3) **RAD MODEL:** RAD stands for Rapid Application Development. It is an incremental software process model that emphasizes short development cycle. It is a version of waterfall model with rapid development. If requirements are well understood and project scope is constrained RAD process enables development team to create a "fully functional system" within a very short time period (60- 90 days). Each major function can be addressed by a separate RAD team and then integrated to form a whole project.

The framework activities of RAD model include:

1. **Communication:** It involves heavy communication and collaboration with customer and other stakeholders. It encompasses requirements gathering and related activities.
2. **Planning:** It plans for Software Engineering work that follows. It describes technical tasks to be conducted, resources that will be required, likely risks, work products to be produced and a work schedule.
3. **Modeling:** This activity focuses on creation of models that allow stakeholders (customer, developer) to better under software requirements and design that will achieve requirements.
4. **Construction:** It combines code generation and testing to uncover errors in the code.
5. **Development:** The software (completed/partial increment) is delivered to the customer for evaluation and feedback of it.

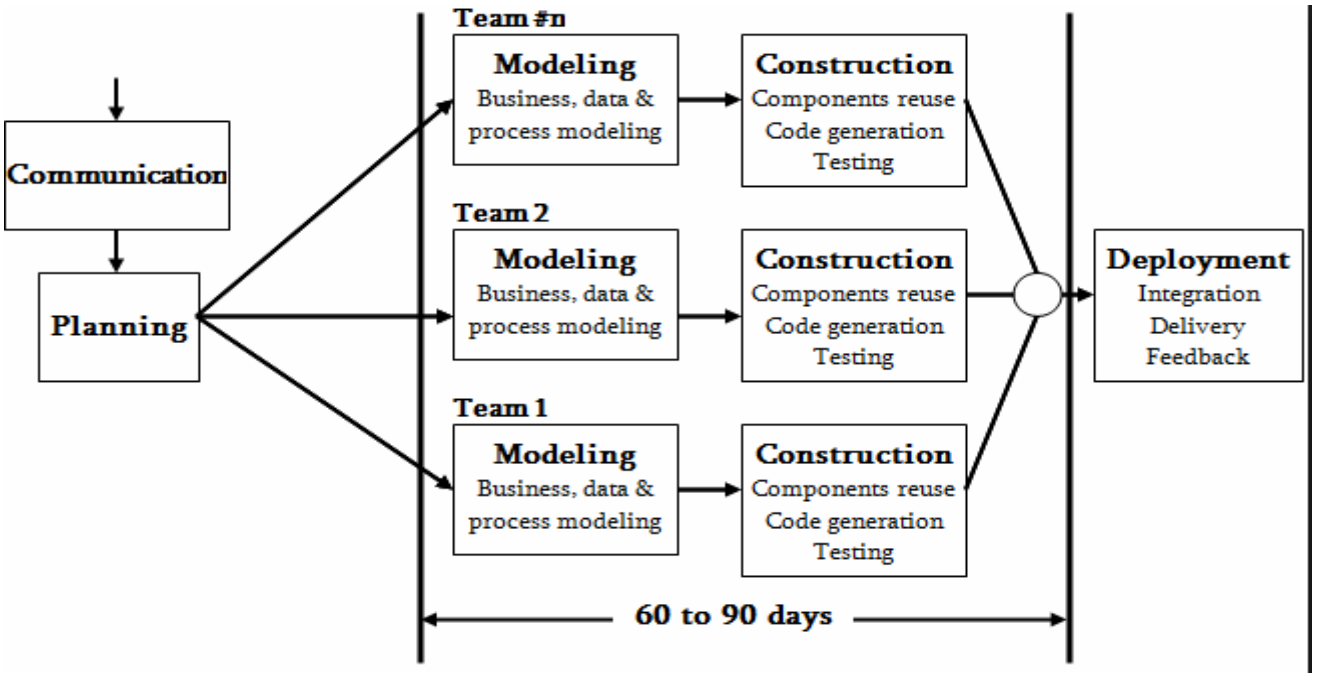

Fig: RAD Process model.

Advantages:

- Project with 2-3 months deadline opt for RAD.
- Task is divided among teams to fasten development process.

