Software Engineering (CSE 415)

Process Models

Process Model

- A software process model is a simplified representation of a software process, presented from a specific perspective.

- A software process model is an abstract representation of a software process.

Process Model

- Attempt to organize the software life cycle by
  - defining activities involved in software production
  - order of activities and their relationships

- Every process model must be adapted so that it is used effectively for a specific software project.

Prescriptive Process Models

- Prescriptive software process models have been applied for many years in an effort to bring order and structure to software development.
- Each of these conventional models suggests a somewhat different process flow, but all perform the same set of *generic framework activities –*
  - Communication
  - Planning
  - Modeling
  - Construction
  - deployment

Prescriptive Process Models

- A prescriptive process model populates a process framework with explicit task sets for software engineering actions.
- Prescriptive process models advocate an orderly approach to software engineering.
- Prescriptive Process Models are NOT static, they should be adapted to the –
  - People
  - Problem
  - Project

Prescriptive Process Models

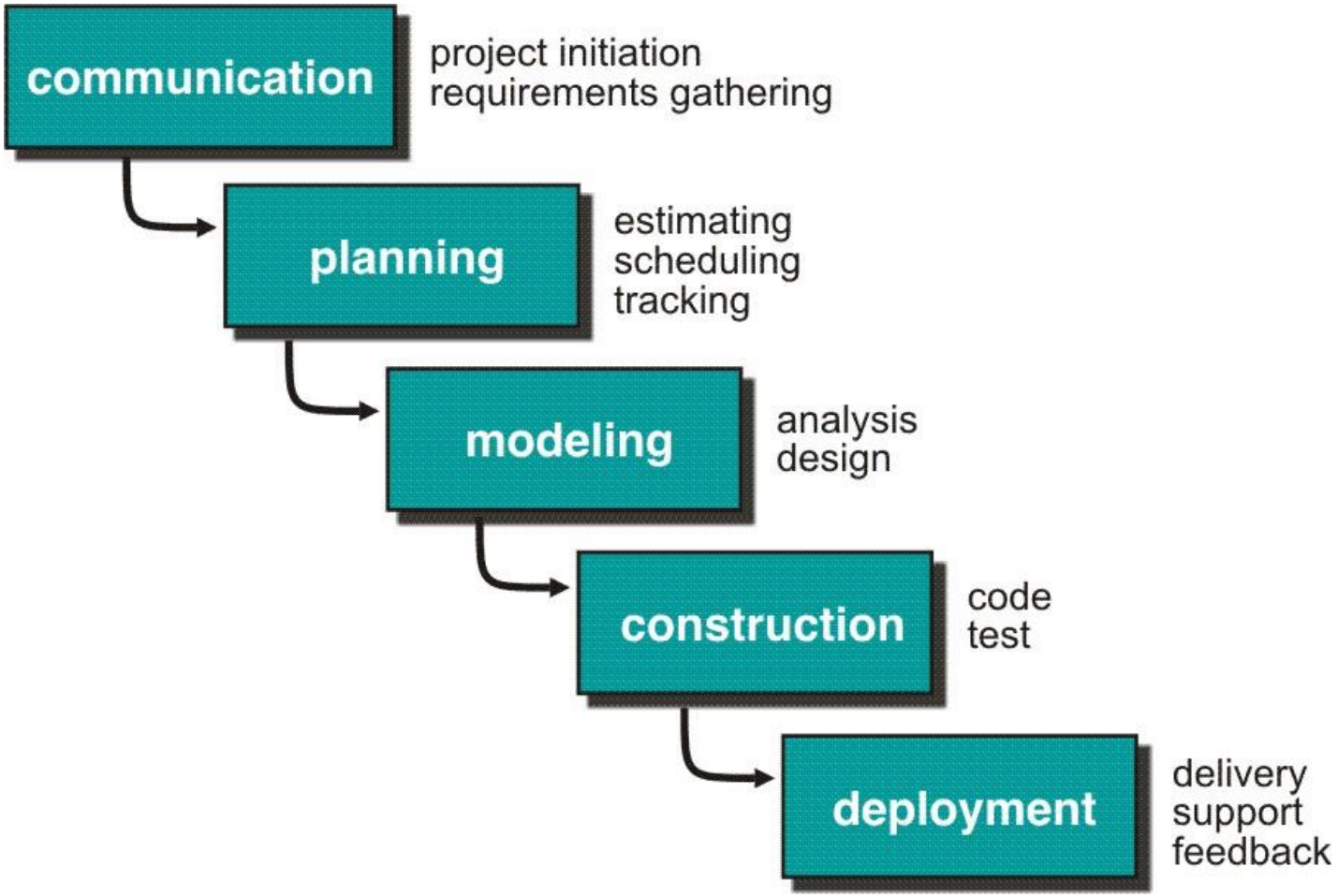
- The Waterfall Model
- Incremental Process Models
- Evolutionary Process Models

