


The background is a deep blue gradient. On the left side, there are several interlocking gears of different sizes, rendered with a glowing, translucent effect. To the right of the gears, a complex network of white lines connects numerous small dots, resembling a molecular structure or a data network. The overall aesthetic is high-tech and digital.

# Software Engineering

---

Dr. Rasha Montaser



# Lec 2

## Process Models

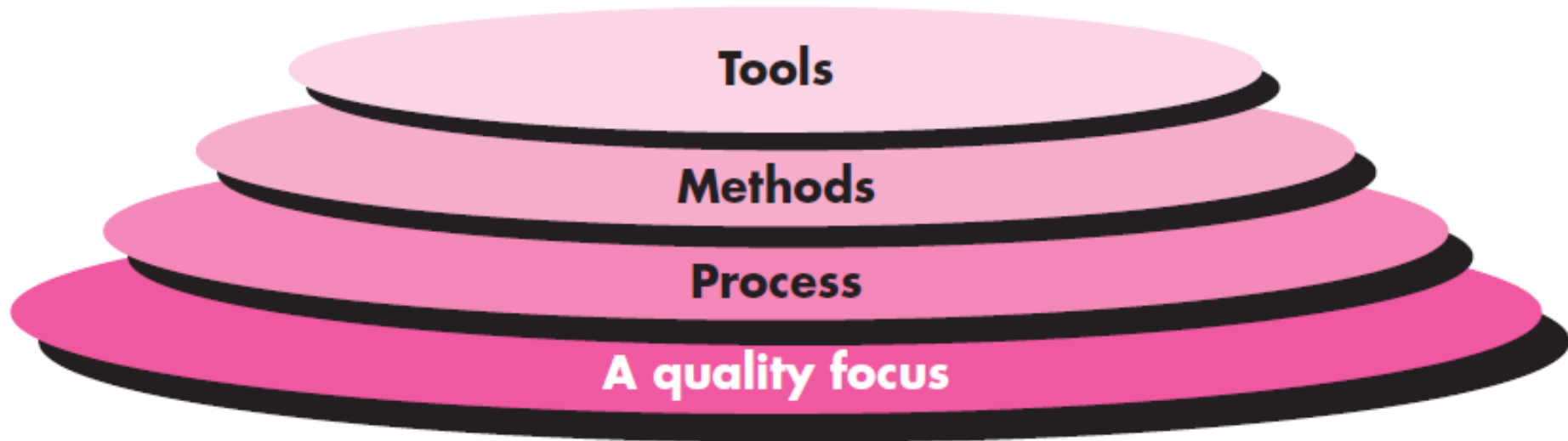
Ref: Chapter 2



# Software Engineering

- The application of a systematic, discipline, qualifiable approach to the development, operation and maintenance of software.

# Software Engineering layers





# Software process

- A series of predictable steps that helps you to create a timely, high-quality software.





# Process framework

- A generic process framework for software engineering encompasses five activities:
  1. **Communication (Requirement Analysis).**
  2. **Planning and Modeling (Design).**
  3. **Construction (Implementation).**
  4. **Testing**
  5. **Deployment (Delivery and Installation).**



# 1- Communication

- Before any technical work can commence, it is critically important to communicate and collaborate with the customer and other Stakeholders
- The intent is:
  - to understand stakeholders' objectives for the project
  - to gather requirements that help define software features and functions.



# The work to be done to accomplish the objective of the software

1. Make a list of stakeholders for the project.
2. Interview each stakeholder separately to determine overall wants and needs.
3. Invite all stakeholders to an informal meeting.
4. Ask each stakeholder to make a list of features and functions required.
5. Discuss requirements and build a final list.
6. Prioritize requirements.
7. Note areas of uncertainty.
8. Note constraints and restrictions that will be placed on the system.
9. Discuss methods for validating the system.





## 2- Planning

- Creates a “map” that helps guide the team as it makes the journey.
- Describe:
  - the technical tasks to be conducted
  - the risks that are likely,
  - the resources that will be required
  - the work products to be produced
  - a work schedule.



## 2- Modeling

- create a “sketch” of the thing so that you’ll understand the big picture and to better understand the problem and how you’re going to solve it.



