


The System and Software Development Process

Instructor: Dr. Hany H. Ammar
Dept. of Computer Science and
Electrical Engineering, WVU



OUTLINE

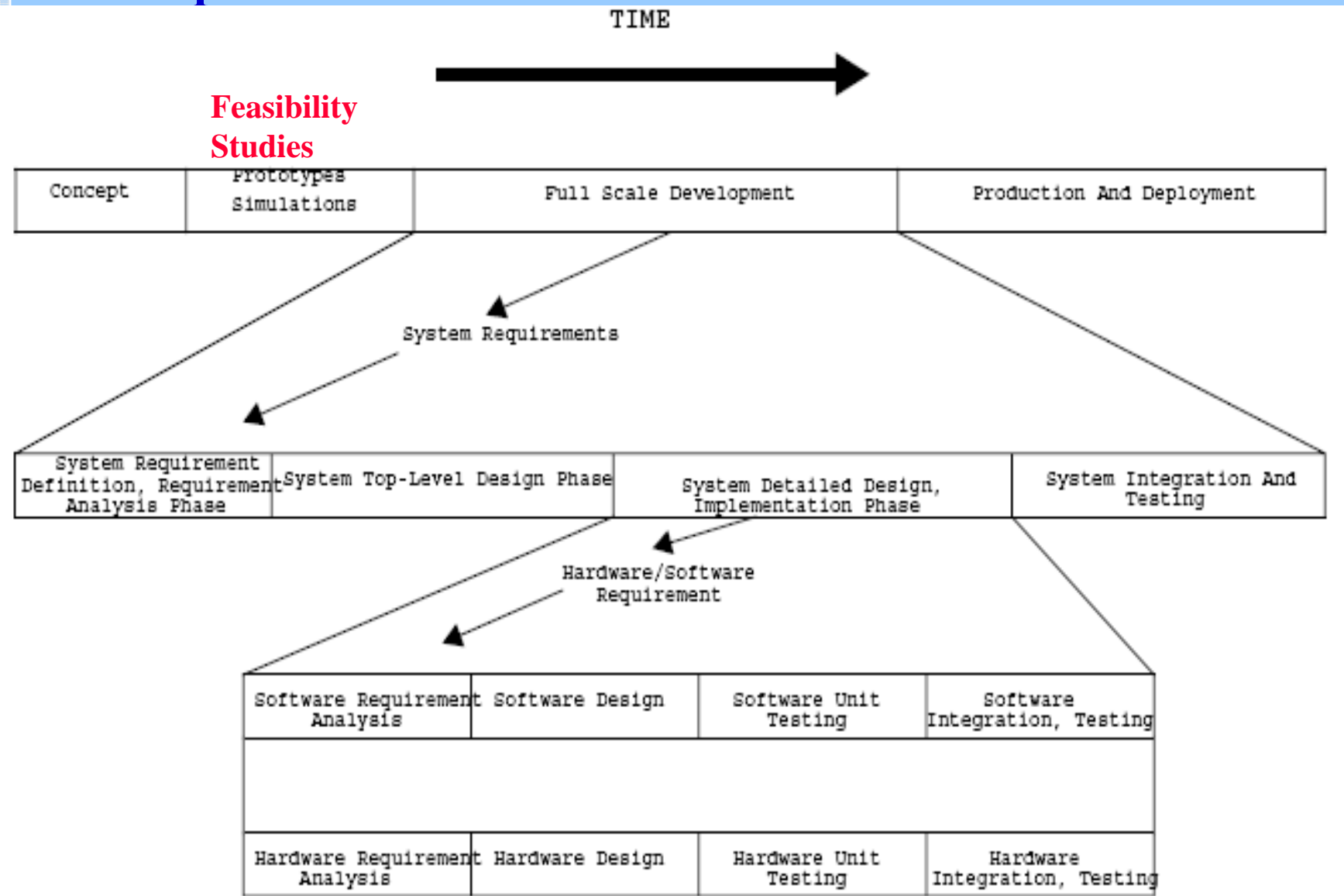
- 
- The System Life Cycle Model and the System Development Process.
 - Software Engineering and the Software Development Process
 - Software Development standards
 - ICASE Environments
 - ICASE Tool: Software through Pictures (StP)



The System Life Cycle Model and the System Development Process


- The system life cycle model is defined as the framework containing the processes, activities, and tasks involved in the development, operation, and support of a system.
- The word “cycle” in the above definition refers to the way a system usually evolves through several cycles of development and enhancement during its life span
- The concept of a development process model discussed next is an important part of the system life cycle model. It covers the activities and tasks starting from the definition of requirements to the deployment of the developed system.

The System Life Cycle Model and the System Development Process





The System Life Cycle Model and the System Development Process

- 
- The system concepts are established by capturing the user needs or requirements, and defining the scope of the system. This phase produces documents with titles such as “*The Mission Needs Statement*”, “*The Operational Requirements*”, or “*The User Requirements Document*”.
 - Capturing the user requirements involves the following:
 - Concept exploration,
 - The use of documented experiences with other systems,
 - The use of specially developed prototypes, and
 - Simulations.



The System Life Cycle Model and the System Development Process

- the full-scale development phase consists of the following basic set of activities:

The system requirements definition and analysis phase: in which a hierarchy of system requirements is produced, and then analyzed for consistency, completeness, & testability

The system top-level design phase: in which the system configuration is designed in terms of hardware and software configuration items or components. These items and their interfaces are specified. Software and hardware requirements for these components are also specified



The System Life Cycle Model and the System Development Process

The system detailed design and implementation phase:

During this phase, each configuration item specified in the previous phase is developed and tested.

