SOFTWARE PROCESS MODELS

NEED FOR MODELING A PROCESS

- Forms common understanding of the activities, resources and constraints involved in software development.
- Helps to find inconsistencies, redundancies and removals in the process, as these problems are noted and corrected the process becomes more effective.
- The model reflects the goals of development and shows explicitly how they can be achieved.
- Each development is different and a process has to be tailored for different situations, the model helps people to understand these differences.

PRESCRIPTIVE MODELS

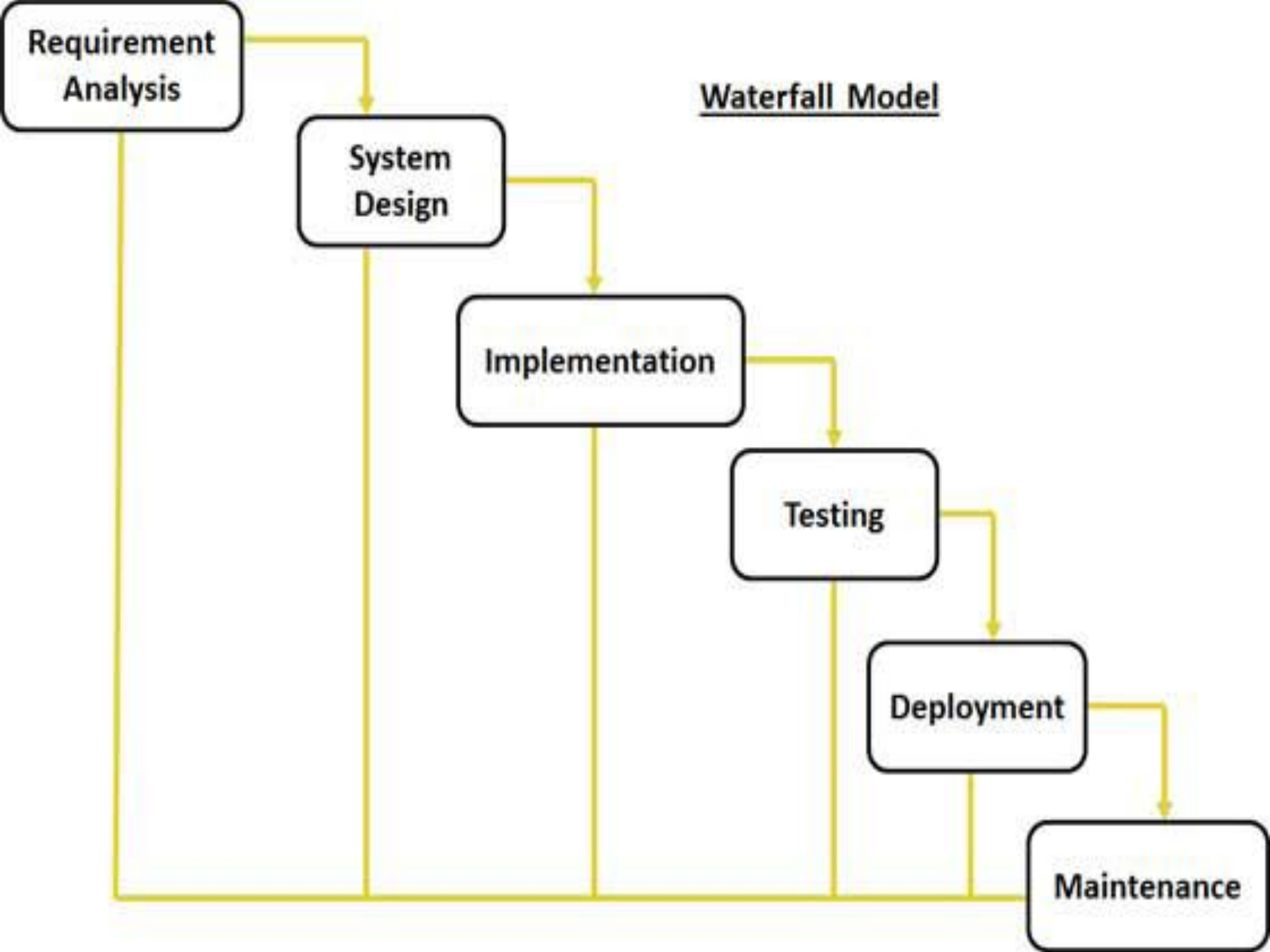
- Advocates an orderly approach to software engineering.
- They prescribes
  - A set of process elements,
  - Framework activities,
  - Software engineering actions,
  - Tasks,
  - Quality assurance and change control mechanism for each project.