## 3- Construction

- This activity combines **code generation** (either manual or automated)



## 4- Testing

- the **testing** that is required to uncover errors in the code.



## 5- Deployment

- The software (as a complete entity or as a partially completed increment) is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.

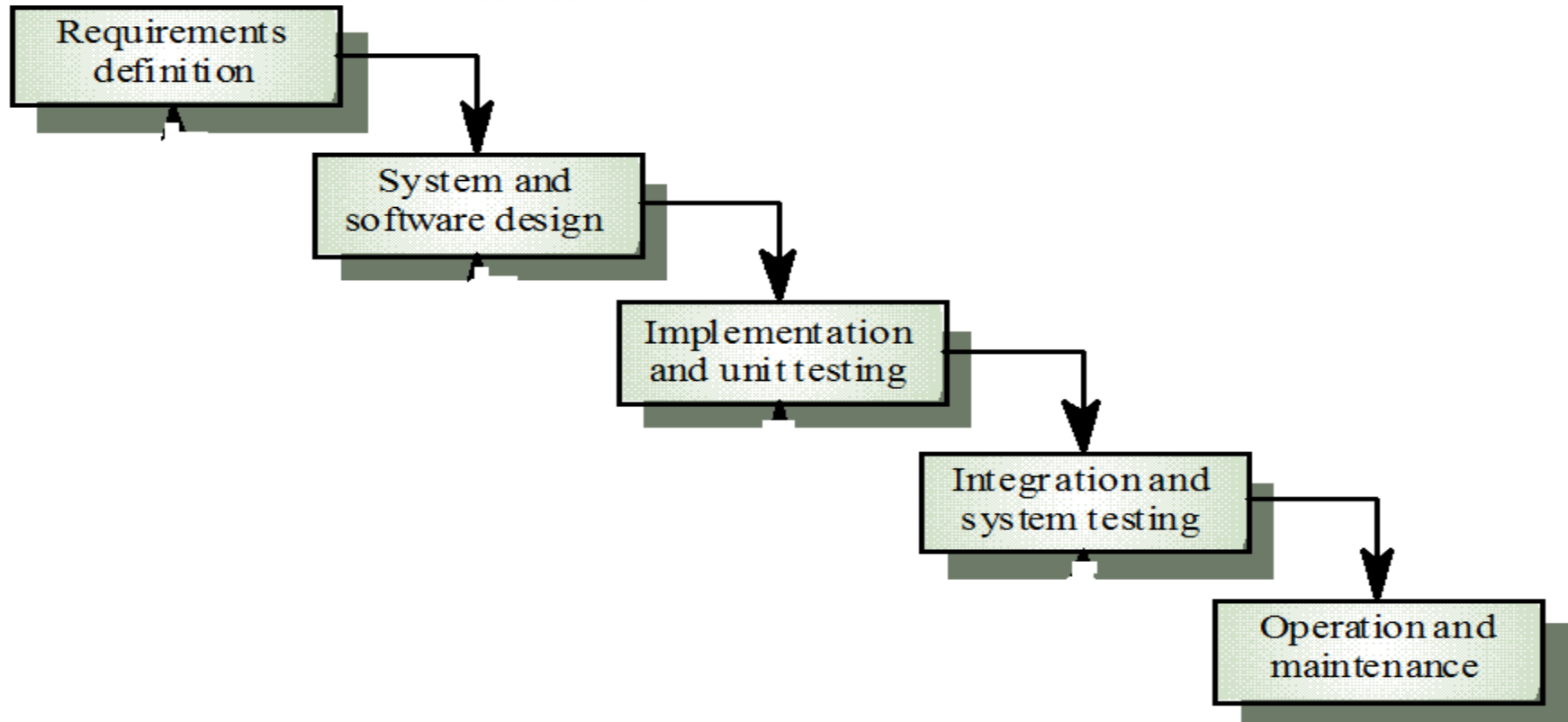


# Process Models

1. Perspective models (standard)
  - a. The waterfall model
2. Incremental process models
  - a. The incremental model
3. Evolutionary process model
  - a. Prototyping
  - b. Spiral model
4. Specialized process model
  - a. Concept based development
  - b. The formal methods model
  - c. Aspect oriented software development



# The waterfall model





# The waterfall model

- When to use?
  - The requirements are well defined and reasonably stable.
  - The requirements are fixed
  - The work proceed to completion in a linear manner.