The Waterfall Model

- The oldest paradigm for software engineering.
- The Waterfall model, sometimes called the *classic life cycle*, suggests a systematic, sequential approach to software development.
- One phase begins when another completes.
- Useful process model in situations where requirements are well-defined and stable, and work is to be proceed to completion in a linear manner.

The Waterfall Model

The Waterfall Model



Advantages of the Waterfall model

- Identifies systems requirements long before programming begins.
- Minimized changes to the requirements as the project proceeds.

Problems of the Waterfall model

- The Waterfall model suggests a linear progression of framework activities that is often inconsistent with modern realities(e.g. continuous change, evolving systems, tight timelines). Real projects rarely follow the sequential flow that the model proposes.
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. One phase has to be completed before moving onto the next phase.
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - It is often difficult for the customer to state all requirements explicitly
  - The customer must be patient

Incremental Process Models

- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
- User requirements are prioritized and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.
- Incremental software process models produce software as a series of increment releases.
  - **The Incremental Model**
  - **The RAD model**

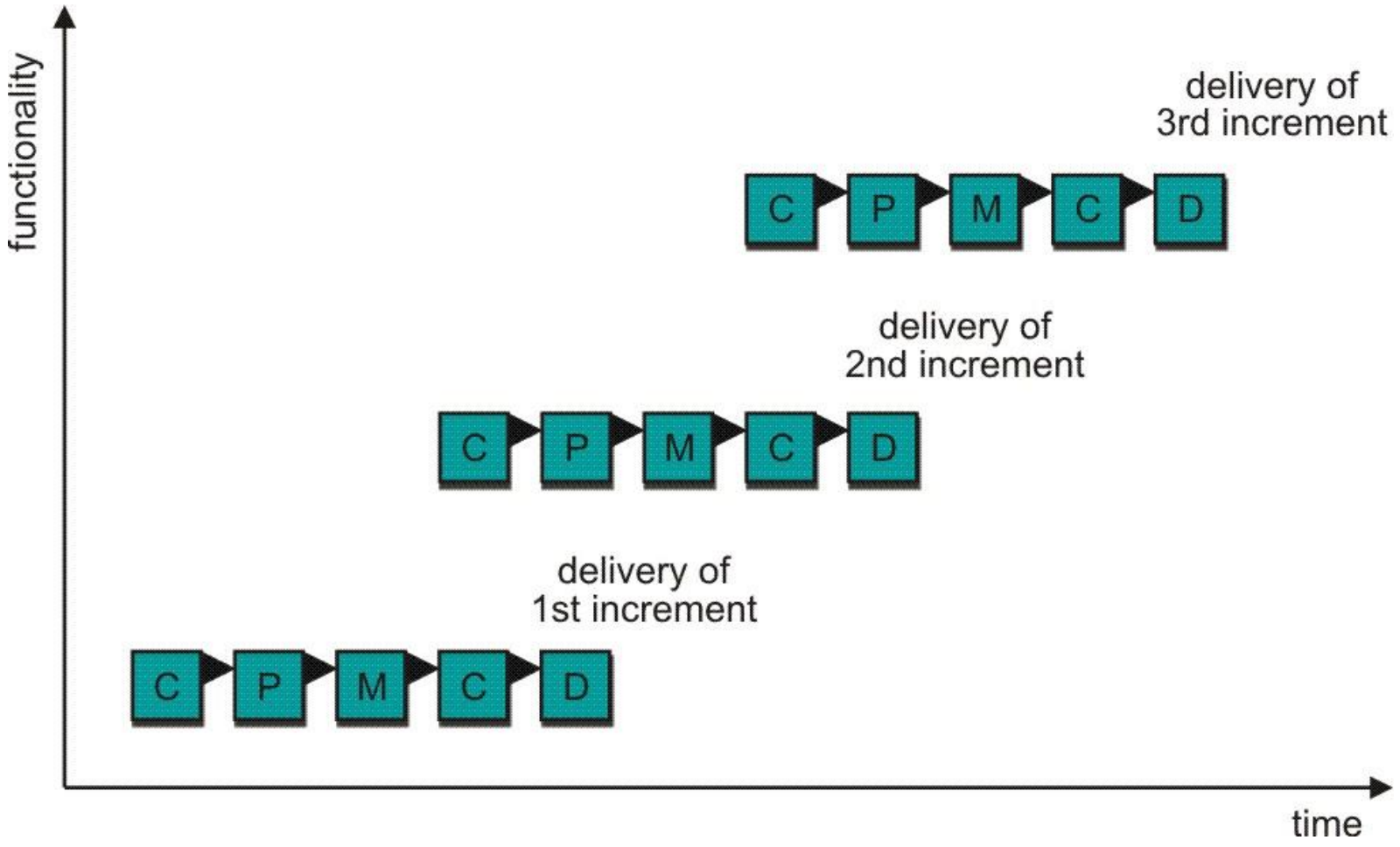
The Incremental Model

- The Incremental Model combines elements of the Waterfall model applied in an iterative fashion.
- Delivers a series of releases , called *increments*, that provide progressively more functionality for the customer as each increment is delivered.
  - Deliver the core product first
  - Add on / refine features
  - Provide a platform for evaluation by user

The Incremental Model

- The incremental process model, like prototyping and other evolutionary approaches, is iterative in nature.
- Unlike prototyping, the incremental model focuses on the delivery of an operational product with each increment. Early increments are “stripped down” versions of the final product.
- Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.
  - Early increments can be implemented with fewer people.
  - If core product is well received, additional staff can be added to implement the next increment.
  - Increments can be planned to manage technical risks

