable(2)
Software configuration management
Software quality assurance
Software subcontract management
Software project tracking and oversight
Software project planning
Requirements management
Initial(1)

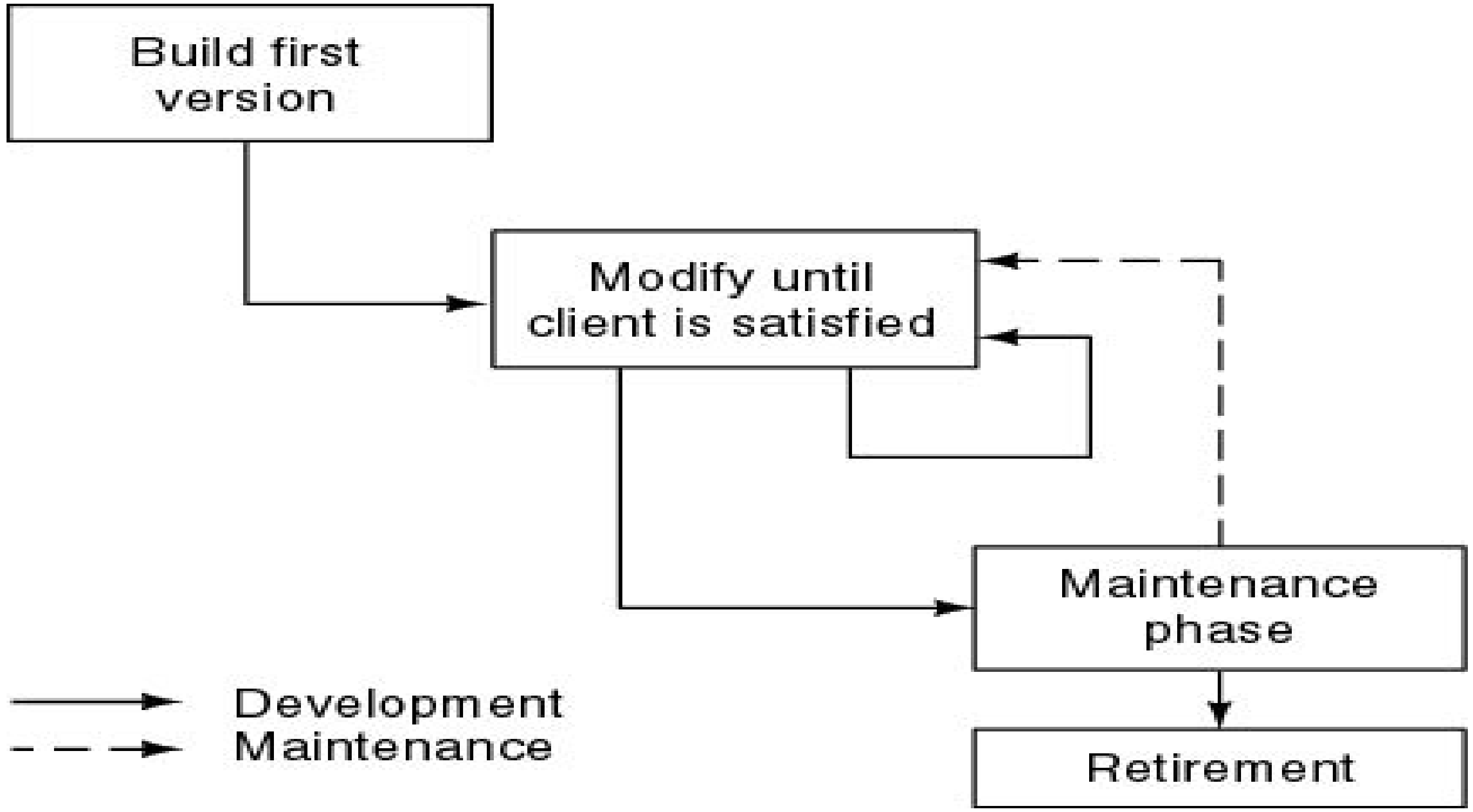
Software process model

- Prescribes a distinct set of activities, actions, tasks, milestones, and work products required to engineer high quality software.
- Software models provide stability, control, and organization to a process that if not managed can easily get out of control
- Software process models are adapted to meet the needs of software engineers and managers for a specific project.
- Definition : it is defined as the abstract representation of a process. The process model can be chosen on the basis of abstract representation of process.
- The software process model is also known as "Software Development Life Cycle (SDLC) Model" or "Software Paradigm".

Why it is required?

- The software development team must decide the process model that is to be used for software product development and then entire team must adhere to it.
- Systematic process.
- Each team member will understand – what is the next activity and how to do it.
- Brings the definiteness and discipline in overall development process.
- If the process model is not followed for software development, then any team member can perform any software development activity, this will ultimately cause chaos and software project will definitely fail.

Build and Fix Model

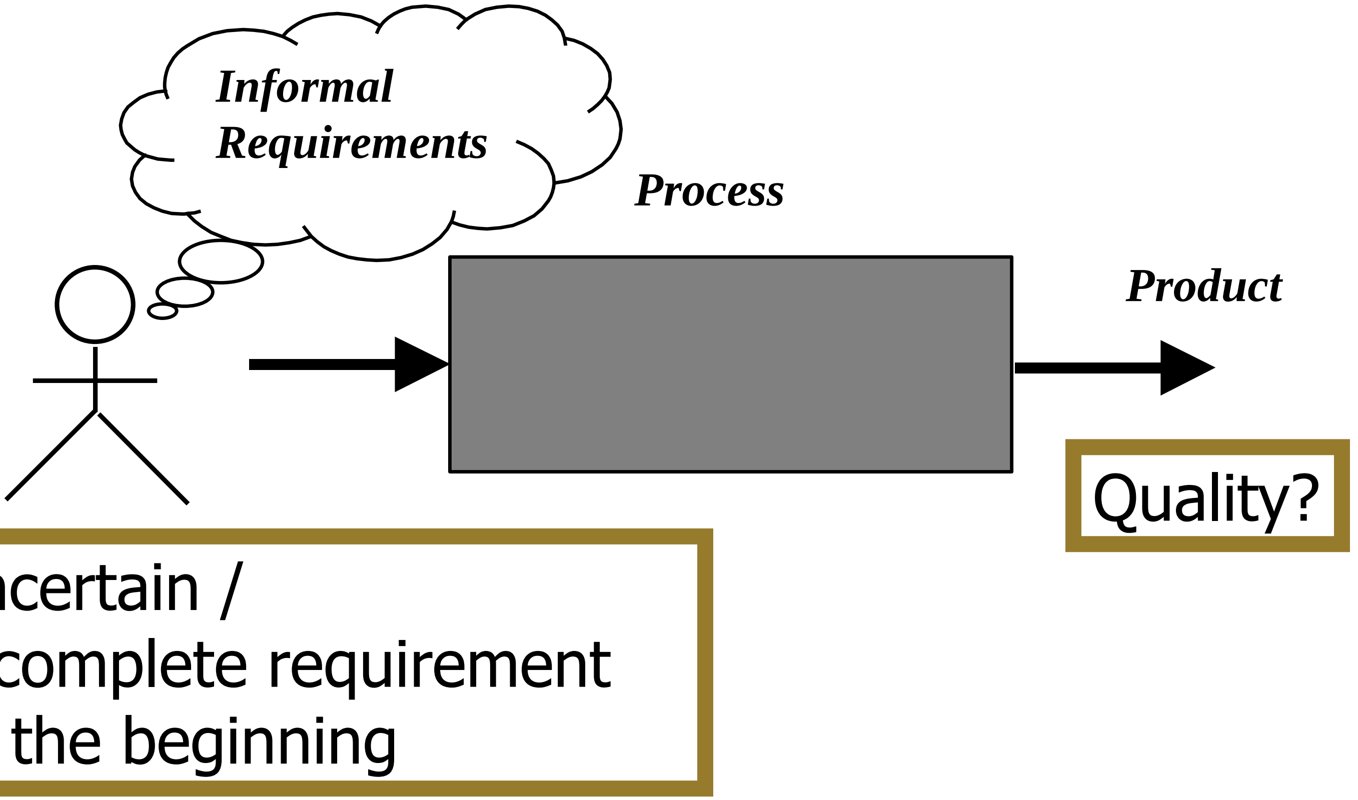


Build and Fix Model

The earlier approach

- Product is constructed without specification or any attempt at design.
- Developers simply build a product that is reworked as many times as necessary to satisfy the client.
- Model may work for small projects but is totally unsatisfactory for products of any reasonable size.
- Maintenance is high.
- Source of difficulties and deficiencies
  - impossible to predict
  - impossible to manage