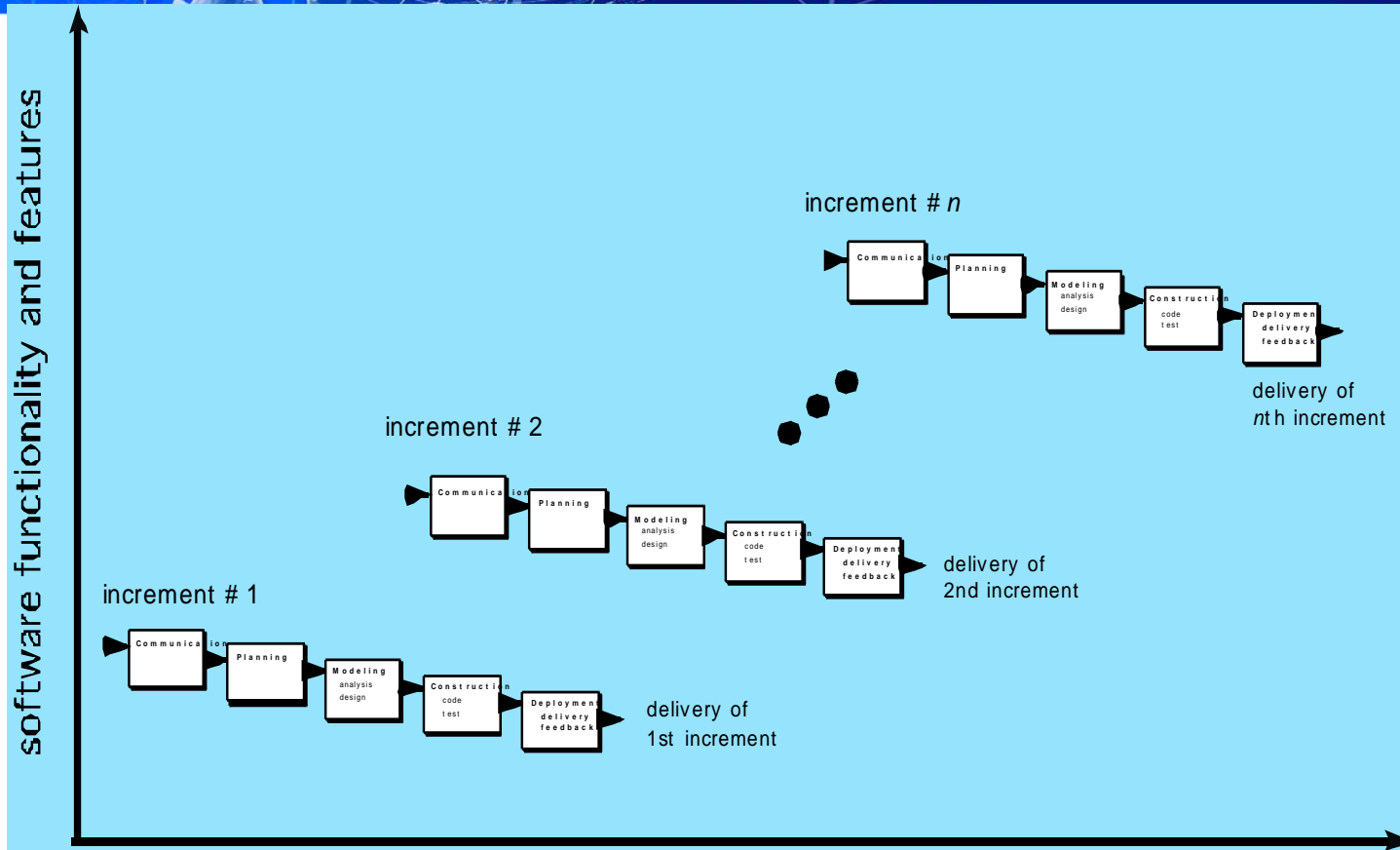


# The waterfall model

- **Problems**

1. Iterations can be accommodated which cause confusion as the project team proceeds.
2. It is often difficult for the customer to state all requirements explicitly.
3. The customer must have patience. A working version of the program(s) will not be available until late in the project time span.
4. the linear nature of the classic life cycle leads to “blocking states” in which some project team members must wait for other members of the team to complete dependent tasks.