The Incremental Model



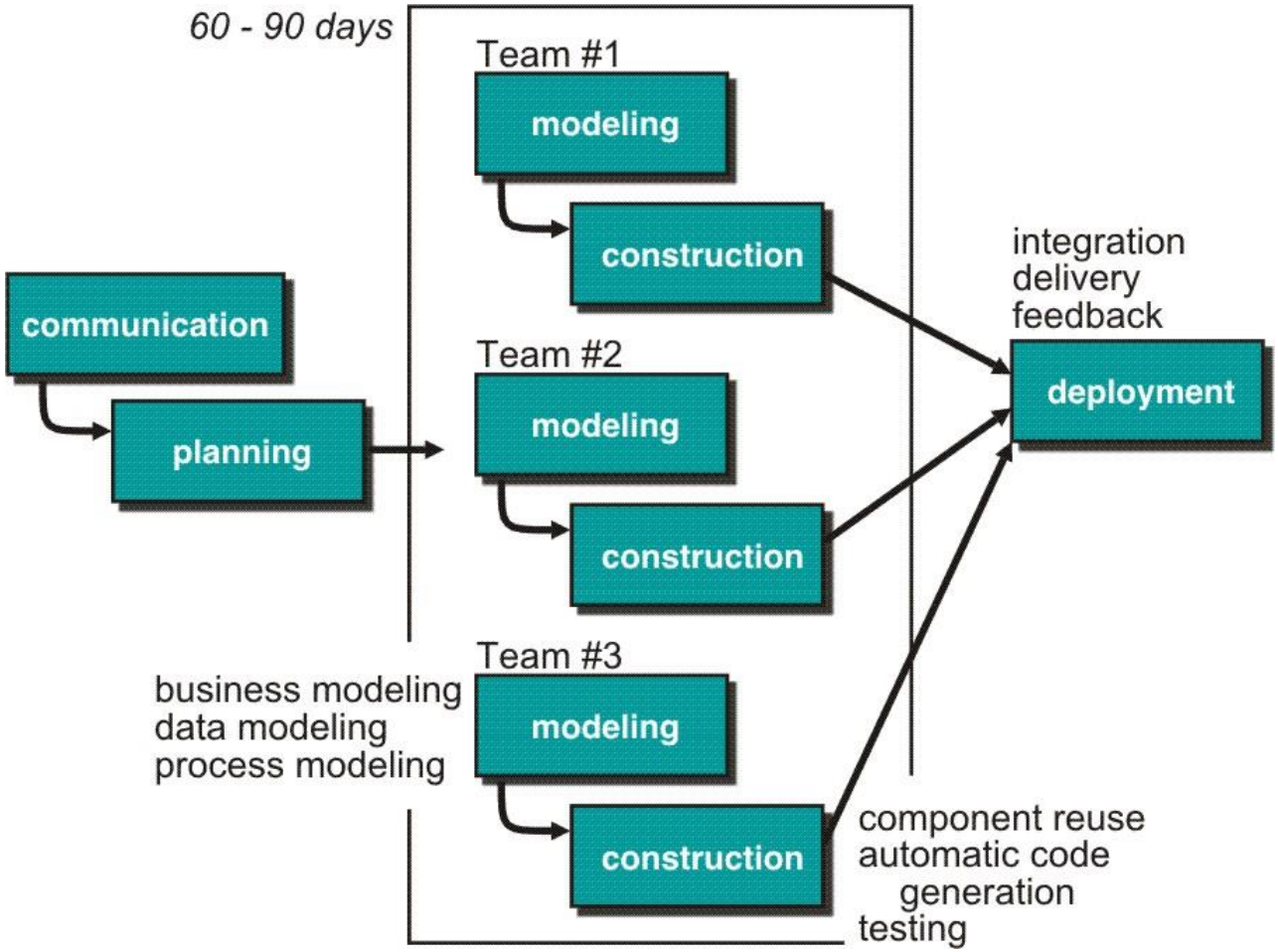
Advantages of Incremental Model

- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

The RAD Model

- Rapid Application Development (RAD) model
- Incremental software process model that emphasizes a short development cycle (as little as 60-90 days)
- “High-speed” adaptation of the Waterfall model
- Rapid development is achieved by using a component-based construction approach
- Each major function handled by separate RAD team, and then integrated to form a whole
- RAD model is designed for larger projects that must be delivered in tight time frames

The RAD Model



Drawbacks of the RAD model

- For large projects, RAD requires sufficient human resources (to create right number of RAD teams).
- RAD projects fails in the event if developers & customers are not committed to rapid-fire activities within short time frame
- If a system cannot be properly modularized, building the components necessary for RAD will be problematic
- Not suitable when technical risks are high, or when high performance needs to be achieved through tuning.

Evolutionary Process Models

- Evolutionary Process Models produce an increasingly more complete version of the software with each iteration.
- Evolutionary models are iterative and are designed to accommodate change.
- Produce incremental work products(or working version of the software) quickly.
- Based on the idea of developing an initial implementation, exposing this to user comment & refining it through many versions until an adequate system has been developed.