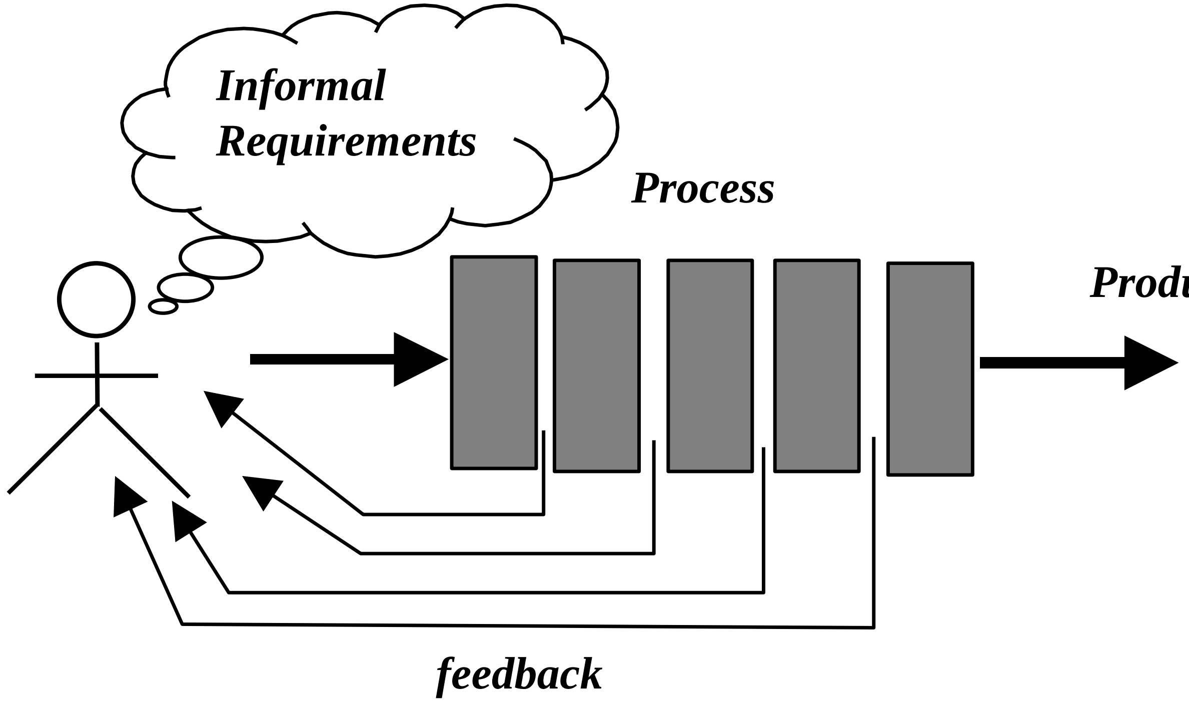
Process as a "black box"



Problems

- The assumption is that requirements can be fully understood prior to development
- Interaction with the customer occurs only at the beginning (requirements) and end (after delivery)
- Unfortunately the assumption almost never holds

Process as a "white box"



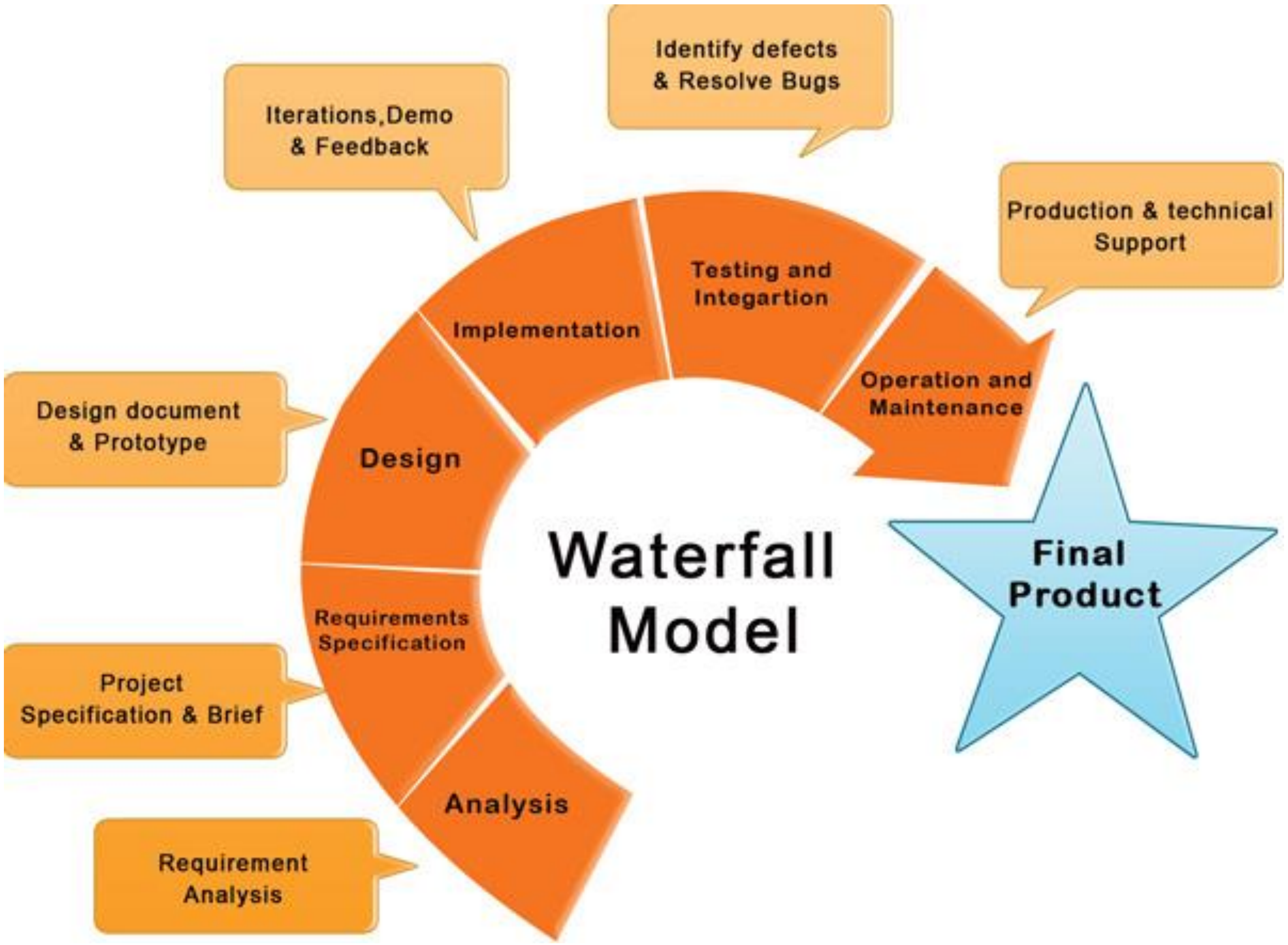
Advantages

- Reduce risks by improving visibility
- Allow project changes as the project progresses
  - Based on feedback from the customer

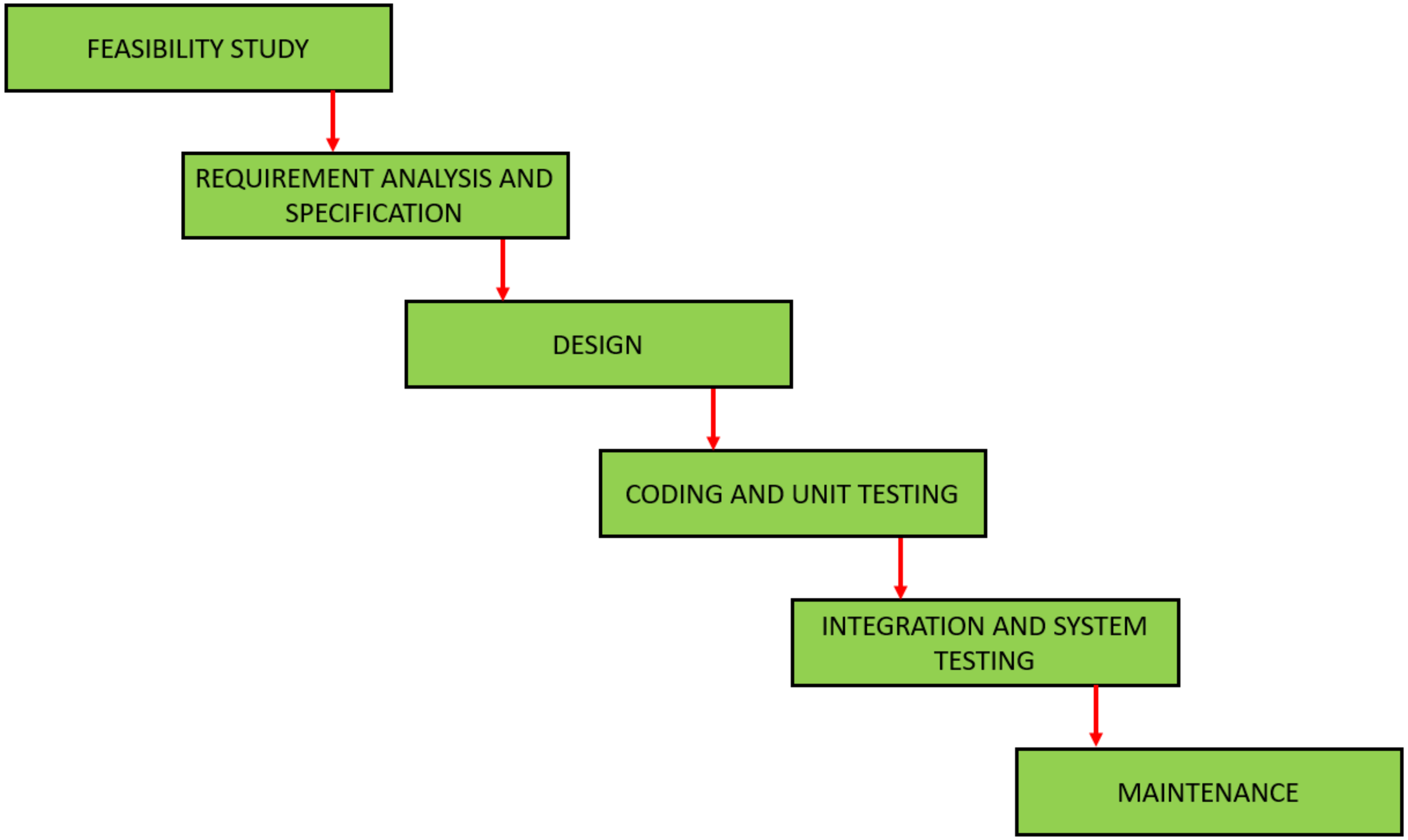
Waterfall model

- Waterfall model is also called as “linear sequential model” or “classical life cycle model”.
- It is oldest software paradigm. This model suggests a systematic, sequential approach to software development.
- The software development starts with requirements gathering phase. Then, progresses through analysis, design, coding, testing and maintenance.

