Chapter 4
Process Models

Software Engineering: A Practitioner's Approach
by Roger S. Pressman and Bruce R. MAXIM

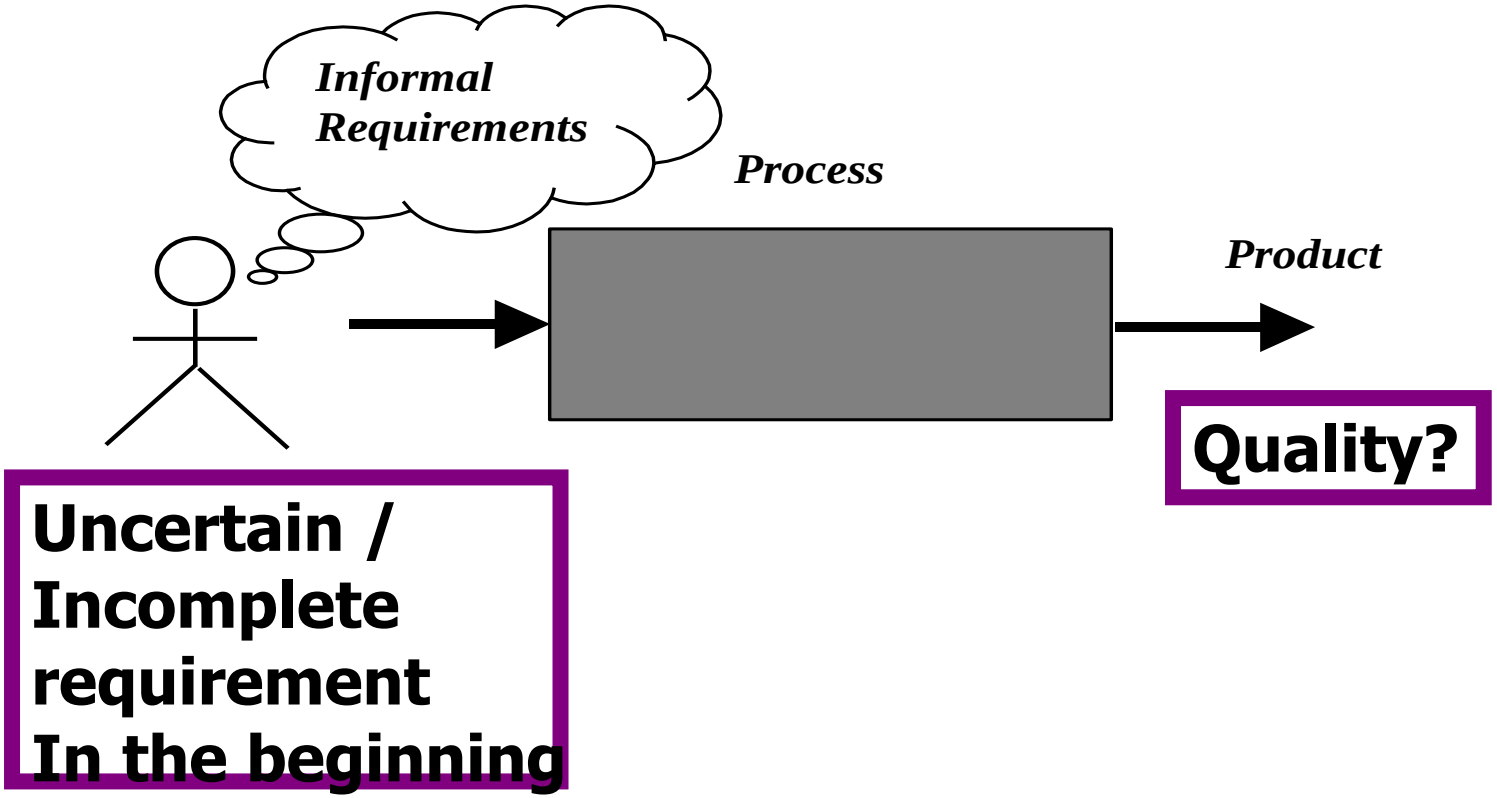
Software process model

- Attempt to organize the software life cycle by
  - defining activities involved in software production
  - order of activities and their relationships
- Goals of a software process
  - standardization, predictability, productivity, high product quality, ability to plan time and budget requirements

