Introduction to SE

Mrs. Purvi D. Sankhe

What is Software ?

Software can define as:

- **Instruction – executed provide desire features, function & performance.**
- **Data structure – to adequately manipulate operation.**
- **Documents – operation and use of the program.**

Software products may be developed for a particular customer or may be developed for a general market.

- **Software products may be**
  - **Generic - developed to be sold to a range of different customers e.g. PC software such as Excel or Word.**
  - **Bespoke (custom) - developed for a single customer according to their specification.**

Changing Nature of Software

- System software Eg.. *Compilers*
- Application software..*point of sale transaction processing*
- Engineering/scientific software..*CAD, system simulations*
- Embedded software ..*keypad control of microwave oven*
- Product-line software (e.g., *inventory control, word processing, multimedia*)
- Web applications
- Artificial intelligence software ..*pattern recognition, game playing*

System Software:

- System software is a collection of programs written to service other programs.
- It is characterized by heavy interaction with computer hardware; heavy usage by multiple users; concurrent operation that requires scheduling, resource sharing, and sophisticated process management; complex data structures; and multiple external interfaces.

Ex. Compilers, operating system, drivers etc.

Application Software :

- Application software consists of standalone programs that solve a specific business need.
- Application software is used to control the business function in real-time.

Engineering /Scientific software:

- Characterized by "number crunching" algorithms.
- Applications range from astronomy to volcano logy, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing.

Ex. Computer Aided Design (CAD), system stimulation etc.

Embedded Software:

- It resides in read-only memory and is used to control products and systems
- Embedded software can perform limited and esoteric functions.

Ex. keypad control for a microwave oven.

Product line software:

- Designed to provide a specific capability for use by many different customers, product line software can focus on a limited and esoteric marketplace.

Ex. Word processing, spreadsheet, CG, multimedia, etc.

Web Applications:

- Web apps can be little more than a set of linked hypertext files.
- It evolving into sophisticated computing environments that not only provide standalone features, functions but also integrated with corporate database and business applications.

Artificial Intelligence software

- AI software makes use of non-numerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis

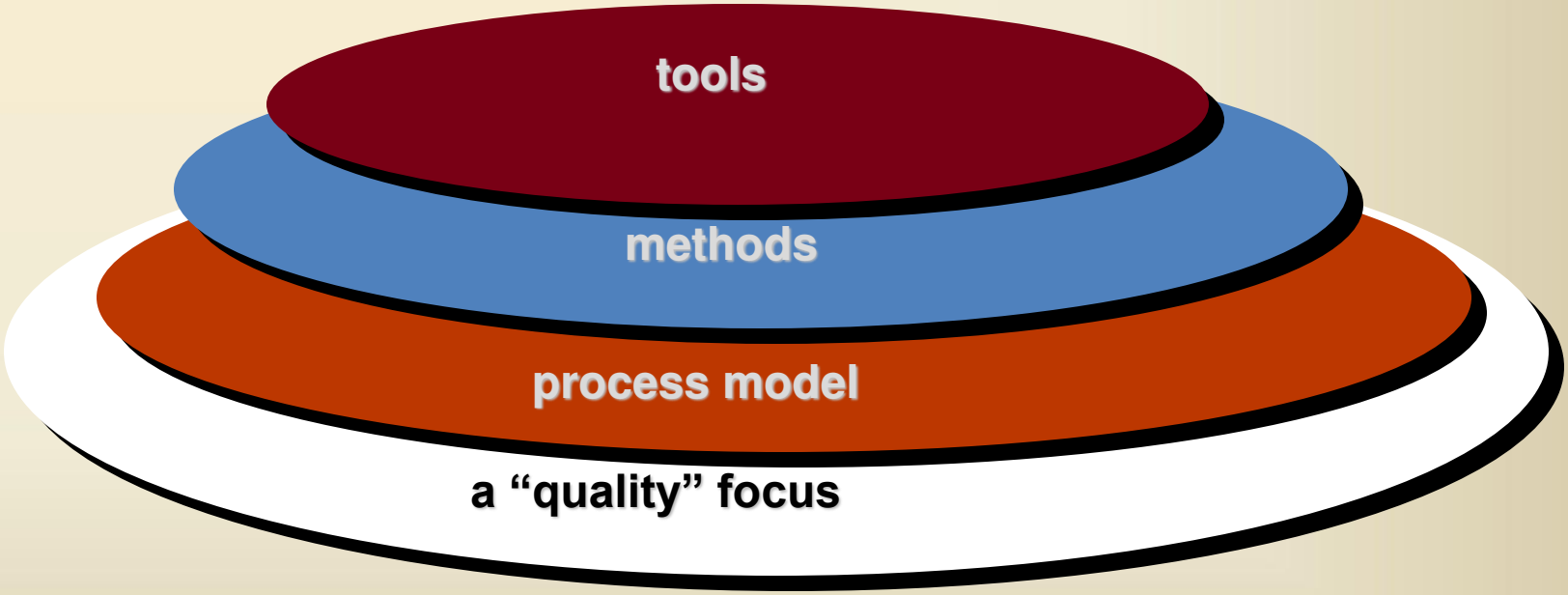
Ex. Robotics, expert system, game playing, etc.

Software Engineering - Defined

- (1969) Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines

- (IEEE) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software

Software Engineering is a Layered Technology



Process, Methods, and Tools

- **Process**
  - Provides the glue that holds the layers together; enables rational and timely development; provides a framework for effective delivery of technology; forms the basis for management; provides the context for technical methods, work products, milestones, quality measures, and change management
- **Methods**
  - Provide the technical "how to" for building software; rely on a set of basic principles; encompass a broad array of tasks; include modeling activities
- **Tools**
  - Provide automated or semi-automated support for the process and methods (i.e., CASE tools)