WATER FALL MODEL

- **ADVANTAGES**
- The advantage of waterfall development is that it allows for departmentalization and control.
- A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.
- Development moves from concept, through design, implementation, testing , installation, troubleshooting , and ends up at operation and maintenance. Each phase of development proceeds in strict order.

WHEN TO USE

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- T here are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- T he project is short.


Requirement Analysis
System Design
Implementation
Testing
Deployment
Maintenance
Waterfall Model

DISADVANTAGES

- Current stage to be completed before moving to 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 makes no allowances for prototyping.
- The entire functionality is developed and then tested all together at the end.
- Major design problems may not be detected till very late.
- The model implies that once the product is finished, everything else is maintenance.
- The disadvantage of waterfall development is that it does not allow for much reflection or revision.
- Once an application is in the testing stag e, it is very difficult to g o back and change something that was not well documented or thought upon in the concept stage.

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model .
- each phase has specific deliverables and a review process.
- Phases are processed and completed one at a
- time.
- Works well for smaller projects where

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing . So risk and uncertainty is high with this process model.

- requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- No working software is produced until late in the life cycle.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big - bang . at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

INCREMENTAL PROCESS MODEL

- Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented.
- At each iteration, design modifications are made and new functional capabilities are added.
- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).
- "During software development, more than one iteration of the software development cycle may be in progress at the same time." and "This process may be described as an "evolutionary acquisition" or "incremental build" approach."

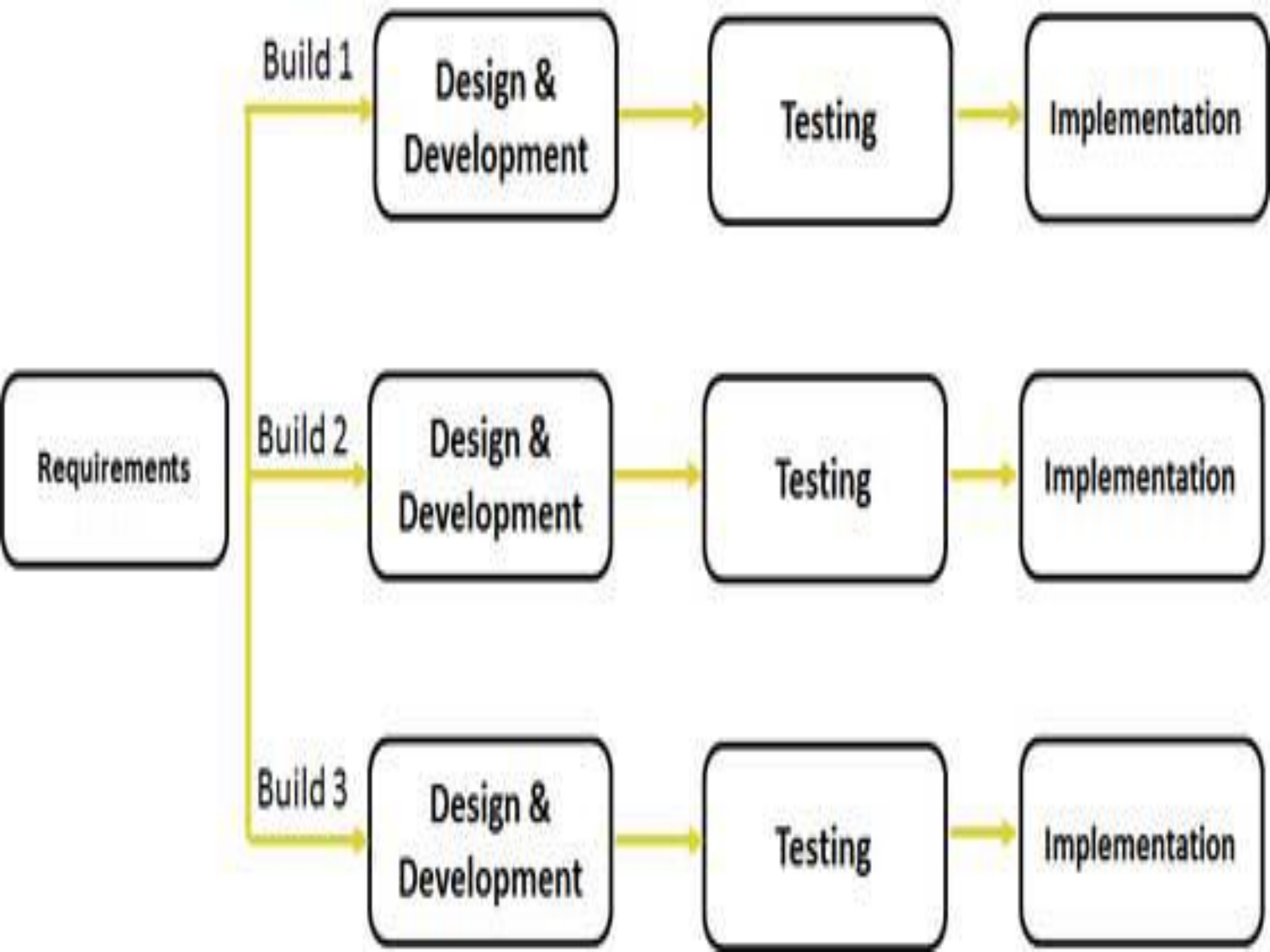
- In incremental model the whole requirement is divided into various builds.
- During each iteration, the development module goes through the requirements, design, implementation and testing phases.
- Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.
- The key to successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model.
- As the software evolves through successive cycles, tests have to be repeated and extended to verify each version of