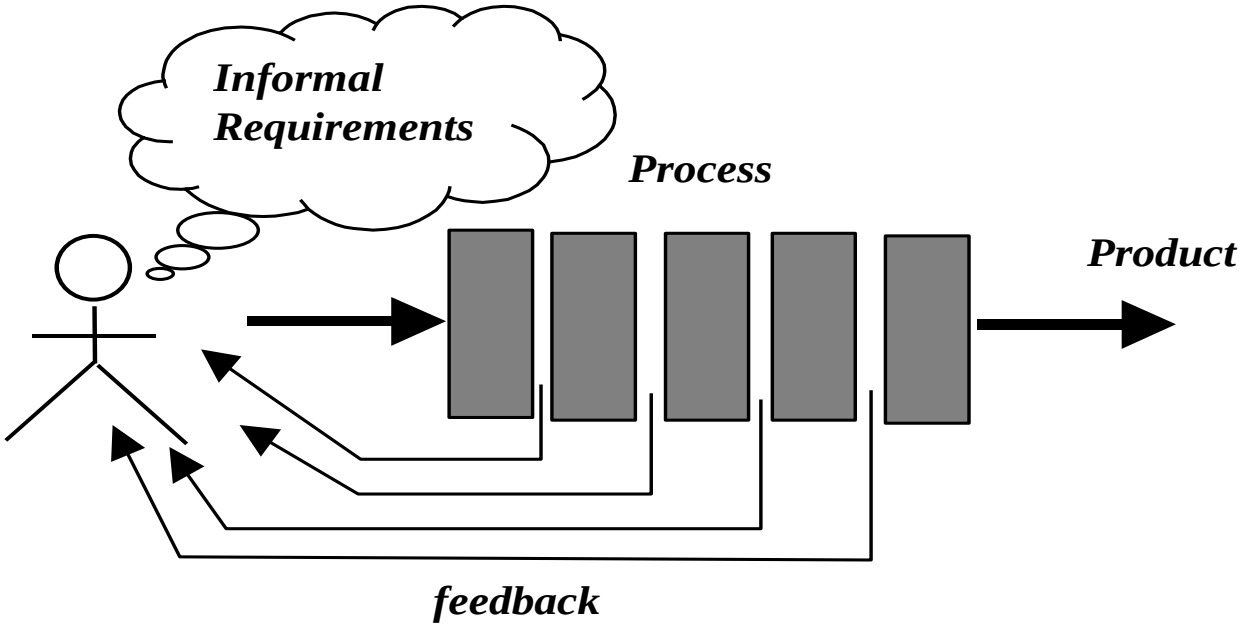
Models are needed

- Symptoms of inadequacy: the software crisis
  - scheduled time and cost exceeded
  - user expectations not met
  - poor quality
- The size and economic value of software applications required appropriate "process models"

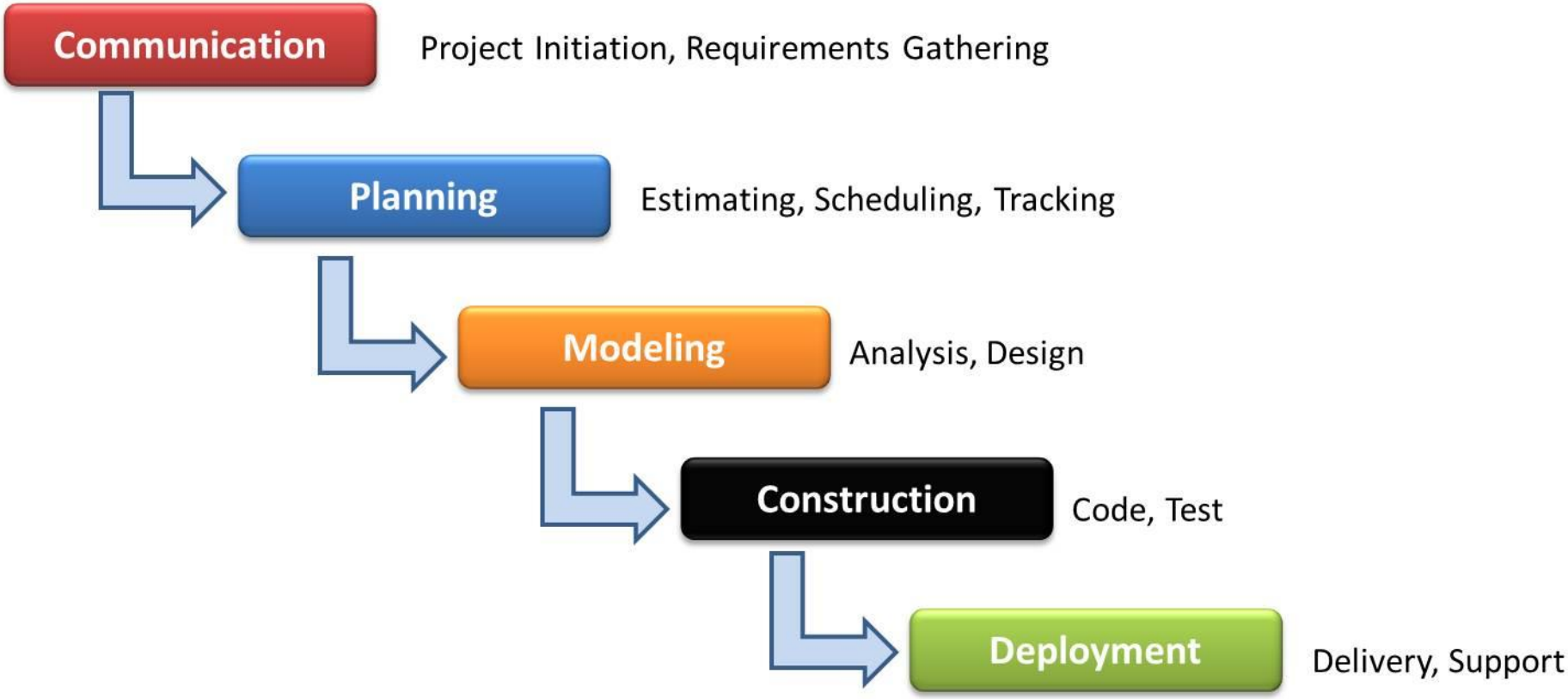
Process as a "black box"



Process as a "white box"



The Waterfall Model



The Waterfall Model: A Traditional Approach of SDLC

Waterfall Model Assumptions

1. It is also referred to as a **linear-sequential life cycle model**.
2. In a waterfall model, each phase must be completed fully before the next phase can begin.
3. This type of model is basically used for the for the small and large project which is and there are no uncertain requirements.
4. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project.
5. In this model the testing starts only after the development is complete.
6. In **waterfall model phases** do not overlap.