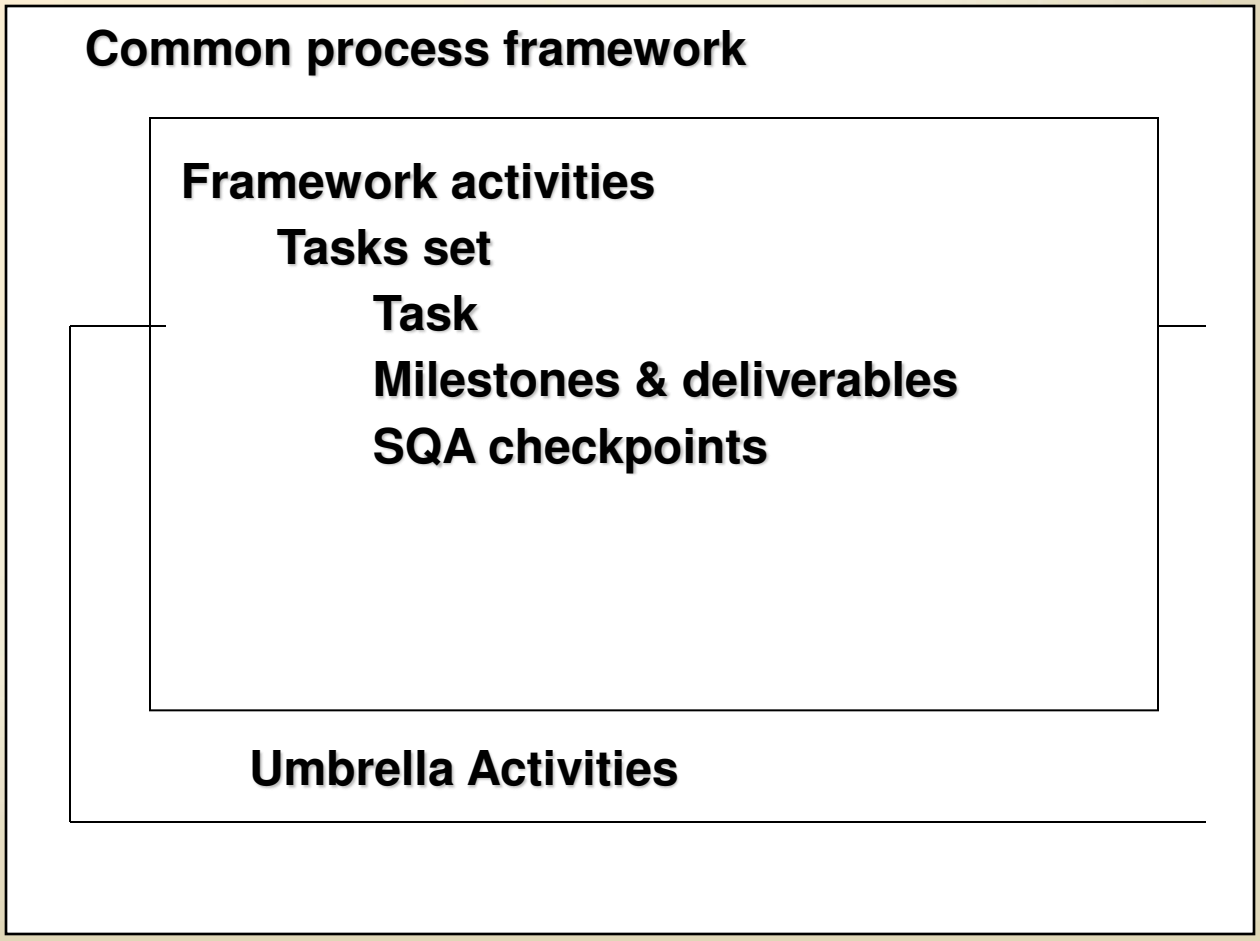
Generic Process Framework

- **Communication**
  - Involves communication among the customer and other stake holders; encompasses requirements gathering
- **Planning**
  - Establishes a plan for software engineering work; addresses technical tasks, resources, work products, and work schedule
- **Modeling (Analyze, Design)**
  - Encompasses the creation of models to better understand the requirements and the design
- **Construction (Code, Test)**
  - Combines code generation and testing to uncover errors
- **Deployment**
  - Involves delivery of software to the customer for evaluation and feedback

Umbrella Activities

- Software requirements management
- Software project planning
- Software project tracking and oversight
- Software quality assurance
- Software configuration management
- Formal technical reviews
- Risk management
- Measurement – process, project, product
- Reusability management (component reuse)
- Work product preparation and production

A Common Process Framework



Generic Process Framework

- **Communication**
  - Involves communication among the customer and other stake holders; encompasses requirements gathering
- **Planning**
  - Establishes a plan for software engineering work; addresses technical tasks, resources, work products, and work schedule
- **Modeling (Analyze, Design)**
  - Encompasses the creation of models to better under the requirements and the design
- **Construction (Code, Test)**
  - Combines code generation and testing to uncover errors
- **Deployment**
  - Involves delivery of software to the customer for evaluation and feedback

What is a Process?

- **(Webster) A system of operations in producing something; a series of actions, changes, or functions that achieve an end or a result**

- **(IEEE) A sequence of steps performed for a given purpose**

What is a Software Process?

- **(SEI) A set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, and user manuals)**
- **As an organization matures, the software process becomes better defined and more consistently implemented throughout the organization**
- **Software process maturity is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective**

Prescriptive Process Models

Contents

- Generic process framework (revisited)
- Traditional process models
- Specialized process models

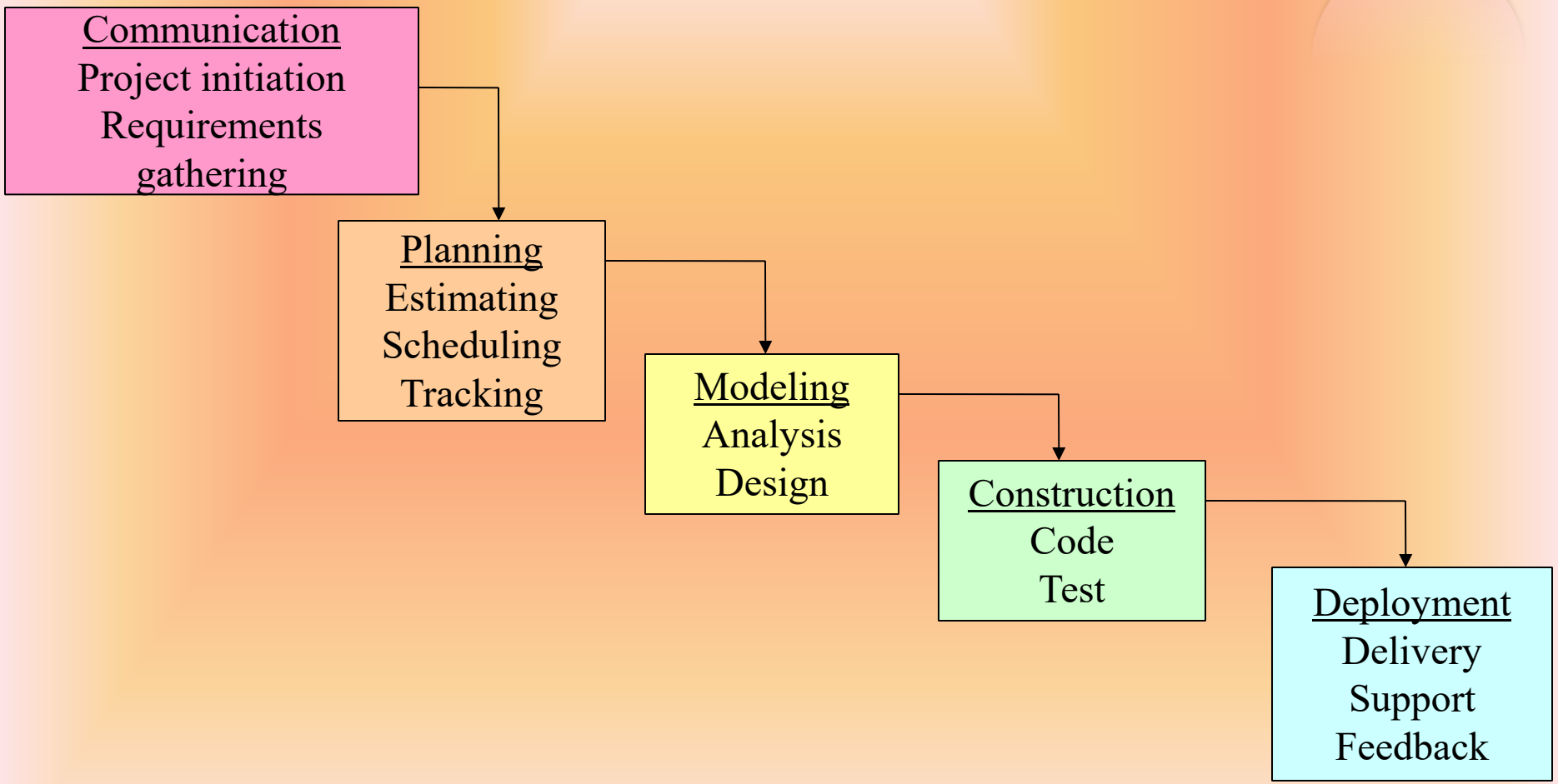
Traditional Process Models

Prescriptive Process Model

- Defines a distinct set of activities, actions, tasks, milestones, and work products that are required to engineer high-quality software
- The activities may be linear, incremental, or evolutionary