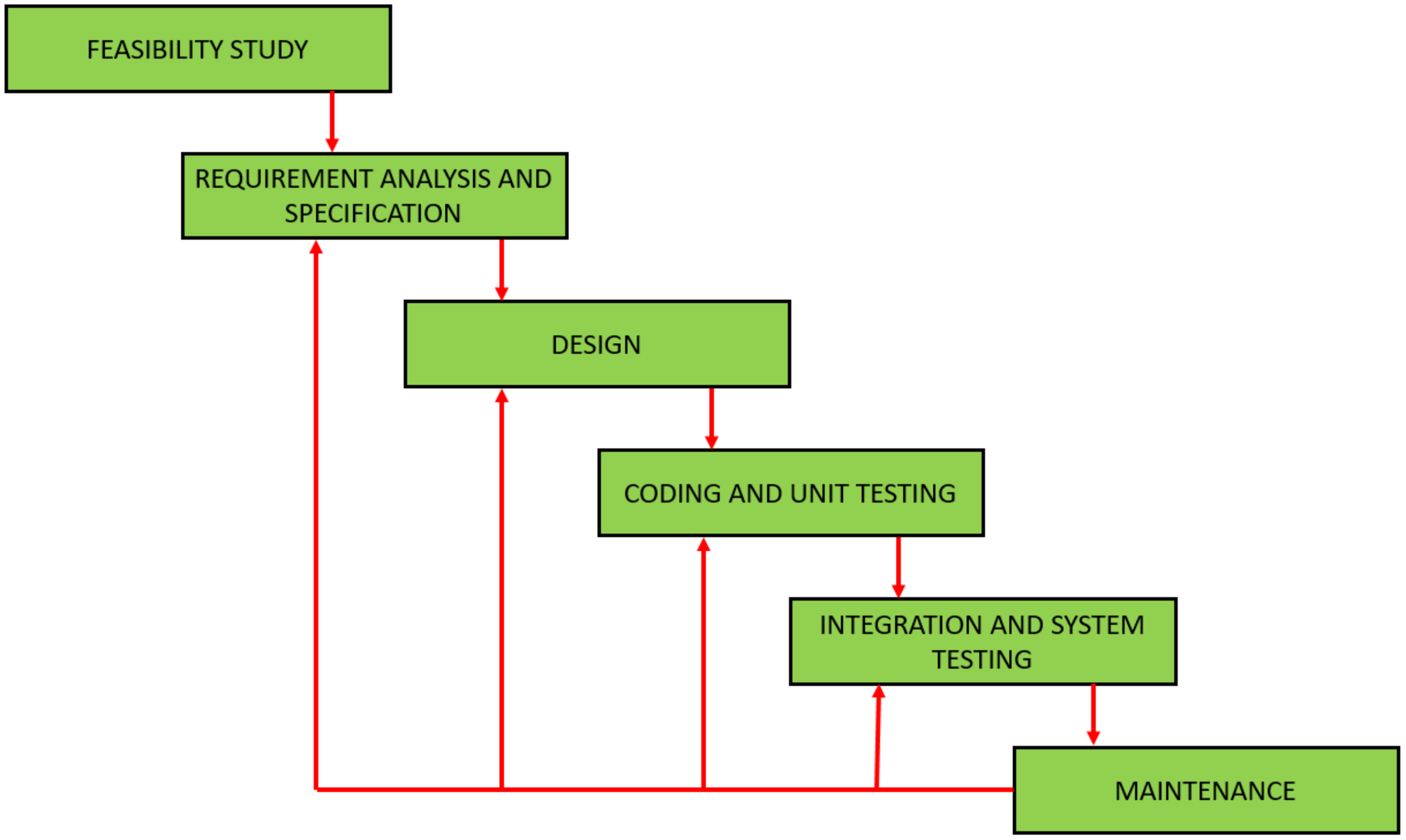
Waterfall model


Identify defects & Resolve Bugs
Iterations,Demo & Feedback
Production & technical Support
Testing and Integartion
Implementation
Operation and Maintenance
Design document & Prototype
Design
Waterfall Model
Final Product
Requirements Specification
Project Specification & Brief
Analysis
Requirement Analysis

Classical Waterfall Model


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

Iterative Waterfall Model


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

Waterfall Model

- Requirement Gathering and Analysis
  - Basic requirements of the system must be understood by software engineer, who is also called Analyst.
  - The information domain, function, behavioral requirements of the system are understood.
  - All these requirements are well documents.
- Design
  - It is the intermediate step between requirements analysis and coding.
  - Focusses on data structure, software architecture, interface representation, algorithmic details.
- Coding
  - Design is translated into machine-readable form. If design is done in sufficient detail, then coding can be done effectively. Programs are created in this phase.

Waterfall Model

- Testing
  - Begins when coding is done. While performing testing, the major focus is on logical internals of the software. The testing ensures execution of all the paths, functional behaviors.
  - The purpose of testing is to uncover errors, fix the bugs and meet the customer requirements.
- Maintenance
  - It is the longest life cycle phase. When the system is installed and put in practical use, then error may get introduced, correcting such errors and putting it in use is the major purpose of maintenance activity.
  - Similarly, enhancing system’s services as new requirements are discovered is again maintenance.

Advantages of Waterfall Model

- Waterfall model is simple to implement.
- For implementation of small systems waterfall model is useful.

Limitations of the waterfall model

- The nature of the requirements will not change very much During development; during evolution
- The model implies that you should attempt to complete a given stage before moving on to the next stage
  - Does not account for the fact that requirements constantly change.
  - It also means that customers can not use anything until the entire system is complete.
- The model implies that once the product is finished, everything else is maintenance.
- Surprises at the end are very expensive
- Some teams sit ideal for other teams to finish
- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.

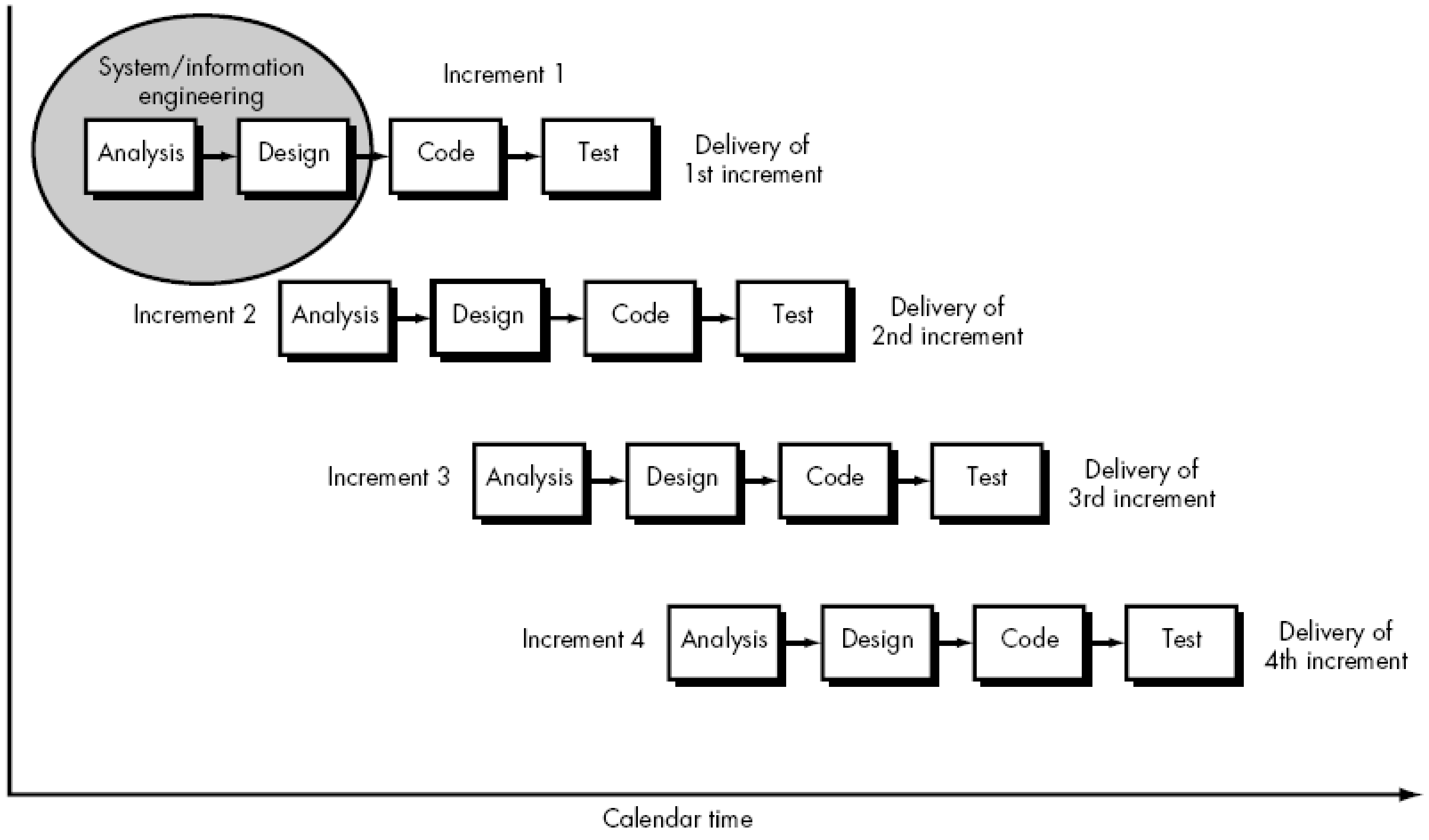
Problems:

1. Real projects rarely follow the sequential model.
2. Difficult for the customer to state all the requirement explicitly.
3. Assumes patience from customer - working version of program will not available until programs not getting change fully.