The system integration and testing phase: in which the developed software and hardware items are integrated and tested. The system requirements are validated in this phase

- Several documents and reviews are prepared and performed during each of the above phases (we will see later the development standards in section 2.3 in the notes).



OUTLINE

- The System Life Cycle Model and the System Development Process.
- Software Engineering and the Software Development Process
 - The Waterfall Model
 - Prototyping Based Models
 - Models based on risk analysis
 - *incremental development*
 - *the evolutionary development*
 - Agile (light-weight) Software Development
 - The Unified process (IBM Rational)



Software Engineering and the Software Development Process

The Waterfall Model

- This model for software development follows the classic life cycle model which assumes a sequential development process consisting of several development phases.
- The process iterates within each phase to correct problems found during reviews and verification activities.
- It also iterates back from the maintenance and operation phase to the early development phases in order to correct problems found during operation or to deal with new requirements.

(See Figure 2.2 in Page 2-9 in the notes)

Software Engineering and the Software Development Process

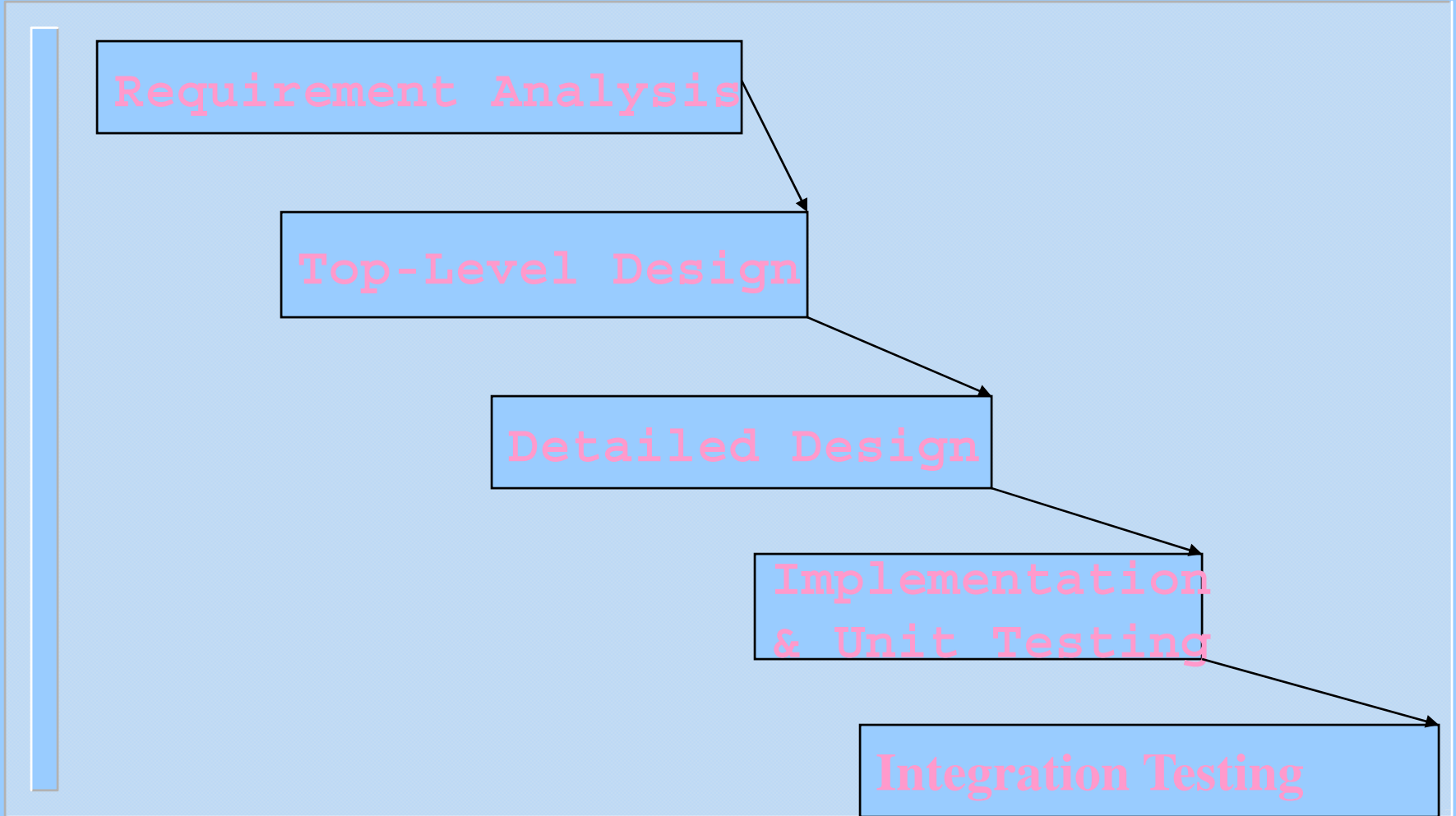
Requirement Analysis

Top-Level Design

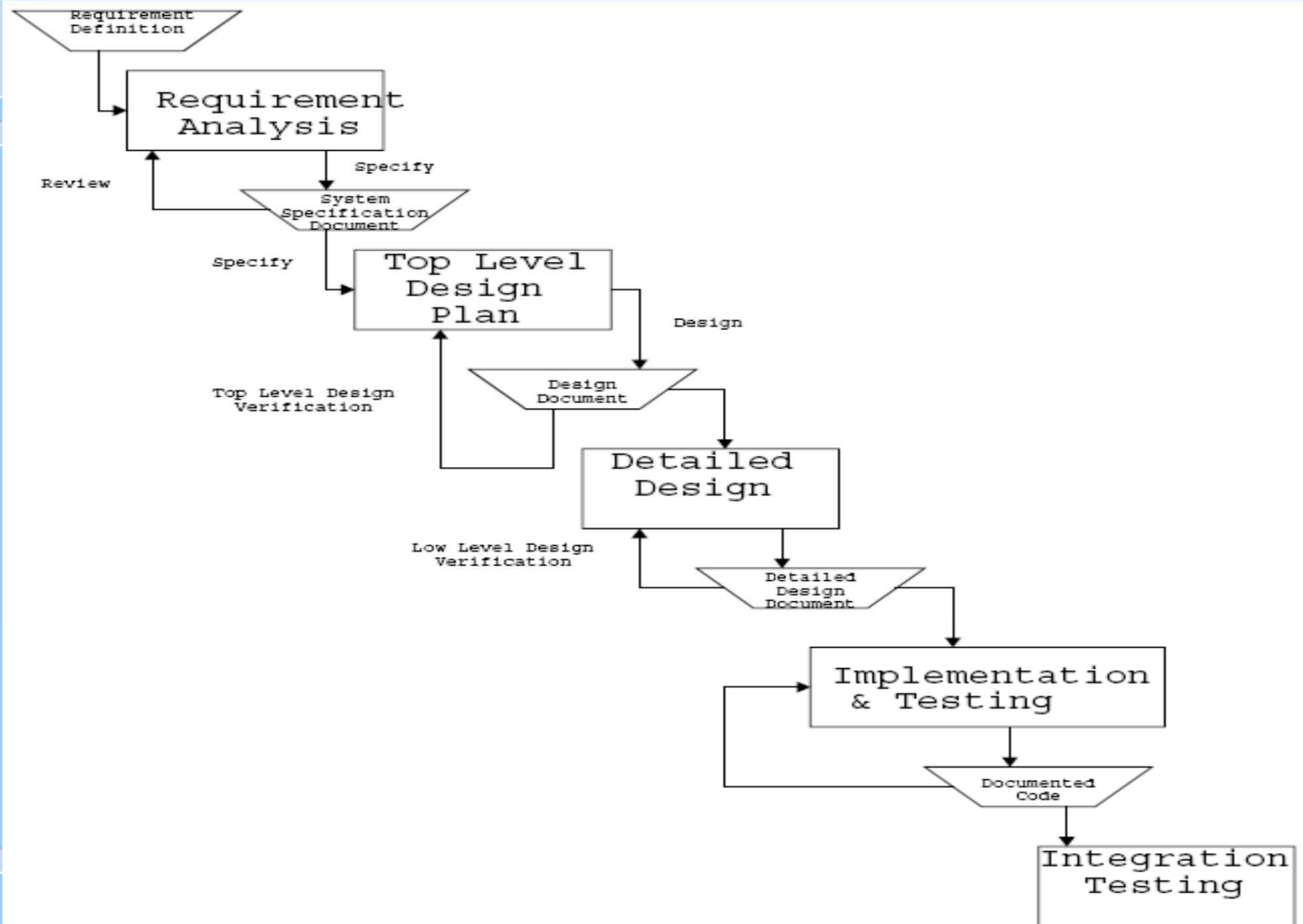
Detailed Design

Implementation
& Unit Testing

Integration Testing




The Waterfall Model (Fig. 2.2)





Software Engineering and the Software Development Process

- Given a set of requirements for a software configuration item (specified in a requirements definition document),
- the software development process starts with the *requirements analysis and specification phase*.