# The Incremental model





# The Incremental model

- The incremental model delivers a series of releases, called increments, that provide progressively more functionality for the customer as each increment is delivered



# Example of Incremental Model

- word-processing software developed using the incremental paradigm
  - deliver basic **file management, editing, and document production functions** in the **first increment**;
  - more **sophisticated editing and document production capabilities** in the **second increment**;
  - **spelling and grammar checking** in the **third increment**;
  - **advanced page layout capability** in the **fourth increment**.





# The Incremental model

- When to use?
  - The initial software requirements are well defined, but the overall scope of the development is unclear.
  - there may be a compelling need to provide a limited set of software functionality to users quickly and then refine and expand on that functionality in later software releases



# The Incremental model

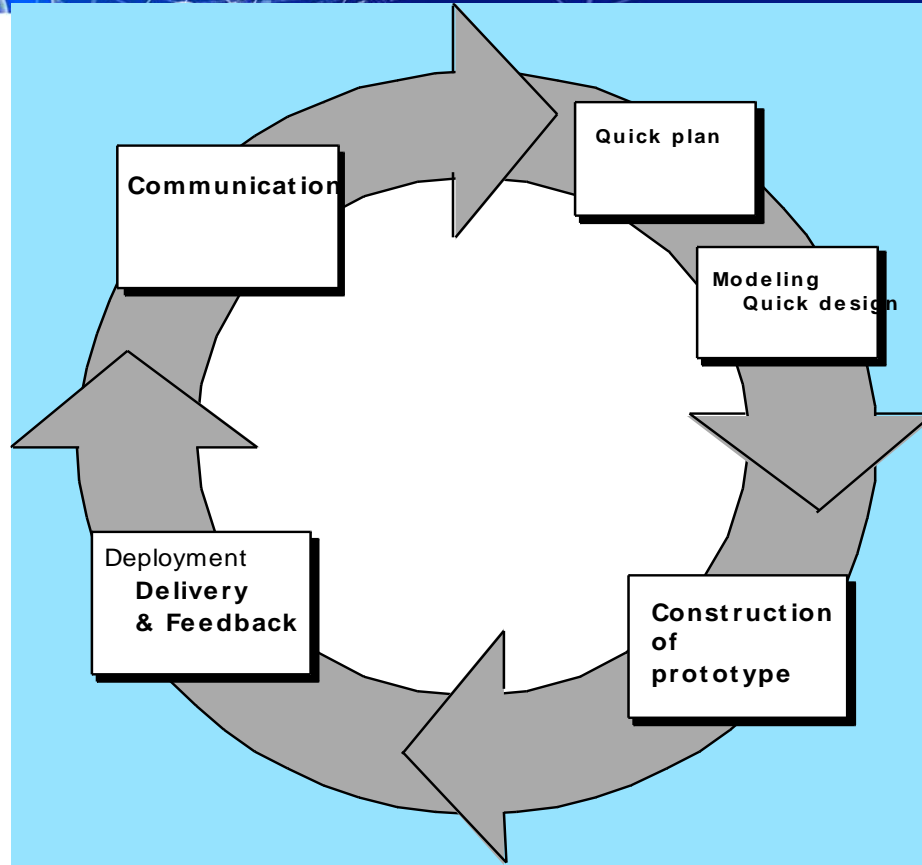
- why to use?
  - *Your customer demands delivery by a date that is impossible to meet. Suggest delivering one or more increments by that date and the rest of the software (additional increments) later.*
  - *Manage technical risks(Ex: major system might require the availability of new hardware that is under development, plan early increments in a way that avoids the use of this hardware, enabling partial functionality to be delivered to end users without inordinate delay).*



# Evolutionary process Models

- When to use?
  - Business and product requirements often change as development proceeds, making a straight line path to an end product unrealistic
- Evolutionary process models produce an increasingly more complete version of the software with each iteration.