Waterfall Model

(Diagram)



Waterfall Model
(Description)

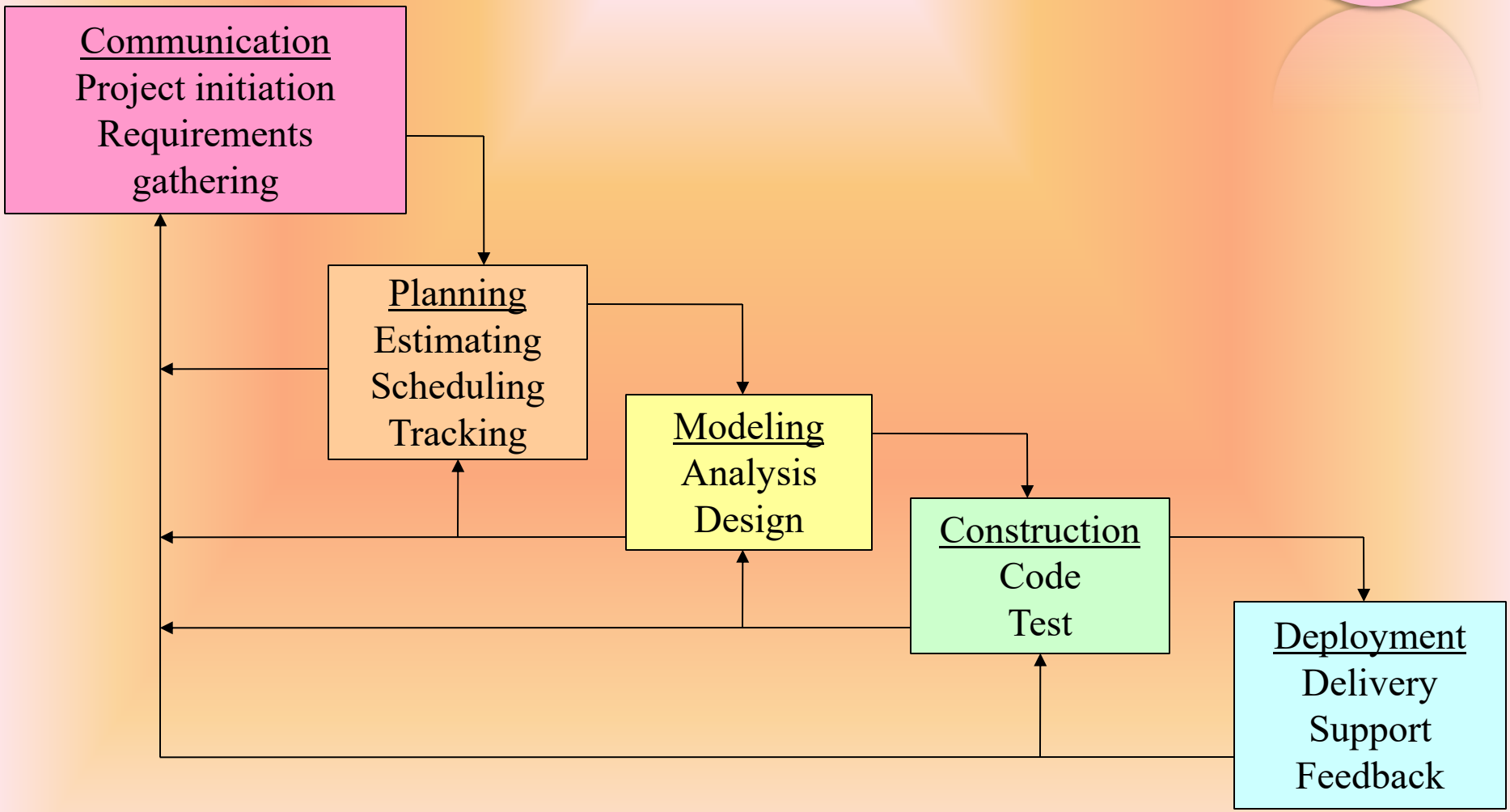
- Oldest software lifecycle model and best understood by upper management
- Used when requirements are well understood and risk is low
- Work flow is in a linear (i.e., sequential) fashion
- Used often with well-defined adaptations or enhancements to current software

Waterfall Model
(Problems)

- Doesn't support iteration, so changes can cause confusion
- Difficult for customers to state all requirements explicitly and up front
- Requires customer patience because a working version of the program doesn't occur until the final phase
- Problems can be somewhat alleviated in the model through the addition of feedback loops (see the next slide)