Incremental model

- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
- The incremental model has same phases that are in waterfall model. But iterative in nature. The incremental model has following phases:
  - Analysis, Design, Code and Test
- It delivers series of releases to the customer. These releases are known as increments. More and more functionality is associated with each increment.
- The first increment is known as core product.
  - In this release, the basic requirements are implemented and then in subsequent increments, new requirements would be added.

Incremental model



The Incremental Model

- User requirements are prioritised and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.
- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

When to choose it?

- This model can be used when the requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- A new technology is being used.

Advantages of incremental model

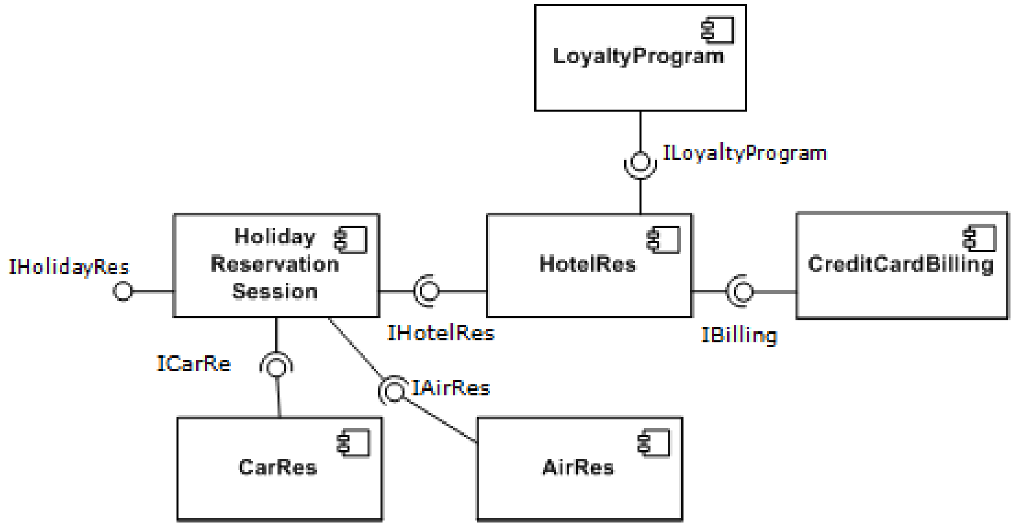
- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during its iteration.

Disadvantages of incremental model

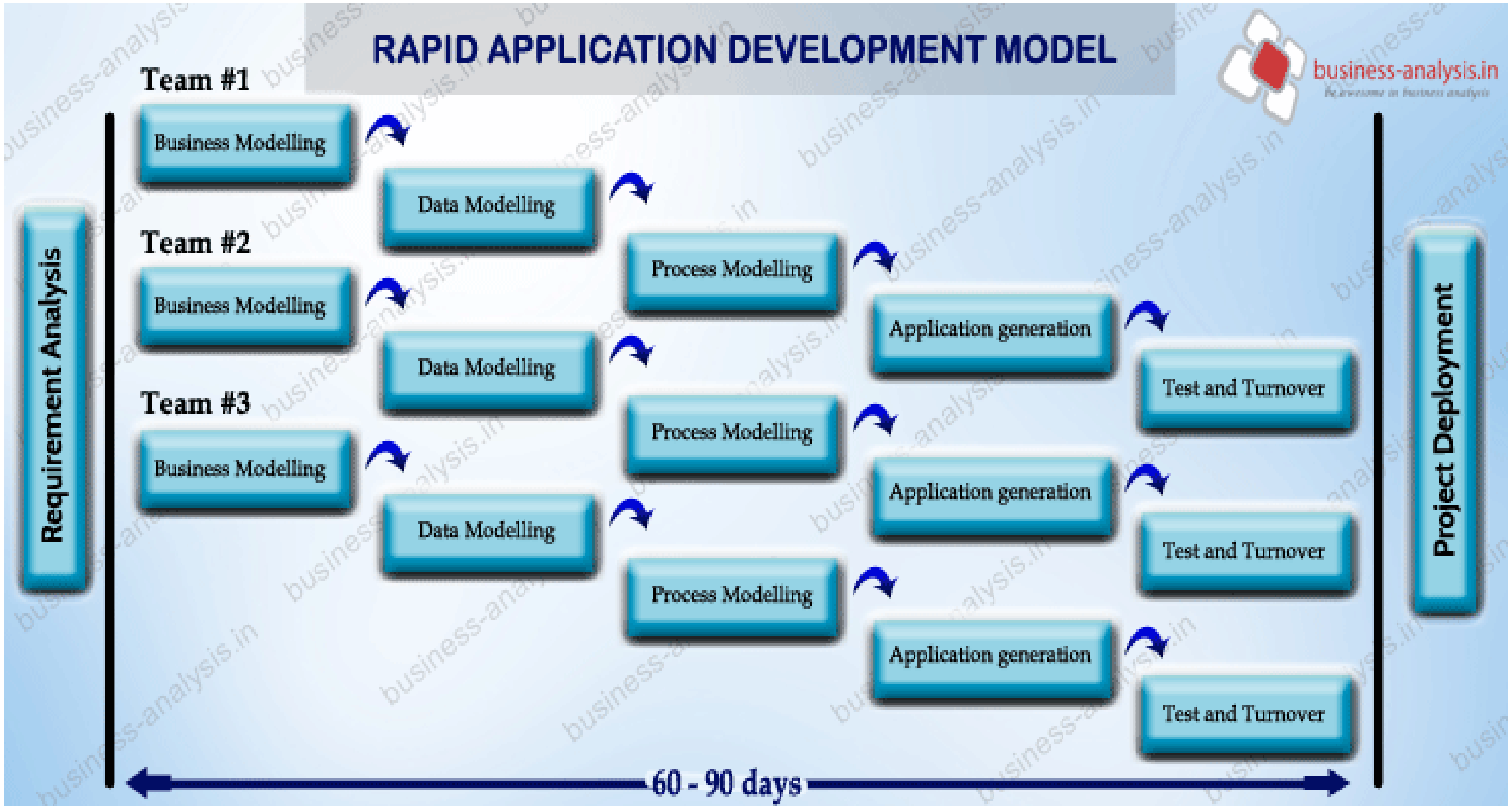
- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- As additional functionality is added to the product, problems may arise related to system architecture which were not evident in earlier prototypes.

RAD (rapid application development) model

- An incremental software process model that focuses on short development cycle time.
- This model is a “high-speed” model which adapts many steps from waterfall model in which rapid development is achieved by using component based construction approach.
- Heavy use of reusable components.
- Requirements are well understood and project scope is constrained, RAD enables a development team to create a “fully functional system” in a short span of 60 to 90 days.



RAD model


RAPID APPLICATION DEVELOPMENT MODEL
business-analysis.in
Team #1
Business Modelling
Data Modelling
Process Modelling
Application generation
Test and Turnover
Team #2
Business Modelling
Data Modelling
Process Modelling
Application generation
Test and Turnover
Team #3
Business Modelling
Data Modelling
Process Modelling
Application generation
Test and Turnover
Requirement Analysis
Project Deployment
60 - 90 days

RAD model

- **Communication** – to understand business problem.
- **Planning** – multiple s/w teams works in parallel on diff. system.
- **Modeling** –
  - **Business modeling** – Information flow among business is working. What kind of information drives? Who is going to generate information? From where information comes and goes?
  - **Data modeling** – Information refine into set of data objects that are needed to support business.
  - **Process modeling** – Data object transforms to information flow necessary to implement business.

RAD model

- **Application Generation** – Highlighting the use of pre-existing software component. Reuse existing program component or create reusable components. Use automated tools to generate the code.
- **Testing and Turnover** – Since the RAD process emphasizes reuse, many of the program components have already been tested. This reduces overall testing time. However, new components must be tested and all interfaces must be fully exercised.

When to use?

- For high budget projects for which automated code generation cost can be achieved.
- When high skilled resources are available.
- When it’s easy to modularize the project.
- If technical risks are low.
- If development needed to complete in specified time.
- RAD Model is suitable if the functionality have less dependencies on other functionality

Advantages of RAD model

- Fast application development and delivery.
- Increases reusability of components.
- Review by the client from the very beginning of development so very less chance to miss the requirements.
- Very flexible if any changes required.
- Good for small projects.