# Prototyping





# Prototyping

- **When to use?**
  - customer defines a set of general objectives for software, but does not identify detailed requirements for functions and features.
  - the prototyping paradigm assists you and other stakeholders to better understand what is to be built when requirements are fuzzy.
  - prototyping can be used as a stand-alone process model, it is more commonly used as a technique that can be implemented within the context of any one of the process models



# Prototyping

- Serves as a mechanism for identifying software requirements.
- Build a pilot system and throw it away.

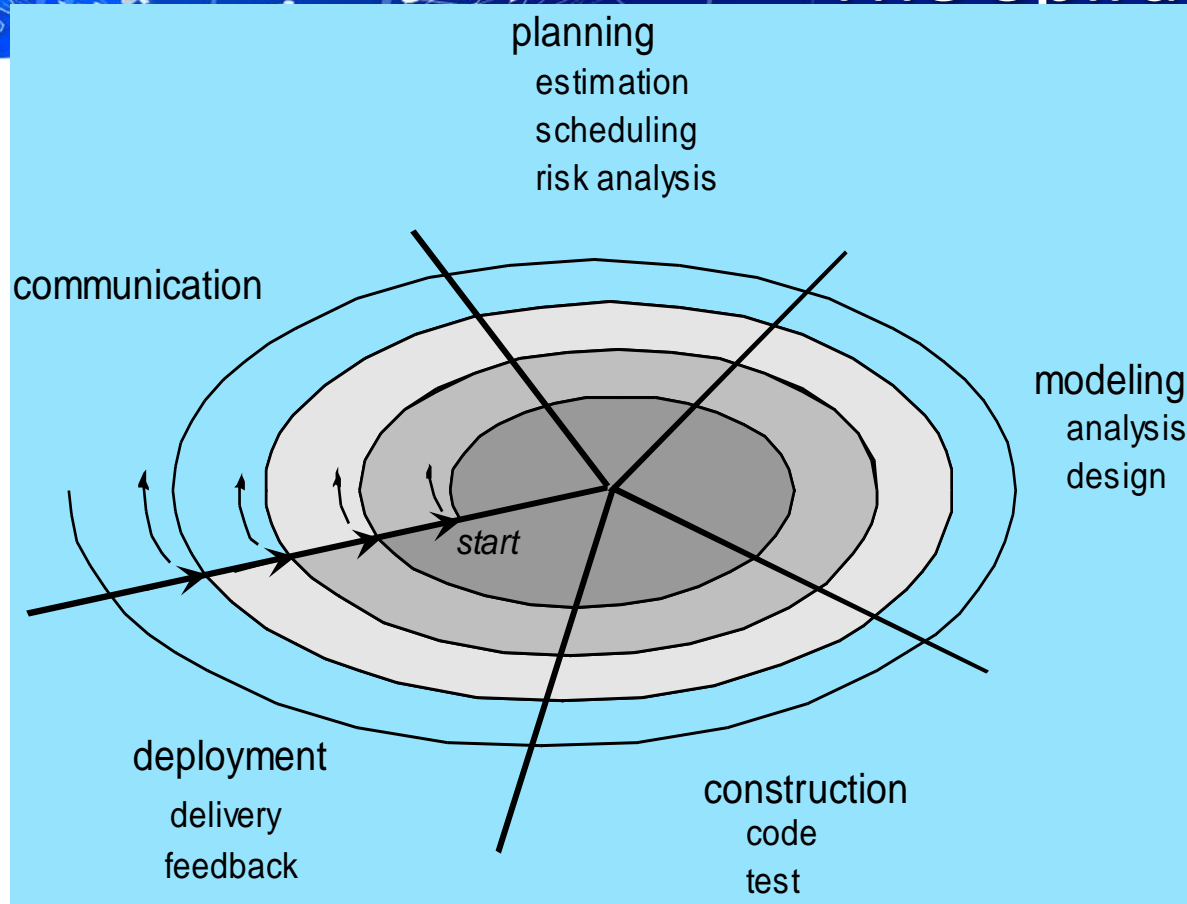




# Problems of prototyping model

1. Stakeholders see what appears to be a working version of the software, unaware that the prototype is held together haphazardly, unaware that in the rush to get it working you haven't considered overall software quality or long-term maintainability. When informed that the product must be rebuilt so that high levels of quality can be maintained, stakeholders cry foul and demand that "a few fixes" be applied to make the prototype a working product. Too often, software development management relents.
2. As a software engineer, you often make implementation compromises in order to get a prototype working quickly. An inappropriate operating system or programming language may be used simply because it is available and known; an inefficient algorithm may be implemented simply to demonstrate capability. After a time, you may become comfortable with these choices and forget all the reasons why they were inappropriate. The less-than-ideal choice has now become an integral part of the system.