WHEN TO USE

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- T here is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill set are not available and are planned to be used on contract basis for specific iterations.
- T here are some high risk features and goals which may change in the future.

ADVANTAGES

- Customers get usable functionality earlier than with waterfall.
- Early feedback improves likelihood of producing a product that satisfies customers.
- The quality of the final product is better
  - The core functionality is developed early and tested multiple times (during each release)
  - Only a relatively small amount of functionality added in each release: easier to get it right and test it thoroughly
  - Detect design problems early and get a chance to redesign

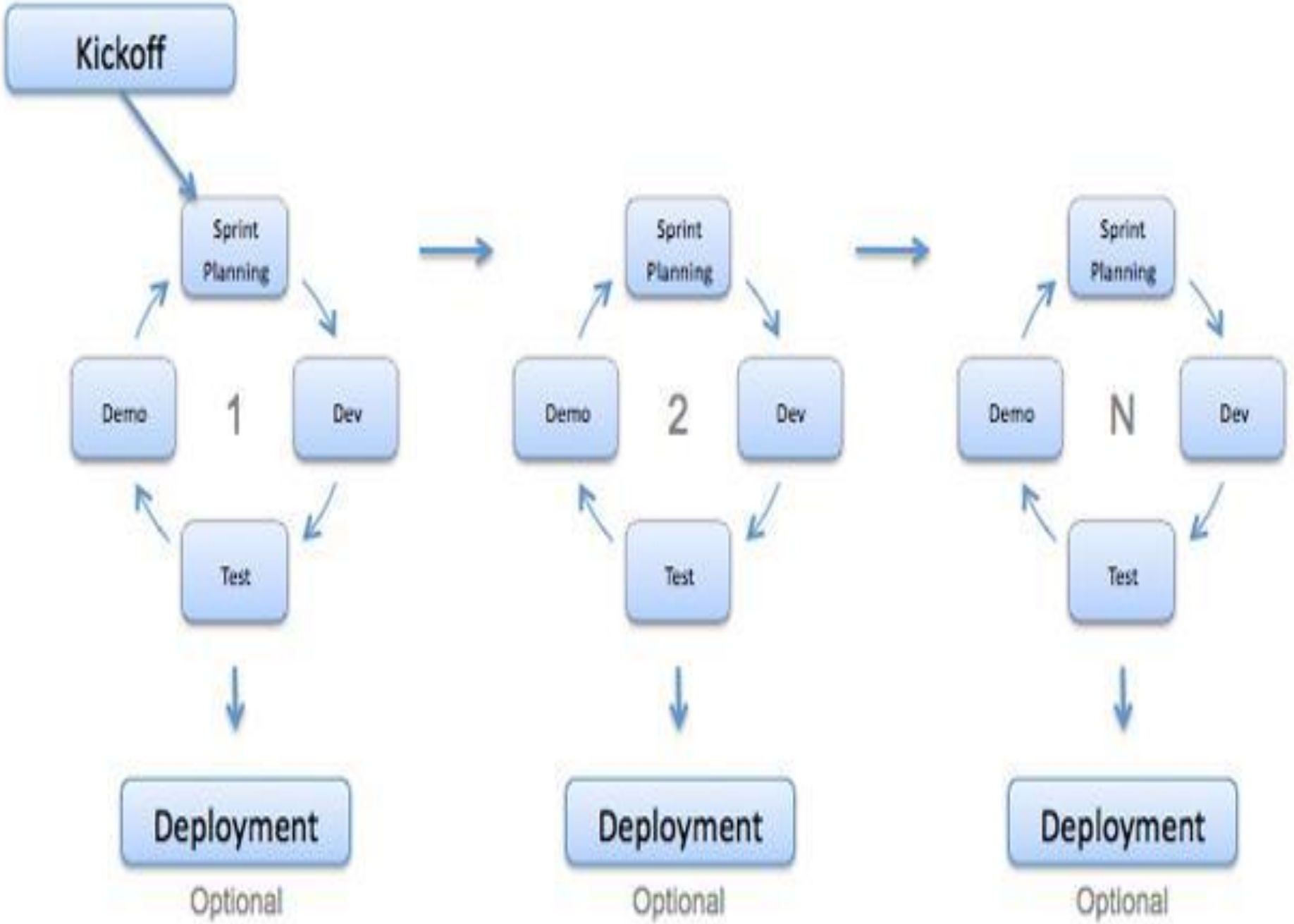

Requirements
Build 1
Design & Development
Testing
Implementation
Build 2
Design & Development
Testing
Implementation
Build 3
Design & Development
Testing
Implementation

DISADVANTAGES

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall.

RAD DEVELOPMENT MODEL

- RAD is an incremental software process model that emphasizes a short development cycle.
- Using Component based construction approach.


Kickoff
Sprint Planning
Demo
1
Dev
Test
Deployment
Optional
Sprint Planning
Demo
2
Dev
Test
Deployment
Optional
Sprint Planning
Demo
N
Dev
Test
Deployment
Optional