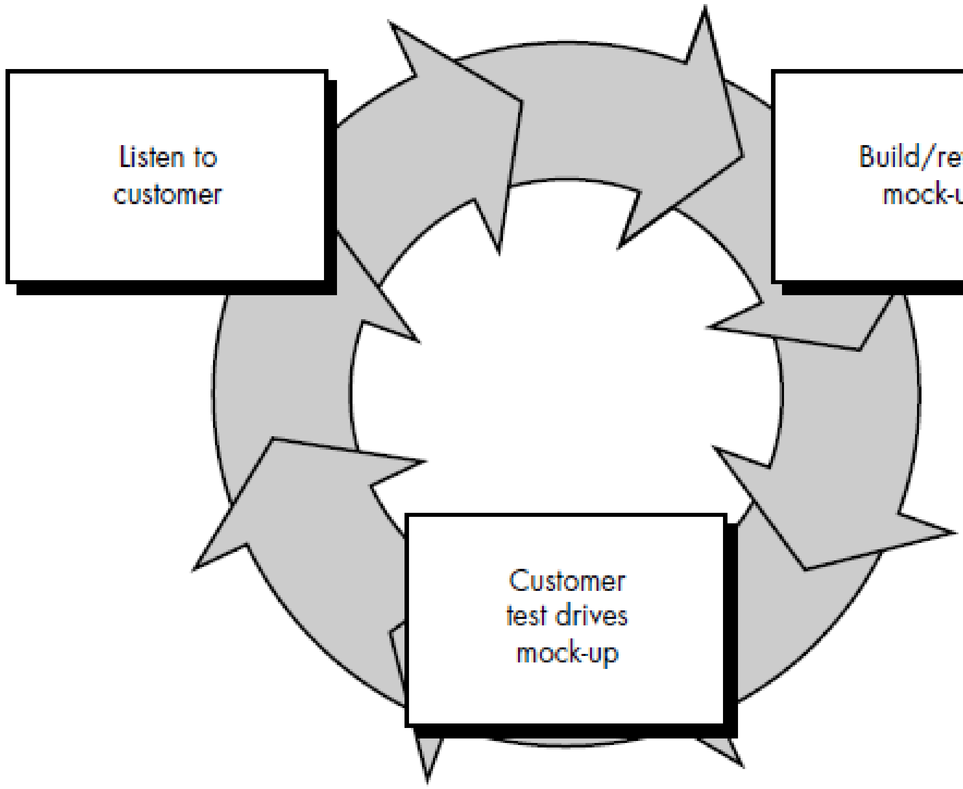
Disadvantages of RAD model

- RAD requires sufficient human resources to create the right number of RAD teams
- If developers and customers are not committed to the rapid , rapid-fire activities necessary to complete the system in a much abbreviated time frame, RAD projects will fail.
- Requires highly skilled developers / designers.
- For RAD system should be properly modularize otherwise it creates lots of problems to the RAD model.
- When technical risks are high, RAD may not be a suitable option.
- Automated code generation is costly.

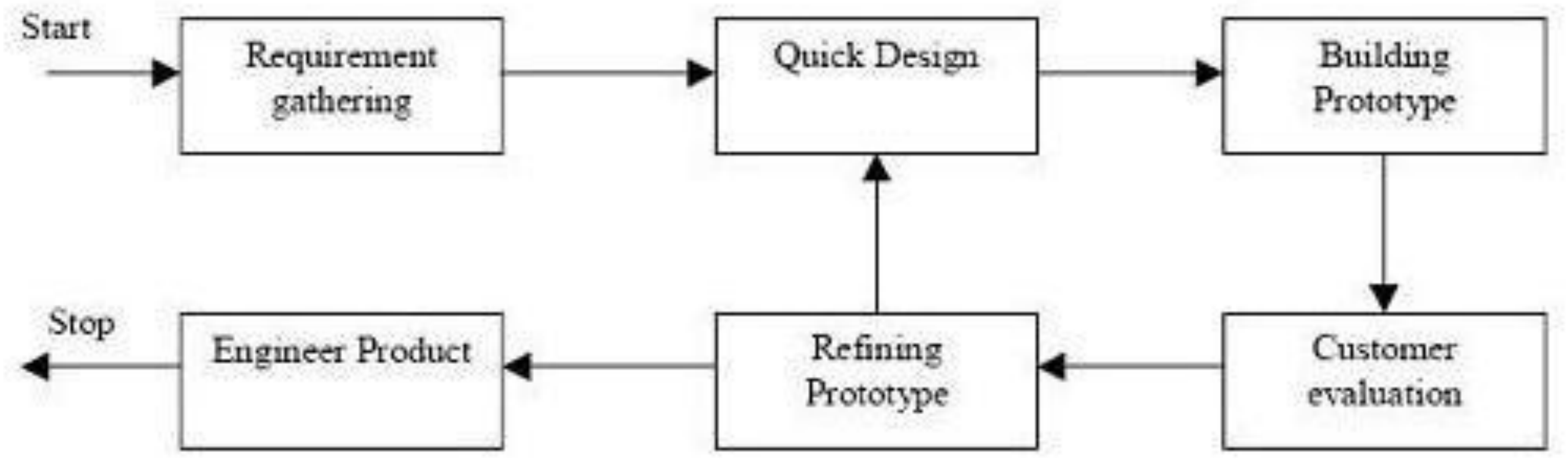
Software prototyping

- Prototype is a working model of software with some limited functionality.
- The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.
- Prototyping is used to allow the users evaluate developer proposals and try them out before implementation.
- It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

Prototype model

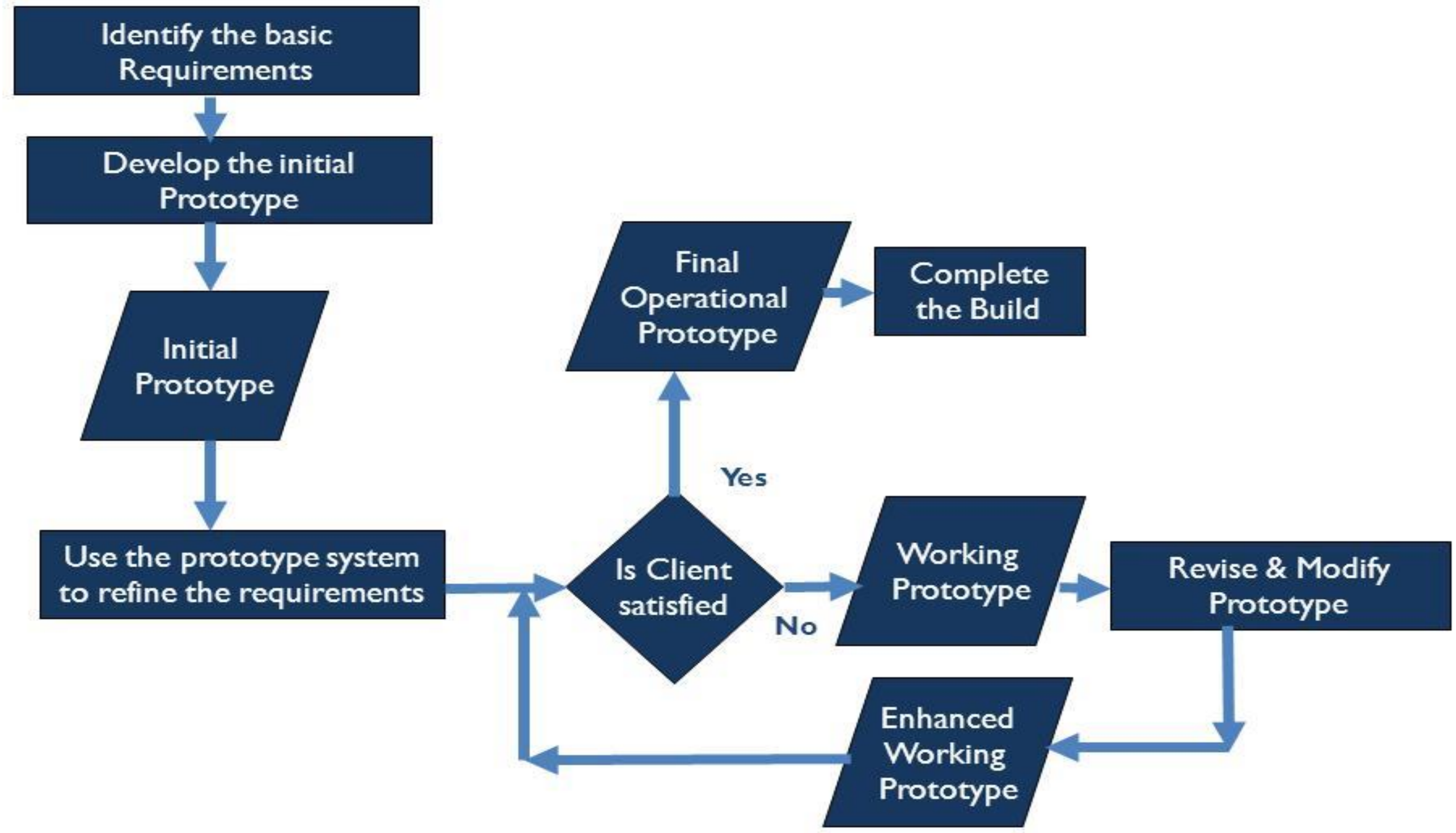


Prototype model



Prototyping Model

Prototype model


Identify the basic Requirements
Develop the initial Prototype
Initial Prototype
Use the prototype system to refine the requirements
Is Client satisfied
Yes
No
Final Operational Prototype
Complete the Build
Working Prototype
Revise & Modify Prototype
Enhanced Working Prototype

When to use?

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model.
- It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system.