Disadvantages:

- Large projects using RAD may not work.
- If high performance is an issue, RAD may not work.
- RAD may not be appropriate when technical risks are high.

EVOLUTIONARY PROCESS MODELS

These models are specially designed to accommodate a product that evolves over time. These are iterative and enable software engineers to develop more complete software versions.

1) **PROTOTYPING:** Prototyping model is used when customer defines a set of objectives, but does not

identify detailed input, processing output requirements, developer is unsure of efficiency of algorithm, adaptability of operating system etc, where phased model is inappropriate.

Prototyping model can be used as a standalone process model. Prototyping paradigm assists the software engineer and customer to better understand what is to be built when requirements are fuzzy. Prototype helps to identify software requirements.

Prototyping paradigm begins with communication, then quickly planning the prototyping iteration, modeling quick design, construction of prototype, and the prototype is deployed and then evaluated by the customer/user. Feedback is used to refine requirements for the software.

Prototype can serve as "the first system", where users get a feel of actual system and developers get to build something immediately.

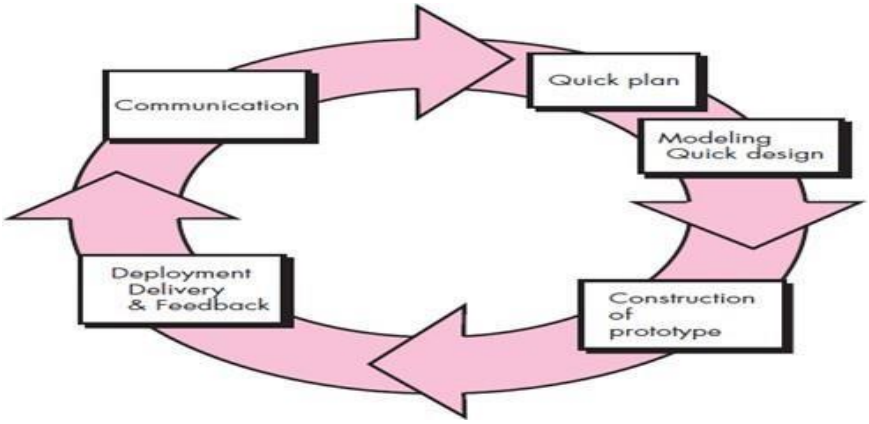

Fig: Prototyping Model

.

2) THE SPIRAL MODEL: The Spiral model is proposed by "Boehm". This model was developed to encompass the best features of waterfall model and prototyping. It is a ***risk-driven*** process model with the risk analysis feature.

Features:

1) Cyclic Approach for increasing system's degree of definition and implementation while decreasing degree of risk-Risk is considered as each revolution is made.
2) Anchor Point Milestone for ensuring stakeholders commitment to feasible and mutually satisfactory systems solution. (Milestone is a combination of work products and conditions).

Spiral model may be viewed as a Meta model, as it can accommodate any process development model. Software is developed as a series of evolutionary releases. Project manager adjusts planned number of iterations to complete the software. During early iterations prototype is generated and during later iterations complete version is developed.