Evolutionary Process Models

- These models can be adopted to apply across all software engineering activities –from concept development to long-term system maintenance.
  - Prototyping
  - Spiral development

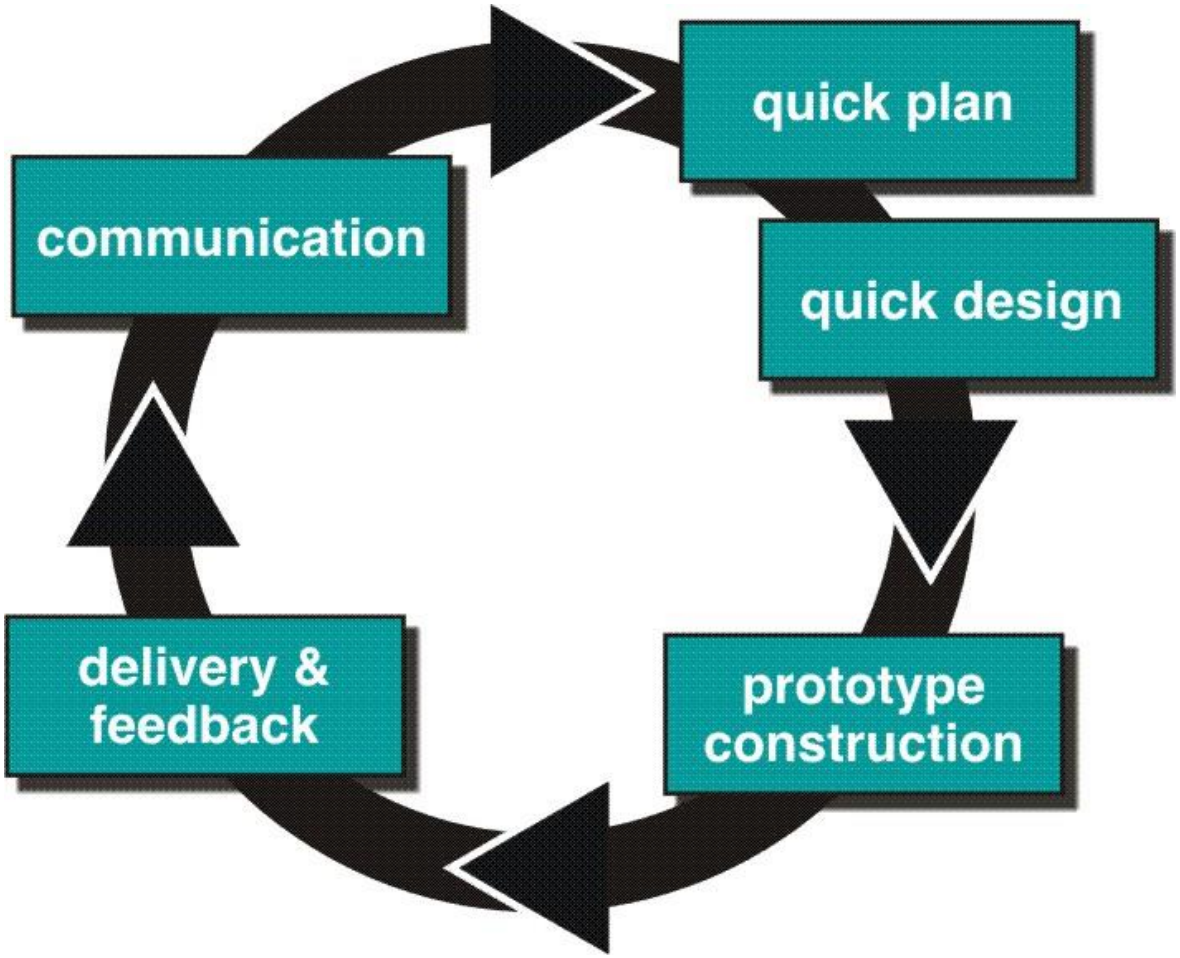
Prototyping

- Designing and building a scaled-down but functional version of a desired system is known as *prototyping*.
- Prototyping is an iterative process of systems development in which requirements are converted to a working system that is continually revised through close collaboration between an analyst and users.
  - Requirements quickly converted to a working system
  - System is continually revised
  - Close collaboration between users and analysts
- Start with a "quick-and-dirty" prototype (that provides minimal functionality), repeat process, refining the prototype each time, stop when prototype is a working system.