- This is followed by a *preliminary design (or high level design) phase*, and then a *detailed design phase*.
- Once the design is finalized, *the coding or implementation phase* begins which is followed by the *testing and validation phase*.



Software Engineering and the Software Development Process

Disadvantages of the water Fall Model

- Although the requirements and design activities influence each other as they develop, the waterfall model assumes a strictly sequential development. Iterations are only allowed within a phase
- The waterfall model does not capture the realistic sequence of activities and tasks required for modern software development.
- The problem comes from requiring the developer to write detailed specifications of vague, ambiguous, imprecise, or poorly understood requirements



Software Engineering and the Software Development Process

- For complex real-time systems in particular, the task of writing detailed adequate, consistent, correct, and complete specifications based on vague, ambiguous, imprecise, contradicting, or incomplete user requirements is almost impossible

Advantages of the model

- The main advantage of the waterfall model is to facilitate the implementation of the management activities of estimating the cost and schedules of the development process.
- This simple sequential development process is also easier to track and tailor to large projects.



OUTLINE

- The System Life Cycle Model and the System Development Process.
- Software Engineering and the Software Development Process
 - The Waterfall Model
 - Prototyping Based Models
 - Models based on risk analysis
 - *incremental development*
 - *the evolutionary development*
 - Agile (light-weight) Software Development
 - The Unified process (IBM Rational)