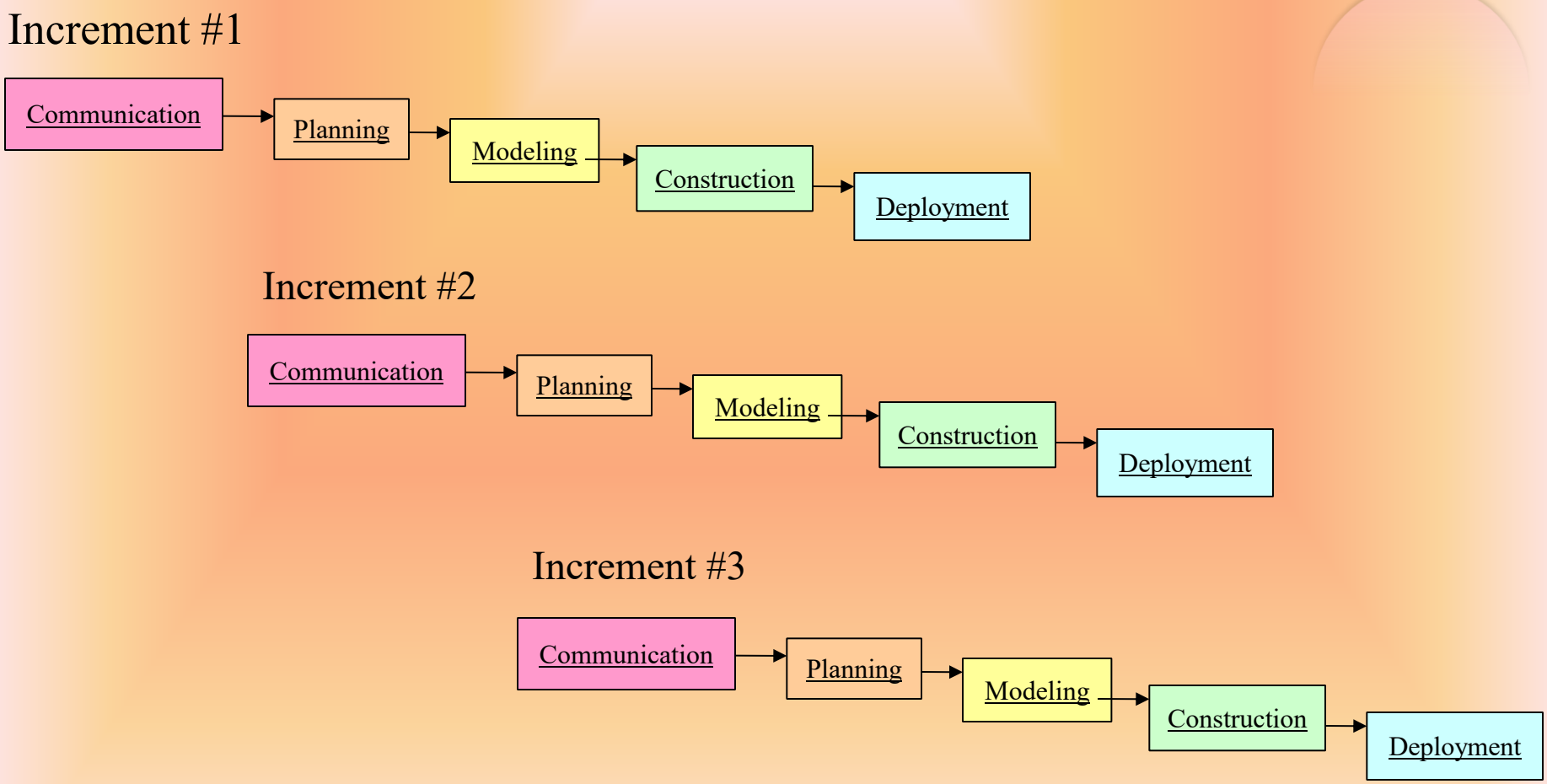
Waterfall Model with Feedback

(Diagram)



Incremental Model

(Diagram)



Incremental Model
(Description)

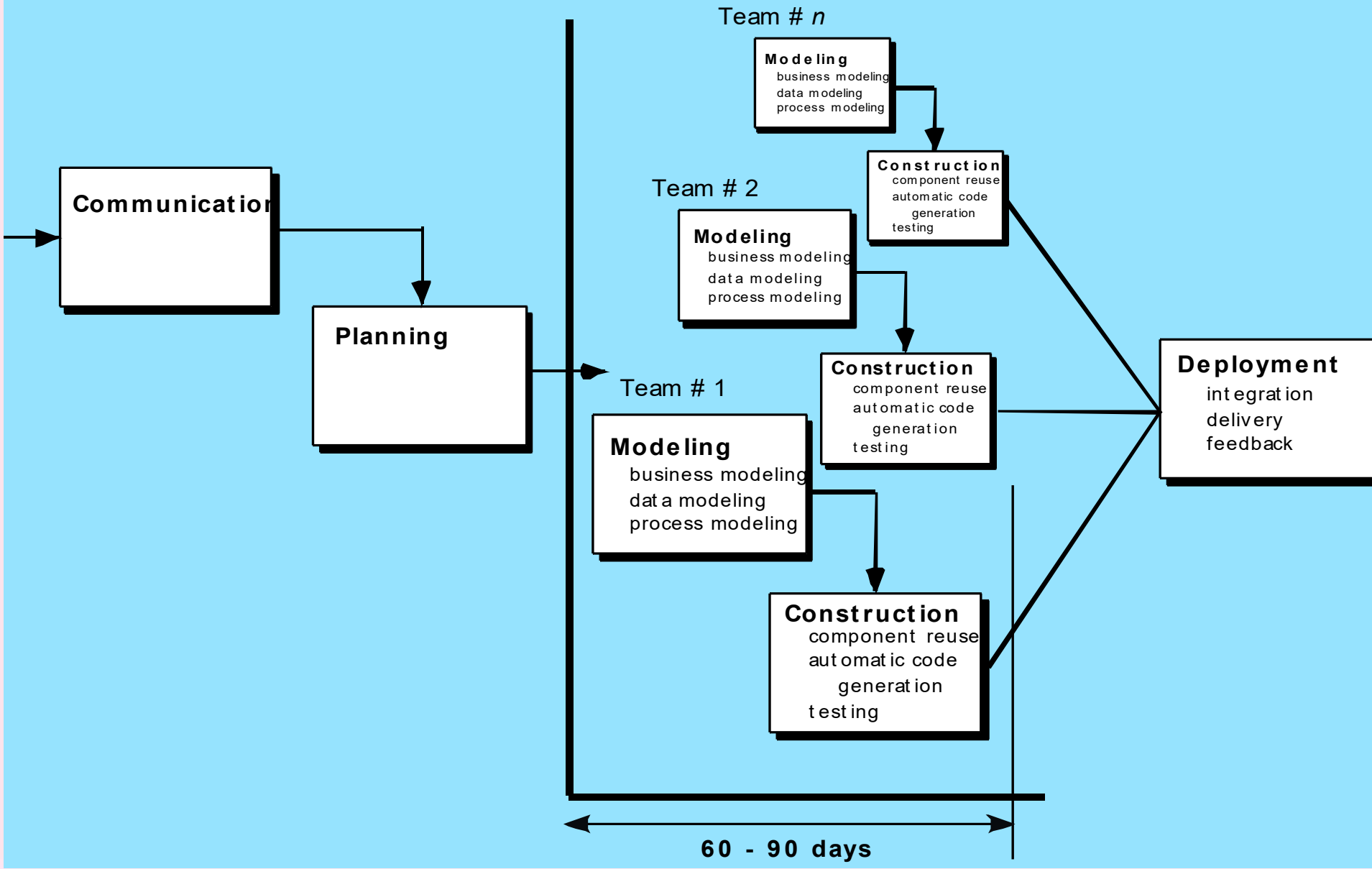
- Used when requirements are well understood
- Multiple independent deliveries are identified
- Work flow is in a linear (i.e., sequential) fashion within an increment and is staggered between increments
- Iterative in nature; focuses on an operational product with each increment
- Provides a needed set of functionality sooner while delivering optional components later
- Useful also when staffing is too short for a full-scale development

Incremental Model

(Example)

- Each linear sequence produces deliverable “increments” of the software.
- Ex: a Word Processor delivers basic file mgmt., editing, in the first increment;
- more sophisticated editing, document production capabilities in the 2nd increment;
- spelling and grammar checking in the 3rd increment.