- Among the problems that are sometimes encountered when the waterfall model is applied are:

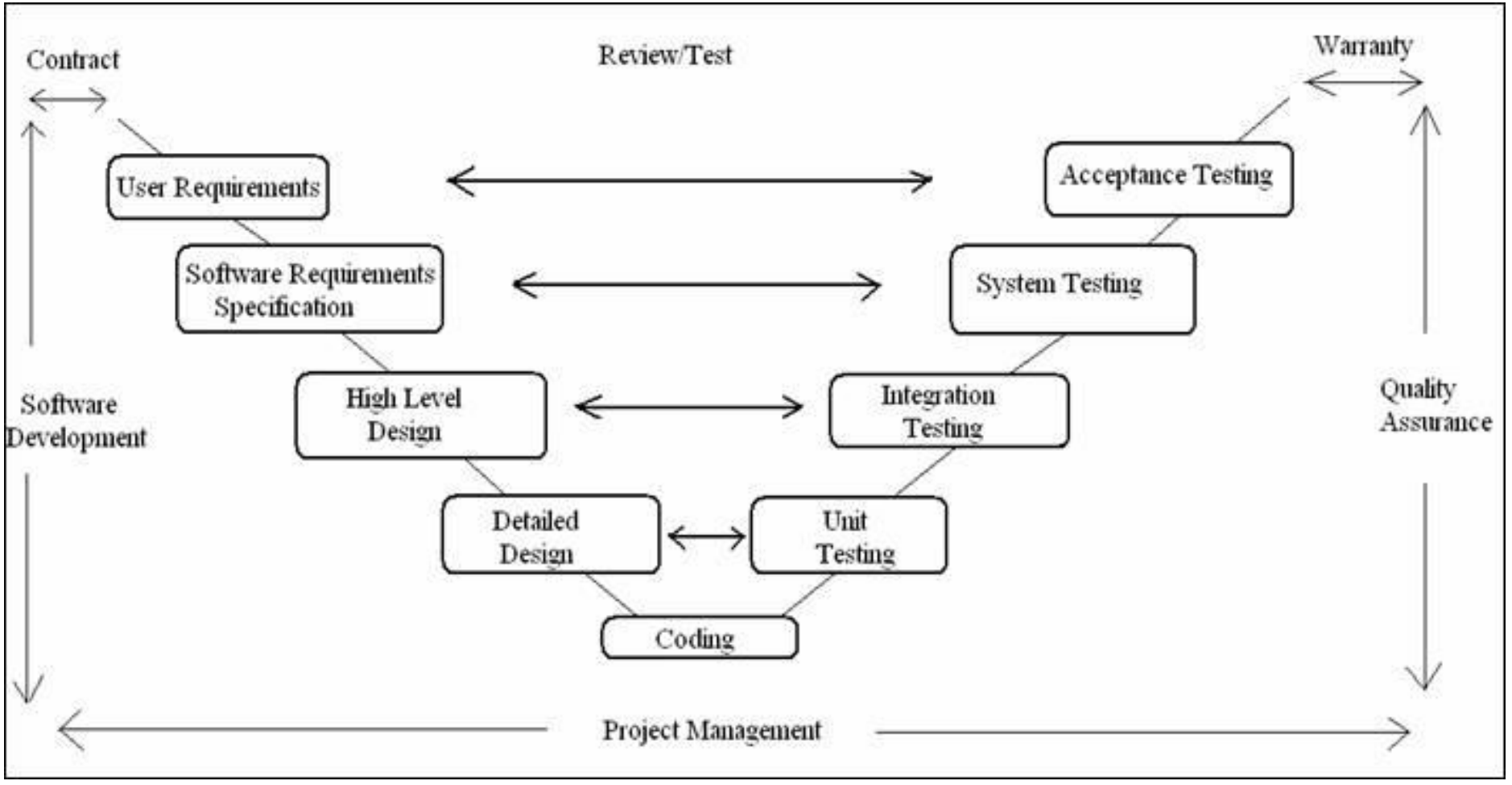
1. Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.

2. It is often difficult for the customer to state all requirements explicitly. The waterfall model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.

3. The customer must have patience. A working version of the program(s) will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed, can be disastrous.

The V Model

- If we rely on testing alone, defects created first are detected last



- The V model is an extension of waterfall model.
- The V-model provides a way of visualizing how verification and validation actions are applied to earlier engineering work.
- As stated, left arm of V model is conventional waterfall model of the development and right arm is for corresponding testing phases. Each verification activity (such as requirement specification verification, functional Design verification etc.) has its corresponding validation activity (such as Functional validation/testing, code validation/testing, System/integration validation etc.)
- The major purpose of acceptance testing is to validate /confirm that system meets business requirements and provides confidence before it is delivered to the end user.
- This testing involves user representative and testing team members.
- Once the entire system has been developed then it has to be tested against the “system specification” to validate if it delivers the features required. In short, system testing is not about checking the individual parts of the design, but about checking the system as a whole.

