软件工程

张爽

东北大学软件学院

9.1

Software Life-Cycle Models

Iterative and Incremental Models

- **In an ideal world, a software product is developed phase by phase clearly, from requirements, analysis, design to implementation.**
- **However, it is considerably different in practice for 2 reasons:**
  - **Software professionals are humans and therefore make mistakes.**
  - **The client's requirements can change while the software is being developed.**

Iterative and Incremental Models

- **As a consequence of both the moving-target problem and the need to correct the inevitable mistakes made while a software product is being developed, the actual life cycle of software production can not be an ideal process.**

Iterative and Incremental Models

- **Iteration**
  - **We produce the first version of the artifact, then we revise it and product the second version, and so on.**
  - **Each version is closer to our target than its predecessor, and finally we construct a version that is satisfactory.**

Iterative and Incremental Models

- **Incrementation**
  - **We divides the target software product into builds / artifacts.**
  - **We concentrate on those aspects that are currently the most important and postpone until later those aspects that are currently less critical.**
  - **Then, we consider further aspects of the problem and add the resulting new pieces to the existing artifact.**

Iterative and Incremental Models

- **Incrementation**
  - **For example, we might construct a requirements document by considering the 7 requirements we consider the most important. Then, we would consider the 7 next most important requirements, and so on.**
  - **This is an incremental process.**

Iterative and Incremental Models

- **In practice, iteration and incrementation are used in conjunction with one another.**
- **An artifact is constructed piece by piece (incrementation), and each increment goes though multiple versions (iteration).**

Iterative and Incremental Models

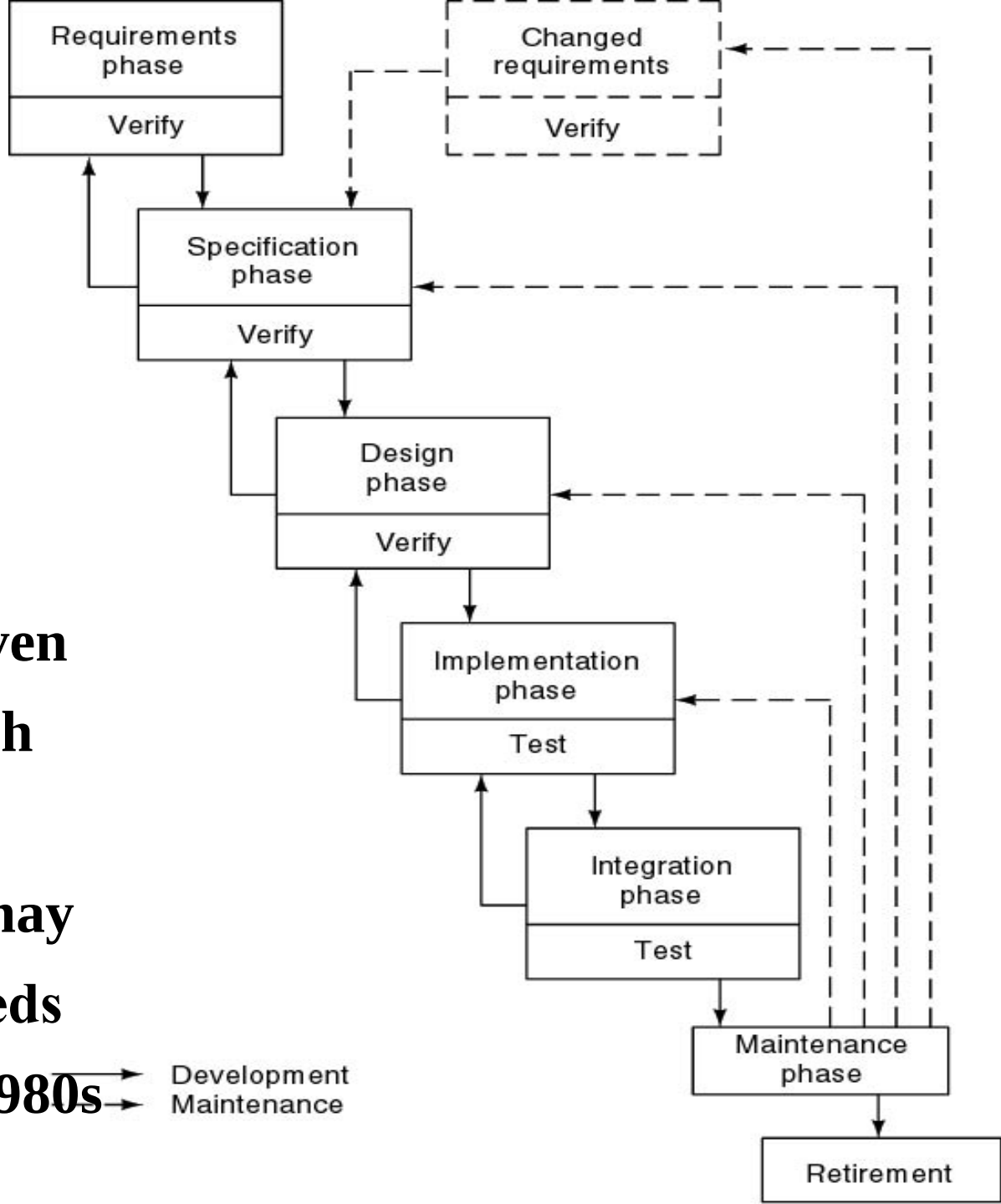
- **Planning and documentation activities are performed throughout the iterative-and-incremental life cycle.**
- **Testing is a major activity during each iteration, and particularly at the end of each iteration.**
- **The software as whole is thoroughly tested once it has been completed.**