RAD model


Team # n
Modeling
business modeling
data modeling
process modeling
Construction
component reuse
automatic code
generation
testing
Team # 2
Modeling
business modeling
data modeling
process modeling
Construction
component reuse
automatic code
generation
testing
Communication
Planning
Team # 1
Modeling
business modeling
data modeling
process modeling
Construction
component reuse
automatic code
generation
testing
Deployment
integration
delivery
feedback
60 - 90 days

RAD model

- is an incremental software development process model
- “high-speed” adaptation of the linear sequential model
- If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within very short time periods

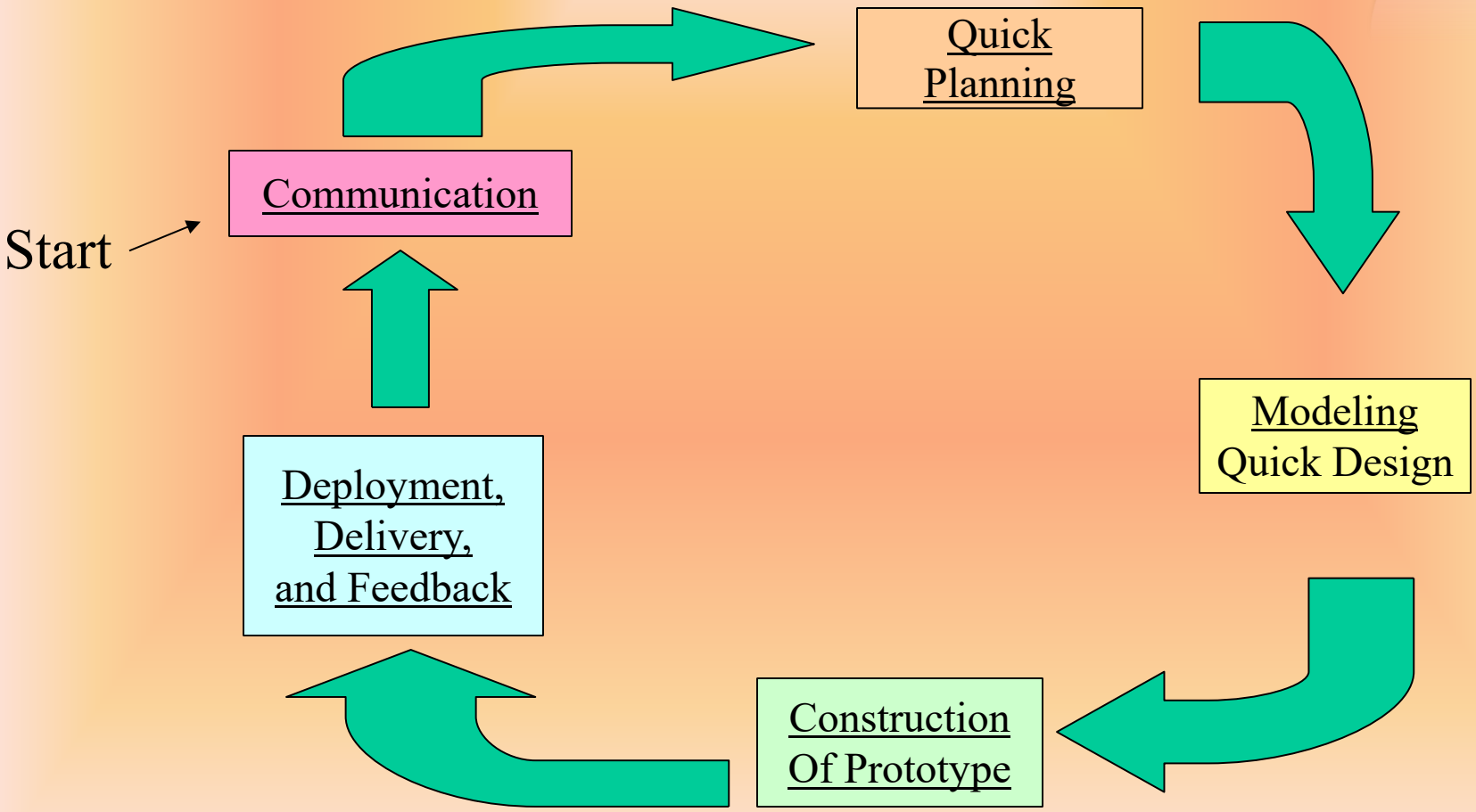
RAD model

- RAD approach encompasses the following phases :
- **Business modeling**. The information flow among business functions is modeled in a way that answers the following questions:
  - What information drives the business process? What information is generated? Who generates it? Where does the information go? Who processes it?
- **Data modeling.** The information flow defined as part of the business modeling phase is refined into a set of data objects
- **Process modeling.** The data objects defined in the data modeling phase are transformed to achieve the information flow
- **Application generation.** RAD assumes the use of fourth generation techniques
- **Testing and turnover.** Since the RAD process emphasizes reuse, many of the program components have already been tested.

RAD model

- **Like all process models, the RAD approach has drawbacks:**
- **For large but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams.**
- **RAD requires developers and customers who are committed to the rapid-fire activities necessary to get a system complete in a much abbreviated time frame. If commitment is lacking from either constituency, RAD projects will fail.**

Prototyping Model
(Diagram)



Prototyping Model
(Description)

- Follows an evolutionary and iterative approach
- Used when requirements are not well understood
- Serves as a mechanism for identifying software requirements
- Focuses on those aspects of the software that are visible to the customer/user
- Feedback is used to refine the prototype

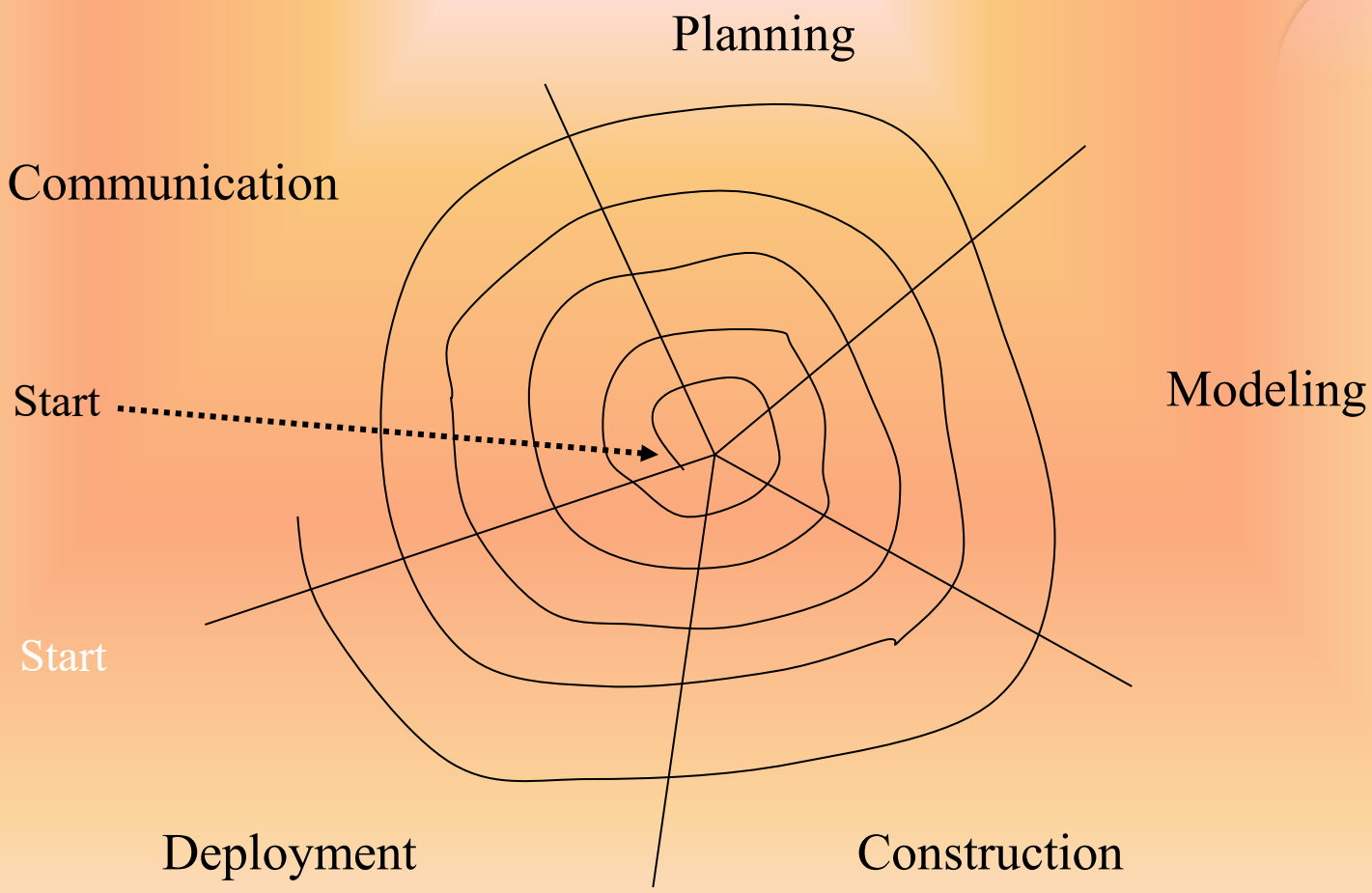
Prototyping Model
(Description)