When to use the V-model:

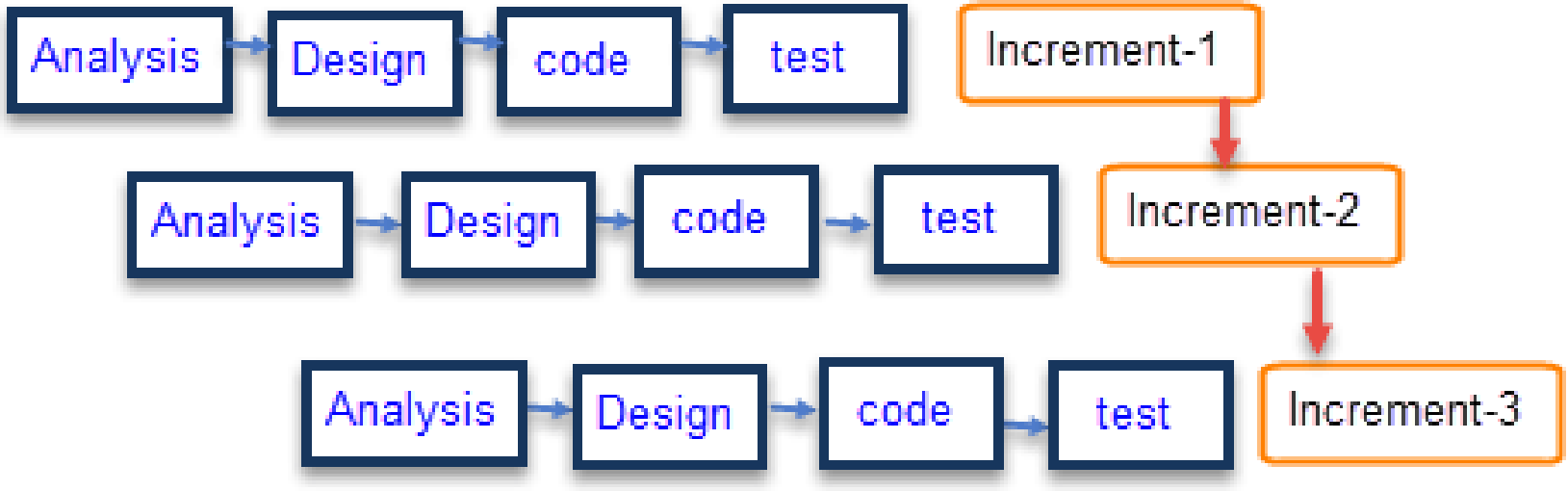
The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.

The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

High confidence of customer is required for choosing the V-Shaped model approach. Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations .

- In an interesting analysis of actual projects, it is found that 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.
- In fact, the time spent waiting can exceed the time spent on productive work! The blocking states tend to be more prevalent at the beginning and end of a linear sequential process.
- Today, software work is fast-paced and subject to a never-ending stream of changes (to features, functions, and information content). The waterfall model is often inappropriate for such work.
- However, it can serve as a useful process model in situations where requirements are fixed and work is to proceed to completion in a linear manner.