# The Spiral Model





# The Spiral Model

- Combines the waterfall model with the prototype model.
- Each phase in spiral model begins with a design goal and ends with the client reviewing the progress.
- starts with a small set of requirement and goes through each development phase for those set of requirements. Then adds functionality for the additional requirement in every-increasing spirals until the application is ready for the production phase.



# When to use Spiral Methodology?

1. When project is large
2. When releases are required to be frequent
3. When creation of a prototype is applicable
4. When risk and costs evaluation is important
5. For medium to high-risk projects
6. When requirements are unclear and complex
7. When changes may require at any time
8. When long term project commitment is not feasible due to changes in economic priorities



# Advantages of spiral model


1. Additional functionality or changes can be done at a later stage
2. Cost estimation becomes easy as the prototype building is done in small fragments
3. Continuous or repeated development helps in risk management
4. Development is fast and features are added in a systematic way
5. There is always a space for customer feedback



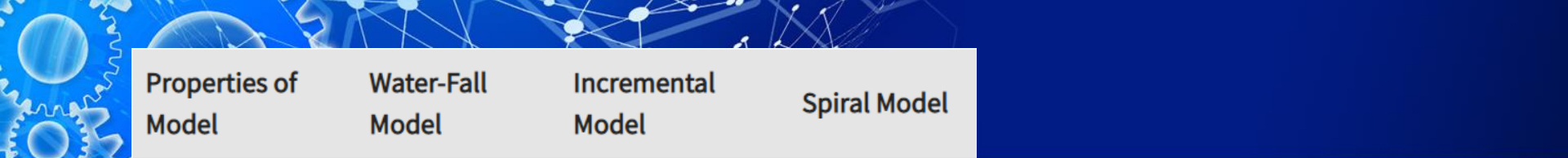
# Disadvantages of spiral model

1. Risk of not meeting the schedule or budget
2. It works best for large projects only also demands risk assessment expertise
3. For its smooth operation spiral model protocol needs to be followed strictly
4. Documentation is more as it has intermediate phases
5. It is not advisable for smaller project, it might cost them a lot





Properties of Model	Water-Fall Model	Incremental Model	Spiral Model
Planning in early stage	Yes	Yes	Yes
Returning to an earlier phase	No	Yes	Yes
Handle Large-Project	Not Appropriate	Not Appropriate	Appropriate
Detailed Documentation	Necessary	Yes but not much	Yes
Cost	Low	Low	Expensive
Requirement Specifications	Beginning	Beginning	Beginning



Properties of Model	Water-Fall Model	Incremental Model	Spiral Model
Flexibility to change	Difficult	Easy	Easy
User Involvement	Only at beginning	Intermediate	High
Maintenance	Least	Promotes Maintainability	Typical
Duration	Long	Very long	Long
Risk Involvement	High	Low	Medium to high risk
Framework Type	Linear	Linear + Iterative	Linear + Iterative
Testing	After completion of coding phase	After every iteration	At the end of the engineering phase

Properties of Model	Water-Fall Model	Incremental Model	Spiral Model
Maintenance	Least Maintainable	Maintainable	Yes
Re-usability	Least possible	To some extent	To some extent
Time-Frame	Very Long	Long	Long
Working software availability	At the end of the life-cycle	At the end of every iteration	At the end of every iteration
Objective	High Assurance	Rapid Development	High Assurance
Team size	Large Team	Not Large Team	Large Team
Customer control over administrator	Very Low	Yes	Yes



# Still Other Process Models

- **Component based development**—the process to apply when reuse is a development objective
- **Formal methods**—emphasizes the mathematical specification of requirements
- **Unified Process**—a “use-case driven, architecture-centric, iterative and incremental” software process closely aligned with the Unified Modeling Language (UML)