Software Engineering and the Software Development Process



Prototyping Based Models

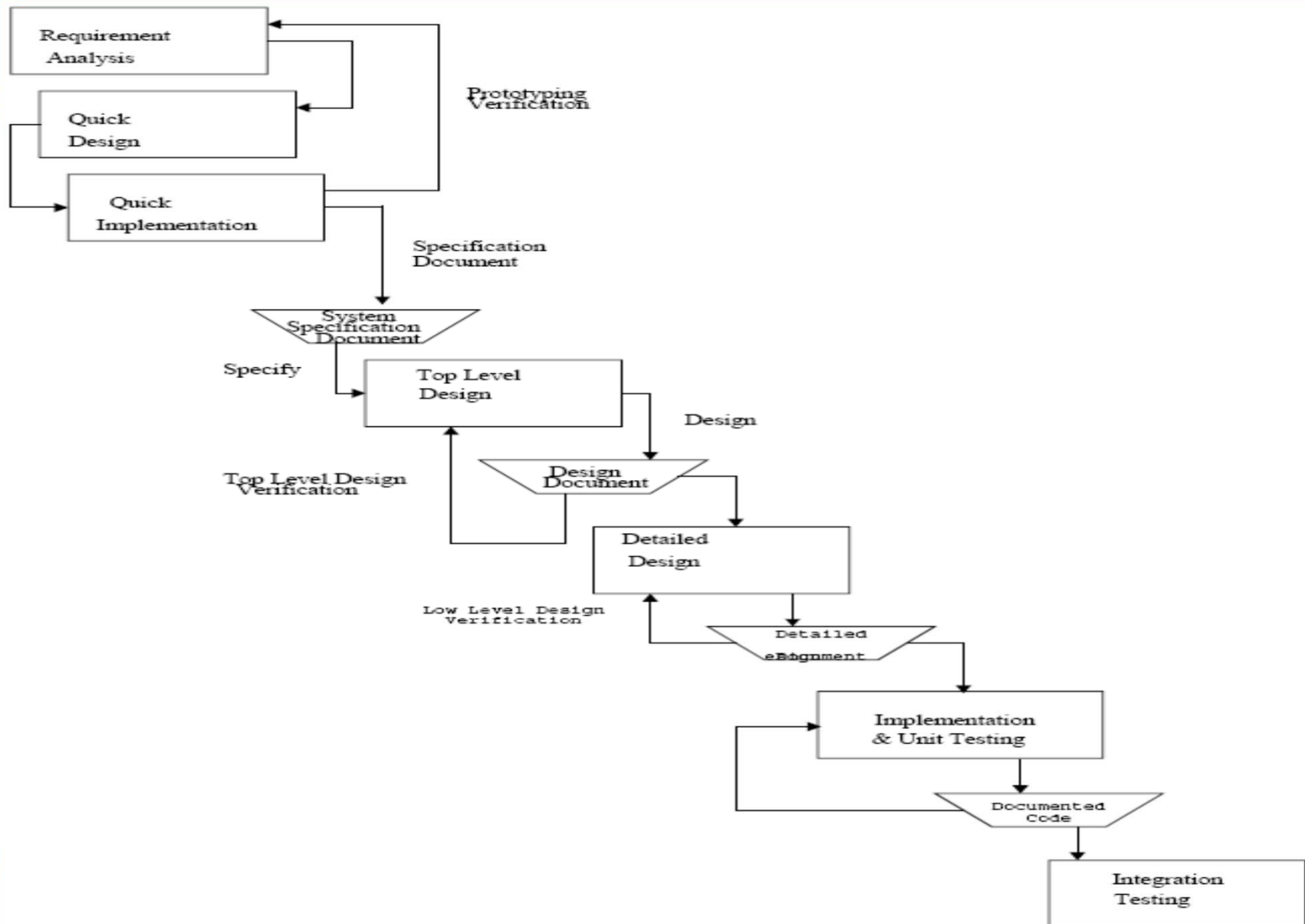
- A working version may be developed through quick analysis, design, and implementation following similar activities as specified in the waterfall model but executed in a much faster pace to come up with a quick, dirty version of the product
- The broad definition of prototypes used in this includes simulation models and executable specifications. These are dynamic models developed using special modeling and simulation tools to study the dynamic behavior of the software as it reacts to external events.
- These models are important to study the timing behavior by simulating the external environment, as well the state transitions which take place as the software executes its functions in reaction to external events.



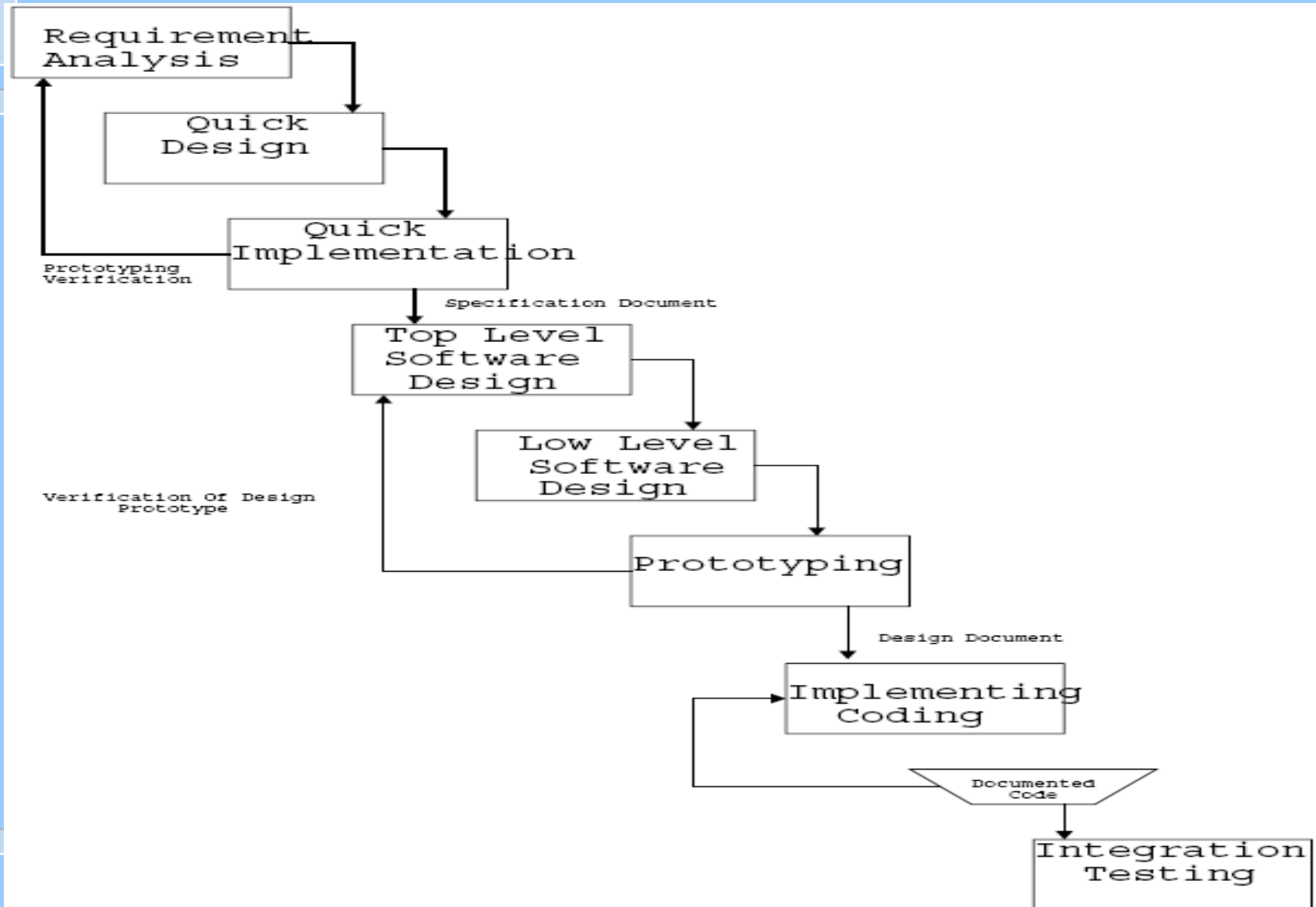
Software Engineering and the Software Development Process