ADVANTAGES & DISADVANTAGES

- RAD reduces the development time
- Reusability of components help to speed up development.
- All functions are modularized so it is easy to work with.
- For large projects RAD require highly skilled engineers in the team.
- Both end customer and developer should be committed to complete the system in a much abbreviated time frame.
- If commitment is lacking RAD will fail.
- RAD is based on Object Oriented approach and if it is difficult to modularize the project the RAD may not work well.

EVOLUTIONARY SOFTWARE

PROCESS MODELS

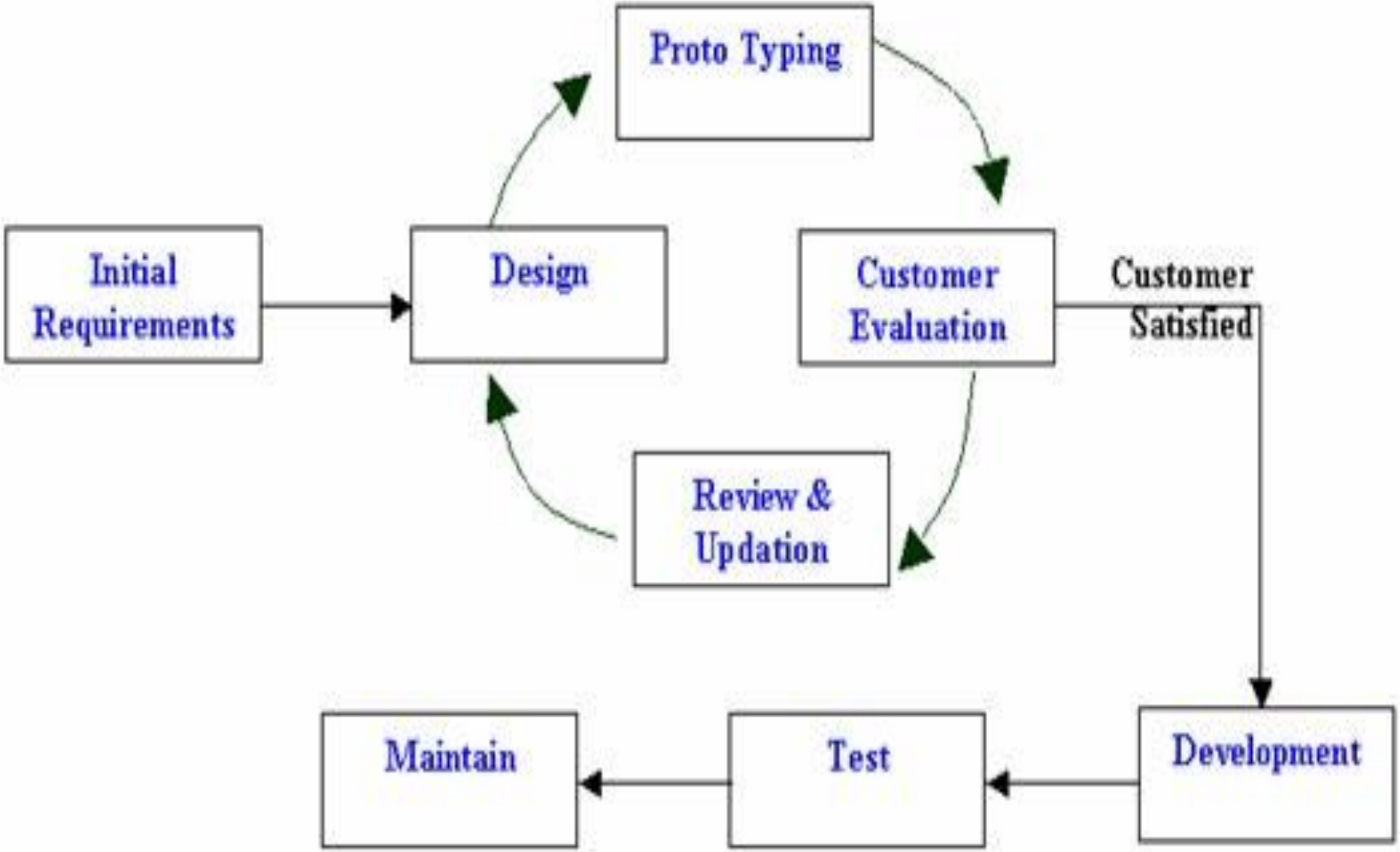
- Business and product requirement often change as development proceed.
- Software engineer need a process model that has been explicitly designed to accommodate a product that evolves over time.
- Evolutionary models are iterative.
- Enables software engineers to develop increasingly more complete version of the software.
- There are two types of evolutionary development:
  - Exploratory development
    - Start with requirements that are well defined
    - Add new features when customers propose new requirements
  - Throw-away prototyping
    - Objective is to understand customer's requirements (i.e. they often don't know what they want, hence poor requirements to start).
    - Use means such as prototyping to focus on poorly understood requirements, redefine requirements as you progress.