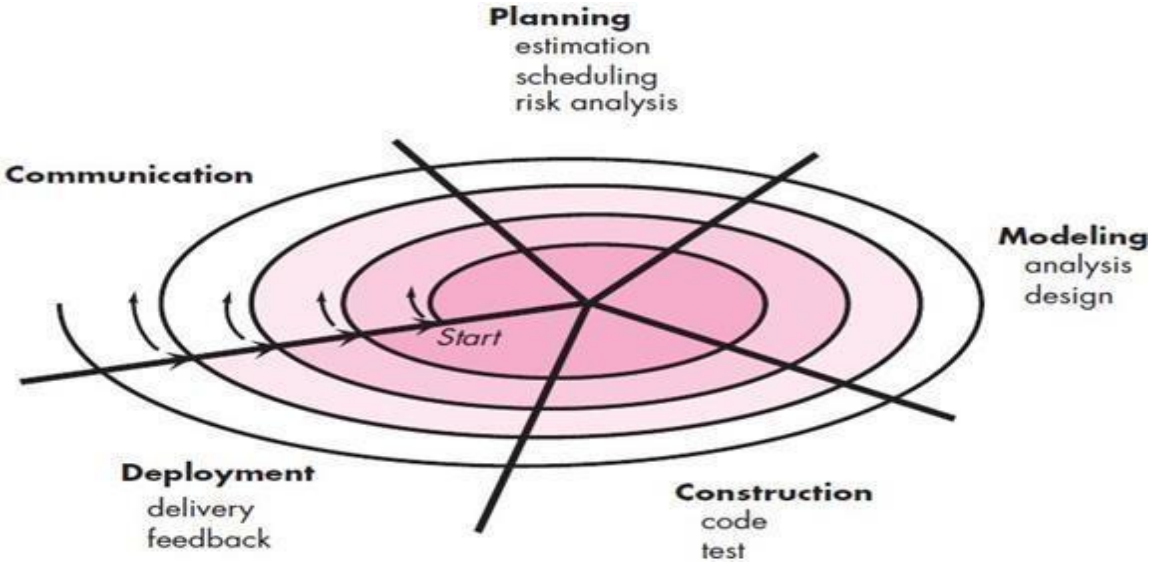

Fig: Spiral Model

In Spiral model, the first circuit around the spiral might result in the development of a product specification; subsequent passes around the spiral model might be used to develop a prototype and then progressively more sophisticated versions of the software. Unlike other process models that end when software is delivered, the spiral model can be adapted to apply throughout the life of the software.

Advantages:

- It is a realistic approach to the development of large scale systems and software. (Software evolves as the process progresses).
- It uses and enables the developer to apply the prototyping approach to any stage in evolution of product.
- Considers technical risks at all the stages of the project, and reduces risks before they become problematic.

❖ Like other paradigms, spiral model is not a panacea (Medicine).It demands considerable risk assessment expertise for success. If a major risk is not covered and managed, problems will occur.

3) **CONCURRENT DEVELOPMENT MODEL:** It is also called as "Concurrent Engineering". This model is represented schematically as a series of framework activities, Software Engineering actions and tasks, and their associated states concurrently. It strives to make all software development activities to be concurrently implemented.

- Ex: "Modeling" activity for spiral model is accomplished by invoking prototyping and/or analysis Modeling and specification and design.

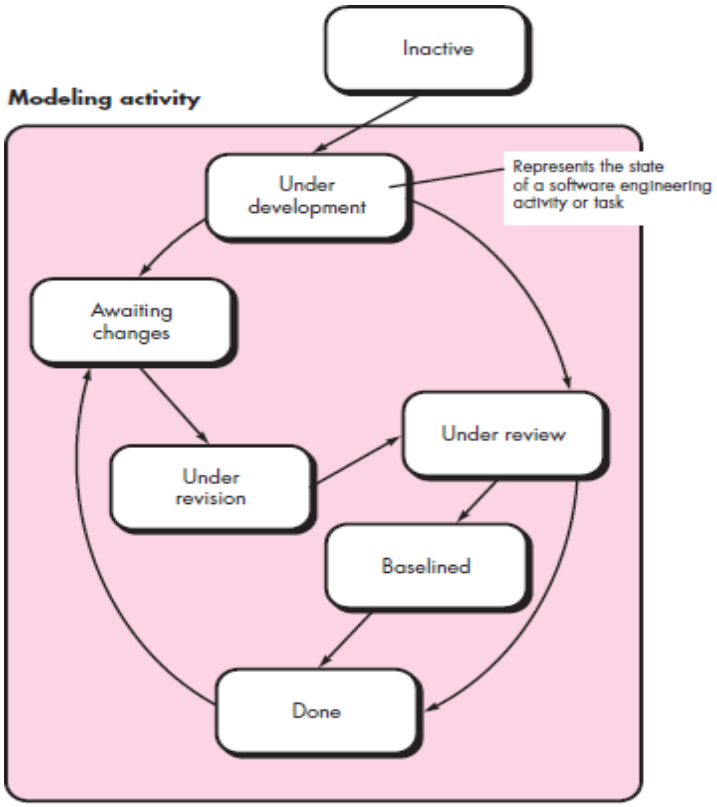

Fig: One element of concurrent model development model.