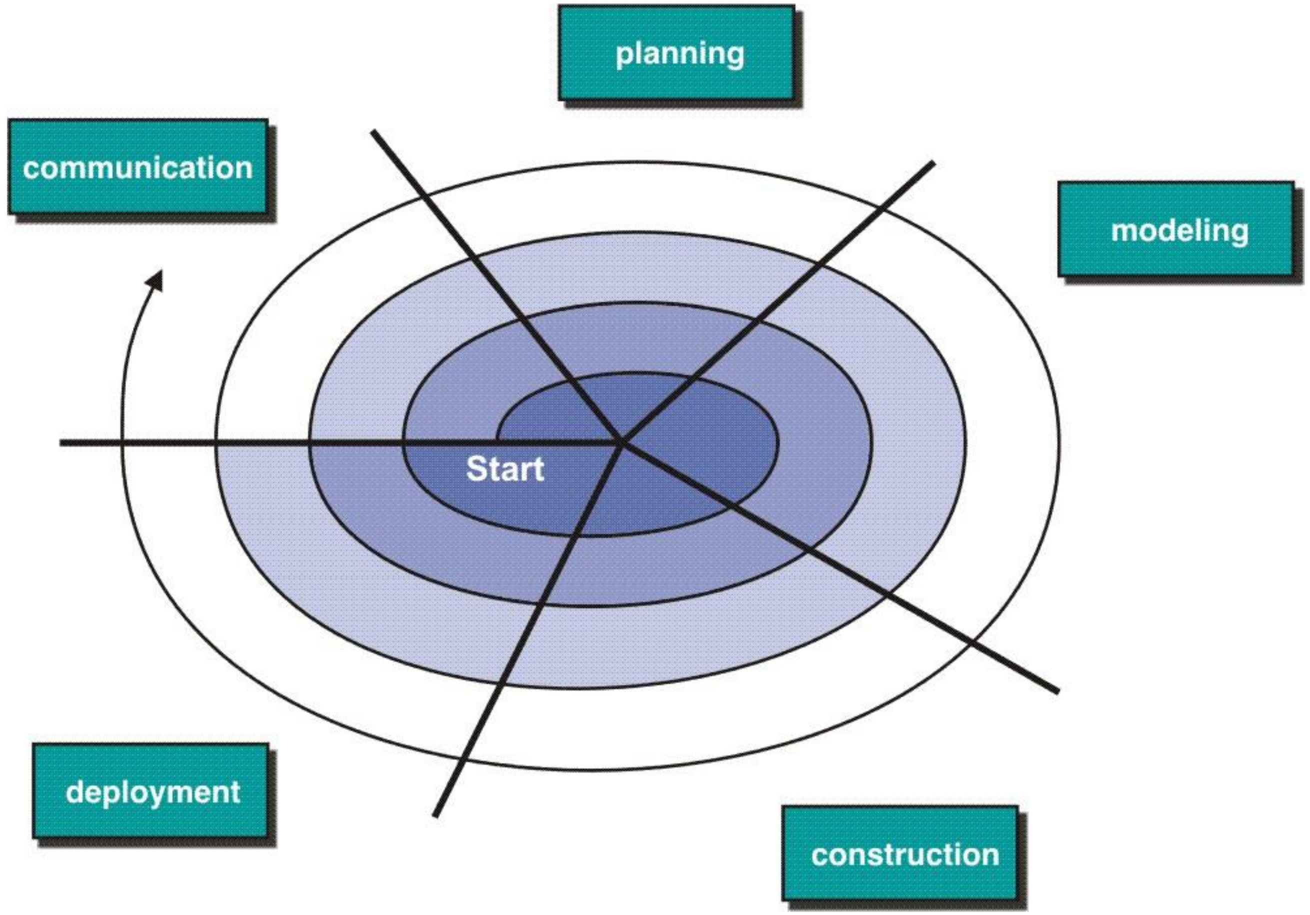
Advantages of prototype model

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily

Disadvantages of prototype model

- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- The effort invested in building prototypes may be too much if not monitored properly.
- Users may get confused in the prototypes and actual systems.
- Developers may try to reuse the existing prototypes to build the actual system, even when its not technically feasible.

Evolutionary Model: Spiral Model



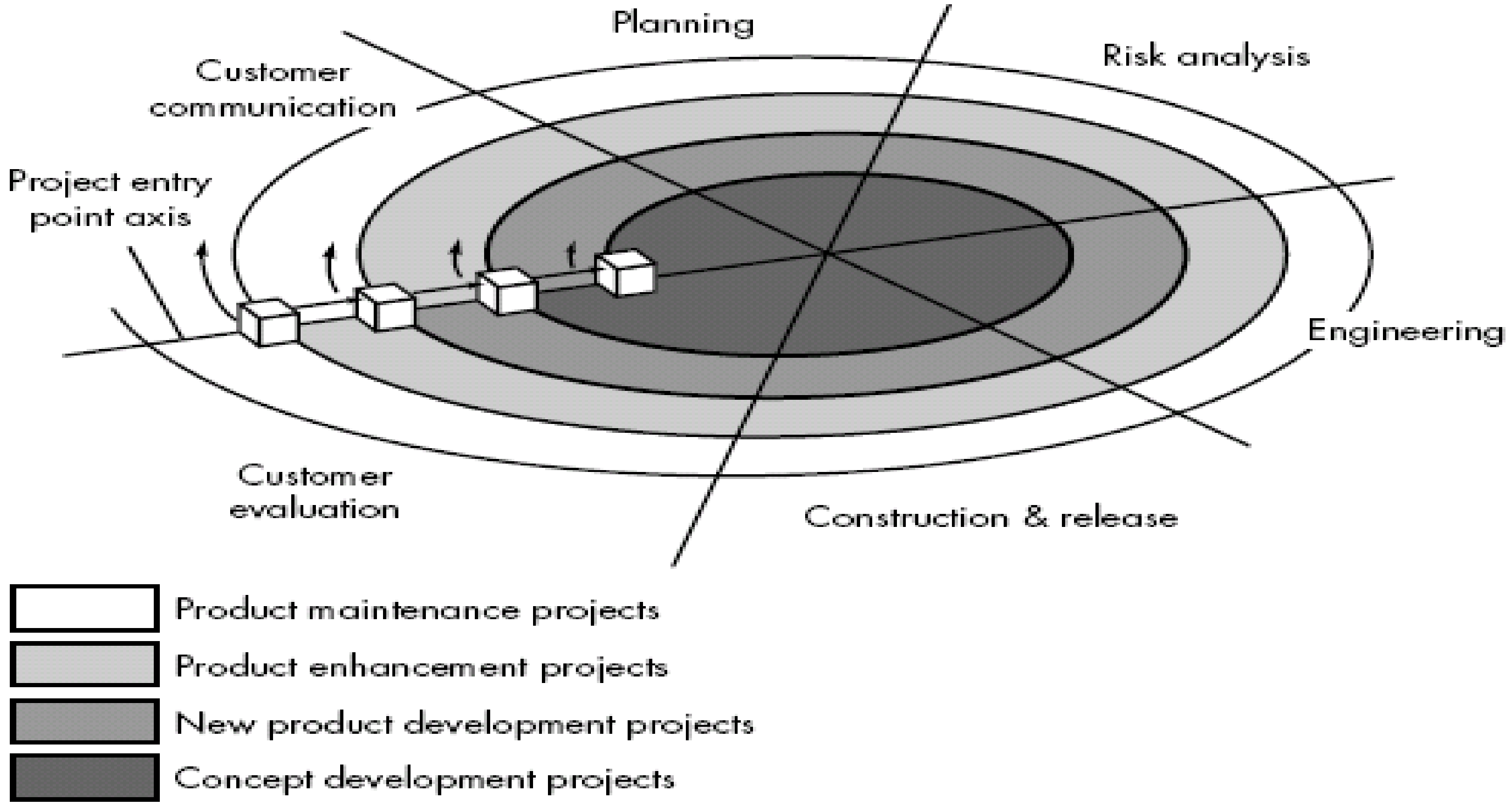
Spiral Model

- Couples iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model **with more emphasis on risk analysis.**
- It provides potential for rapid development of increasingly more complete version of the software.
- Using spiral, software developed in as series of evolutionary release.
  - Early iteration, release might be on paper or prototype.
  - Later iteration, more complete version of software.

Spiral model

- Divided into framework activities. Each activity represent one segment.
- Evolutionary process begins in a clockwise direction, beginning at the center risk.
- First circuit around the spiral might result in development of a product specification. Subsequently, develop a prototype and then progressively more sophisticated version of software, unlike other process models that end when software is delivered.
- It can be adapted to apply throughout the life of software.