Iterative and Incremental Models

- For each increment in turn, the requirements, analysis, design, and implementation phases (in that order) are repeatedly performed on that increment until it is clear the no further iteration is needed.
- The project is broken up into a series of waterfall mini projects.

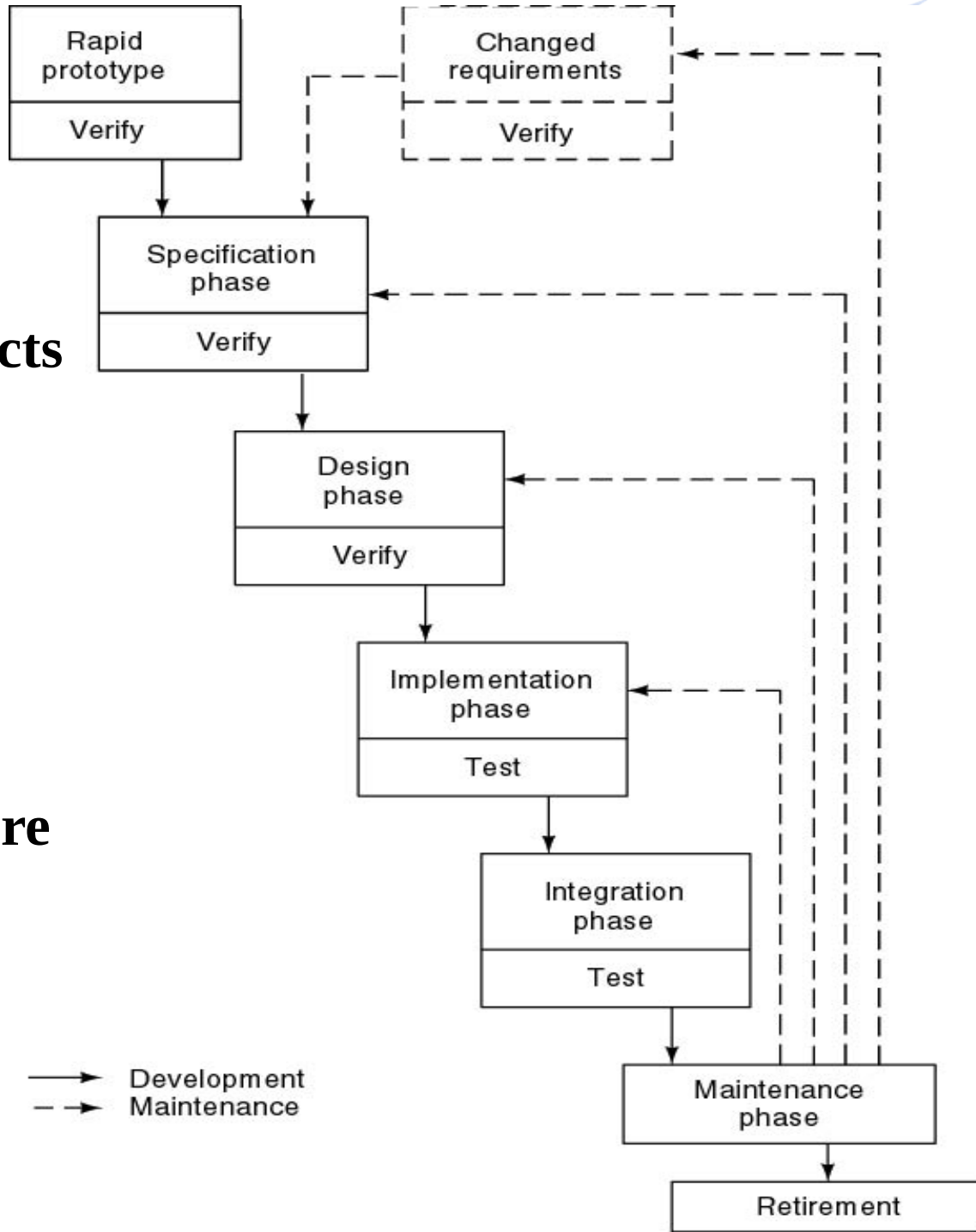
Waterfall Model

- Characterized by
  - Feedback loops
  - Documentation
- Advantages
  - Documentation-driven
  - Disciplined approach
- Disadvantages
  - Delivered product may not meet client's needs
- Widely used before 1980s



Rapid Prototyping Model

- A **rapid prototype** reflects the functionality to be provided to the future users, such as input screens and reports.
- Let the clients and future users to interact and experiment with it.
- No feedback-loops



Rapid Prototyping Model

- **Strength**
  - **Ensures that the delivered product meets the client's needs.**
- **Weaknesses**
  - **Not yet proven beyond all doubt.**
- **Rapid prototyping may replace the specification phase, but never the design phase.**