All activities (communication/modeling/construction etc) exist concurrently but reside in different states. ***State*** is an externally observable mode of behavior). For example, early in a project, the communication activity has completed its first iteration and exists in the awaiting changes state. Modeling activity which was in none state will now move to under development state.

- ❖ This model defines a series of events which will trigger transition from state to state for each of Software Engineering activities, actions/tasks.

Advantages:

- ✓ Applicable to all types of software development, provides accurate picture of current state of a project.
- ✓ The software engineering activities, tasks and actions are defined as a network of activities, rather than sequence of events.

Evolutionary Models Drawbacks:

- Prototyping poses a problem to project planning because of uncertain number of cycles required to construct product.
- Do not establish maximum speed of evolution.
- May not give flexibility and extensibility for the software process.

SPECIALIZED PROCESS MODELS

These models are used when a narrowly defined Software Engineering approach is chosen.

1. COMPONENT-BASED DEVELOPMENT: Commercial Off-The-Shelf (COTS) software components are used when software is to be build. These components provide targeted functionality so that component to be integrated into the software.

- ➢ It incorporates many characters of the spiral model. It is evolutionary in nature, composes applications from pre-packaged (COTS) software components.

Steps:

1. Available component-based products are researched and evaluated for the application domain in question.
2. Component integration issues are considered.
3. Software architecture is designed to accommodate the components.
4. Components are integrated into the architecture.
5. Comprehensive testing is used (conducted to ensure proper functionality).

Advantage: Software reuse-Important to produce high quality software.

2. FORMAL METHODS MODEL: Specialized software development approach that uses mathematical based techniques for specifying, developing and verifying the computer softwares. Formal Methods Model helps the software developers to apply correct mathematical notations to create the issue of insufficiency, inconsistency and uncertainty of the software by applying mathematical analysis.

During design phase, formal methods model acts as a program verifier and help Software Engineers to detect and correct these errors, which are otherwise very difficult to be detected. This model assures defect free software.

Drawbacks:

1. Time consuming and expensive.
2. Software Engineers need extensive training to apply this model.
3. Clients needed to be technically sound for proper communication.

- ❖ Because of these reasons Formal Methods Models are used only in development of high integrity software applications where safety and security is of atmost importance.

3. ASPECT-ORIENTED SOFTWARE DEVELOPMENT (AOSD): As modern computer based systems become more sophisticated and complex, some concerns (security, fault tolerance, memory management etc) span the entire architecture. When concern cut across multiple system functions, features and information, they are referred as crosscutting concerns. Aspectual Requirements define those crosscutting concerns that have impact across the software architecture.

AOSD, often referred to as Aspect-Oriented Programming (AOP), is a relatively new software engineering paradigm which provides a process for defining, specifying, designing and constructing aspects (crosscutting concerns).

Presently there is no distinct Aspect-Oriented Process. If such an approach is developed, then it must integrate the characteristics of both spiral and concurrent model, because of their evolutionary and parallel natures respectively.

THE UNIFIED PROCESS MODEL:

This model is also referred as RUP (Rational Unified Process). Unified Process refers to a methodology of extracting the most essential activities of conventional software development phases (communication, planning, Modeling, construction and deployment) and characterizing them, so that they can be applied in the Agile (highly valued) software development.

History: Jacobson, Rumbaugh and Greedy Booch developed the Unified Process, a framework for Object-Oriented Software Engineering using UML. Today, Unified Process and UML are used on Object- Oriented projects of all kinds.

- The iterative, incremental model proposed by the Unified Process can and should be adapted to meet specific project needs.

Phases of the Unified Process:

1. **Inception Phase:** The Inception Phase encompasses both customer communication and planning activities. By collaborating with customer and end-users, business requirements are identified and described through a set of use cases [sequence of actions that are performed by an actor (Ex: A person, machine, and another system). As actor interacts with the software, use cases provide project scope.

The Inception Phase must:

i. Produce a business case.

ii. Identify business requirements, business and process risks.

iii. Give overall vision for the project, as the outputs result in various documents/work products.