Incremental Models: Incremental



Incremental Model

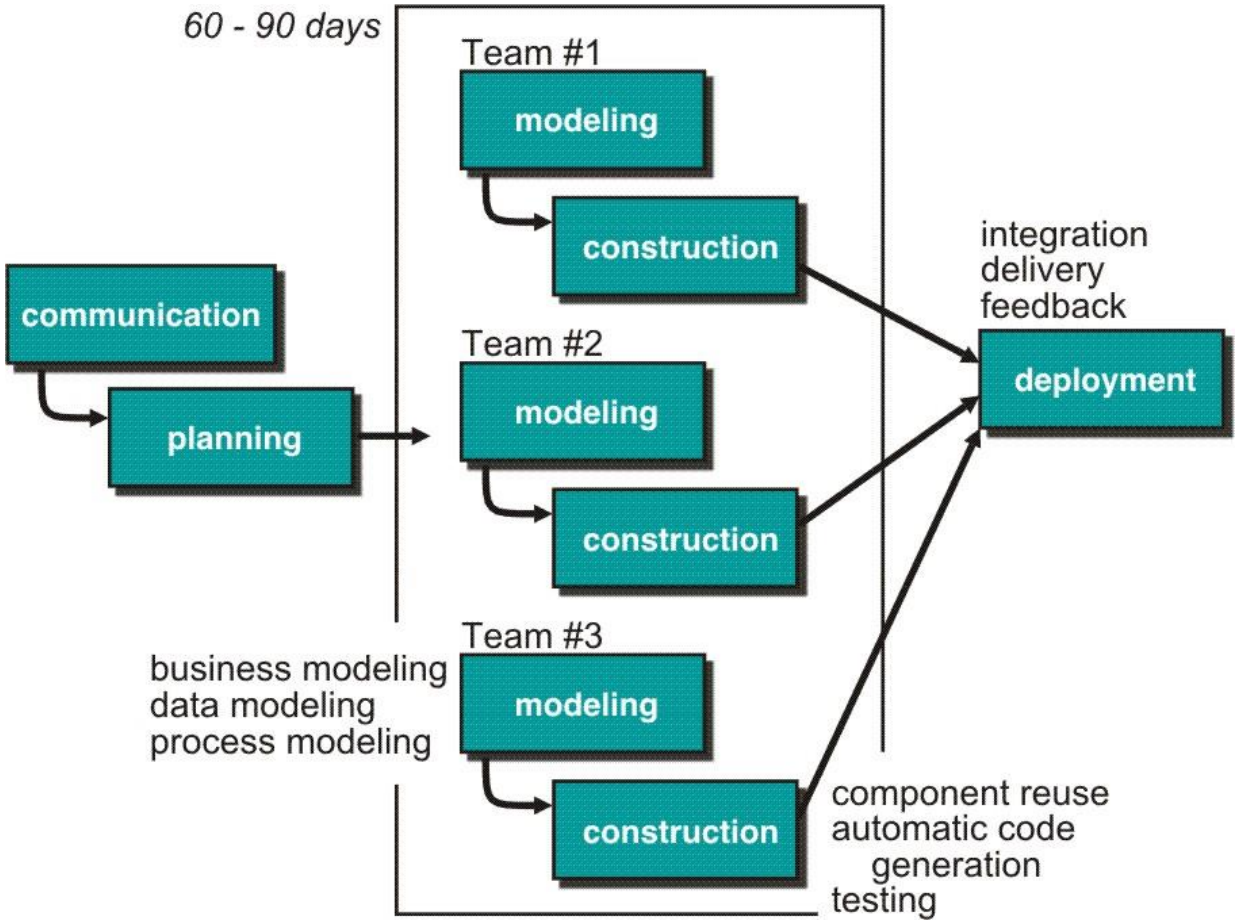
- The system is put into production when the first increment is delivered.
- The first increment is often a core product where the basic requirements are addressed, and supplementary features are added in the next increments.
- Once the core product is analyzed by the client, there is plan development for the next increment.
- System development is broken down into many mini development projects
- Partial systems are successively built to produce a final total system
- Highest priority requirement is tackled first

Total cost is higher than waterfall.

When to use the Incremental model:

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
Resources with needed skill set are not available
There are some high risk features and goals.

Incremental Models: RAD Model



- **Business modeling:** The information flow is identified between various business functions.
  Data modeling: Information gathered from business modeling is used to define data objects that are needed for the business.
  Process modeling: Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.
  Application generation: Automated tools are used to convert process models into code and the actual system.
  Testing and turnover: Test new components and all the interfaces.

.

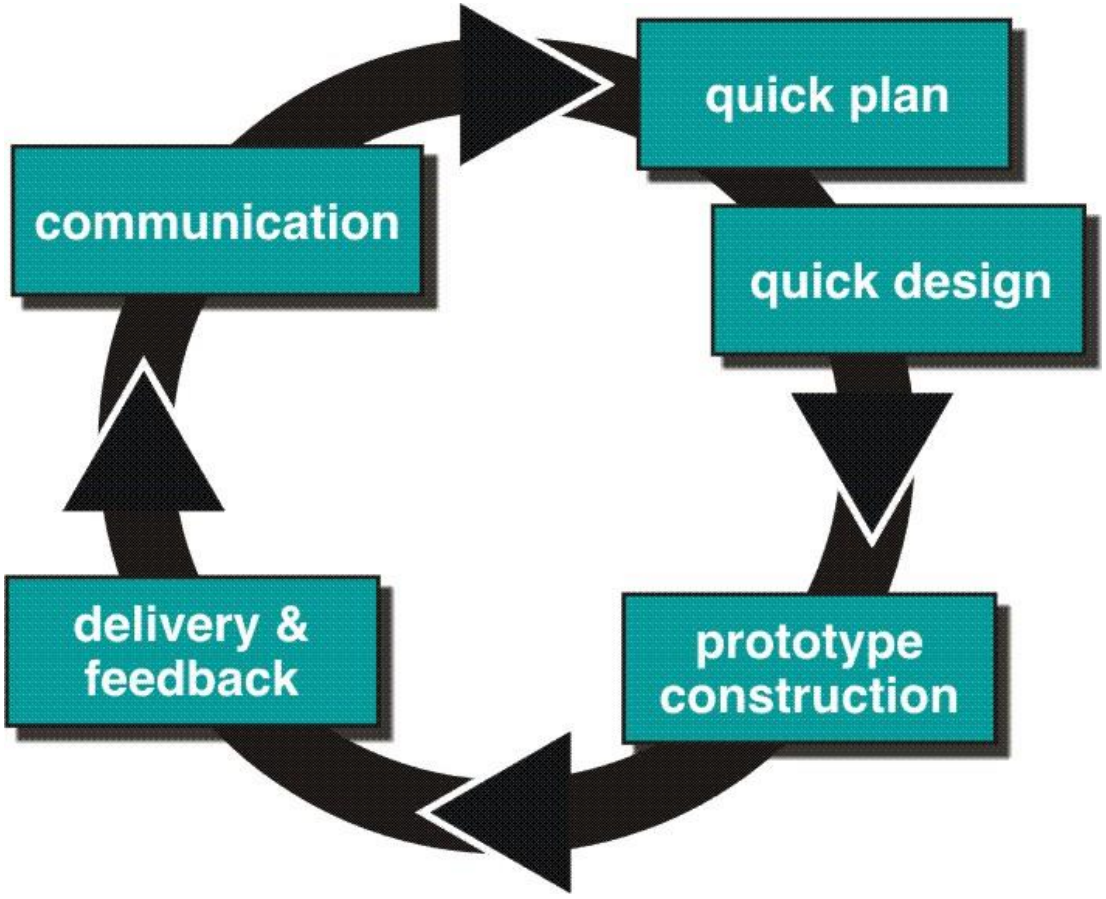
When to use RAD model:

RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.

It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.

RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

Evolutionary Models: Prototyping



Need for Prototype in software development:

- There are several uses of a prototype. An important purpose is to illustrate the input data formats, messages, reports, and the interactive dialogues to the customer. This is a valuable mechanism for gaining better understanding of the customer's needs:
  - how the screens might look like
  - how the user interface would behave
  - how the system would produce outputs

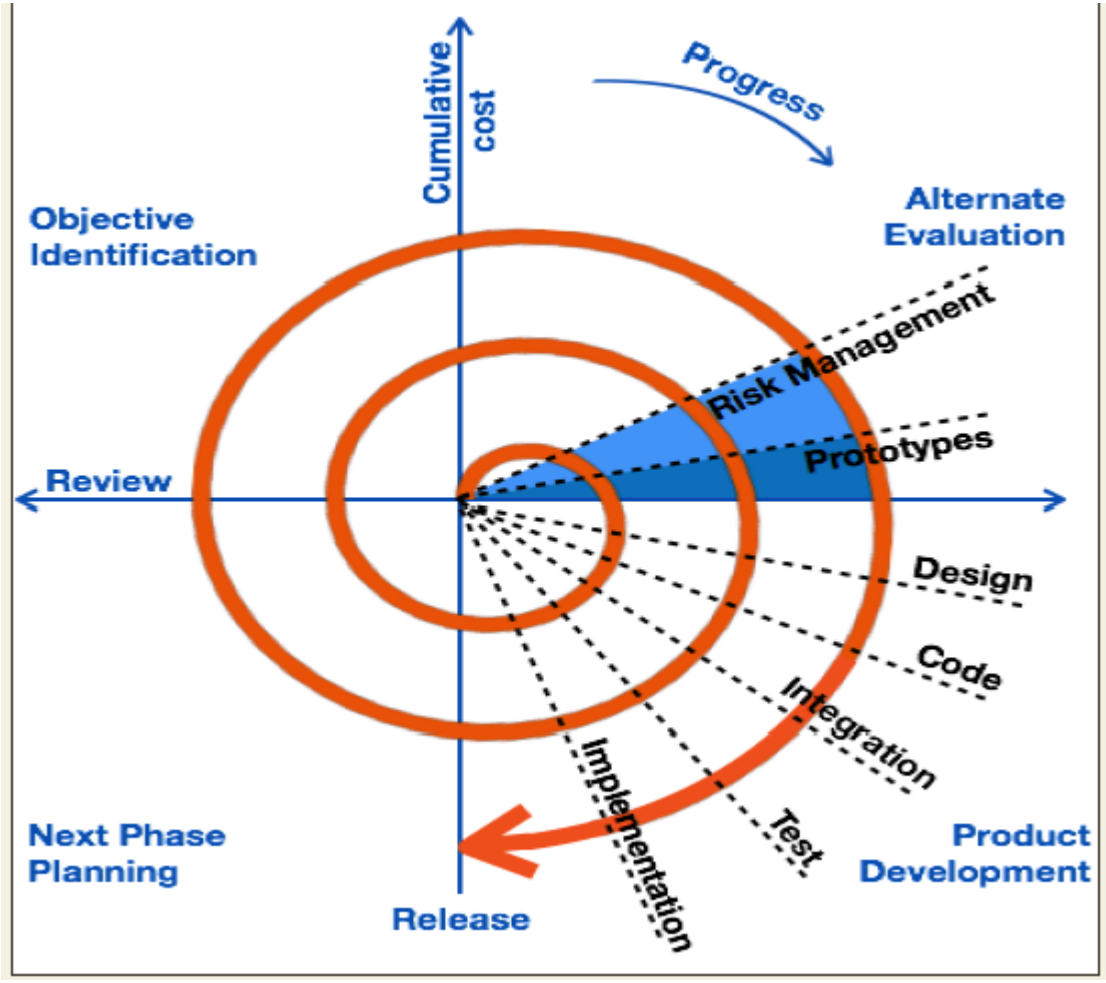
- Another reason for developing a prototype is that it is impossible to get the perfect product in the first attempt.
- Many researchers and engineers advocate that if you want to develop a good product you must plan to throw away the first version.
- The experience gained in developing the prototype can be used to develop the final product.