Prototyping

- Objective is to understand the customer's requirements & hence develop a better requirements definition for the system.
- Should start with poorly understood requirements to clarify what is really needed.
  - The prototype serves as a mechanism for identifying software requirements

Prototyping



Prototyping: Pros & Cons

Pros	Cons
Gets working system to users quickly	Fast paced. Hard to conduct careful, methodical analysis
Reassures users that the project is progressing	Initial design decisions have long term staying power
Quickly refines true requirements	Problems may come to light late in design, requiring re-design

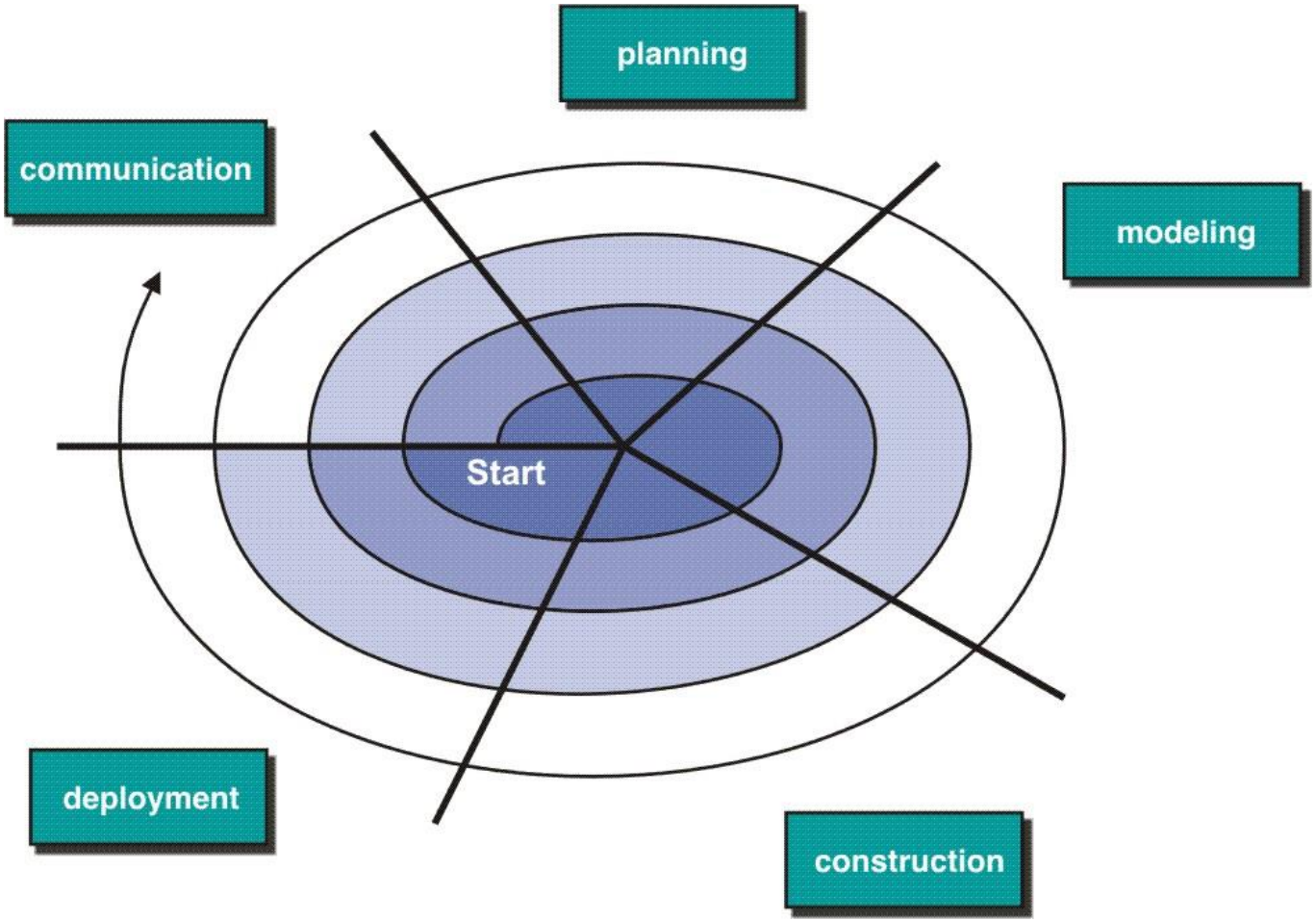
The Spiral Model

- An evolutionary software process model that couples the iterative nature of prototyping with the controlled & systematic aspects of the Waterfall model.
- Unlike other process models that end when software is delivered, the spiral model can be adapted to apply throughout the life of the computer software.
- Realistic approach to the development of large-scale software.
- Software is developed in a series of evolutionary releases.