Work Products:

1) Vision Documents
2) Initial use-case model
3) Initial Project Glossary
4) Initial Business case
5) Initial Risk Assessment
6) Project Plan [Phases and Interaction]
7) Business Model (if necessary)
8) One or more Prototypes

2. **Elaboration Phase:** The Elaboration Phase encompasses planning and Modeling activities. This phase refines and expands preliminary use-cases that were developed in inception phase. The Elaboration Phase expands the architectural representation to five views: 1) Use case Model, 2) Analysis Model, 3) Design Model, 4) Implementation Model, 5) Deployment Model.

Elaboration Phase creates an "executable architectural baseline" that represents "first cut" executable system-prototype. Architectural baseline provides viability of the project but not all features and functions required to use the system. Modifications to the plan may be made at this time.

Work Products:

1) Use case Model
2) Supplementary Requirements
3) Analysis Model
4) Software Architecture Prototype
5) Executable Architecture Description
6) Preliminary Design Model
7) Revised Risk List
8) Project plan, includes
a) Iteration Plan
b) Adapted workflow
c) Milestones
d) Technical work products
9) Preliminary User Manual

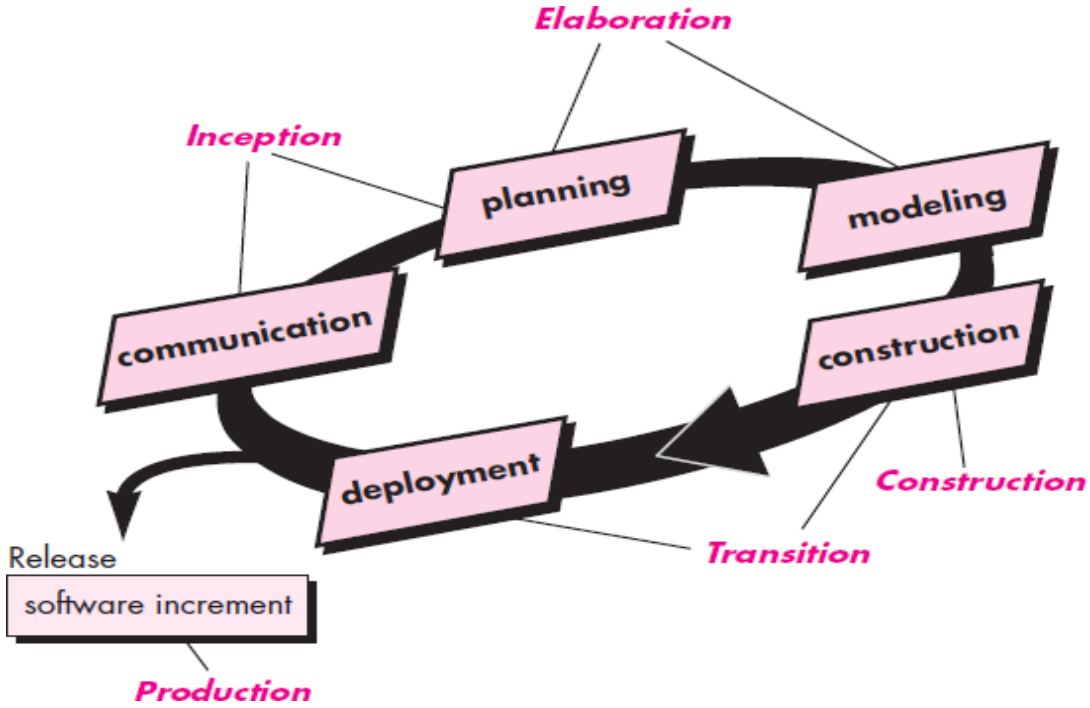

Fig: Unified Process Model.

3. **Construction Phase:** The Construction Phase is same as construction activity, where the application is coded and tested. The Construction Phase develops suitable code for each component of the software. To do this, analysis and design models started in the elaboration phase are completed to reflect the final version of software increment.
4. **Transition Phase:** The Transition Phase encompasses latter stages of construction and first part of deployment activities. Software is given to end-users for beta testing and user feedback about both defects and necessary changes. Software team creates necessary support information (Ex: user manuals, installation procedures) required for release.
5. **Production Phase:** In the Production Phase, on-going use of software is monitored, support for operating environment is provided and defect reports and request for changes are submitted and evaluated.

❖ Construction, Transition and Production phases are being conducted concurrently sometimes. So, five Unified Process phases do not occur in a sequence.