ADVANTAGES & DISADVANTAGES

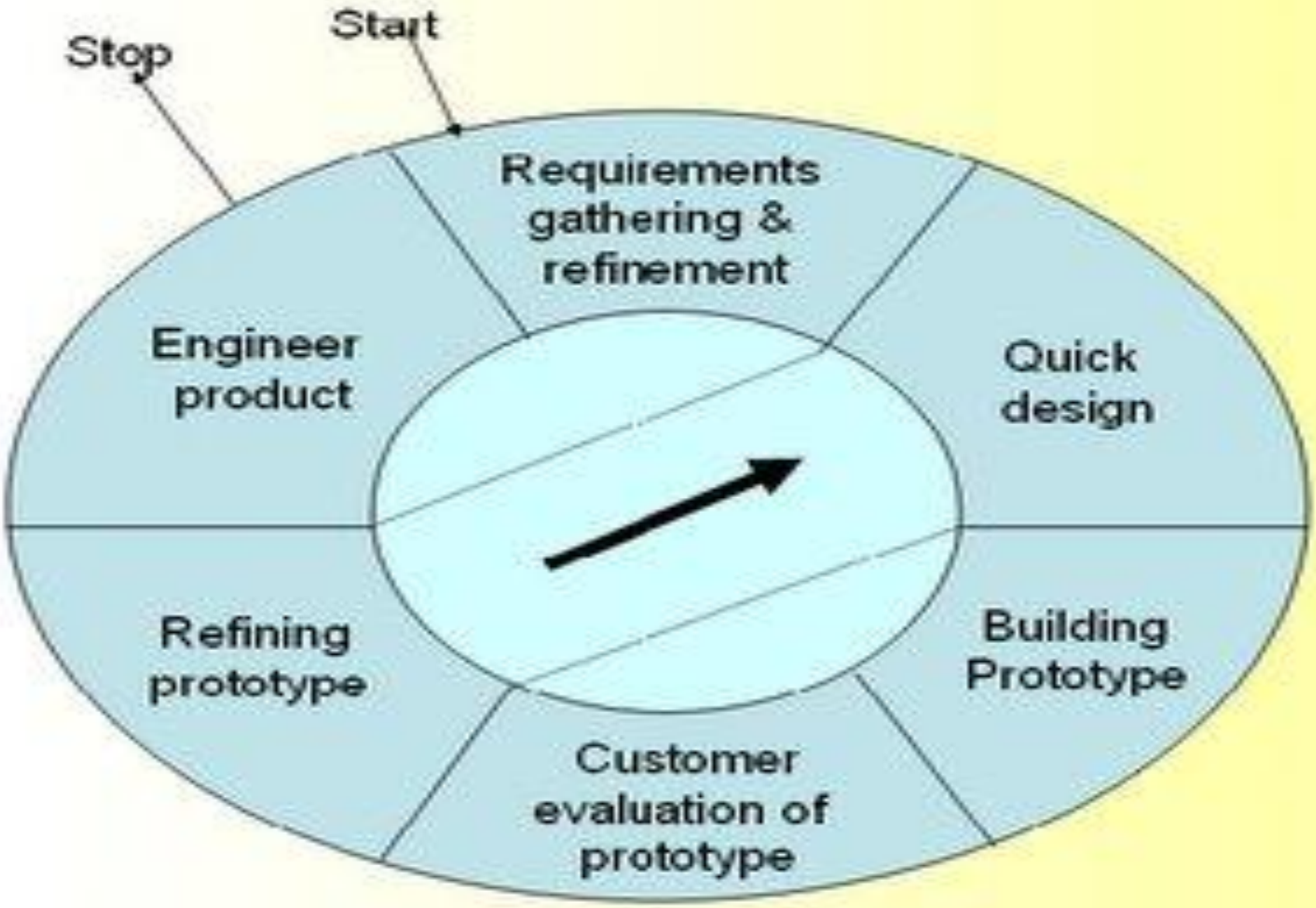
- Advantages:
  - Happier customers since we help them by defining requirements
  - Flexibility in modifying requirements
  - Prototypes are very visual, hence no ambiguities
- Disadvantages:
  - Hard to trace the “process” due to the ad-hoc nature
  - Systems are often poorly structured
  - Special tools and techniques may be required (for rapid development) that may be incompatible
  - Not cost-effective to produce documents