Synchronize-and-Stabilize Model

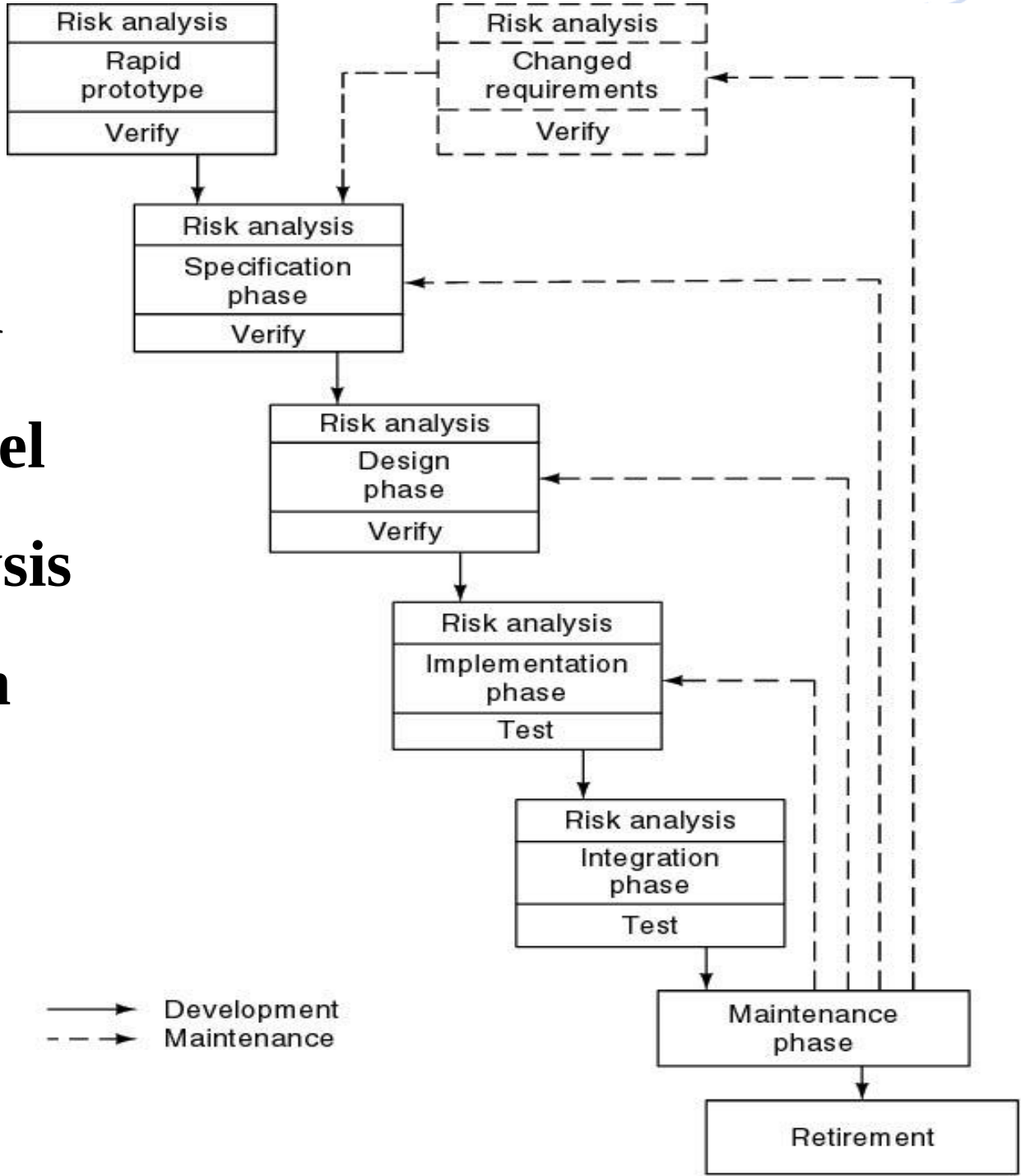
- **Microsoft's life-cycle model**
- **Requirements analysis — interview potential customers**
- **Draw up specifications**
- **Divide project into 3 or 4 builds**
- **Each build is carried out by small teams working in parallel**
- **At the end of the day — synchronize (test and debug)**
- **At the end of the build — stabilize (freeze build)**

Synchronize-and-Stabilize Model

- **Strengths**
  - **Future users' needs are met.**
  - **Ensures that the components can be successfully integrated.**
- **Weaknesses**
  - **Has not been widely used other than at Microsoft**