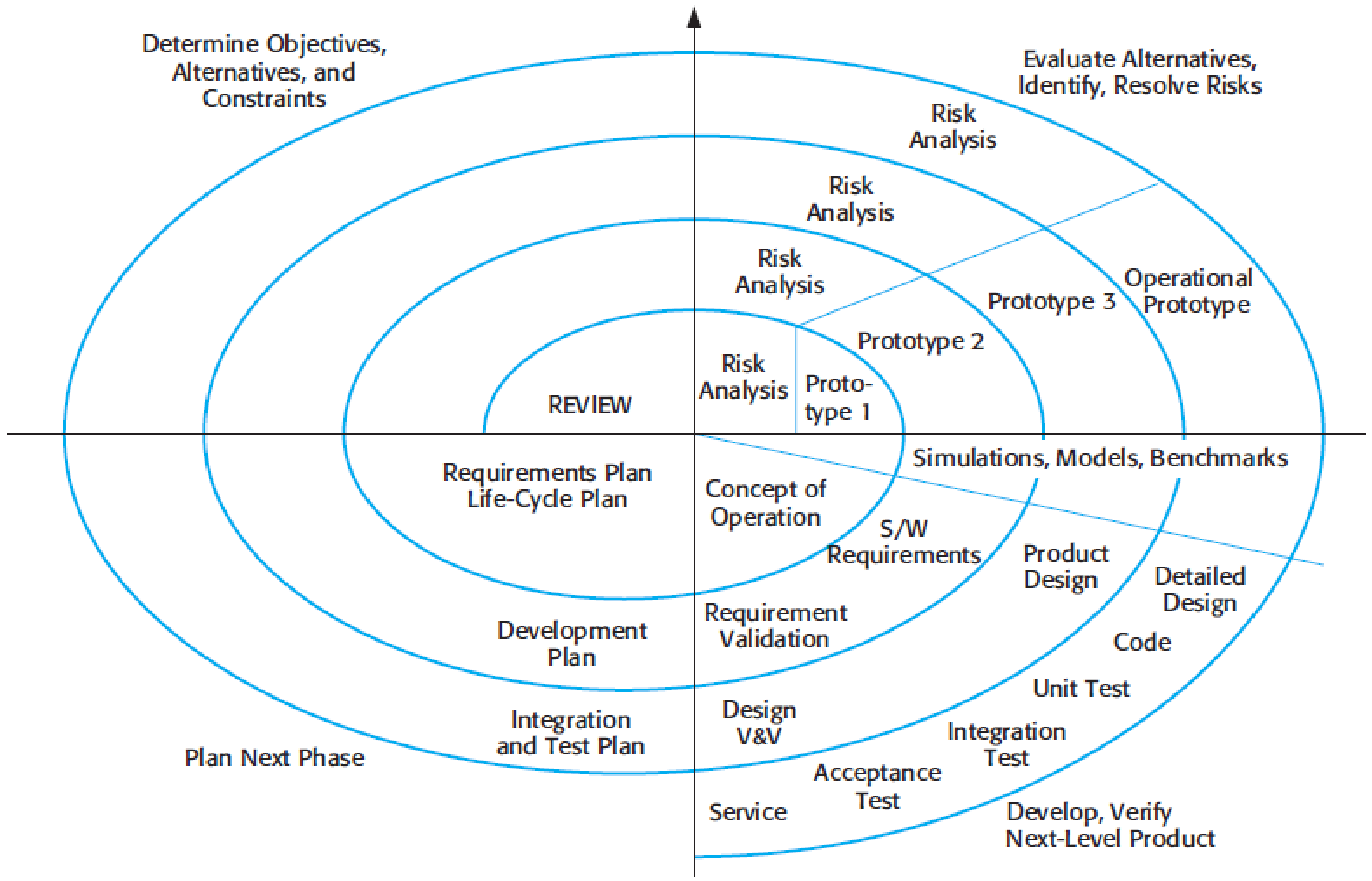
Spiral Model (cont.)



Spiral model

- Spiral models uses prototyping as a risk reduction mechanism but, more important, enables the developer to apply the prototyping approach at each stage in the evolution of the product.
- It maintains the systematic stepwise approach suggested by the classic life cycle but also incorporates it into an iterative framework activity.
- If risks cannot be resolved, project is immediately terminated

Spiral model



Spiral Model (cont.)

Concept Development Project:

- Start at the core and continues for multiple iterations until it is complete.
- If concept is developed into an actual product, the process proceeds outward on the spiral.

New Product Development Project:

- New product will evolve through a number of iterations around the spiral.
- Later, a circuit around spiral might be used to represent a “Product Enhancement Project”

Product Enhancement Project:

- There are times when process is dormant or software team not developing new things but change is initiated, process start at appropriate entry point.

When to use?

- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

Advantages of Spiral model

- Changing requirements can be accommodated.
- Allows for extensive use of prototypes
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.

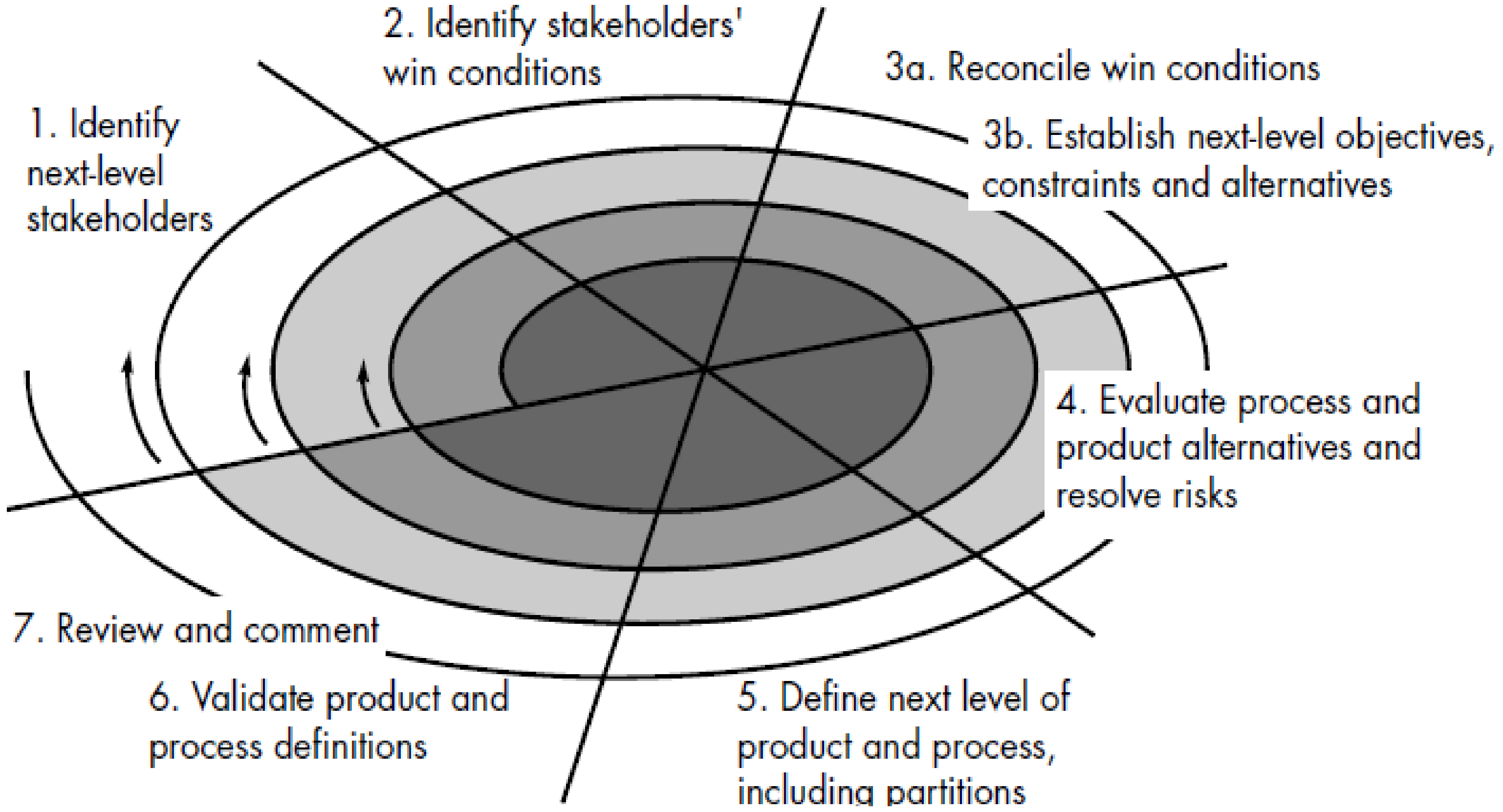
Disadvantages of Spiral model

- Management is more complex.
- End of project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go indefinitely.
- Large number of intermediate stages requires excessive documentation.

Winwin spiral model

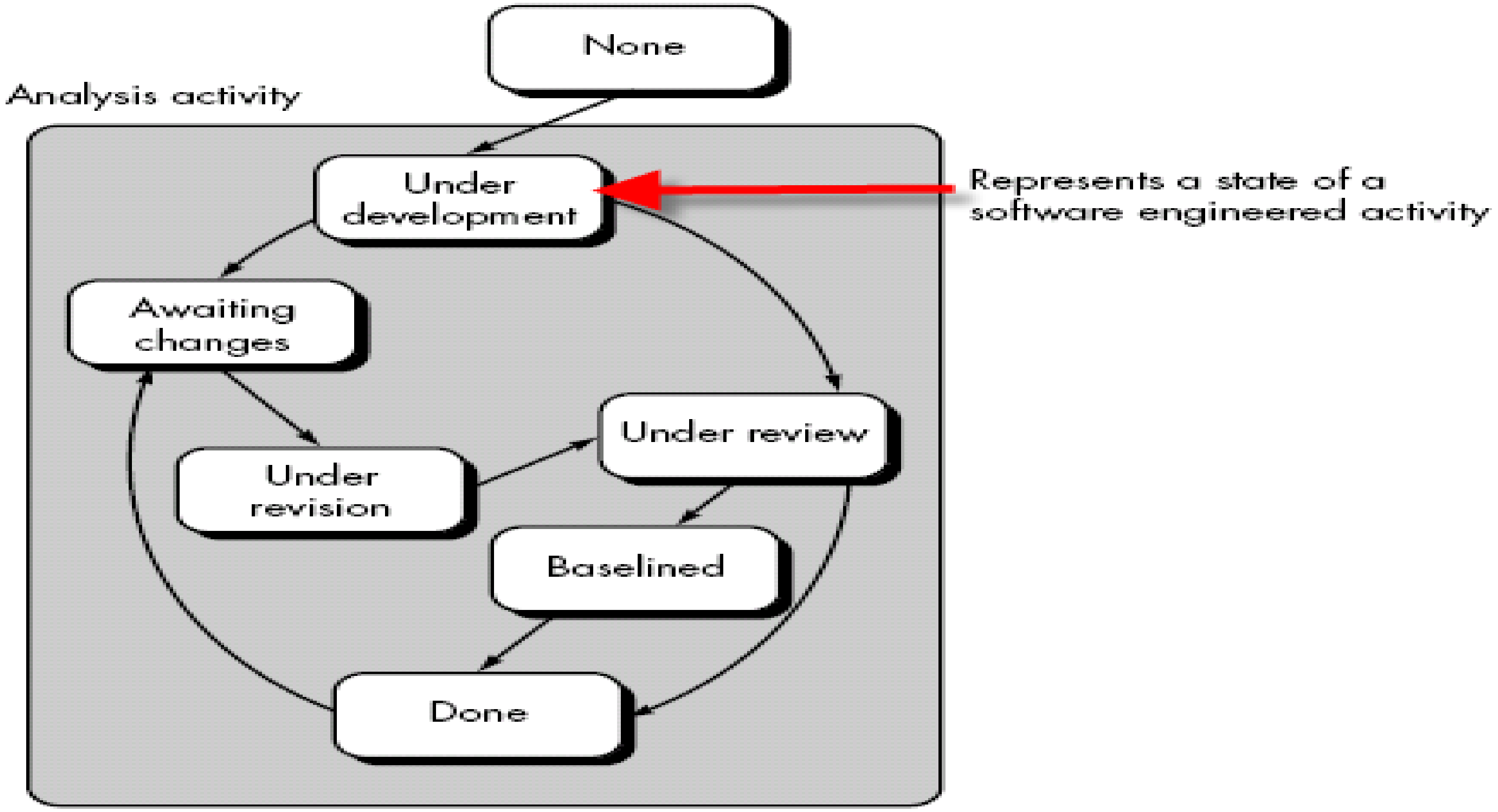
- The objective of this activity is to elicit project requirements from the customer.
- In an ideal context, the developer simply asks the customer what is required and the customer provides sufficient detail to proceed.
- Unfortunately, this rarely happens.
- In reality, the customer and the developer enter into a process of negotiation, where the customer may be asked to balance functionality, performance, and other product or system characteristics against cost and time to market.
- The best negotiations strive for a “win-win” result.