PROTOTYPING MODEL

- Customer defines a set of general objectives for software but doesn’t identify the detail.
- Assist the software engineer and the customer to better understand what is to be built when requirement are fuzzy.



Proto Type Model



PROTOTYPE MODEL

SPRIAL PROCESS MODEL

- A hybrid model where the development of the system spirals outward from an initial outline through to the final developed system.
- Each loop in the spiral represents a phase of the software process.
- Like the innermost loop might be concerned with system feasibility, next loop with system requirements, next loop with system design and so on.
- Each loop in the spiral is split into four sectors:
  - Object Setting: set specific object for that phase.
  - Risk assessment and reduction.
  - Development and validation: select a development model based on risk levels.
  - Planning: decide if a next loop is required

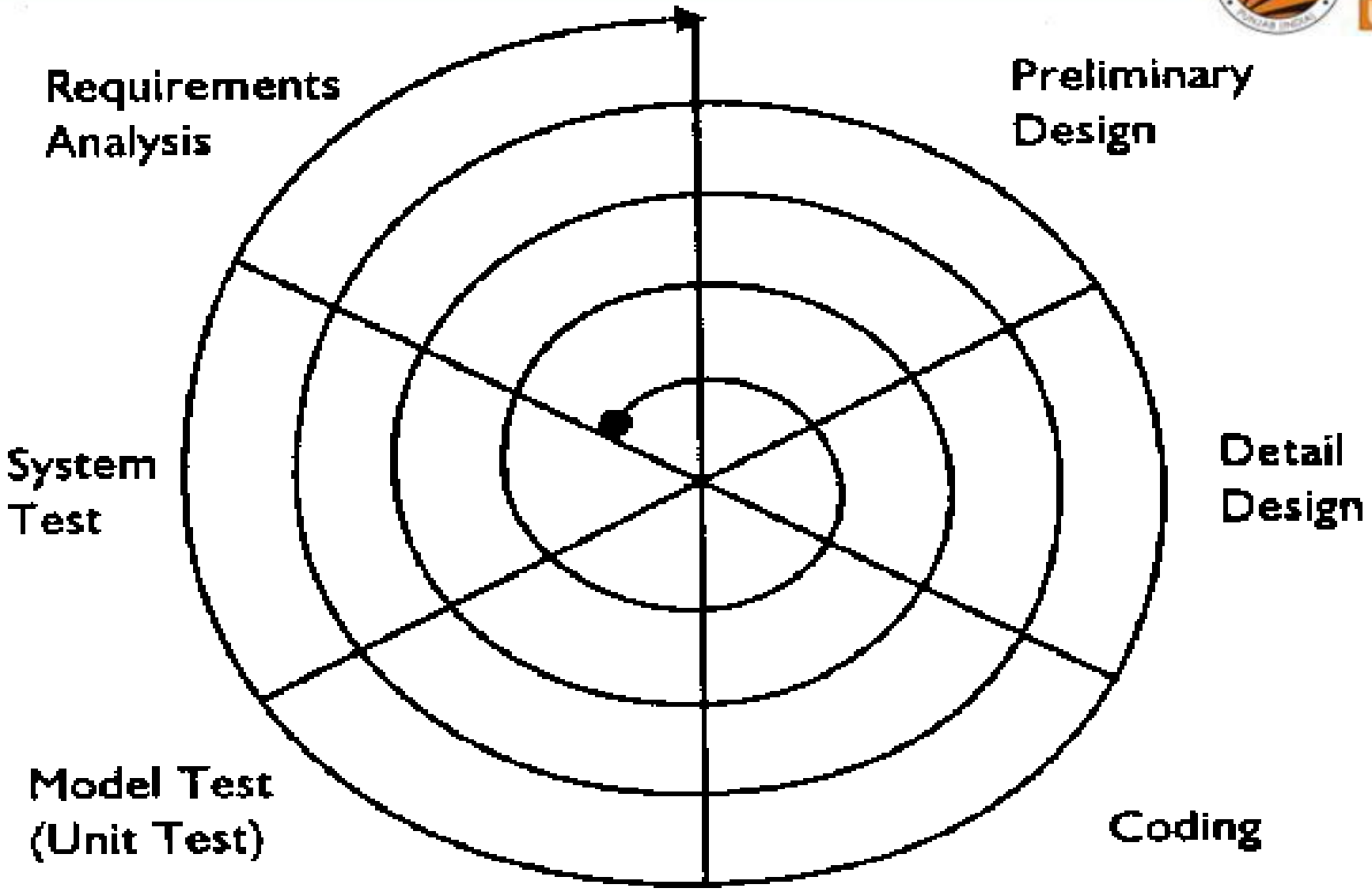


FIGURE 3: SPIRAL MODEL ADAPTED FROM [10].

ADVANTAGES & DISADVANTAGES

- Advantages
  - Explicit consideration of risks (alternative solutions are evaluated in each cycle).
  - More detailed processes for each development phase.
- Disadvantages
  - Cost is high.
  - Sometime difficult to implement or too time consuming.

FOURTH GENERATION TECHNIQUES

- The term fourth generation techniques (4GT) encompasses a broad array of software development technique.
- Each enables the software engineer to specify some characteristic of software at a high level.
- The tool then automatically generates source code based on the developer's specification.
- The 4GT paradigm for software engineering focuses on the ability to specify software using specialized language forms or a graphic notation that describes the problem to be solved in terms that the customer can understand.

- A software development environment that supports the 4GT paradigm includes some or all of the following tools:
- Nonprocedural languages for database query, report generation, data manipulation, screen interaction and definition, code generation.
- High-level graphics capability
- Spreadsheet capability, and automated generation of HTML and similar languages used for Web-site creation using advanced software tools.

- Many of the tools noted previously were available only for very specific application domains, but today 4GT environments have been extended to address most software application categories.