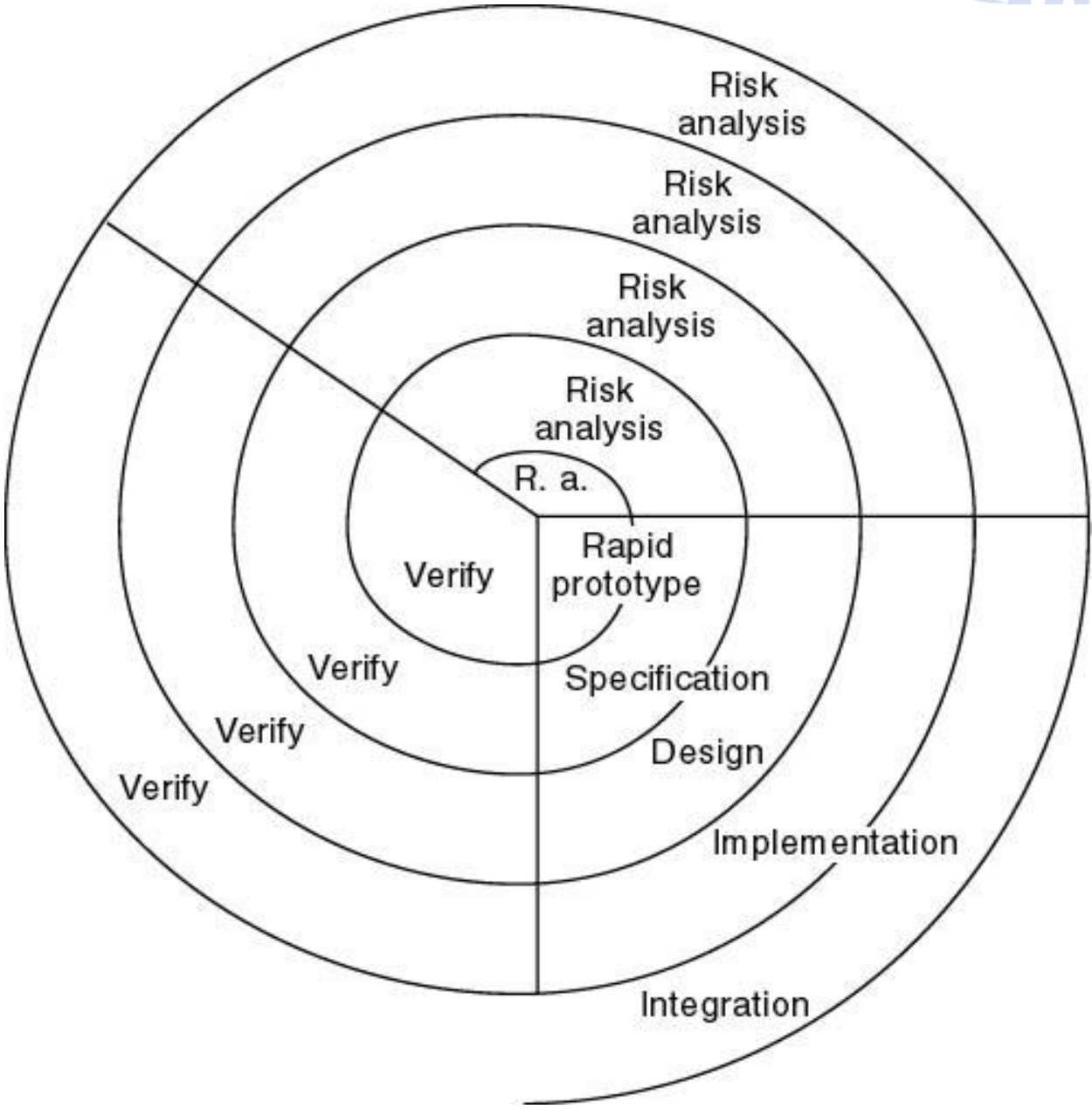
Spiral Model

- **Simplified form**
  - **Waterfall model plus risk analysis preceding each phase**



Spiral Model

- **View of spiral**
- **If all risks cannot be resolved, the project is immediately terminated.**



Spiral Model

- **Strengths**
  - **Risk driven**
- **Weaknesses**
  - **For large-scale internal (in-house) software only**
  - **Developers have to be competent in risk analysis and risk resolution.**

Conclusions

- Different life-cycle models
  - Each with its own strengths
  - Each with its own weaknesses
- Criteria for deciding on a model include
  - The organization
  - Its management
  - Its employees
  - The nature of the product
- Best suggestion
  - “Mix-and-match” life-cycle model