- A development process model may require the use of a prototype for developing a clear, consistent, and complete specification. This activity is based on getting feedback from users and other software developers.
- The process would then proceed in a sequential fashion following the development phases in the waterfall model as shown in Figure 2.3.
- Prototyping may also be used during design and implementation. In this case dynamic simulation models are used to guide the development activities and get feedback from users and other developers on the current status of the evolving product as shown in Figure 2.4.

Prototyping Based Models (Fig. 2.3)



Prototyping Based Models (Fig. 2.4)





OUTLINE

- The System Life Cycle Model and the System Development Process.
- Software Engineering and the Software Development Process
 - The Waterfall Model
 - Prototyping Based Models
 - Models based on risk analysis
 - *incremental development*
 - *the evolutionary development*
 - Agile (light-weight) Software Development
 - The Unified process (IBM Rational)



Software Engineering and the Software Development Process

Models based on risk analysis

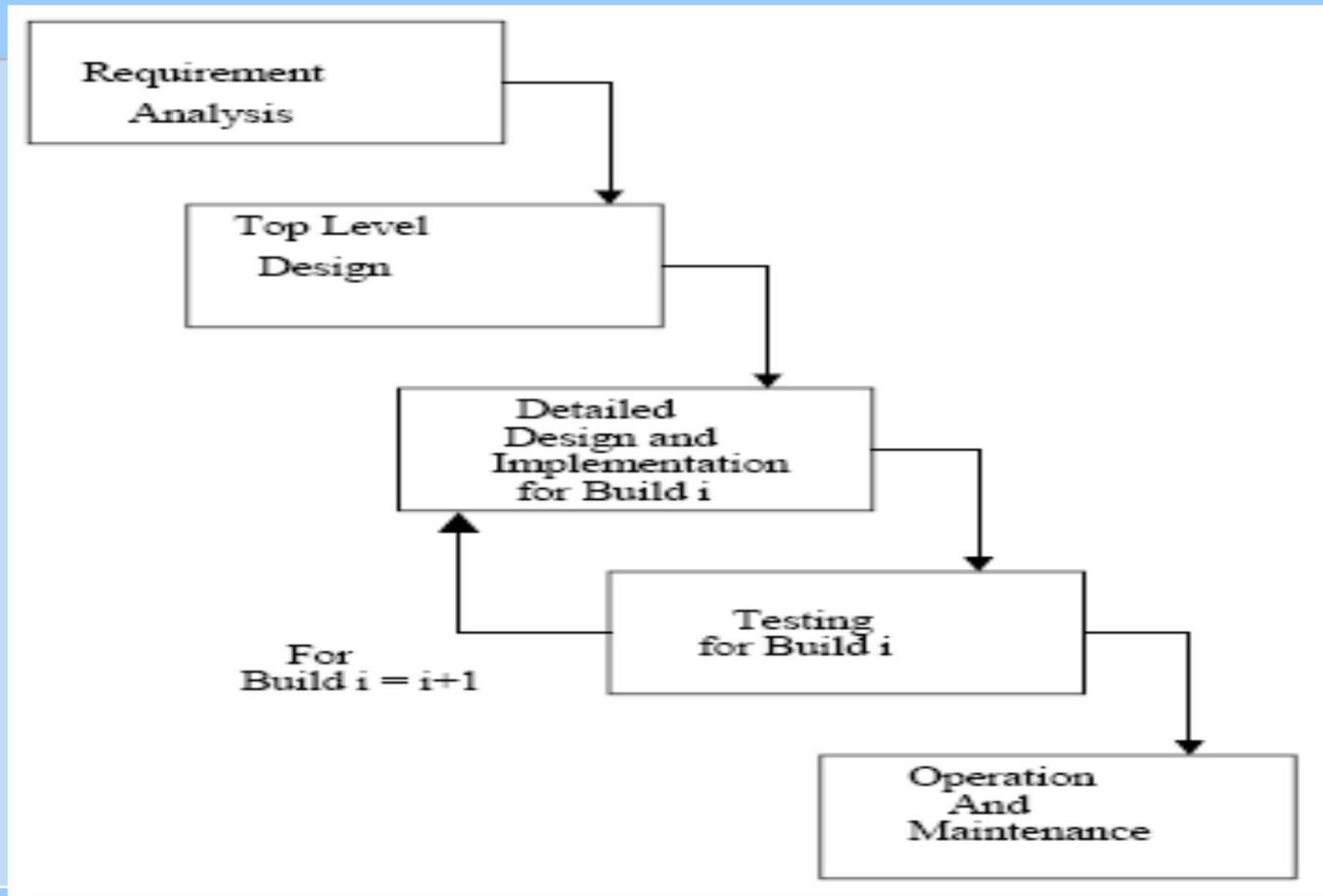
- The term risk factor has an implied definition of risk as a measure of uncertainty in achieving the project goals such as developing a product which satisfies users needs while meeting the project deadlines.
- Some of the important factors affecting the overall project risk factor are based on uncertainty in:
 - understanding users needs,
 - assessing the difficult technical problems which might show up during design and implementation,
 - handling changing requirements which arise due to technological advances or changes in the user needs.