- The prototyping paradigm begins with *communication* where requirements and goals of Software are defined.
- Prototyping iteration is *planned* quickly and modeling in the form of quick design occurs.
- The *quick design* focuses on a representation of those aspects of the Software that will be visible to the customer "Human interface".
- The quick design leads to the *Construction of the Prototype*.
- The prototype is *deployed* and then *evaluated* by the customer.
- *Feedback* is used to refine requirements for the Software.
- Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while enabling the developer to better understand what needs to be done.
- The prototype can serve as the "first system". Both customers and developers like the prototyping paradigm as users get a feel for the actual system, and developers get to build Software immediately.

Prototyping Model
(Potential Problems)

- The customer sees a "working version" of the software, wants to stop all development and then buy the prototype after a "few fixes" are made
- Developers often make implementation compromises to get the software running quickly (e.g., language choice, user interface, operating system choice, inefficient algorithms)

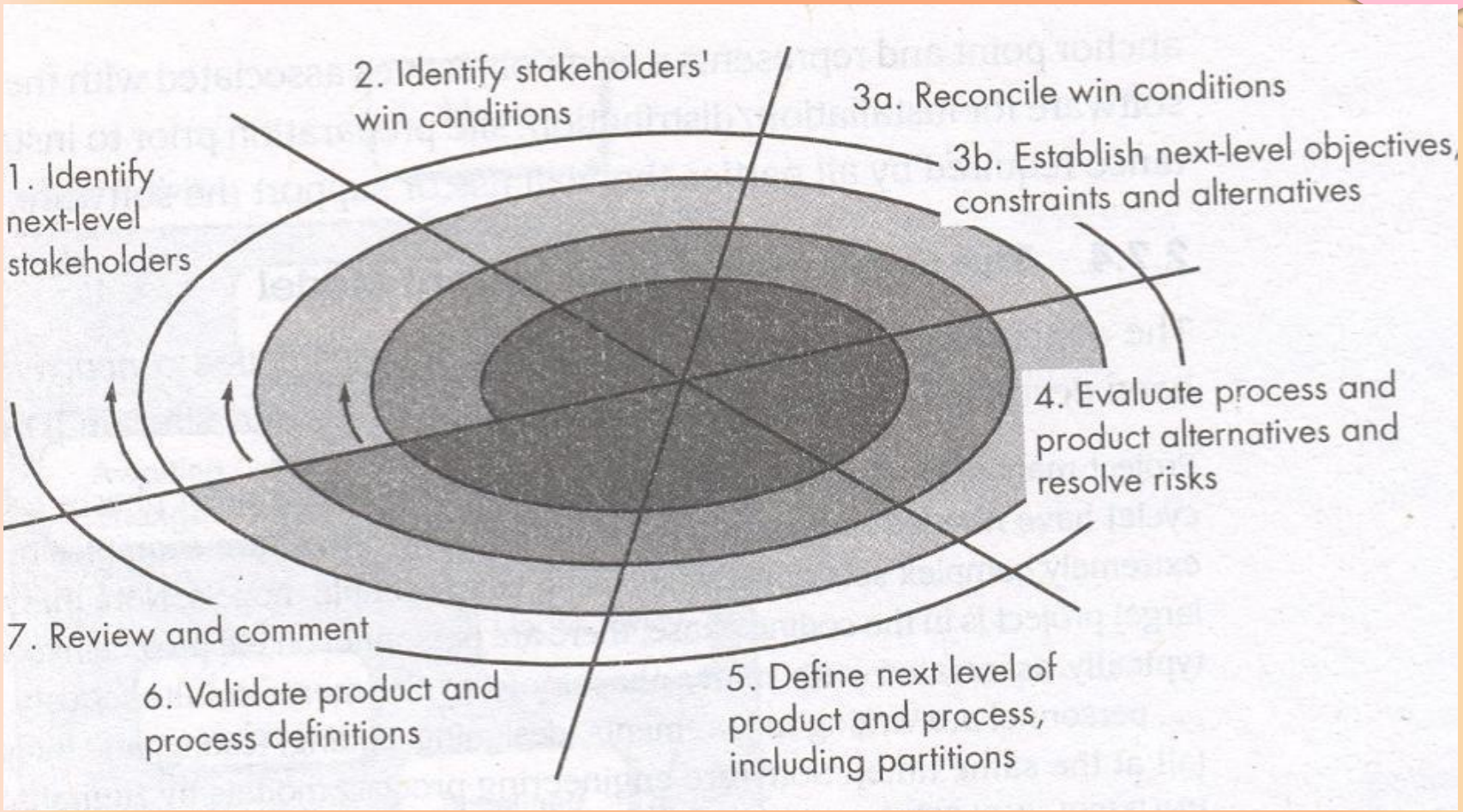
Spiral Model
(Diagram)



Spiral Model
(Description)

- Invented by Dr. Barry Boehm in 1988
- Follows an evolutionary approach
- Used when requirements are not well understood and risks are high
- **Inner spirals** focus on identifying software requirements and project risks; may also incorporate prototyping
- **Outer spirals** take on a classical waterfall approach after requirements have been defined, but permit iterative growth of the software
- Operates as a risk-driven model…a go/no-go decision occurs after each complete spiral in order to react to risk determinations
- Requires considerable expertise in risk assessment
- Serves as a realistic model for large-scale software development

WinWin Spiral Model



WinWin Spiral Model contd..