Winwin spiral model


2. Identify stakeholders' win conditions
3a. Reconcile win conditions
1. Identify next-level stakeholders
3b. Establish next-level objectives, constraints and alternatives
4. Evaluate process and product alternatives and resolve risks
7. Review and comment
6. Validate product and process definitions
5. Define next level of product and process, including partitions

The customer wins by getting the system or product that satisfies the majority of the customer’s needs and the developer wins by working to realistic and achievable budgets and deadlines.

Concurrent Development Model



Concurrent Development Model

- It represented schematically as series of major technical activities, tasks, and their associated states.
- It is often more appropriate for system engineering projects where different engineering teams are involved.
- The activity-modeling may be in any one of the states for a given time.
- All activities exist concurrently but reside in different states.

Concurrent Development Model

E.g.

- The *analysis* activity (existed in the **none** state while initial customer communication was completed) now makes a transition into the **under development** state.
- *Analysis* activity moves from the **under development** state into the **awaiting changes** state only if customer indicates changes in requirements.
- Series of event will trigger transition from state to state.

E.g. During initial stage there was inconsistency in design which was uncovered. This will triggers the analysis action from the **Done** state into **Awaiting Changes** state.

Concurrent Development Model

- Visibility of current state of project
- It defines network of activities
- Each activities, actions and tasks on the network exists simultaneously with other activities ,actions and tasks.
- Events generated at one point in the process network trigger transitions among the states.

Advantages of Concurrent Development Model

- Focus on concurrent engineering activities in a software engineering process such as prototyping, analysis modeling, requirements specification and design.
- Represented schematically as a series of major technical activities, tasks and their associated states.

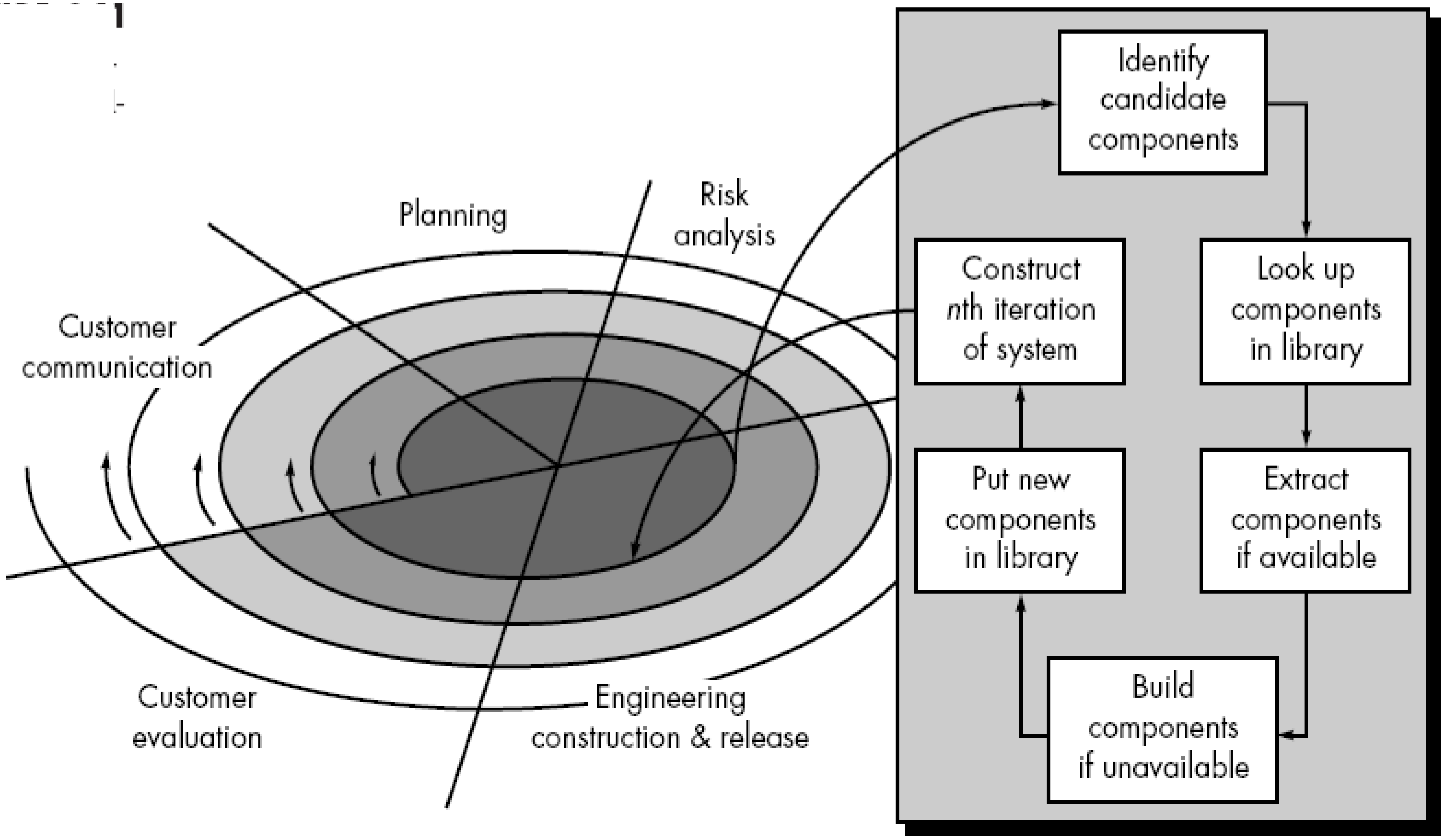
Disadvantages of Concurrent Development Model

- Because all of the stages are working simultaneously, any changes to user requirements halts the work on any component dependent on the component being changed. This can lead to a much longer development cycle than originally planned for.
- Additionally, this model requires excellent and current communication between all teams, which is sometimes not easily feasible.

Component Based Development

- Component-based development (CBD) model incorporates many of the characteristics of the spiral model.
- It is evolutionary by nature and iterative approach to create software.
- CBD model creates applications from prepackaged software components (called *classes).*

Component Based Development


Planning
Risk analysis
Customer communication
Customer evaluation
Engineering construction & release
Identify candidate components
Construct *n*th iteration of system
Look up components in library
Put new components in library
Extract components if available
Build components if unavailable

Component Based Development

- Modeling and construction activities begin with identification of candidate components.
- Classes created in past software engineering projects are stored in a class library or repository.
- Once candidate classes are identified, the class library is searched to determine if these classes already exist.
- If class is already available in library extract and reuse it.
- If class is not available in library, it is engineered or developed using object-oriented methods.
- Any new classes built to meet the unique needs of the application.
- Now, process flow return to the spiral activity.

Component Based Development

- CBD model leads to software reusability.
- Based on studies, CBD model leads to 70% reduction in development cycle time.
- 84% reduction in project cost.
- Productivity is very high.

Prescriptive Process Model

- A prescriptive process model is a model that describes "how to do" according to a certain software process system.
- Calling this model as “Prescribe” because it recommend a set of process elements, activities, action task, work product & quality.
- A prescriptive model prescribes how a new software system should be developed. Prescriptive models are used as guidelines or frameworks to organize and structure how software development activities should be performed, and in what order.

1. Waterfall model
2. Incremental model
3. RAD model

Evolutionary Process Model

- Evolutionary Process Models are iterative. They are characterized in a manner that enables software engineers to develop increasingly more complete versions of the software.

1. Prototype model
2. Spiral model
3. Win-win spiral model
4. Concurrent development model

Questions!!

- Explain which process model is applicable for the development of the following systems:
  - Waterfall — (a) University accounting system that replaces an existing system
  - Prototype — (b) Interactive system that allows railway passengers to find time and other information from the terminals installed in the station.
  - Spiral — (c) A system to control anti-lock breaking in car.
  - Prototype — (d) A data entry system for office staff that has never used computers before. The user interface and user-friendliness are extremely important.
  - Waterfall — (e) A well understood data processing application.
  - Incremental + Prototype — (f) A Web-site for an on-line store which has a long list of desired features it wants to add, and it wants a new release with new features to be done very frequently.

Exercise

- Comparison of all process models

Thank You!!!

Any questions???