Software Engineering and the Software Development Process

- In order to decrease the project risk factor, two models have been suggested in practice.
- The first model is based on an *incremental development approach*, where the software is developed in multiple builds (or versions), each with increased functionality and capability.
- In this model, following the architectural design (or high-level design) phase, the rest of the phases in the waterfall model are executed for each build.
- This approach is necessary for large projects since a single build may not be practical.
- The disadvantage of this model is the increased amount of testing required to confirm that existing capabilities of the software are not impaired by any new function in the new build.

The Incremental Development Model (Fig. 2.5)

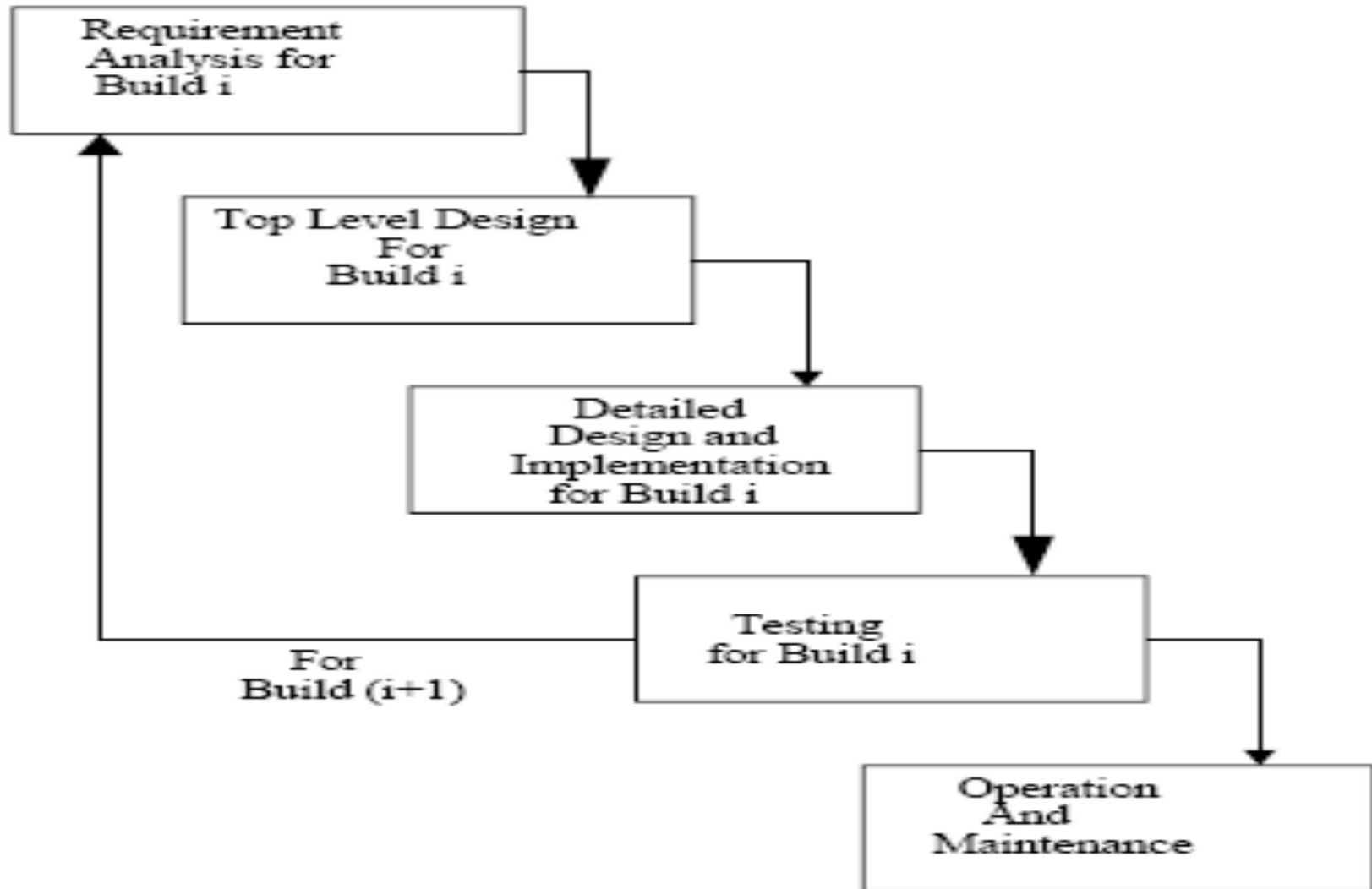




Software Engineering and the Software Development Process

- The second model follows the *evolutionary development approach* in which all phases of the life cycle are performed to produce a release.
- The project plan in this case specifies the development of multiple releases. Each release incorporates the experience of earlier ones.
- One major reason for such an approach is that the user requirements are incomplete to start with.
- One major disadvantage of this approach is that the initial software architectural design may not be easily adaptable to bear the changes necessary for later evolutions. In this case much more time may be spent in redesigning the software architecture in subsequent releases.