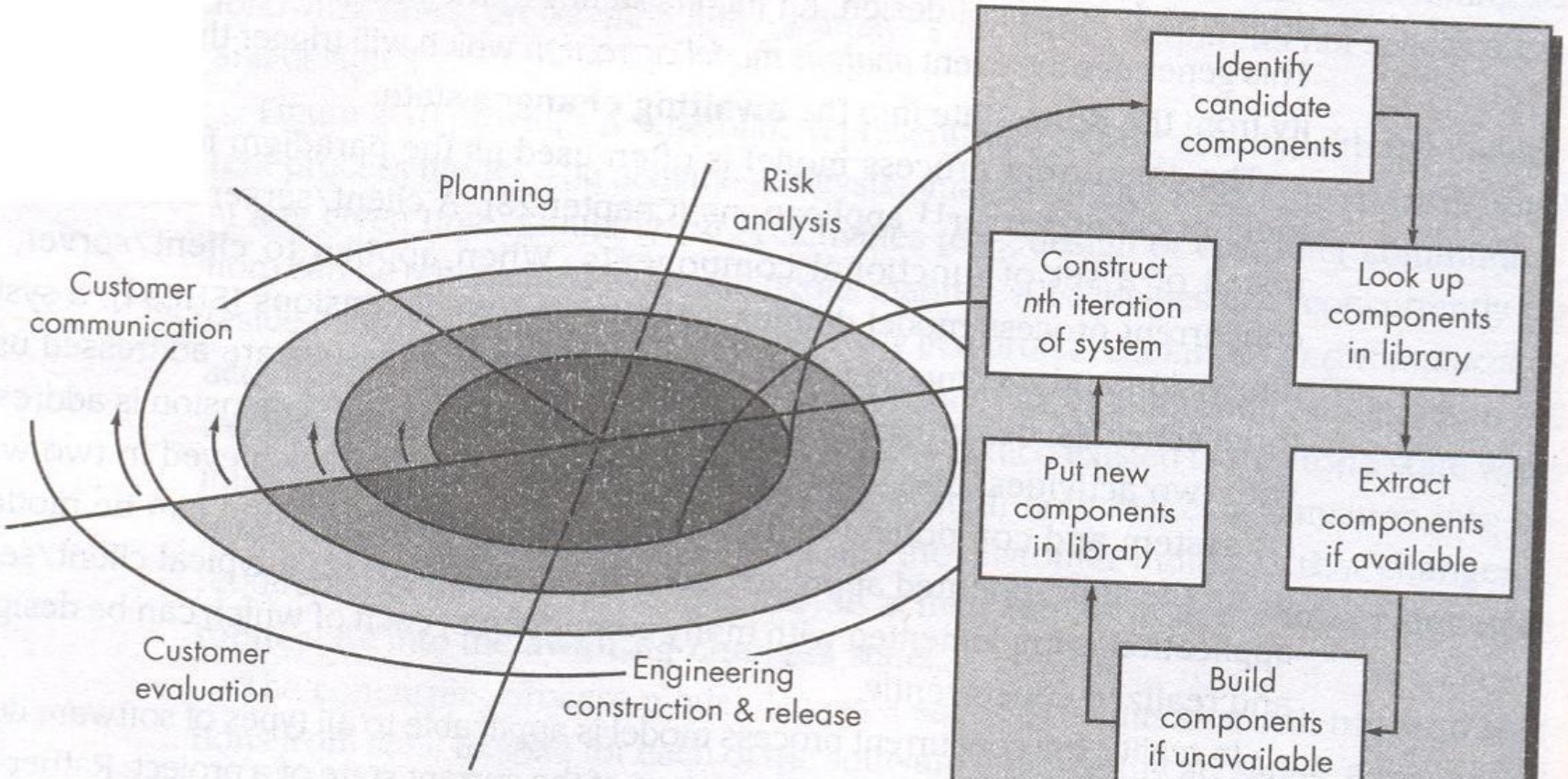
- **The customer and the developer enter into a process of negotiation, where the customer may be asked to balance functionality, performance, and other product or system characteristics against cost and time to market.**
- **The best negotiations strive for a “win-win” result.**
- **The customer wins by getting the system or product that satisfies the majority of the customer’s needs**
- **The developer wins by working to realistic and achievable budgets and deadlines.**

Specialized Process Models

Component-based Development Model

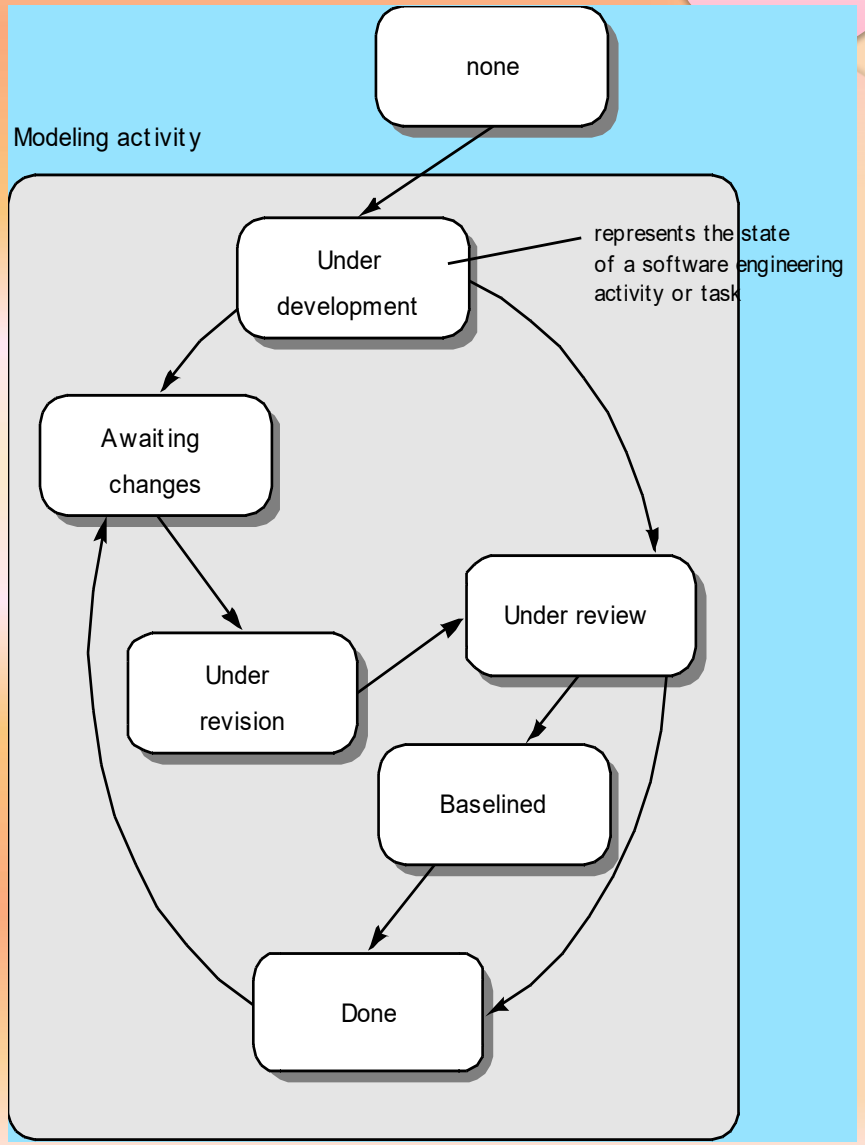
- Consists of the following process steps
  - Available component-based products are researched and evaluated for the application domain in question
  - Component integration issues are considered
  - A software architecture is designed to accommodate the components
  - Components are integrated into the architecture
  - Comprehensive testing is conducted to ensure proper functionality
- Relies on a robust component library
- Capitalizes on software reuse, which leads to documented savings in project cost and time

Component Assembly Model



Concurrent Process Model

- **The concurrent development model, sometimes called concurrent engineering, can be represented schematically as a series of framework activities, Software engineering actions of tasks, and their associated states.**



The concurrent model is often more appropriate for system engineering projects where different engineering teams are involved.