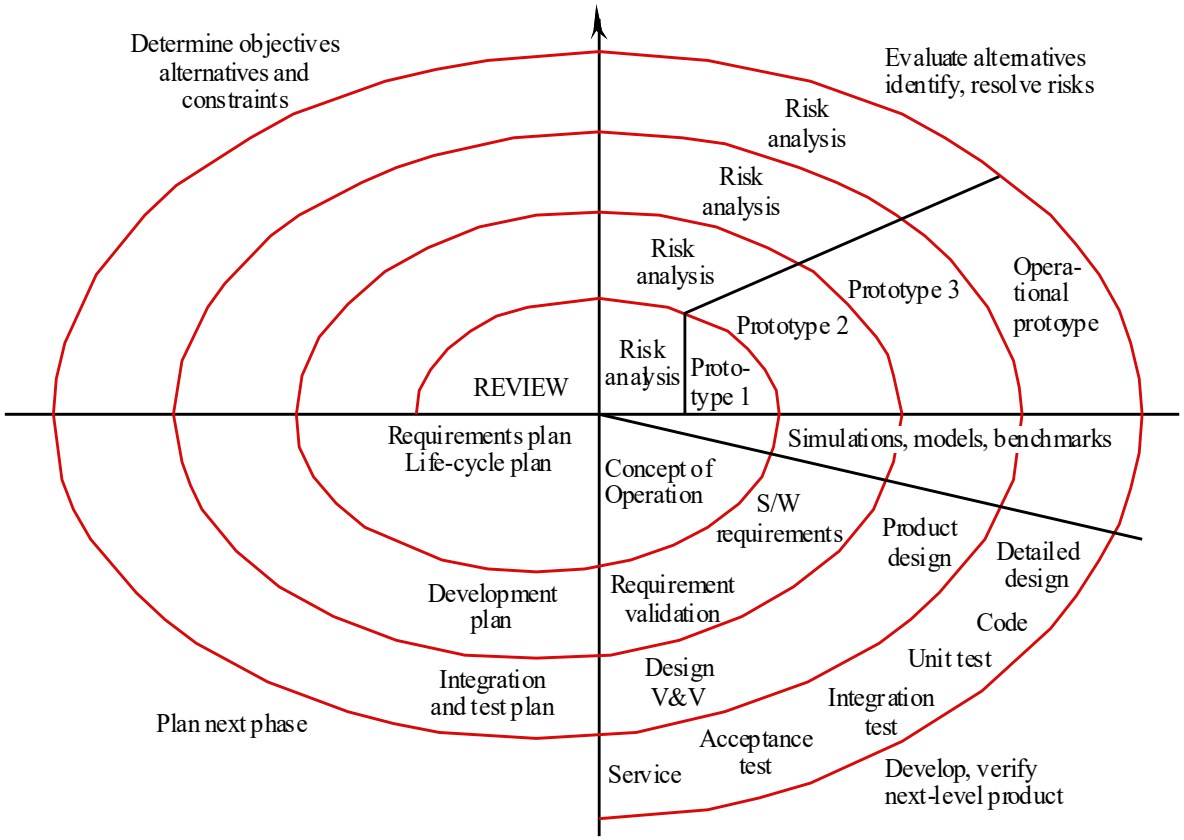
The Spiral Model

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- *Each loop* in the spiral represents a *phase* in the process.
- No fixed phases – loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.

The Spiral Model



The Spiral Model



The Spiral model sectors

- Each loop in the spiral is split into four sectors:

1) Objective setting
   - Specific objectives for the phase are identified

2) Risk assessment and reduction
   - Risks are assessed and activities put in place to reduce the key risks

3)Development and validation
   - A development model for the system is chosen which can be any of the generic models

4) Planning
   - The project is reviewed
   - A decision is made whether to continue with a further loop of the spiral
   - If decided to continue, plans are drawn up for the next phase of the project

Weaknesses of the Spiral model

- Only suitable for large-scale software
- Only suitable for internal (in-house) software
- Not suitable for fixed-budget development
- It may be difficult to convince customers that the evolutionary approach is controllable.
- Demands considerable risk assessment expertise & relies on this expertise for success.
- Projects fail if major risks are not uncovered & managed properly.

Specialized Process Model

- Component-Based Development Model
- Formal methods Model
- Aspect-Oriented Model

Specialized Process Model

- Component-Based Development Model
  - Component-Based Model emphasizes component reuse and assembly
  - COTS software components can be used when software is to be built. These components provide targeted functionality with well-defined interfaces that enable the component to be integrated into the software.
  - Component-based development model incorporates many of the characteristics of the spiral model. It is evolutionary in nature, demanding an iterative approach to the creation of software
  - The model composes applications from prepackaged software components

Specialized Process Model

- *Component-based development model* incorporates the following steps –
  1) Available component-based products are researched and evaluated for the application domain
  2) Component integration issues are considered