The Evolutionary Development Model





Software Engineering and the Software Development Process

- The incremental approach learns from the evaluation and testing of the previous build to improve the quality and functionality of the product in the subsequent builds.
- With this approach, it is assumed that the user requirements are completely defined before the development of the first build can start.
- The evolutionary approach, on the other hand, assumes that the user requirements are only partially specified and it evolves with the user experiences with each release.

	Water Fall		Incremental		Evolutionary	
Risk-Item no.	Risk Item (Reasons against this strategy)	Risk Level	Risk Item (Reasons against this strategy)	Risk Level	Risk Item (Reasons against this strategy)	Risk Level
1	Requirements are not well understood	H	Requirement are not well understood	H	User prefer all capabilities at first delivery	M
2	System too large to do all at once	M	User prefers all capabilities at first delivery	M		
3	Rapid changes in mission technology anticipated--may change the requirements	H	Rapid changes in mission technology are expected --may change the requirements	H		
4	Limited staff or budget available now	M				
Opp-Item no.	Opportunity Item (Reasons to use this strategy)	Opp. Level	Opportunity Item (Reasons to use this strategy)	Opp. Level	Opportunity Item (Reasons to use this strategy)	Opp. Level
1	User prefers all capabilities at first delivery	M	Early capability is needed	H	Early capabilities is needed	H
2	User prefers to phase out old system all at once	L	System breaks naturally into increments	M	System breaks naturally into increments	M
3			Funding/staffing will be incremental	H	Funding/staffing will be incremental	H
4					User feedback and monitoring of technology changes is needed to understand full requirements	H
					Decision: Use this strategy	

Table 2.1: Sample risk analysis for determining the appropriate program strategy. Risks and