WORKING UNDER 4GT

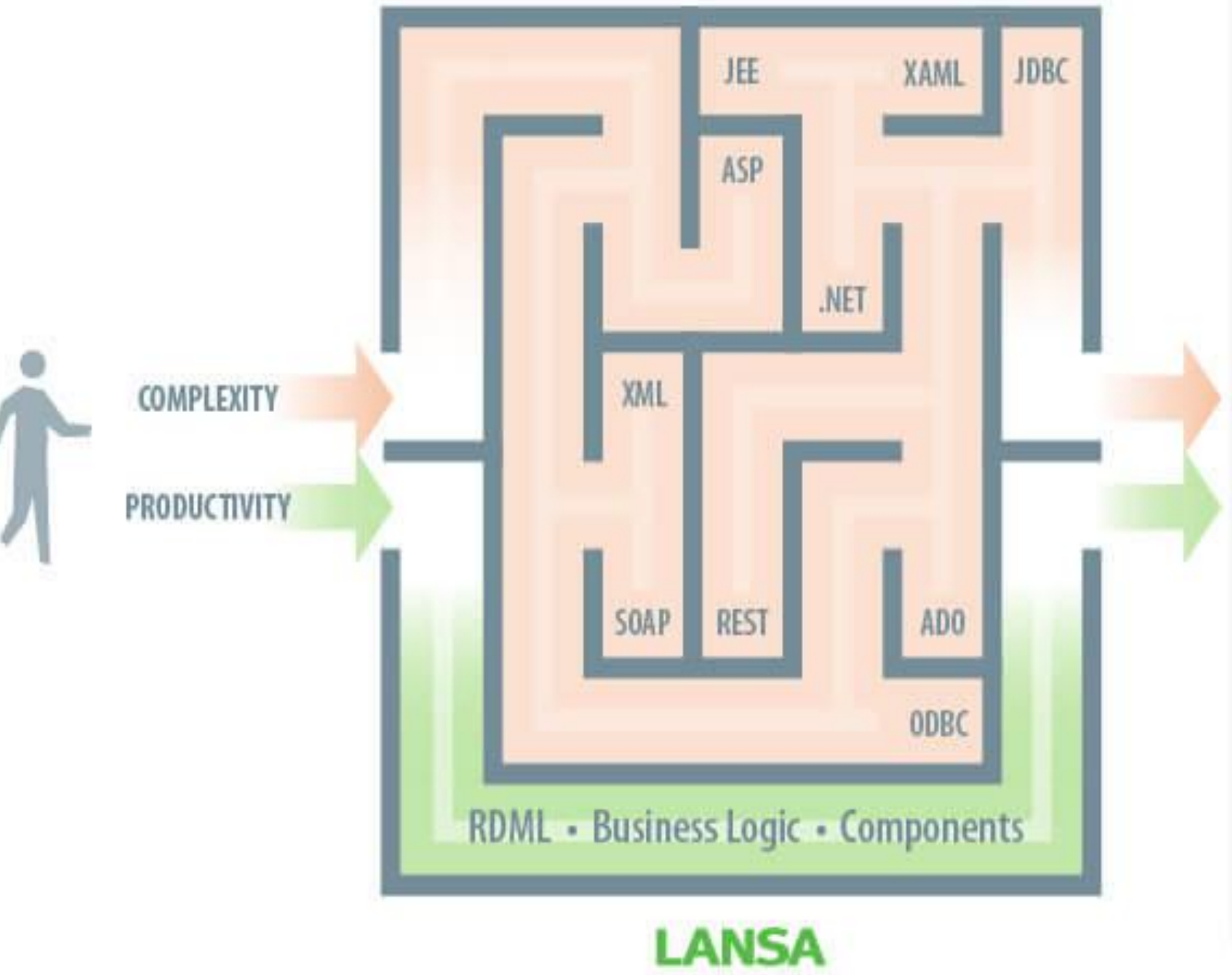
- 4GT begins with a requirements gathering step. Ideally, the customer would describe requirements and these would be directly translated into an operational prototype.
- Customer might be ambiguous in specifying facts about things he need, For this reason, the customer/developer dialog described for other process models remains an essential part of the 4GT approach.

- For small applications
  - It may be possible to move directly from the requirements gathering step to implementation using a nonprocedural fourth generation language (4GL) or a model composed of a network of graphical icons.
- For larger applications
  - It is necessary to develop a design strategy for the system, even if a 4GL is to be used.
  - The use of 4GT without design (for large projects) will cause the same difficulties (poor quality, poor maintainability, poor customer acceptance) that have been encountered when developing software using conventional approaches.
- Implementation using a 4GL enables the software developer to represent desired results in a manner that leads to automatic generation of code to create those results.

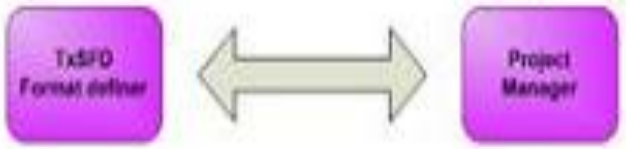
- The developer must conduct thorough testing, develop meaningful documentation, and perform all other solution integration activities that are required in other software engineering paradigms for implementing product developed using 4GT.
- In addition, the 4GT developed software must be built in a manner that enables maintenance to be performed efficiently.

CURRENT STATE OF 4GT

- Is a viable approach for many different application areas. Coupled with computer-aided software engineering tools and code generators, 4GT offers a credible solution to many software problems.
- Data collected from companies that use 4GT indicate that the time required to produce software is greatly reduced for small and intermediate applications and that the amount of design and analysis for small applications is also reduced.
- For large software development efforts demands as much or more analysis, design, and testing to achieve substantial time savings that result from the elimination of coding.


JEE
XAML
JDBC
ASP
.NET
XML
SOAP
REST
ADO
ODBC
COMPLEXITY
PRODUCTIVITY
RDML • Business Logic • Components
LANSA
Windows
5250
Web
Mobile

4th Generation



Client

Server

1st	2nd	3rd	4th	5th
Machine Language	Assembly Language	Procedural Languages	Nonprocedural Languages	Intelligent Languages
0–1 Long, difficult programming	Assemble repetitive instructions, shorter code	Include commands, shorter code	Application generators, commands specify results	Natural language processing

Progress

Thanks