  3) A software architecture is designed to accommodate the components
  4) Components are integrated into the architecture
  5) Comprehensive testing is conducted to ensure proper functionality

Specialized Process Model

- Formal Methods Model
  - Encompasses a set of activities that leads to formal mathematical specification of computer software
  - Encourages a mathematically based approach to software development & verification
  - A variation on this approach, called *clean-room software engineering*, is currently applied by some software development organizations.
  - Provide a mechanism for eliminating many of the problems that are difficult to overcome using other SE paradigms. Ambiguity, incompleteness, and inconsistency can be discovered & corrected more easily through the application of mathematical analysis.

Specialized Process Model

- Formal methods model offers the promise of defect-free software. BUT, it is not widely used –
  - Development of formal models is quite time-consuming & expensive
  - Because few developers have the necessary background to apply formal methods, extensive training is required
  - It is difficult to use the models as a communication mechanism for technically unsophisticated customers.

Specialized Process Model

- Aspect-Oriented Model
  - Accommodates cross-cutting concerns that spans the entire system architecture
  - AOSD defines “aspects” that express customer concerns that cut across multiple system functions, features, and information

Unified Process Model

- The Unified Process is a "*use-case driven, architecture-centric, iterative and incremental*" software process designed as a framework for UML methods and tools.
- UP phases are similar in intent to the generic framework activities .
- A modern process model that is organized into five phases but that separates activities from these phases.

Five Phases of Unified Process Model

1) **Inception**
   - Establish the business case for the system.
2) **Elaboration**
   - Develop an understanding of the problem domain and the system architecture.
3) **Construction**
   - System design, programming and testing.
4) **Transition**
   - Deploy the system in its operating environment.
5) **Production**
   - On-going monitoring & support

Unified Process Model