- A prototyping model can be used when technical solutions are unclear to the development team.
- A developed prototype can help engineers to critically examine the technical issues associated with the product development.
- Often, major design decisions depend on issues like the response time of a hardware controller, or the efficiency of a sorting algorithm, etc. In such circumstances, a prototype may be the best or the only way to resolve the technical issues.

A prototype of the actual product is preferred in situations such as:

- User requirements are not complete
- Technical issues are not clear

Evolutionary Models: The Spiral



- The diagrammatic representation of this model appears like a spiral with many loops.
- The exact number of loops in the spiral is not fixed.
- Each loop of the spiral represents a phase of the software process. For example, the innermost loop might be concerned with feasibility study, the next loop with requirements specification, the next one with design, and so on.
- Each phase in this model is split into four sectors (or quadrants) as shown in fig.. The following activities are carried out during each phase of a spiral model

First quadrant (Objective Setting)

• During the first quadrant, it is needed to identify the objectives of the phase.

• Examine the risks associated with these objectives.

Second Quadrant (Risk Assessment and Reduction)

• A detailed analysis is carried out for each identified project risk.

• Steps are taken to reduce the risks. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

Third Quadrant (Development and Validation)

- • Develop and validate the next level of the product after resolving the identified risks.

Fourth Quadrant (Review and Planning)

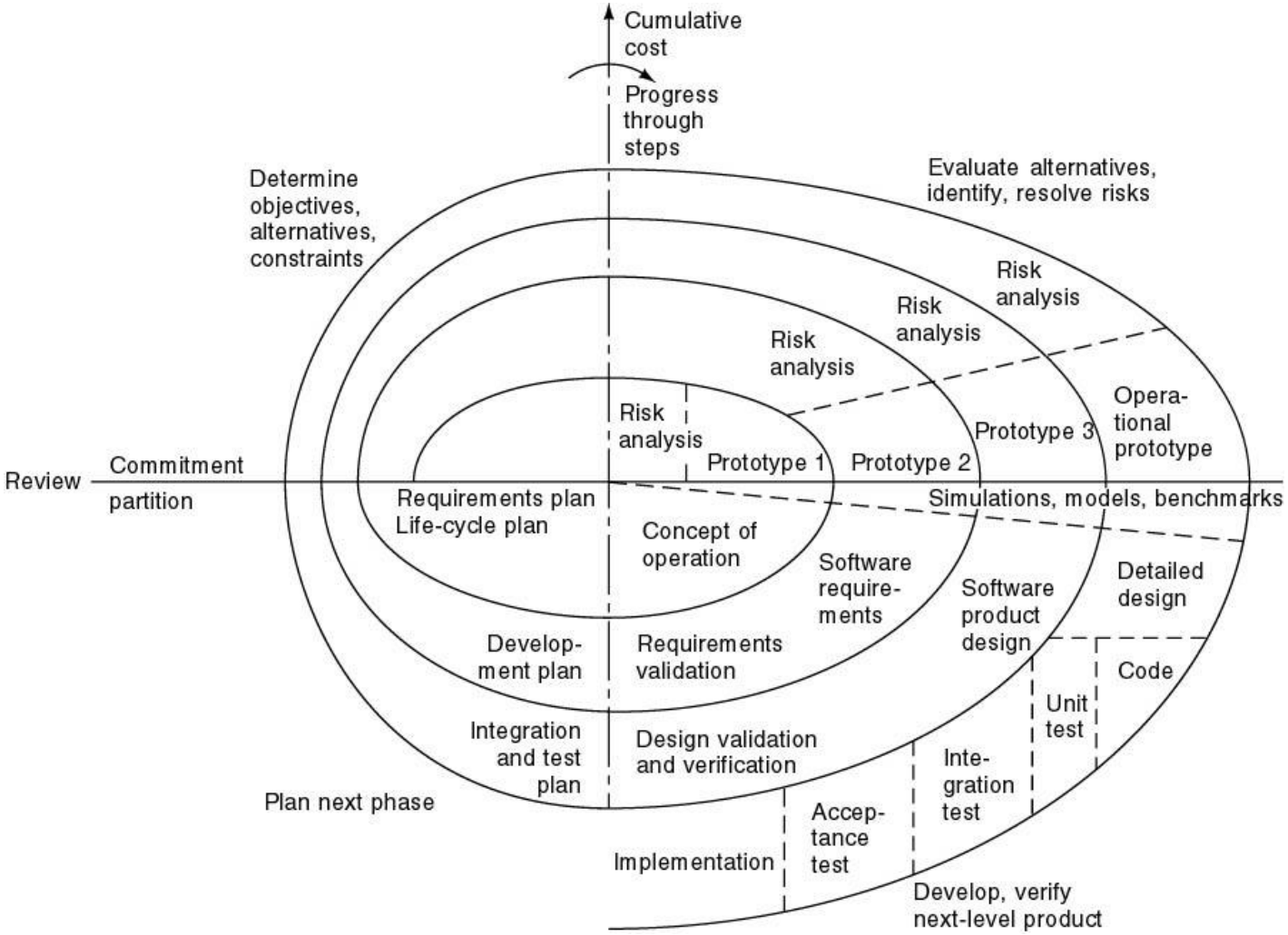
• Review the results achieved so far with the customer and plan the next iteration around the spiral.

• Progressively more complete version of the software gets built with each iteration around the spiral.