The Spiral Model of Software Development

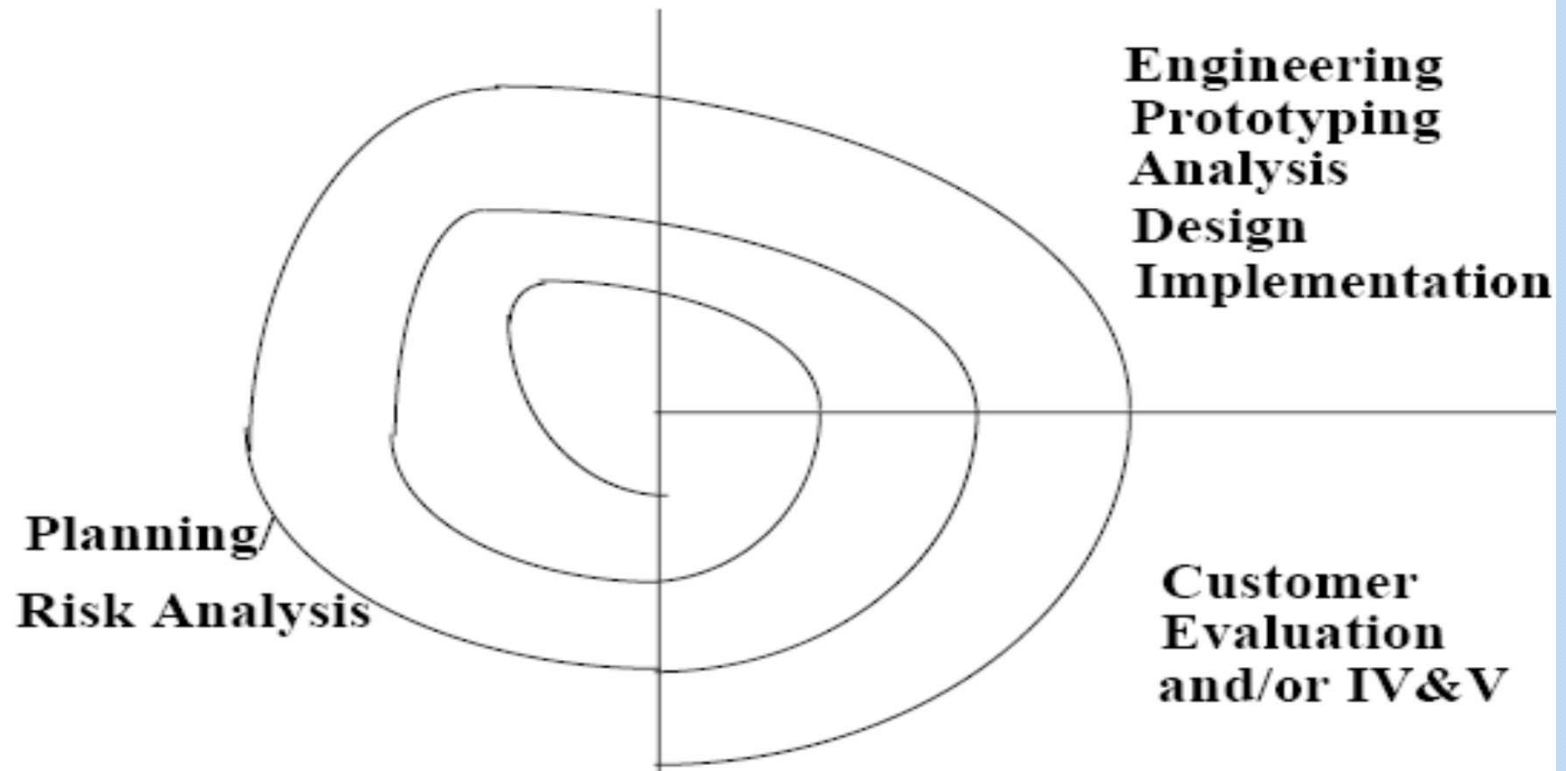


FIGURE 2.7 Spiral model




OUTLINE

- The System Life Cycle Model and the System Development Process.
- Software Engineering and the Software Development Process
 - The Waterfall Model
 - Prototyping Based Models
 - Models based on risk analysis
 - *incremental development*
 - *the evolutionary development*
 - Agile (light-weight) Software Development
 - The Unified process (IBM Rational)




Agile (light-weight) Software Development

- 
- Individuals and interactions over processes and tools
 - Working software over comprehensive documentation
 - Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, the approach values the items on the left more




Agile (light-weight) Software Development (continue)

- 
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
 - Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale

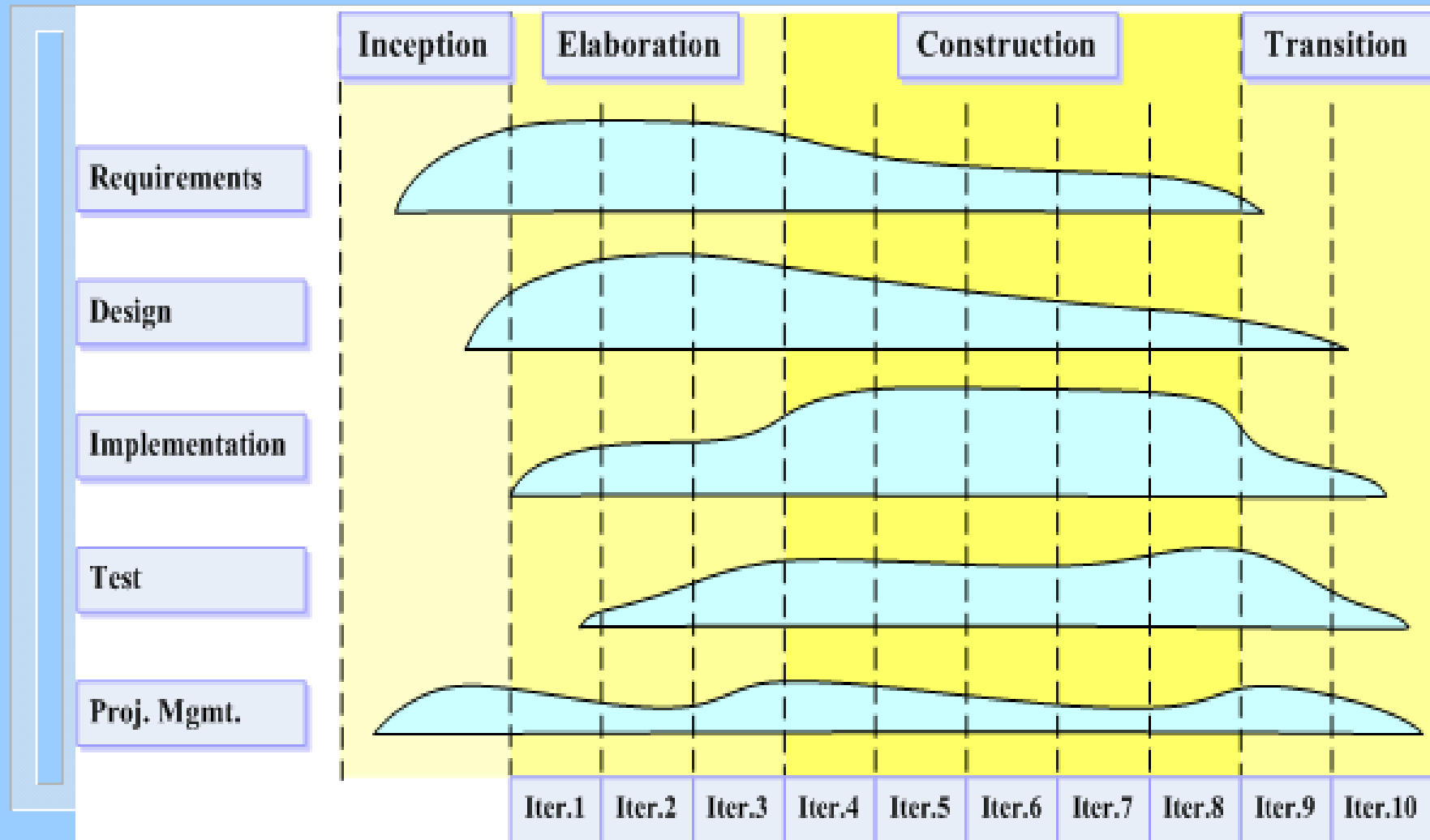


Agile (light-weight) Software Development (continue)

- 
- Agile methods differ from iterative methods in that their time period is measured in weeks rather than months and work is performed in a highly collaborative manner.
 - Agile methods have much in common with the "Rapid Application Development" techniques from the 1980's

The Unified process (IBM Rational)

Iterative and Incremental



The Unified process (Cont.)