Figure above provides a schematic representation of one Software engineering task within the modeling activity for the concurrent process model. The activity – modeling- may be in any one of the states noted at any given time.

All activities exist concurrently but reside in different states.

For example, early in the project the communication activity has completed its first iteration and exists in the awaiting changes state. The modeling activity which existed in the none state while initial communication was completed now makes a transition into underdevelopment state.

If, however, the customer indicates the changes in requirements must be made, the modeling activity moves from the under development state into the awaiting changes state.

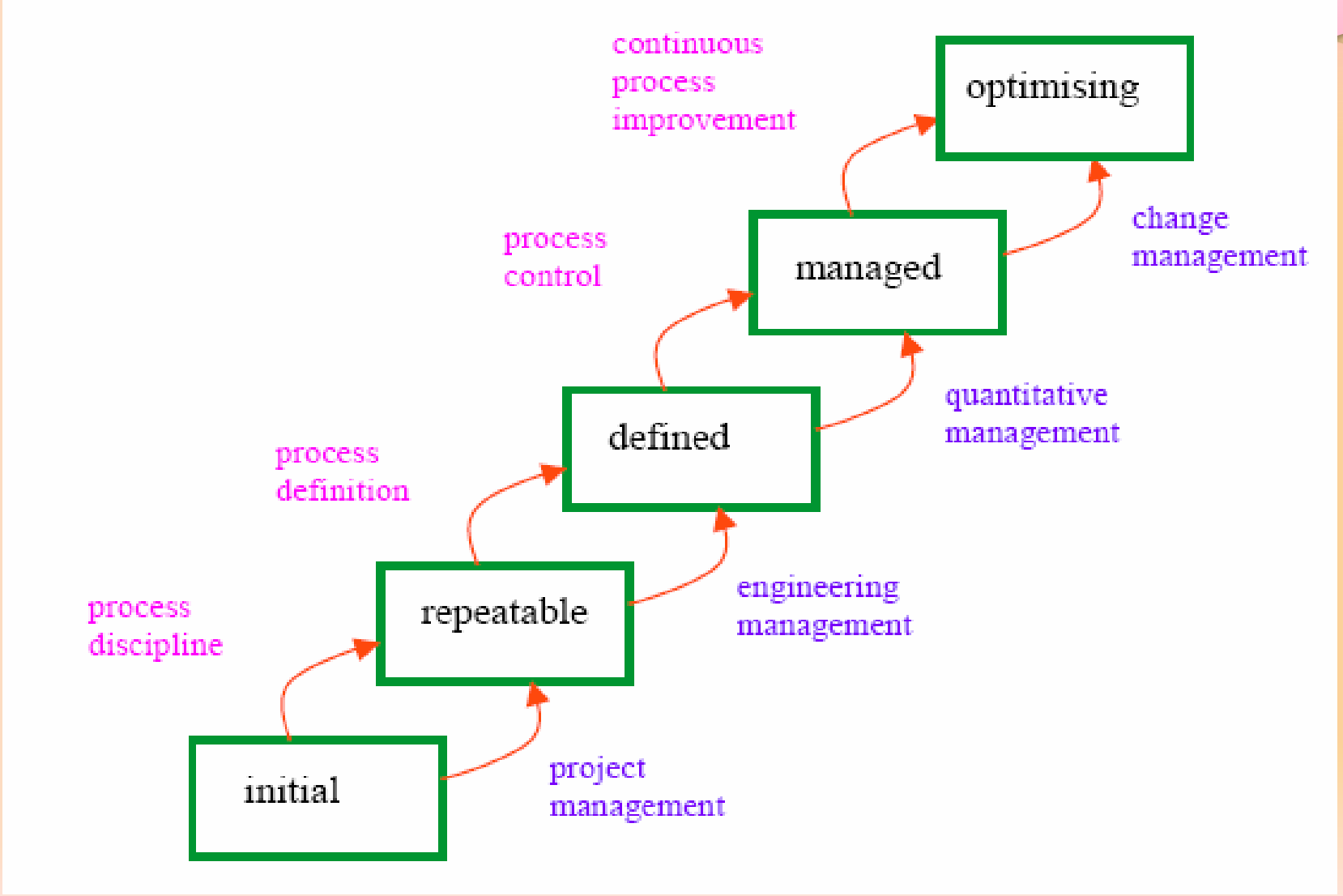
Comparing Models

Model	Strengths	Weaknesses
Waterfall	Disciplined approach Document driven	Final product may not meet client's real needs
Prototyping	Helps to detail user's requirements	The prototype may outlive its usefulness
Iterative	Get early results Promotes maintenance	May degenerate into CABTAB
Spiral	Incorporates features from other models	Only suited to large-scale in-house development

Capability Maturity Model (CMM)

- **Developed in 1987 by the Software Engineering Institute (SEI) at Carnegie-Mellon University under the sponsorship of DARPA**
- **Described in the book Managing the Software Process in 1989 by Watts Humphrey**
- **Published as a separate document: Capability Maturity Model for Software in 1991**

Five Levels of Software Process Maturity



Characteristics of Each Level

- **Initial Level (Level 1)**
  - Characterized as ad hoc, and occasionally even chaotic
  - Few processes are defined, and success depends on individual effort
- **Repeatable (Level 2)**
  - Basic project management processes are established to track cost, schedule, and functionality
  - The necessary process discipline is in place to repeat earlier successes on projects with similar applications

Characteristics of Each Level (continued)

- **Defined (Level 3)**
  - The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization
  - All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software
  - level most aimed for by software developers via standards such as ISO 9001
- **Managed (Level 4)**
  - Detailed measures of the software process and product quality are collected
  - Both the software process and products are quantitatively understood and controlled

Characteristics of Each Level (continued)

- **Optimized (Level 5)**
  - **Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies**
  - **reached by very few developers**

The Capability Maturity Model (CMM) is a way to develop and refine an organization's processes

CMMI

- **SEI CMMI is a process improvement approach that provides organizations with the essential elements of effective processes.**
- **CMMI can help you make decisions about your process improvement plans.**
- **CMM is a method to evaluate and measure the maturity of the software development process of an organization.**
- **CMM measures the maturity of the software development process on a scale of 1 to 5.**

CMMI

- **Levels are:**
- **Level 0: Incomplete: doesn't achieve goal**
- **Level 1: Performed: work tasks are being conducted**
- **Level 2: Managed: well managed ..resources, stakeholders, monitoring, reviewing, evaluation**
- **Level 3: Defined: mgmt. & eng processes are documented, standardized& integrated**
- **Level 4: Quantitatively Managed: measures are used to quantitatively understanding**
- **Level 5: Optimizing: enabling continuous process improvement, testing innovative ideas.**

Difference between CMM and CMMI

- **CMM is a reference model of matured practices in a specified discipline like Systems Engineering CMM, Software CMM, People CMM, Software Acquisition CMM etc., but they were difficult to integrate as and when needed.**

- **CMMI is the successor of the CMM and evolved as a more matured set of guidelines and was built combining the best components of individual disciplines of CMM(Software CMM, People CMM, etc.). It can be applied to product manufacturing, people management, software development, etc.**

- **CMM describes about the software engineering alone where as CMM Integrated describes both software and system engineering. CMMI also incorporates the Integrated Process and Product Development and the supplier sourcing**