- Radial dimension: cumulative cost to date
- Angular dimension: progress through the spiral

The radial dimension in Figure represents the cumulative cost incurred in accomplishing the steps to date; the angular dimension represents the progress made in completing each cycle of the spiral.

Full Spiral Model (contd)



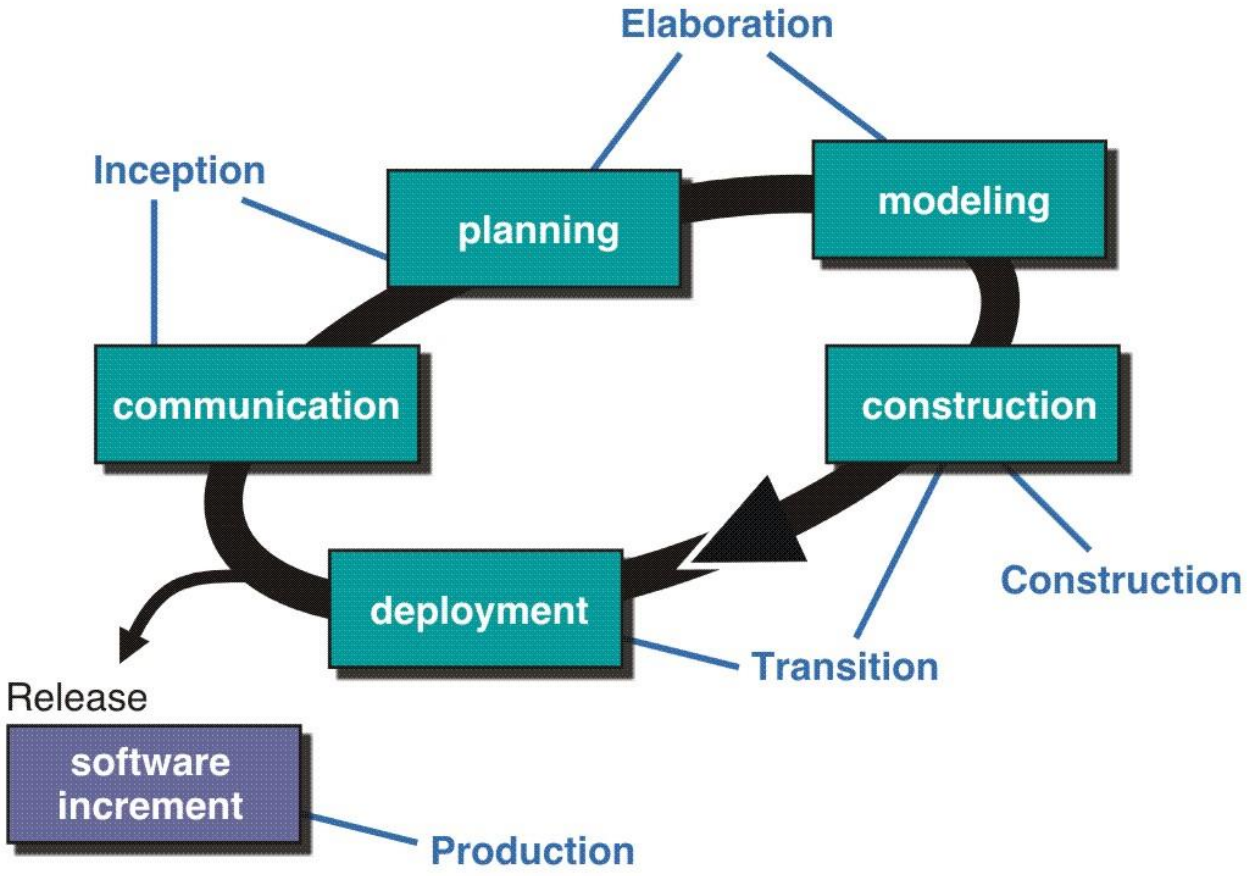
Unified Process Model

A software process that is:

- use-case driven
- architecture-centric
- iterative and incremental

Closely aligned with the
Unified Modeling Language (UML)

The Unified Process (UP)



Inception Phase

The following are typical goals for the Inception phase.

- Establish a justification or business case for the project
- Establish the project scope and boundary conditions
- Outline the use cases and key requirements that will drive the design tradeoffs
- Outline one or more candidate architectures
- Identify risks
- Prepare a preliminary project schedule and cost estimate
- Feasibility

Elaboration Phase

- The primary goals of Elaboration are to address known risk factors and to establish and validate the system architecture. Common processes undertaken in this phase include the creation of use case diagrams, conceptual diagrams (class diagrams with only basic notation) and package diagrams (architectural diagrams).
- The architecture is validated primarily through the implementation of an Executable Architecture Baseline. This is a partial implementation of the system which includes the core, most architecturally significant, components. It is built in a series of small, time boxed iterations.

Construction Phase

Construction is the largest phase in the project. In this phase the remainder of the system is built on the foundation laid in Elaboration. System features are implemented in a series of short, time boxed iterations. Each iteration results in an executable release of the software.

Transition Phase

The final project phase is Transition. In this phase the system is deployed to the target users. Feedback received from an initial release (or initial releases) may result in further refinements to be incorporated over the course of several Transition phase iterations